



**UNIVERSIDAD
SAN SEBASTIAN**

**ESCUELA DE INGENIERÍA CIVIL INFORMÁTICA
FACULTAD DE INGENIERÍA Y TECNOLOGÍA
SEDE SANTIAGO, BELLAVISTA.**

Sistema de Visualización con Proyecciones para Convergencia de Túneles a Gran Profundidad.

Memoria para optar al Título de Ingeniero Civil Informático

Autor: Héctor Alfredo Saavedra Casanova

Profesor Guía: Dr. Mauricio Sepúlveda

Santiago, Chile, abril de 2019

Sistema de Visualización con Proyecciones para Convergencia de Túneles a Gran Profundidad.

Héctor Alfredo Saavedra Casanova



Memoria para optar al título de Ingeniero Civil Informático

Escuela de Ingeniería Civil Informática

Facultad de Ingeniería y Tecnología

UNIVERSIDAD SAN SEBASTIAN

Abril 2019

Calificación De La Memoria

En Santiago, el de de, los abajo firmantes dejan constancia que el alumno **HÉCTOR ALFREDO SAAVEDRA CASANOVA** de la carrera de **INGENIERÍA CIVIL INFORMÁTICA** ha aprobado la Memoria para optar al Título de **INGENIERO CIVIL INFORMÁTICO**, con una nota de

Profesor: Profesor:

Firma: Firma:

Profesor:

Firma:

© Héctor Alfredo Saavedra Casanova

Se autoriza la reproducción parcial o total de esta obra, con fines académicos, por cualquier forma, medio o procedimiento, siempre y cuando se incluya la cita bibliográfica del documento.

Sistema de Visualización con Proyecciones de Convergencia para túneles a Gran Profundidad.

Santiago, Chile, abril de 2019.

“Esto es para los locos, los inadaptados, los excéntricos, los problemáticos, los de forma abstracta en un mundo cuadrado, los que ven las cosas de manera diferente. Para quienes no siguen las reglas y no sienten ningún respeto por el status quo. Personas a las que puedes citar, discrepar, glorificar o villanizar. Pero a quienes lo único que no puedes hacer, es ignorar, porque ellos cambian las cosas, empujan a la raza humana hacia adelante. Y mientras algunos los ven como locos, deberían ser vistos como genios. Porque sólo las personas que están tan locas como para pensar que pueden cambiar el mundo, son las que lo hacen.”

Dedicado a mis Padres...

Dedicated to my Parents...

Dédié à mes Parents...

Meine Eltern Gewidmet...

Dedykowane moim Rodzicom...

Dedicato ai miei Genitori...

Agradecimientos

Primeramente, debo agradecer a toda mi familia, a mis padres y a mi hermana. Pilares fundamentales en mi vida, cuyo esfuerzo y apoyo incondicional son quienes han me han impulsado y permitido tomar y realizar todas las oportunidades que se me han presentado, junto a todos los riesgos y costes que eso conlleve. Sin ellos este trabajo que ustedes leen en este momento, no existiría.

Quiero agradecer a mis amigos, por todos los buenos momentos, bullying y risas. Como también por toda la ayuda brindada a lo largo de estos años. Sin ninguna duda puedo decir que la amistad que construimos en la Universidad seguirá fuera de esas cuatro paredes y por mucho tiempo más. Ya que son contadas las veces que me vieron dentro de ella.

Quiero agradecer a mi mentor el Dr. Mauricio Sepúlveda, por aceptarme como su alumno a guiar, proponerme este tema y generar los contactos para la realización de esta investigación. Recibirme siempre que lo necesité, incluso cuando por mis irresponsabilidades llegué tarde a nuestras reuniones.

Al Profesor Rodolfo Cabezas por trabajar conmigo en esta investigación entregando sus conocimientos en el tema base de esta investigación, como lo es la minería y entregando mediciones para realizar las pruebas.

Quiero agradecer a mi director de carrera don Dagmar Pearce, quien dio su voto de confianza en mí desde el inicio, aceptándome como alumno de admisión especial. Por el apoyo por todo el proyecto que construimos durante los años que fui miembro de la directiva del centro de estudiantes de la carrera, como en los eventos y actividades que logramos concretar y las competencias que logramos ganar. Siempre con el fin de obtener notoriedad y ser un referente de la Facultad. También por recibirme siempre que lo necesité en su oficina y dispuesto a escuchar y ayudar, aunque él estuviera muy ocupado. Por haber generado las gestiones y recomendaciones que me permitieron realizar un intercambio estudiantil.

Para terminar, agradecer a toda la Facultad de Ingeniería y a los docentes que me formaron.

Tabla de Contenido

CALIFICACIÓN DE LA MEMORIA.....	III
AGRADECIMIENTOS.....	VII
LISTA DE FIGURAS.....	X
LISTA DE TABLAS.....	XI
LISTA DE ECUACIONES.....	XI
RESUMEN.....	XII
ABSTRACT.....	XIII
GLOSARIO.....	XIV
INTRODUCCIÓN.....	1
CAPÍTULO 1. ANTECEDENTES DEL TEMA DE INVESTIGACIÓN.....	2
1.1. Tema de Investigación.....	2
1.2. Descripción de la investigación.....	2
1.3. Planteamiento del Problema.....	2
1.4. Objetivos.....	3
1.4.1. <i>Objetivo General</i>	3
1.4.2. <i>Objetivos Específicos</i>	3
1.5. Justificación e Importancia.....	4
1.6. Alcances y Limitaciones.....	4
1.6.1. <i>Alcances</i>	5
1.6.2. <i>Limitaciones</i>	5
CAPÍTULO 2. MARCO TEÓRICO.....	7
2.1. Mercado Actual.....	7
2.1.1. <i>Beneficios que se Esperan con el Desarrollo del Proyecto</i>	7
2.2. Bases Teóricas.....	8
2.2.1. <i>Uso de Sistema LiDAR para Recopilación de Datos en Excavación</i>	8
2.2.2. <i>Aceleración Grafica</i>	11
2.2.2.1. <i>Funcionamiento de la Aceleración Grafica</i>	12
2.2.3. <i>Medidas de Convergencias</i>	14
CAPÍTULO 3. SOLUCIÓN AL PROBLEMA.....	15
3.1. Diseño de Prototipo.....	15

3.1.1. <i>Caso de Uso</i>	15
3.2. Sistema de utilización de procesamiento gráfico, para análisis masivo de datos.	16
3.2.1. <i>Web Graphics Library (WebGL)</i>	17
3.2.2. <i>Three.js</i>	19
3.2.3. <i>Procesamiento de nubes de punto</i>	20
3.2.3.1 Árbol Octal Anidado Modificable (MNO).....	22
3.2.3.2 Atributo de Coloreado de Puntos.	23
3.2.4. <i>Creación del Escenario de Renderizado</i>	25
3.2.4.1. Escena.	26
3.2.4.2. Cámara.	26
3.2.4.3. Renderer.....	27
3.2.4.4. Generando la Escena.	27
3.2.5. <i>Patrón de Arquitectura</i>	28
3.2.6. <i>Sistema Multiplataforma</i>	29
3.2.6.1. Electron.	29
3.2.6.2. Configuración Inicial Electron.	31
3.2.7. <i>Arquitectura de la Aplicación</i>	34
3.3. Propuesta de Apoyo a Calculo de Medidas de Convergencia entre Puntos de Control de una Excavación.	35
3.3.1. <i>Árbol Cuaternario (Quadtree)</i>	36
3.3.2. <i>Comparación de mediciones LiDAR</i>	37
3.3.2.1. Algoritmo Iterative Closest Point (ICP) y Nearest Neighbors Search (NNS).	38
3.3.2.2. Cálculo de distancias.....	39
3.3.3. <i>Proceso para el cálculo de distancias entre secciones medidas en el tiempo</i>	41
CAPÍTULO 4. PRUEBAS Y RESULTADOS.....	42
4.1. Equipo de Pruebas.	42
4.2. Carga de Archivo.....	43
La selección de los distintos métodos en si son indiferentes y no representan en estos momentos impacto alguno en los tiempos de carga y ejecución. Ya que en estas pruebas la lectura del archivo siempre se realizó dentro del mismo equipo. Por lo tanto, el factor de carga corre por parte de la velocidad de lectura/escritura del disco duro.....	44
4.3. Tiempo de Carga.	44
4.4. Resolución Visual.....	46
4.5. Coloración.	49
4.6. Alineamiento en árbol cuaternario.....	50

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES.....	52
REFERENCIAS BIBLIOGRÁFICAS.....	53

Lista de Figuras

Figura 1: La imagen presenta el principio de como los instrumentos LiDAR determinan la posición en el espacio 3D. (Haugland 2010).	9
Figura 2: Visibilidad de las fracturas en relación con la línea de visión del escáner. (Lato and Diederichs 2010).	10
Figura 3: Catálogo de Aplicaciones con Aceleración por GPU (Nvidia)	12
Figura 4: Como funciona la aceleración gráfica en una aplicación. (Nvidia).	13
Figura 5: Apreciación de la diferencia entre la arquitectura entre CPU (izquierda) y GPU (derecha). (Nvidia).	13
Figura 6: Caso de Uso de Usuario	16
Figura 7: Compatibilidad por Dispositivos para WebGL (WebGLStats).....	18
Figura 8: Usuarios por Navegador Web medidos para compatibilidad por WebGL (WebGLStats).	18
Figura 9: Usuarios de Sistema Operativos medidos para compatibilidad con WebGL (WebGLStats).	18
Figura 10: Los nodos de bajo nivel(izquierda) contienen modelos dispersos en grandes regiones. Cada nivel aumenta exponencialmente la cantidad de puntos y detalle. (Scheiblaue 2014).	21
Figura 11: Formato en como los puntos son recibidos.	24
Figura 12: Arquitectura de un punto de control identificado.....	24
figura 13: Generación básica de escena con Three.js	27
figura 14: Etapas proceso de renderizado, desde ingreso de datos cargados desde la nube de puntos, hasta que la figura es mostrada en pantalla.	28
figura 15: Representación visual del patrón de diseño Modelo-Vista-Controlador	28
figura 16: Procesos de Electron y comunicación entre ellos (Electron).	31
figura 17: Estructura package.json.....	32
figura 18: Código del main.js.....	33
figura 19: Arquitectura General del Sistema.	34
figura 20: Representación espacial (izquierda) y representación lógica (derecha) de un árbol cuaternario. (Apple)	37
figura 21.: Correspondencia estimada entre nube de puntos (P) y nube de puntos (Q).....	39
figura 22: Distancia medida respecto a nodo más cercano según algoritmo NNS y distancia calculada aplicando la ecuación de Hausdorff.	40
figura 23: Flujo del proceso para el cálculo de distancias entre secciones medidas en el tiempo.....	41
figura 24: Cargar nube de puntos almacenada dentro de la aplicación.....	43
figura 25: Cargar Nube de puntos directamente desde disco duro.	43
figura 26: Cargar nube de puntos utilizando el método de drag&drop.....	44
figura 27: Renderizado 3D del túnel al 10% de densidad de datos.	47
figura 28: Renderizado 3D del túnel al 100% de densidad de datos.	47

figura 29: Renderizado 3D con tamaño de puntos al máximo. (10% densidad).....	48
figura 30: Renderizado 3D con tamaño de puntos al máximo. (100% densidad).....	48
figura 31: Coloración RGB de sección según tabla de rangos de índice.	49
figura 32: Representación espacial de árbol cuaternario.	50
figura 33: Alineamiento de secciones mediante la ejecución de algoritmo ICP y NNS.....	51

Lista de Tablas

Tabla 1: Comparación de librerías WebGL.	19
Tabla 2: Tabla de coloración de punto, según su valor de índice.	25
Tabla 3: Especificaciones técnicas equipo de pruebas.	42
Tabla 4: Archivos de nubes de punto a analizar.	45
Tabla 5: Resultados de rendimiento calculado.	45

Lista de Ecuaciones

Ecuación 1: Función de Error	38
Ecuación 2: Distancia de Hausdorff	40

Resumen

En la presente memoria se expone una solución informática y se desarrolla un software prototipo que permite visualizar un renderizado 3D de una excavación, conformada por la totalidad de los puntos medidos de las paredes de un túnel. Junto con la proyección de secciones, lo que permite observar el daño medido al interior del macizo rocoso. Además de una propuesta que permita el cálculo de convergencia para realizar un análisis del comportamiento del macizo rocoso durante el ejercicio de la actividad minera en excavaciones a gran profundidad.

Este trabajo nace desde la investigación que se basa en encontrar una relación directa entre las medidas de convergencias con el daño en profundidad que sufre el macizo luego de ser excavado o tronado y que lidera el Sr Rodolfo Cabezas, candidato a Doctor en Minería.

Este trabajo tiene altas proyecciones científicas y comerciales porque el procesamiento de datos obtenidos gracias a la implementación de tecnología LiDAR para evaluar deformaciones en excavaciones a gran profundidad, automatiza la observación lo que permite disminuir el riesgo, en cierta medida, de realizar el proceso tradicional en la medición manual de los índices de deformación en locaciones con un alto riesgo de colapso.

PALABRAS CLAVE: LiDAR; Renderizado 3D; Convergencia; Excavación; Túnel.

Abstract

In the present report a computer solution is exposed and a prototype software is developed that allow to visualize a 3D rendering of an excavation, formed by all the measured points of the walls of a tunnel. Together with the projection of sections, it allows to observe the damage measured within the rock mass. In addition to a proposal that allows the calculation of convergence to perform an analysis of the behaviors of the rock mass during the exercise of mining activity in deep excavations.

This work is thought from the research that is based on finding a direct relationship between the measures of convergence with the damage in depth suffered by the rock mass after being excavated or thundered and lead Mr. Rodolfo Cabezas, Doctor candidate in Mining.

This work has high scientific and commercial projections because the processing of data obtained thanks to the implementation of LiDAR technology to evaluate deformation in deep excavations, automates the observation which reduces the risk, to a certain extent, of carrying out the traditional process in the manual measurement of deformation rates in locations with a high risk of collapse.

KEY WORDS: LiDAR; 3D Rendering; Convergence; Excavation; Tunnel.

Glosario

Aliasing: aplicado al campo de la imagen digital, se refiere a un efecto indeseado resultante de la degradación de la calidad de la imagen.
API (<i>Application Programming Interface</i>): conjunto de reglas y especificaciones, que las aplicaciones pueden seguir para comunicarse entre ellas.
Árbol abstracto de sintaxis (AST): estructura de datos usada extensamente en compiladores, debido a su propiedad de representar la estructura del código de un programa.
Árbol cuaternario (<i>Quadtree</i>): estructura de datos para la organización de objetos en función a su ubicación en un espacio en bidimensional.
Convergencia: medida de los movimientos relativos entre dos puntos fijos colocados en la superficie de un espacio subterráneo. Según si los puntos se aproximen o se alejen entre sí. Su cuantificación permite graficar.
CPU (<i>Unidad central de procesamiento</i>): procesador que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.
Desprendimiento: movimientos de inestabilidad
Desplazamiento: punto en el cual la deformación deja de ser elástica y pasa a ser irreversible (plástica).
Diagnóstico: análisis que se realiza para determinar cualquier situación, se realiza sobre datos y hechos recogidos y ordenados, que permiten juzgar mejor qué es lo que está pasando.
ECMAScript: es una especificación de lenguaje de programación publicada por <i>ECMA International</i> . El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por <i>Netscape Communications Corporation</i> . Actualmente está aceptado como el estándar <i>ISO 16262</i>
Electron: es un <i>framework</i> de código abierto, permite el desarrollo de aplicaciones gráficas de escritorio usando componentes del lado del cliente y del servidor originalmente desarrolladas para aplicaciones web.
Excavación: operación minera en el túnel de carácter destructiva, es decir se remueve o destruye material rocoso del macizo con el fin de mayor exploración y ampliación de la obra.
Fallamiento: En geología, una falla es una fractura o zona de fracturas a lo largo de la cual ha ocurrido un desplazamiento relativo de los bloques paralelos a la fractura (Bates y Jackson, 1980). Esencialmente, una falla es una discontinuidad que se forma debido a la fractura de grandes bloques de rocas en la Tierra cuando las fuerzas tectónicas superan la resistencia de las rocas.
Fractura: es una grieta o discontinuidad del terreno producida por fuerzas tectónicas. Se forman cuando se supera la resistencia mecánica del terreno a la deformación y se rompe.

Framework: es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
GPU (<i>Unidad de procesamiento gráfico</i>): coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador en aplicaciones como videojuegos o aplicaciones 3D. De esta forma, mientras gran parte de lo relacionado con los gráficos lo procesa la GPU. La CPU puede dedicarse a otro tipo de cálculos.
Grieta: abertura larga y estrecha producto de la separación de dos materiales.
Incidencia: relación entre el número de casos a lo largo de un periodo de tiempo en que se diagnostican deformaciones, fallas, fracturas, etc. En un macizo rocoso.
Investigación: proceso metódico, riguroso, cuidadoso y sistemático que se aplica al estudio de un fenómeno, para intentar medirlo o entenderlo. Busca resolver problemas o preguntas científicas, mediante la construcción de nuevos conocimientos.
Java: lenguaje de programación de propósito general, concurrente, orientado a objetos.
JavaScript: lenguaje de programación interpretado, dialecto del estándar ECMAScript.
LiDAR (<i>Light Detection and Ranging o Laser Imaging Detection and Ranging</i>): es un dispositivo que permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser. La distancia al objeto se determina midiendo el tiempo de retraso entre la emisión del pulso y su detección a través de la señal reflejada.
Macizo rocoso: conjunto tanto por bloques de matriz rocosa y distintos tipos de discontinuidades que afectan al medio rocoso. Para el estudio del comportamiento mecánico del macizo rocoso se debe analizar las propiedades de la matriz rocosa y de las discontinuidades.
Mediciones: resultados obtenidos a través de monitoreo.
Offline: plataforma que no requiere de una conexión constante a internet para su funcionamiento y manejo de datos.
Open Source: sistema de código abierto que se puede redistribuir y modificar,
OpenGL: especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.
Operación Minera: proceso en que se reconocen las características del yacimiento minero y se decide extraerlo, se inicia el desarrollo y construcción de la mina, para luego pasar al proceso de extracción del mineral.
Nube de Puntos (<i>PointCloud</i>) conjunto de vértices en un sistema de coordenadas tridimensional. Estos vértices se identifican habitualmente como coordenadas (x, y, z) y son representaciones de la superficie externa de un objeto.
Renderizado (<i>Rendering</i>): anglicismo usado en informática para referirse al proceso de conversión automática de modelos 3D en imágenes 2D con efectos 3D.
Resistencia: esfuerzo que resiste la roca bajo ciertas condiciones, deformaciones.
Toma de decisión: proceso en realiza elecciones entre diferentes operaciones o situaciones.

WebGL: especificación estándar que define una API implementada en JavaScript para la renderización de gráficos en 3D dentro de cualquier navegador web.

Introducción

Chile es conocido mundialmente por su actividad minera, teniendo excavaciones a más de mil metros de profundidad, siendo la mina El Teniente, la mina subterránea más grande del mundo, que se espera que alcance los 1880 metros de profundidad durante el año 2019 (Codelco 2016). Y así mismo tanto en Chile como al rededor del mundo se hace cada vez más común proyectos de minería que se adentran a más profundidad. Por ejemplo, la mina de Chuquicamata que, siendo la mina a rajo abierto más grande del mundo, planea una gigante operación subterránea que permitirá explorar parte de los recursos que quedarán bajo el actual yacimiento, esperando que sea una de las más grandes, modernas y eficientes del mundo (Codelco 2017).

Sin embargo, las operaciones a tales profundidades conllevan un alto riesgo, en una excavación subterránea es el mismo macizo rocoso el que mantiene la estructura. Y por tanto cargas que le afectan pueden comprometer la capacidad de resistencia de la roca, como por ejemplo las actividades de excavación.

Accidentes ocasionados por colapso en túneles por la deformación de paredes producto de cambios en la integridad del macizo rocoso, tienen consecuencias graves, tanto a nivel operacional, logístico, económico y humano. Es importante contar con procesos de control para hacer seguimiento del comportamiento de las paredes del túnel en el macizo en el tiempo, como también metodologías que permitan predecir de acuerdo a ese comportamiento, el estado interno del mismo.

Este estudio se basa en apoyo a esa investigación, entregando y proponiendo herramientas informáticas que faciliten el computo, presentación de datos, y permitan el análisis de convergencias en las paredes del macizo.

Capítulo 1. Antecedentes del Tema de Investigación.

1.1. Tema de Investigación.

Sistema de visualización con proyecciones para convergencia de túneles a gran profundidad.

1.2. Descripción de la investigación.

La investigación consta en diseñar un sistema que utilice las nubes de puntos obtenidas mediante LiDAR, en una excavación minera a gran profundidad. Además, utilizando la unidad de procesamiento gráfico (GPU) para realizar una representación tridimensional de la excavación medida y de la proyección de la sección hacia la profundidad del macizo. Proponiendo también un método que permita ver el comportamiento de las paredes de las secciones en el macizo rocoso en el tiempo, utilizando las medidas de convergencia para observar y analizar las fallas, desprendimientos, deformaciones, etc. En la profundidad del macizo, provocados por la actividad minera.

1.3. Planteamiento del Problema.

Como se presentaba con anterioridad, en operaciones mineras a grandes profundidades, distintos factores pueden afectar el comportamiento del macizo rocoso, que puede repercutir en desprendimientos, grietas y fallas que se pueden propagar al interior de las paredes del túnel, entre otros eventos.

Predecir las inestabilidades en las excavaciones subterráneas es importante porque permite idear estrategias de operación para el control y refuerzo de estas. Con el fin de minimizar los riesgos.

1.4. Objetivos.

Se plantearán los objetivos que delimitarán las etapas de realización para la elaboración de esta investigación, y lo que se pretende proponer y realizar con la misma.

1.4.1. Objetivo General.

El objetivo general de esta investigación es la propuesta de una plataforma que realice el renderizado tridimensional de una excavación minera, que permita mediante el procesamiento de una nube de puntos, apoyo en el análisis de las medidas de convergencias en el tiempo.

1.4.2. Objetivos Específicos.

Se definen los siguientes objetivos específicos:

1. Estudio teórico sobre funcionamiento de recopilación de datos mediante LiDAR.
2. Diseño de sistema utilizando recursos de procesamiento grafico para análisis masivo de datos.
3. Planteamiento, desarrollo y pruebas de un software prototipo, el cual contenga las siguientes características:
 - a. Genere un modelo 3D al recibir un archivo de puntos coordenados.

- b. Aplicación multiplataforma, que permita su ejecución tanto en los sistemas operativos Windows, Mac, Linux, en modalidad offline.
- 4. Generar una propuesta con la utilización de algoritmos y ecuaciones que den una solución al cálculo de medidas de convergencia entre los puntos de control, para distintas mediciones en el tiempo de una sección.

1.5. Justificación e Importancia.

En relación a la problemática para abordar la predicción de inestabilidades es que se han llevado diversos estudios, con metodologías, teorías y criterios aceptados y aplicados. Más aún es un campo en constante estudio, los factores del ambiente como composición del macizo, geometría del túnel, extensión, profundidad, etc. Presentan distintos escenarios muy diversos para analizar.

Para realizar una medición más exhaustiva es que se utilizan estaciones LiDAR para el escaneo de las paredes de la excavación, obteniendo nubes de punto con una cantidad masiva de datos.

Las mediciones de LiDAR son utilizadas en la mecánica de rocas, para identificar orientación, apertura, persistencia y rugosidad de discontinuidades.

Sin herramientas informáticas especializadas es imposible analizar la magnitud de datos obtenidos, más los que se deben ser generados, luego de aplicados los criterios geo mecánicos, para el cálculo de proyecciones en el macizo.

1.6. Alcances y Limitaciones.

A continuación, se presentan los alcances y limitaciones relacionadas al ejercicio teórico y de desarrollo de esta investigación.

1.6.1. Alcances.

- La solución se centrará en mecanismos de renderizado tridimensional de una excavación minera, cuyos datos fueron proporcionados. Se desarrollará un prototipo funcional con las capacidades de representación de la excavación mencionada.
- Para la presentación de la solución de Quadtree en el cálculo de deformaciones en el tiempo, los datos presentados serán una exageración de la realidad para explicar el trasfondo teórico del método, al no contar con mediciones reales en distintos instantes de tiempo.
- Existiendo más alternativas, las opciones de desarrollo fueron pensadas para obtener un programa que no necesite de instalación de paquetes ni librerías previas, en ninguno de los tres sistemas operativos compatibles.

1.6.2. Limitaciones.

Este sistema cuenta con la capacidad de hacer una representación tridimensional, tanto de las paredes de la excavación como de sus proyecciones interiores, mas no cuenta con la integración de las formulas y criterios necesarios para realizar dichos cálculos.

Se considera que el software será ejecutado en equipos que cuenten con hardware de procesamiento gráfico capaz de realizar las operaciones de renderizado o de cálculo. Por lo que puede presentar latencia y overheating en equipos que cuenten con chipset gráfico dependiente del CPU.

Capítulo 2. Marco Teórico.

El capítulo entregará la definición de los conceptos y la base teórica, con la que se trabajará en el proyecto, Estos estarán directamente relacionado con las tecnologías de la información utilizadas en la resolución de la solución.

2.1. Mercado Actual.

En el mercado existen empresas focalizadas en el desarrollo de software para minería o construcción civil que puede ser utilizado para realizar los estudios de observación del comportamiento del macizo, pero estas no son exclusivamente diseñadas para efectuar análisis en excavaciones a gran profundidad. Aunque muchas herramientas si pueden ser programadas para cumplir tal función (ITASCA), la documentación para el aprendizaje del lenguaje de programación está restringida y este al ser propietario de esta suite de programas, tiene limitaciones sobre realizar peticiones de recursos al sistema operativo como podría realizarse con un lenguaje más estandarizado.

Otro factor importante a considerar es el alto costo de las licencias.

2.1.1. Beneficios que se Esperan con el Desarrollo del Proyecto.

Los beneficios que se esperan obtener con el desarrollo de este sistema de visualización con proyecciones para convergencias de túneles a gran profundidad, es que permita identificar de

forma temprana anomalías que estén ocurriendo en las paredes del túnel y en el interior del macizo rocoso, para realizar las operaciones de reparación de las mismas. Y el seguimiento correspondiente a las zonas comprometidas y así evitar derrumbes. Además, se espera dar pie a una siguiente etapa de desarrollo, en la que conste la integración de ambas investigaciones con el fin de automatizar y utilizar los recursos del motor de procesamiento gráfico para realizar los cálculos de las proyecciones de una excavación en tiempo real. De esta manera, el usuario podrá obtener información importante para el diagnóstico de incidencias, como también apoyo en la toma de decisiones.

2.2. Bases Teóricas.

Se procederá a realizar un análisis y presentación de los conceptos teóricos necesarios para la resolución y creación del sistema, en base a los objetivos planteados en el Capítulo 1.

2.2.1. Uso de Sistema LiDAR para Recopilación de Datos en Excavación.

Si bien esta investigación no se basa en la obtención de los datos, es necesario conocer la teoría y los conceptos matemáticos detrás del sistema de escáner LiDAR, y la opción para hacer un modelo representativo de una superficie escaneada.

En general, las principales características que identifican a un sistema LiDAR son: mayor distancia de observación (respecto a otro tipo de equipos de medición equivalente), la velocidad de exploración, calidad medida en términos de precisión, exactitud y repetitividad. Su proceso de

funcionamiento se basa en un rayo láser que se va desviando sobre una rejilla angular muy precisa, y calcula las coordenadas para cada punto medido en un marco de referencia interno (Haugland 2010). Las coordenadas esféricas se obtienen calculando la distancia (d) al objeto reflectante junto con la componente angular horizontal (φ) y vertical (α) de la dirección del rayo láser, como se aprecia en Figura 1.

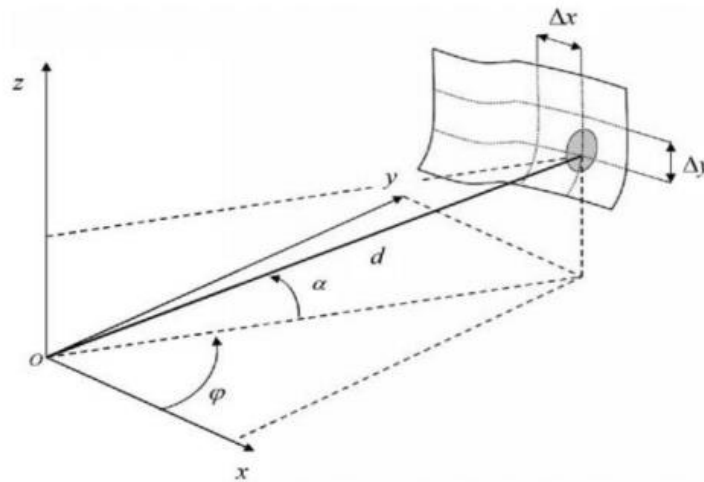


Figura 1: La imagen presenta el principio de como los instrumentos LiDAR determinan la posición en el espacio 3D. (Haugland 2010).

Los puntos se convierten luego al sistema de coordenadas cartesiano (x, y, z). Todos los puntos son medidos en relación con la posición del escáner, que se define como el origen (0,0,0). Adicionalmente, el láser mide la intensidad (i), basado en el poder del rayo reflejado. La señal posterior dispersada depende de la humedad, el color y la rugosidad de la superficie. Los puntos reflejados al escáner, consisten en la información de coordenadas e intensidad dando como resultado una “nube de puntos” que forma un modelo 3D de la superficie escaneada (Kemeny and Turner 2008).

Como ya se comentó la capacidad del escáner para identificar y medir la intensidad al retornar es dependiendo del material en el cual es dirigido el rayo. También dependen del ángulo de ataque a la superficie y los objetos que bloquean la línea de visión. Si la medición se realiza en una estructura compleja donde la superficie varia en diferentes orientaciones, se recomienda realizar dos o tres exploraciones de la misma superficie. Las zonas oscuras resultantes se conocen como oclusiones y se producen cuando no se pueden muestrear partes de la superficie de la roca, debido a orientaciones desfavorables en relación con el escáner y la línea de visión (Lato and Diederichs 2010). Esto puede provocar vacíos en el *dataset* de la nube de puntos. Lo anteriormente mencionado se ilustra en la Figura 2.

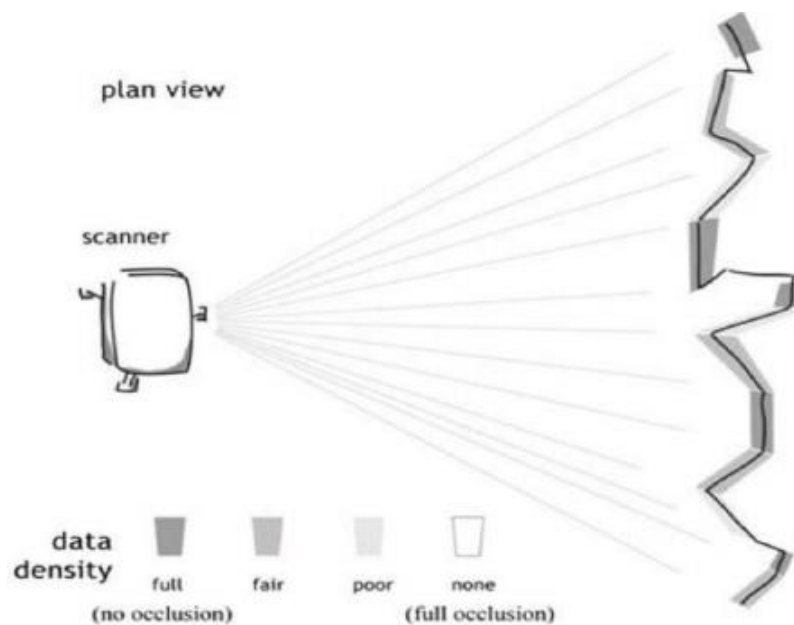


Figura 2: Visibilidad de las fracturas en relación con la línea de visión del escáner. (Lato and Diederichs 2010).

2.2.2. Aceleración Gráfica.

Desarrollada e impulsada por Nvidia para aplicaciones visuales desde el 2007, la computación acelerada gráficamente es el uso de la GPU junto con la CPU para acelerar aplicaciones. Esto permite a los científicos de datos e investigadores abordar algunos de los problemas más desafiantes del mundo de distintas magnitudes, más rápido de lo posible con las arquitecturas de procesamiento tradicionales.

La aceleración gráfica está revolucionando el alto rendimiento de la computación a gran escala. Según Wu, R., Zhang, B. & Hsu, M. (2009), en las primeras aplicaciones de ciencia de datos los procesos acelerados que se realizaron con la memoria de GPU, se obtuvieron velocidades entre 200 a 400 veces más rápido que ejecutándolos en un solo núcleo de CPU, y aproximadamente entre 50 y 88 veces más rápido que aplicaciones altamente optimizadas para ejecución en CPU en un solo núcleo.

La utilización de esta tecnología ha progresado dentro de diferentes áreas de la ingeniería, analítica y matemática, como lo es la ciencia de datos, machine learning y deep learning, defensa e inteligencia, videojuegos, supercomputación y visualización científica.

Nvidia mantiene un catálogo de aplicaciones con aceleración gráfica y la documentación para que los desarrolladores implementen esta tecnología en sus aplicaciones.

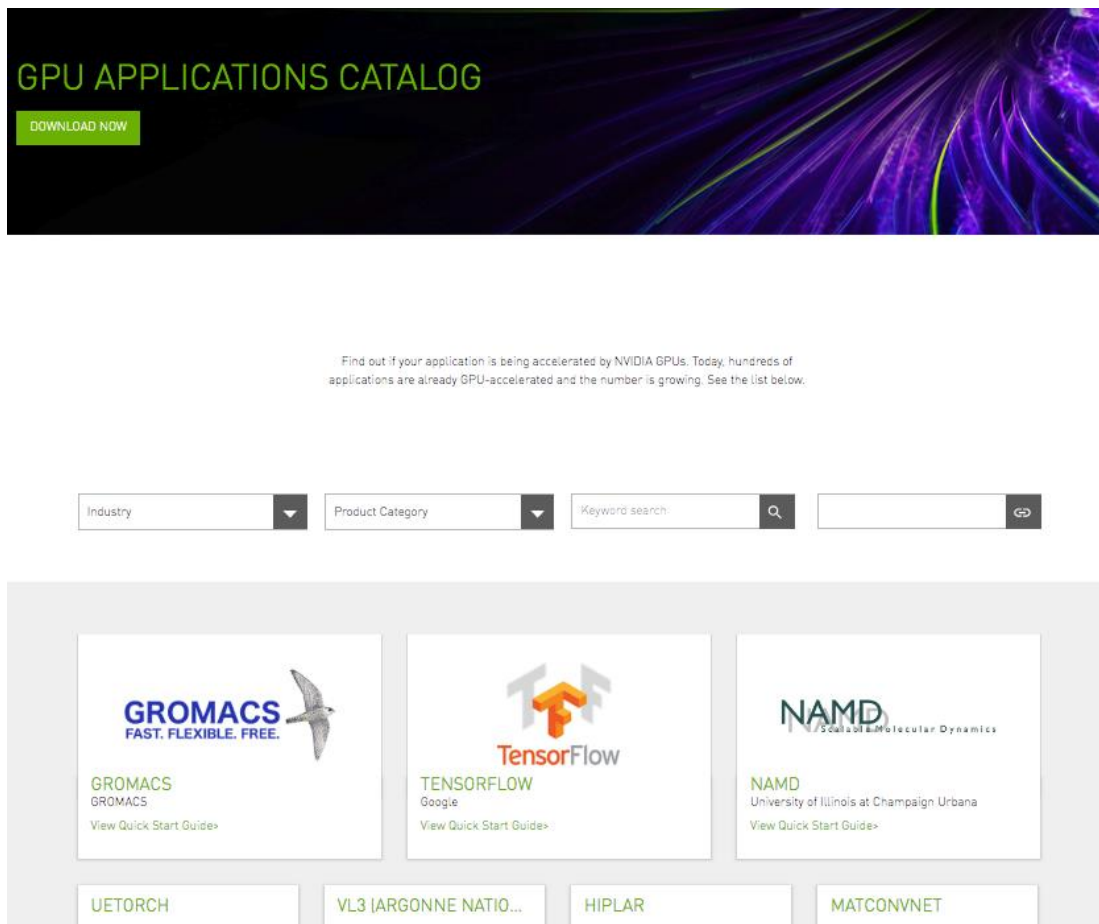


Figura 3: Catálogo de Aplicaciones con Aceleración por GPU (Nvidia)

2.2.2.1. Funcionamiento de la Aceleración Gráfica.

La computación acelerada por GPU, se basa en el enfoque *divide y vencerás* descargando partes de uso intensivo de computo de una aplicación a la GPU, mientras el resto del código aún se ejecuta en la CPU, como puede ser apreciado en la figura 4. Desde la perspectiva del usuario, las aplicaciones habilitadas para utilizar procesamiento gráfico sólo se ejecutan mucho más rápido que las aplicaciones de CPU. Una forma sencilla de entender la diferencia entre la GPU y CPU es comparando como procesan las tareas. La CPU se conforma en pequeños núcleos optimizados

para realizar procesamiento secuencial en serie. Mientras que la GPU posee una masiva arquitectura paralela, albergando miles de pequeños y más eficientes núcleos, diseñados para ejecutar múltiples tareas de manera simultánea como se observa en la figura 5.

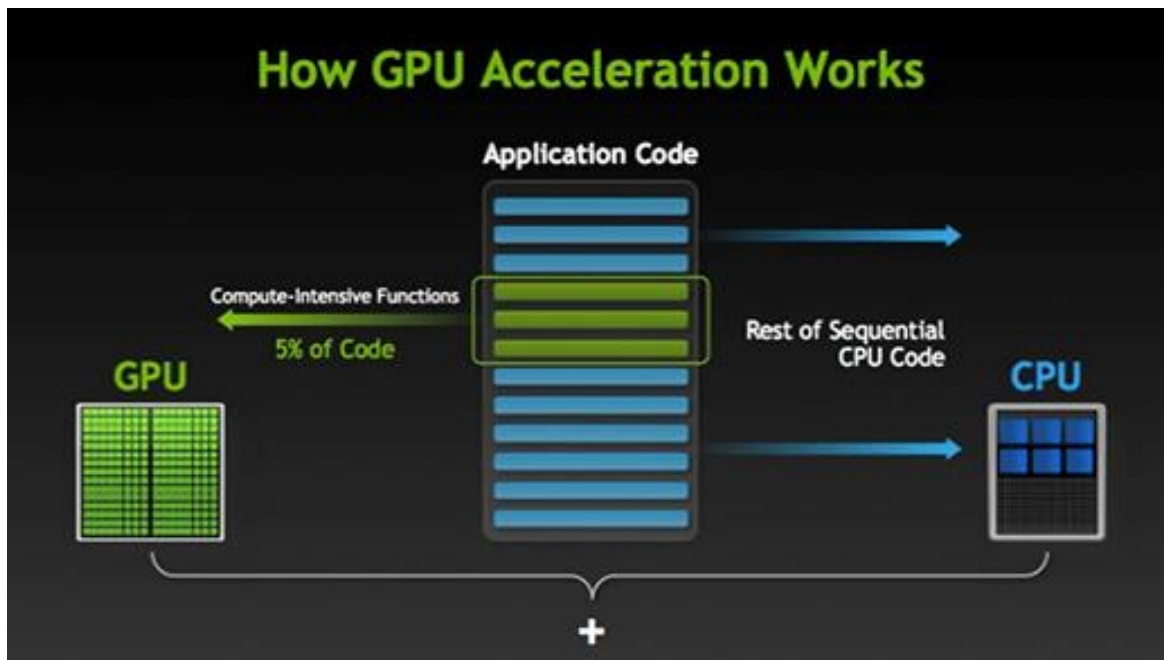


Figura 4: Como funciona la aceleración gráfica en una aplicación. (Nvidia).

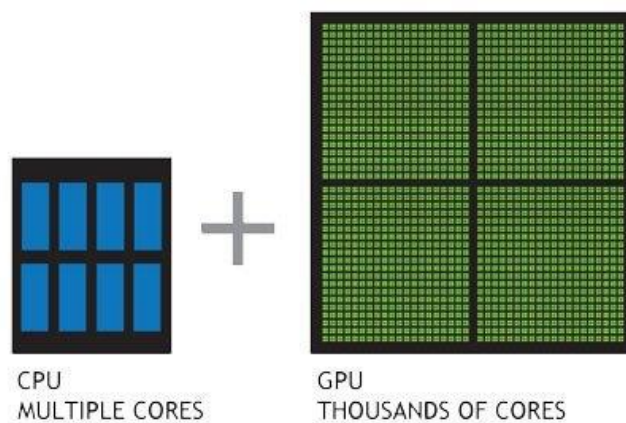


Figura 5: Apreciación de la diferencia entre la arquitectura entre CPU (izquierda) y GPU (derecha). (Nvidia).

Es por esta razón que las GPU son ideales para realizar trabajo analítico de cargas de computo intensivas en grandes conjuntos de datos, fundamentalmente para la exploración, medición y visualización de datos.

2.2.3. Medidas de Convergencias.

Durante el ejercicio de las actividades de excavación en un macizo rocoso, se provocan inestables desequilibrios. Por consiguiente, desplazamientos convergentes que se deben controlar sistemáticamente. Estos desplazamientos convergentes o convergencias se les llama a los movimientos relativos producidos entre dos puntos del interior del túnel.

Las medidas de convergencia son consideradas una metodología económica para conocer el comportamiento del material (R. Cabezas 2019).

Cabe mencionar que en cada caso los puntos seleccionados para realizar el análisis de convergencias del túnel varían, esto relacionado con la geometría que presenta el propio túnel. Además de la geometría, existen otros factores que provocan que la deformación en un macizo sea diferente en cada caso. Y por lo tanto eso repercute en que las mediciones de convergencia en cada excavación tengan a su vez diferentes magnitudes (R. Cabezas 2019).

Capítulo 3. Solución al Problema.

En este capítulo se llevará a cabo desde el planteamiento y diseño de un software prototipo, hasta el razonamiento y creación de una propuesta para el apoyo del cálculo de medidas de convergencias.

3.1. Diseño de Prototipo.

El software prototipo en primera instancia está basado en realizar un renderizado tridimensional de un archivo, cuyo contenido está formado por puntos coordenados (x, y, z). Además, contiene una cuarta variable que representa al índice de resistencia del material, este valor viene calculado y es utilizado para el análisis del daño progresivo, aunque para efectos del prototipo será utilizado como referencia para la coloración de puntos.

3.1.1. Caso de Uso.

Como ya se comentó, el prototipo busca ser sencillo de utilizar, donde se pretende desplegar un modelo visual más que una interacción mayor entre el usuario y el software. Por lo tanto, existe un solo perfil de interacción como puede ser observado en la figura 6 que se presenta a continuación.

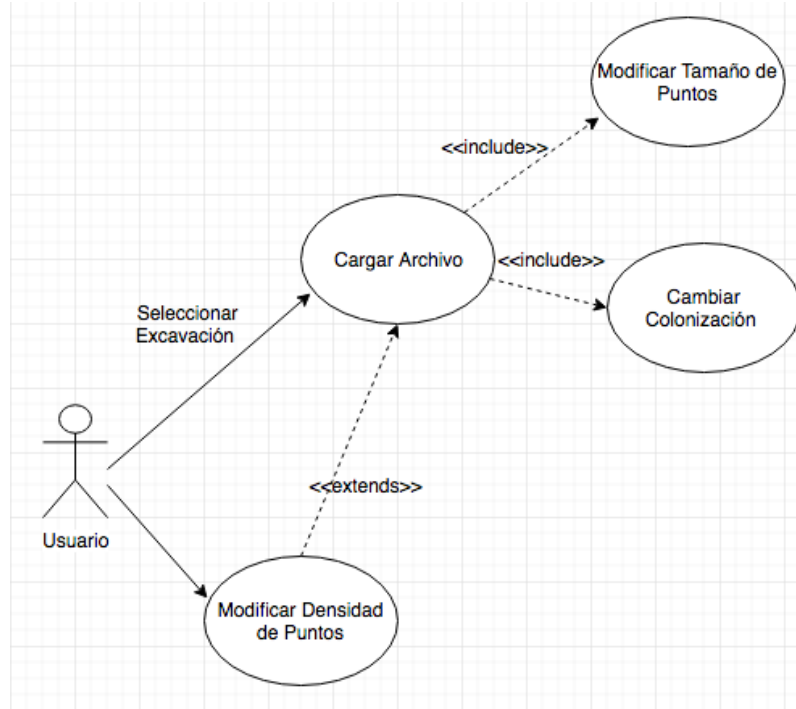


Figura 6: Caso de Uso de Usuario

3.2. Sistema de utilización de procesamiento gráfico, para análisis masivo de datos.

De las librerías gráficas disponibles para realizar el procesamiento de los datos medidos por LIDAR, podemos optar por Direct3D y OpenGL (Khronos 2013), ambas son librerías gráficas robustas a implementar para ejecutar aceleración de hardware gráfico (GPU).

Si bien a simple vista no presentan grandes diferencias en tema de rendimientos. Bajo la superficie si difieren bastante. Mientras Direct3D está basada en la interfaz binaria Windows COM (Component Object Model). OpenGL está basado en una API gráfica parecida a C de bajo nivel. Lo que significa que no depende de ningún lenguaje de programación particular e incluso las librerías de OpenGL están llevadas a los lenguajes más comunes. Direct3D es y siempre será una

API propietaria, lo que significa que el código fuente de la API no está disponible para el usuario final.

Ya que no es Open Source ni multiplataforma, para efectos de esta investigación, utilizaremos OpenGL en su versión web como librería gráfica.

3.2.1. Web Graphics Library (WebGL).

WebGL es una API que está diseñada para realizar renderizado 3D en un navegador web. No precisa de plumines adicionales en cualquier plataforma que soporte OpenGL 2.0 u OpenGL ES 2.0. WebGL es una API DOM, significa que puede ser utilizada desde cualquier lenguaje compatible con DOM, como JavaScript o Java. Es multiplataforma de licencia y código libre. Además, es plug-in free, eso quiere decir que tiene la habilidad de ser implementado dentro de cualquier componente web del navegador sin el uso de software externo. Esta característica puede ser vista por las siguientes figuras 7,8 y 9 a continuación, las que representan según mediciones realizadas cada 30 días por la (WebGLStats), en donde a medida que se desarrollan nuevas librerías u optimizaciones para la API de WebGL, se mide la compatibilidad entre los distintos dispositivos, sistemas operativos y navegadores disponibles.

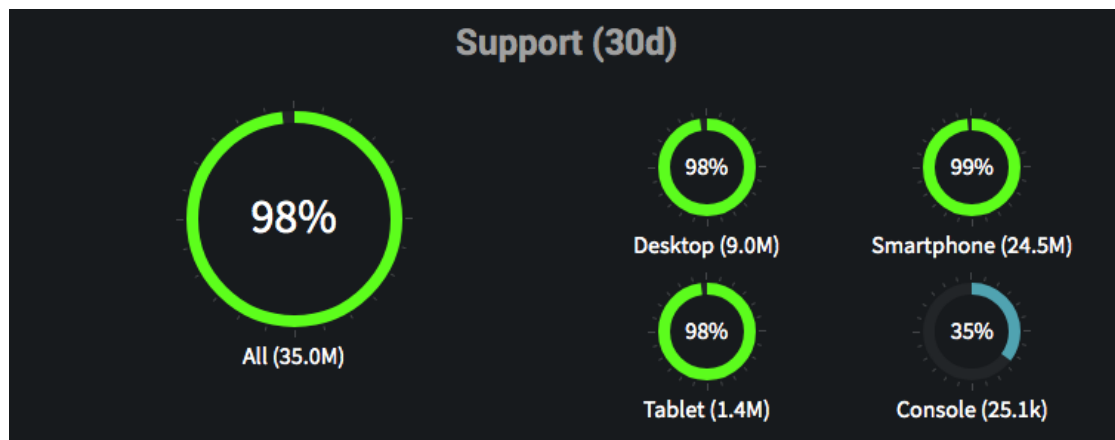


Figura 7: Compatibilidad por Dispositivos para WebGL (WebGLStats).

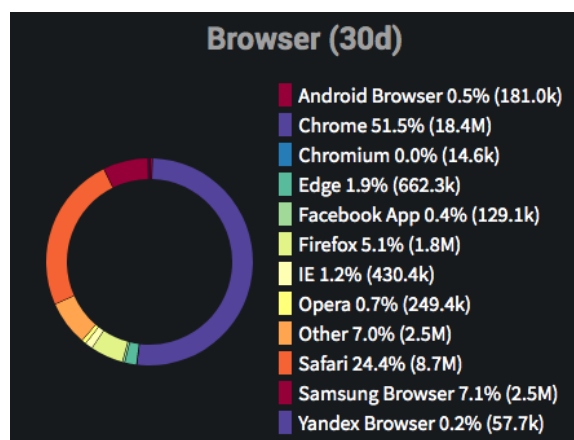


Figura 8: Usuarios por Navegador Web medidos para compatibilidad por WebGL (WebGLStats).

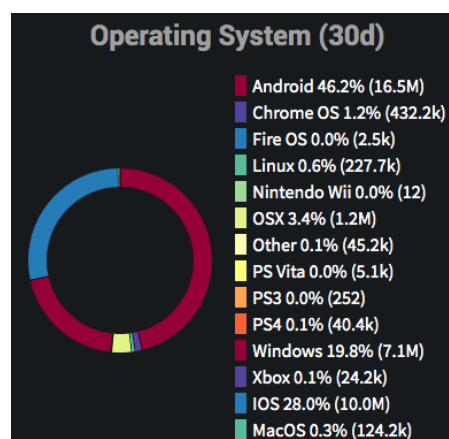


Figura 9: Usuarios de Sistema Operativos medidos para compatibilidad con WebGL (WebGLStats).

La API de WebGL es una tecnología diseñada para trabajar directamente con la GPU, y puede resultar demasiado tedioso y difícil de usar directamente (en comparación con otros estándares

web más accesibles) sin algunas librerías de utilidad. La carga de escenas gráficas y objetos 3D en los formatos convencionales de la industria no están tampoco previstos directamente. Por ello, se han creado bibliotecas JavaScript para aportar esta funcionalidad adicional.

De entre todas ellas Three.js es la más popular por número de usuarios. Es ligera y tiene un bajo nivel de complejidad en comparación con la especificación WebGL original.

A continuación, se presenta una tabla comparativa de las 3 principales librerías de WebGL.

	X3DOM	Three.js	Babylon.js
Estrellas de GitHub	237	50299	1698
Contribuidores	35	1065	44
Tamaño minificado	700kb	401KB	635KB
Licencia	MIT y GPL	MIT	Apache License 2.0

Tabla 1: Comparación de librerías WebGL.

De la tabla anterior concluimos en trabajar con Three.js para el desarrollo de modelado tridimensional. Puesto que tiene una comunidad superior y más activa eso se demuestra por la cantidad de estrellas en GitHub y la cantidad de contribuidores a la librería. Además de licencia MIT, lo que significa que es un proyecto de código abierto.

3.2.2. Three.js.

Es una librería de JavaScript que está enfocada al renderizado de gráficos 3D en un navegador web. Cuenta con dos renderizadores diferentes: Canvas y WebGL.

La librería simplifica el proceso de configuración de un entorno de representación en gran medida, ofreciendo características como matemáticas matriciales, escenografía, importación de modelos 3D y soporte para efectos de sombreado.

3.2.3. Procesamiento de nubes de punto.

A diferencia de representación de figuras poligonales. Las nubes de puntos no contienen información de conectividad y representan un conjunto de puntos en una superficie en lugar de una superficie cerrada. Los puntos generalmente se representan como rectángulos, círculos o píxeles individuales en pantalla, puesto que estas primitivas se procesan rápidamente en las GPU. El manejo de grandes cantidades de puntos, que generalmente no encajan en la memoria, requiere algoritmos de memoria externa que transmitan, procesen y rendericen solo un pequeño subconjunto de todos los datos. La mayoría de los métodos emplean variaciones de estructuras jerárquicas de partición de datos, también denominadas estructuras de resolución múltiple, como kd-trees, octrees o quadrees, y pueblan todos los nodos de datos que representan el modelo original en diferentes resoluciones. Algunos de estos métodos redistribuyen los datos originales dentro de la estructura jerárquica, mientras que otros almacenan los datos originales solo en los nodos de hoja y los promedios reducidos en los nodos internos.

Las nubes de puntos a menudo son demasiado grandes para caber en la memoria como un todo. De tal manera que, deben procesarse utilizando algoritmos de memoria externa. Una posible opción para realizar este proceso es dividir los datos en varios mosaicos y procesar uno o unos cuantos mosaicos a la vez. Este enfoque funciona bien para el procesamiento, pero para las

visualizaciones a menudo se desea mostrar todo el conjunto de datos y no solo unos pocos mosaicos a la vez.

El almacenamiento de varios niveles de detalle del modelo original en una estructura de datos jerárquica de partición del espacio permite que el renderizador de la nube de puntos cargue y muestre rápidamente las partes relevantes de una nube de puntos. Las regiones que están cerca de la cámara se representan con un nivel de detalle más alto que las regiones distantes, y las regiones que están fuera de la vista. Se descartan por completo. Las variaciones de Árbol octal (*octree*) y Árbol Kd (*kd-tree*) son dos estructuras populares de partición de espacio para la representación de grandes nubes de puntos. Algunas variaciones almacenan subdivisiones de la nube de puntos original. Otros almacenan los datos originales en distintos nodos de hoja y promedios reducidos. Para la realización de este prototipo se ha seleccionado una estructura que subdivide la nube de puntos original, ya que no crea nuevos puntos que requiere espacio en disco adicional, y porque permite al usuario realizar la selección de puntos y las mediciones en datos originales, sin alterar, en cualquier nivel de zoom, sin tener que esperar a que cargue un nodo hoja. Esta estructura es un árbol octal anidado modificable (MNO) creado por (Scheiblauer 2014). La figura 10 muestra una nube de puntos esférica que se ha dividido en un MNO.

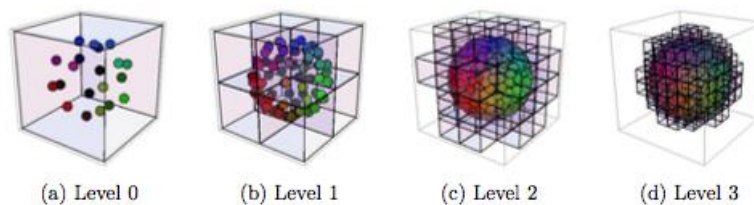


Figura 10: Los nodos de bajo nivel(izquierda) contienen modelos dispersos en grandes regiones. Cada nivel aumenta exponencialmente la cantidad de puntos y detalle. (Scheiblauer 2014).

3.2.3.1 Árbol Octal Anidado Modificable (MNO)

La estructura MNO tiene la particularidad sobre otro tipo de Árbol Octal porque realiza un método de subdivisión diferente y una partición de la jerarquía en fragmentos más pequeños y de fácil transmisión. La resolución de un nodo es definida por la propiedad de espaciado, que es la distancia mínima entre puntos. El espaciado se calcula inicialmente para el nodo raíz, según el tamaño del cuadro delimitador, y luego se reduce a la mitad en cada nivel.

La estructura MNO obtiene sus subdivisiones a través de una cuadrícula tridimensional de 128^3 celdas (Scheiblauer 2014). Inicialmente, los puntos se agregan al nodo raíz y un punto ocupará la primera celda en la que cae. Si un punto cae en una celda ya ocupada, y el número total de puntos en el nodo está por debajo de un umbral, entonces el punto se asignará a este nodo de todos modos, pero se almacenará dentro de una matriz de relleno en lugar de la cuadrícula. La matriz de relleno contiene otros puntos en el mismo nodo, además de la cuadrícula, para evitar la necesidad de crear inmediatamente un nuevo nodo secundario para almacenar estos puntos. Los nuevos nodos secundarios se crean tan pronto como se acumulan suficientes puntos que caen en un posible nuevo nodo secundario. Por lo tanto, los nuevos nodos secundarios se llenan inmediatamente con una cantidad mínima de puntos. Este enfoque de subdivisión conduce a densidades de puntos variables en diferentes niveles del árbol octal, y también evita la mayoría de los nodos vacíos porque los nuevos nodos solo se crean después de que se haya acumulado una cantidad mínima de puntos para un nuevo hijo. Sin embargo, no garantiza una cierta distancia mínima entre los puntos. Las celdas de cuadrícula adyacentes pueden contener puntos que están arbitrariamente cerca uno del otro (Scheiblauer 2014).

3.2.3.2 Atributo de Coloreado de Puntos.

Las nubes de puntos pueden colorearse utilizando los atributos de puntos que incluyen, entre otros, RGB. Los atributos que no sean RGB deben asignarse a un color RGB en tiempo de ejecución. Los atributos de los puntos se almacenan en la nube de puntos para cada punto además de las coordenadas, en el caso de la elevación, como uno de los ejes de coordenadas.

Dependiendo de los dispositivos de escaneo que se usaron para capturar los datos y de los algoritmos de procesamiento posterior utilizados para aumentarlos, diferentes tipos de atributos se almacenan en una nube de puntos. Los escáneres láser suelen proporcionar al menos intensidad, mientras que la fotogrametría suele proporcionar al menos datos RGB. Algunos atributos, como la clasificación, requieren pasos de procesamiento posterior una vez que se han capturado los datos.

En este proyecto y por las condiciones en donde son tomadas las mediciones ese atributo RGB no se requiere. Pero si debemos realizar un coloreado del modelo tridimensional en función al índice de resistencia medido por el laser. Para esto es necesario conocer el formato en como es recibida la nube de puntos antes de ser procesada figura 11, además es necesario identificar de esos datos recibidos, la estructura del archivo figura 12.

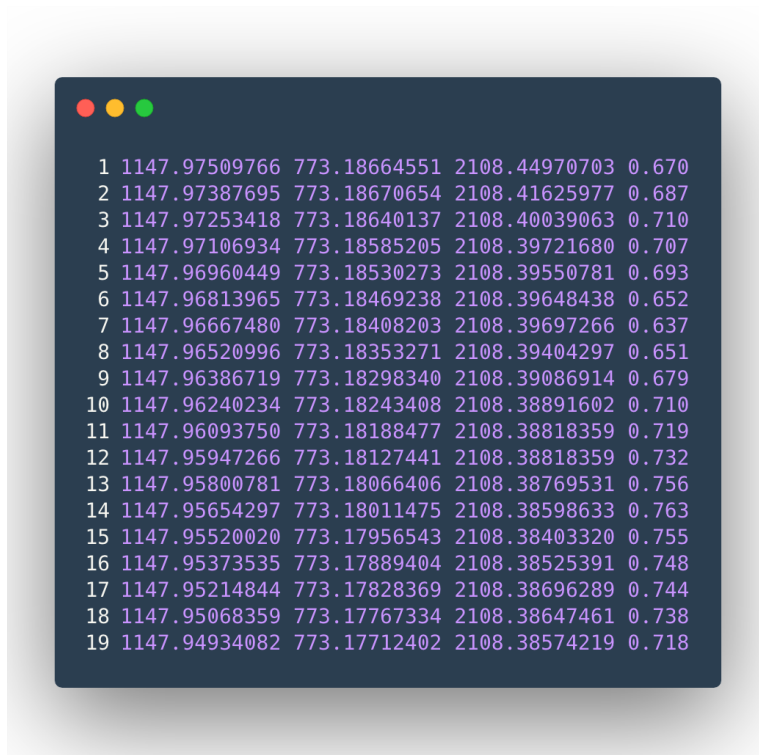


Figura 11: Formato en como los puntos son recibidos.

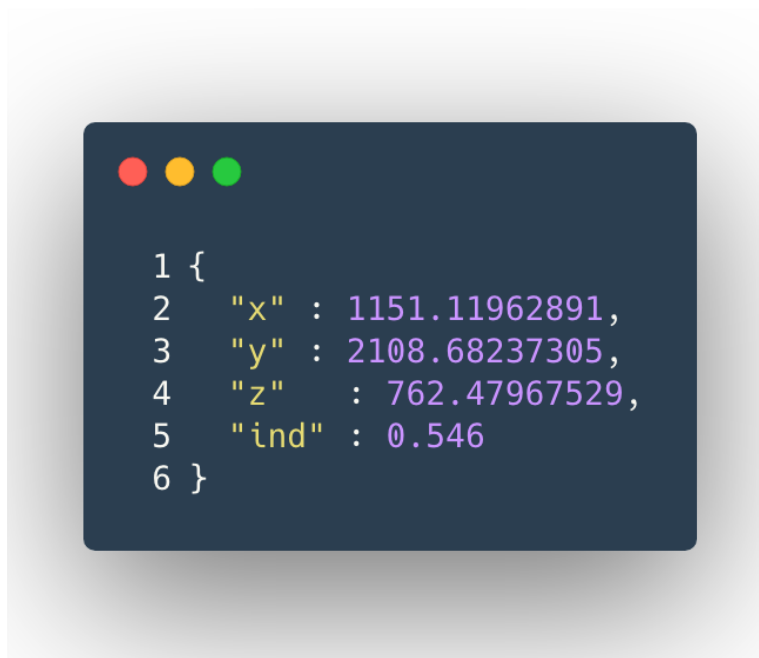


Figura 12: Arquitectura de un punto de control identificado.

De la figura 12, podemos identificar los 3 ejes coordenados y una 4 variable que representa al atributo de resistencia, que determina la calidad de la roca en el punto medido. Los valores de este atributo son presentados a una escala entre 1 y 0. Donde valores cercanos a 1 representan una calidad de roca superior y cercanos a 0 representan a un tipo de roca de inferior calidad de resistencia. A esta clasificación de colores al valor del atributo es presentada en la tabla 2. Donde se muestran de acuerdo al valor de índice del atributo, los rangos en que es clasificada la roca, el color y el valor RGB asociado al color presentado.







Rango Valor Índice	Color (Visual)	Color(RGB)
1 – 0.840		(0,0,255)
0.839 – 0.690		(0,255,255)
0.689 – 0.440		(0,255,0)
0.439 – 0.290		(255,255,0)
0.289 – 0.150		(255,140,0)
0.149 - 0		(255,0,0)

Tabla 2: Tabla de coloración de punto, según su valor de índice.

3.2.4. Creación del Escenario de Renderizado.

El escenario de renderizado es el espacio en que será presentado el modelo tridimensional del túnel.

Las herramientas necesarias para efectuar este proceso son las contenidas dentro de la librería de Three.js, llamándola desde el objeto THREE, nos brinda acceso a todos los elementos necesarios. Principalmente para poder mostrar cualquier objeto con Three.js, se necesitan tres cosas: escena, cámara y renderer.

3.2.4.1. Escena.

Espacio tridimensional en el que se puede añadir objetos e interactuar con los mismos y desplazarse sobre este.

3.2.4.2. Cámara.

Es el punto virtual en que se puede ver la escena y los objetos presentes en esta, es un tipo de objeto que puede ser añadido a la escena.

Una escena puede contener tantas cámaras como se requiera, pero solo se puede utilizar una de ellas para renderizar la escena en un momento dado. La cámara puede ser posicionada, rodada según sea necesario. Para definir una cámara se debe tener en cuenta el tipo de proyección. Three.js proporciona las siguientes.

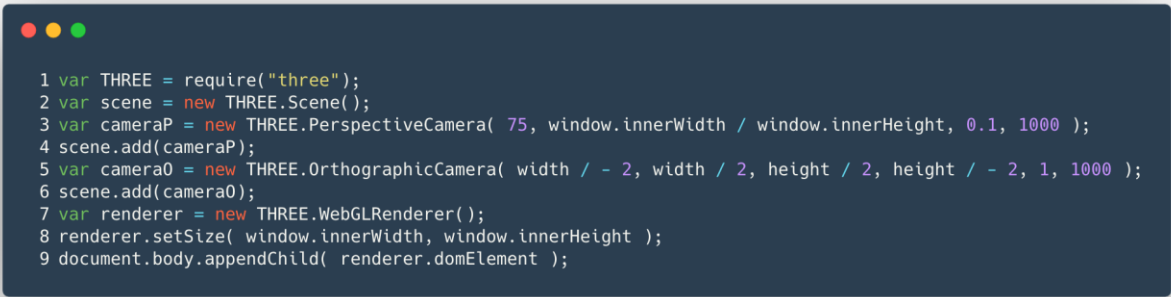
- Proyección Paralela (*OrthographicCamera*): En este modo de proyección, el tamaño de un objeto en la imagen renderizada permanece constante independientemente de la distancia desde la cámara.
- Proyección Cónica (*PerspectiveCamera*): Deforma los objetos de acuerdo a la distancia y posición a la que se encuentre de la cámara. Generando una vista cónica que realiza una mímica a la vista humana.

3.2.4.3. Renderer.

Objeto WebGL donde la tarjeta gráfica procesa todos los gráficos, la escena 3D puede ser enorme, pero en la pantalla únicamente se mostrará lo que quede dentro del encuadre de la cámara.

3.2.4.4. Generando la Escena.

Realizamos entonces el llamado a la librería de three.js, junto a las herramientas para generar una escena que permita generar un modelo a mostrar, como se puede apreciar en la figura 13. Mostrando una configuración básica que permita generar el proceso de renderizado observado en la figura 14. La cual muestra las 4 etapas en el proceso de renderizado.



```
1 var THREE = require("three");
2 var scene = new THREE.Scene();
3 var cameraP = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );
4 scene.add(cameraP);
5 var camera0 = new THREE.OrthographicCamera( width / - 2, width / 2, height / 2, height / - 2, 1, 1000 );
6 scene.add(camera0);
7 var renderer = new THREE.WebGLRenderer();
8 renderer.setSize( window.innerWidth, window.innerHeight );
9 document.body.appendChild( renderer.domElement );
```

figura 13: Generación básica de escena con Three.js

La primera etapa de renderizado es, existiendo una escena, el ingreso de los datos a ser mostrados, en este caso el punto coordenado. Luego en la segunda etapa consta de la población total por los datos en la escena, para luego en la tercera etapa incluir una cámara que permita el punto de mira inicial. De esta forma y finalmente culminar con el proceso de renderizado mostrando en pantalla una vista de la figura renderizada.

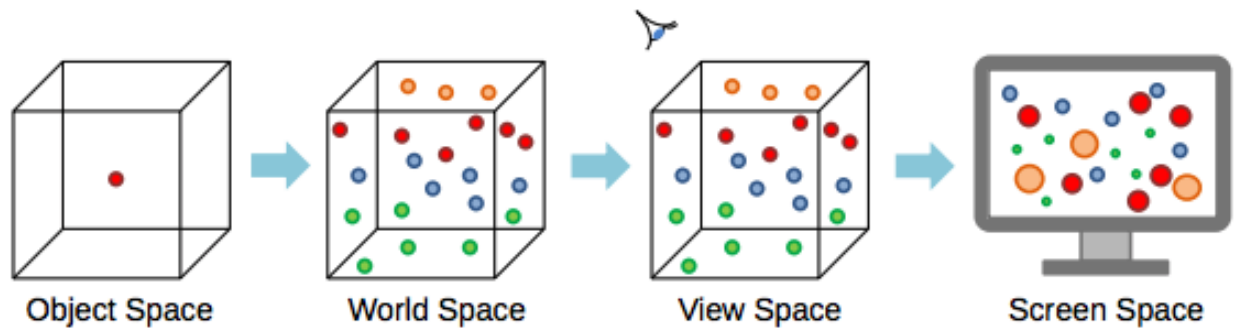


figura 14: Etapas proceso de renderizado, desde ingreso de datos cargados desde la nube de puntos, hasta que la figura es mostrada en pantalla.

3.2.5. Patrón de Arquitectura.

Dado que el proyecto está dirigido a una facilidad de uso y la capacidad de expansión, se decidió seguir el patrón de arquitectura de software Modelo-Vista-Controlador (MVC), mostrado en la figura 15. El patrón MVC separa el modulo visual (Vista) desde el modulo que genera los datos a ser procesados (Modelo), lo que es para ser “visto”. El usuario final está separado de los datos por el controlador. A través del cual envía comandos al módulo de datos.

Esto permite que la aplicación se pueda reutilizar y expandir fácilmente. Por ejemplo, el diseño de la plataforma puede modificarse completamente con un nuevo diseño sin tener que volver a escribir cualquiera de los códigos para la visualización de los macizos.

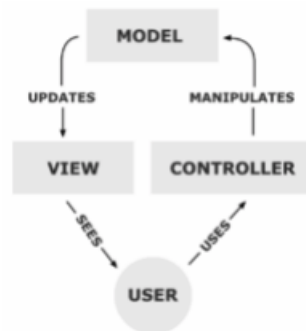


figura 15: Representación visual del patrón de diseño Modelo-Vista-Controlador

3.2.6. Sistema Multiplataforma.

El desarrollo de aplicaciones de escritorio tiene aproximadamente el mismo enfoque que el desarrollo móvil. Más específicamente, es necesario construir una aplicación para cada sistema operativo generalmente, .Net (basado en C#) se usa para sistemas Windows y Swift, Objective-C para OSX. Para otras plataformas es posible usar el lenguaje de programación Java (Litayem 2015).

A lo largo de toda la investigación las decisiones de tecnologías que se seleccionaron previas a esta sección fueron hacia un mismo enfoque. Obtener un sistema que sea multiplataforma sin la necesidad de realizar distintos desarrollos para cada sistema operativo. La tecnología que mayor afinidad en relación a las especificaciones del proyecto es Electron.

3.2.6.1. Electron.

Electron es un framework de código abierto para el desarrollo de aplicaciones multiplataforma de escritorio, usando JavaScript, HTML y CSS como se indica en la documentación oficial de Electron.

El sistema ya lo desarrollamos en el punto anterior en un ambiente Web. La ventaja de usar Electron es que usamos el mismo desarrollo como aplicación de escritorio que funciona tanto en Windows, Mac y Linux. Sin necesitar instalación previa. Se otorga un acceso a las APIs nativas a la aplicación en Electron, mientras que, como regla general, las aplicaciones web están restringidas al acceso de este tipo. Además, la aplicación Electron tiene acceso ilimitado a todos los eventos del sistema operativo. Por ejemplo, puede controlar archivos de gran tamaño como lo

son los archivos de nubes de punto y enviar notificaciones del sistema operativo. Al mismo tiempo Electron puede utilizar cualquier módulo de Node.js, incluyendo los módulos desarrollados por terceros. (Jasim, 2017).

Electron combina Chromium y Node.js en un solo tiempo de ejecución para proporcionar aplicaciones multiplataforma de escritorio. Electron puede verse como un navegador minificado con la capacidad de interactuar con el sistema operativo, y este navegador es parte del empaquetado de la aplicación.

Cuando se inicia, Electron ejecuta el proceso principal (main.js) y el renderizador, este último proceso, introduce diferentes ventanas, que representan diferentes vistas/páginas. El proceso principal puede comunicarse con los procesos de representación mediante una herramienta llamada comunicación de procesos internos. Sin embargo, la comunicación entre los procesos del renderizador solo es posible a través del proceso principal. Como se indica en la figura 16, el proceso principal (proceso del navegador) crea las ventanas de la aplicación y las controla, mientras el proceso del renderizador carga la página web, soportando scripts de JavaScript y estilos CSS. Así que, técnicamente, la aplicación de Electron no es una aplicación de escritorio propiamente tal.

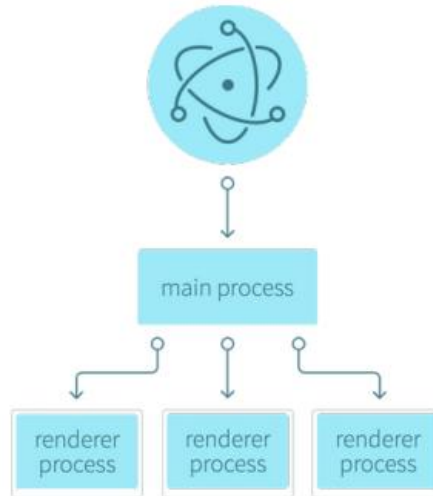


figura 16: Procesos de Electron y comunicación entre ellos (Electron).


Para construir la aplicación Electron con éxito es requerido conocimientos básicos de JavaScript, CSS, HTML y Node.js

3.2.6.2. Configuración Inicial Electron.

La aplicación comienza leyendo el archivo package.json. Este archivo contiene las propiedades principales del proyecto como lo son: nombre, version, autor, dependencias de desarrollo, etc.

El campo “main” contiene el nombre de la ruta del archivo donde Electron ejecutará el script del proceso principal cuando la aplicación inicie.

Una representación de cómo es la estructura de este formato, se puede observar en la figura 17 a continuación.



```
1 {
2   "name": "HSaavedraTesis",
3   "version": "0.0.1",
4   "description": "Visualizador de Puntos",
5   "author": "Hector Saavedra Casanova",
6   "main": "main.js",
7   "devDependencies": {
8     "browserify": "^14.5.0",
9     "electron-prebuilt": "^1.4.13",
10    "chai": "^4.1.2",
11    ...
12    ...
13    ...
14  }
```

figura 17: Estructura package.json.

Electron ejecuta el proceso principal, que es el archivo `main.js` que está definido en el `package.json`. Una vez que se ejecuta el proceso principal, crea su proceso renderizador. El proceso del renderizador contiene `index.html`, `style.css` y archivos JavaScript. Los archivos JavaScript incluyen el modelo de objetos de la aplicación.

El archivo `index.html` es un archivo HTML5 que refiere al archivo CSS y carga los archivos JavaScript para ejecutar código en este proceso.

Por lo tanto, la ejecución del archivo `main.js` no proporciona la UI de la aplicación. La interfaz de usuario es creada usando la API de Electron llamada `BrowserWindow`. Entonces el módulo `BrowserWindow` carga el `index.html` para mostrar la interfaz de usuario de la aplicación. Cuando se corta la instancia de `BrowserWindow`, el proceso renderizador se termina. El proceso principal y el proceso de renderizado también pueden comunicarse entre sí usando el módulo IPC. El

modulo IPC permite enviar y recibir mensajes entre remitente y destinatario. Una parte del código de main.js puede ser vista en la figura 18 a continuación.

```
1 const electron = require('electron');
2 const app = electron.app;
3 const BrowserWindow = electron.BrowserWindow;
4
5 const path = require('path');
6 const url = require('url');
7
8 // Mantener una referencia global al objeto window, sino, la ventana podría
9 // cerrarse automáticamente cuando el objeto JavaScript se termine.
10
11 let mainWindow;
12 function createWindow () {
13   // Crea la ventana BrowserWindow.
14   const {width, height} = electron.screen.getPrimaryDisplay().workAreaSize;
15   mainWindow = new BrowserWindow({width, height});
16   // y carga el index.html de la app.
17   mainWindow.loadURL(url.format({
18     pathname: path.join(__dirname, 'index.html'),
19     protocol: 'file:',
20     slashes: true
21   }));
22   // Se emite al cerrar la ventana.
23   mainWindow.on('closed', function () {
24     mainWindow = null
25   });
26 }
27
28 app.on('ready', createWindow);
29
30 // Cerrar cuando se cierran todas las ventanas.
31 app.on('window-all-closed', function () {
32   // En OS X es común para aplicaciones y sus barras de menú
33   // mantenerse activas hasta que el usuario ejecute Cmd + Q para cerrar.
34   if (process.platform !== 'darwin') {
35     app.quit()
36   }
37 });
```

figura 18: Código del main.js.

En el código de la figura 18 se aprecia el método ready que tiene una función de callback conocida como createWindow. Esta función define un BrowserWindow y establece el tamaño inicial. Luego, el archivo index.html se carga en él para mostrar la GUI de la aplicación.

3.2.7. Arquitectura de la Aplicación.

A lo largo del capítulo se comentaron todas las tecnologías utilizadas en la creación de la aplicación, y como interactuaban entre ellas. Por lo tanto, en la figura 19 se puede observar la arquitectura general del sistema.

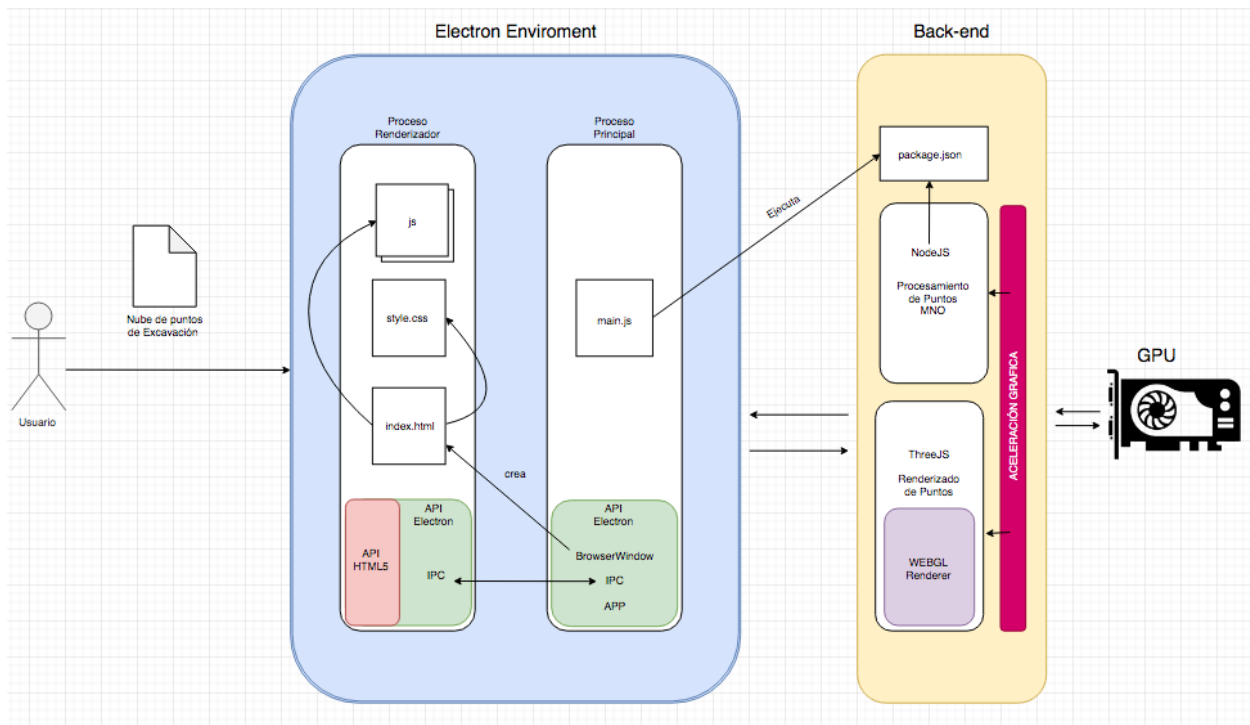


figura 19: Arquitectura General del Sistema.

3.3. Propuesta de Apoyo a Calculo de Medidas de Convergencia entre Puntos de Control de una Excavación.

A lo largo de los capítulos de esta investigación hemos descrito el proceso de medidas de convergencia y también explicado la importancia de aplicarlo para observar el comportamiento del macizo rocoso durante las actividades mineras.

La actividad de realización de estas mediciones, mediante la utilización de los datos recopilados por LiDAR, es en lo que se basa la siguiente propuesta. Y de la misma forma en que se comentó con anterioridad que para el correcto uso de las nubes de punto y sus datos asociados, por magnitud y por distribución de datos, es necesario realizar un pre-procesamiento.

Existe una gran cantidad de estructuras de datos que se pueden utilizar para almacenar y procesar conjuntos de datos de puntos. Por ejemplo (Samet) ofrece una visión general estructurada de las variantes más conocidas. La decisión, respecto a que estructura se utilizará depende del uso específico de los datos a trabajar. Por lo general una sola estructura no es la mejor opción para todas las tareas. Otra distinción entre las diferentes variantes son sus complejidades de implementación. Para esta investigación, y este caso en específico se requieren operaciones de ordenamiento, segmentación y calculo. Aunque los conjuntos de datos de puntos utilizados en esta memoria de investigación existen en el espacio 3D y por lo mismo para su renderización en ese espacio se utilizó la estructura de datos de árbol octal anidado modificable, para realizar las operaciones necesarias en este caso se trabajará con su versión en el espacio de 2D por conveniencia, llamada quadtree o árbol cuaternario.

3.3.1. Árbol Cuaternario (Quadtree).

Una estructura de datos simple para ordenar y buscar rápidamente los puntos, es el árbol cuaternario (Finkel R. 1974). Un árbol cuaternario se puede interpretar como una estructura de datos jerárquica que divide el espacio, donde en cada nodo el espacio subyacente se subdivide en cuatro subnodos (hijos) a lo largo de líneas paralelas al eje que pasan por un punto de datos determinado. Los puntos se almacenan en todos los nodos de la jerarquía, es decir, nodos internos y nodos hoja. Si no hay un punto disponible para un nodo hoja, se deja vacío. La figura 20 muestra una representación visual de un árbol cuaternario. A la izquierda se muestra como una estructura de datos de partición de espacio y a la derecha como una estructura de datos de árbol con el nodo raíz en la parte superior. Cada nodo interno del árbol cuaternario tiene exactamente 4 hijos, cada uno de ellos puede estar vacío, tener un punto o un enlace a sus propios hijos. La búsqueda o inserción de un punto desencadena una búsqueda de complejidad en el tiempo $O(\log(N))$ desde el nodo raíz hasta el nodo requerido. Donde N es el número total de puntos insertados. Eliminar un punto es bastante complicado, ya que la jerarquía también debe mantenerse cuando los puntos de los nodos interiores se han eliminado, y el método más simple para hacerlo es reconstruir la estructura de datos desde cero. Como ya se mencionó con anterioridad la extensión de un árbol cuaternario al espacio 3D es el árbol octal. Así como el árbol octal generó el modelo tridimensional de la excavación, podemos considerar al llevar el modelo al campo 2D, el eje de profundidad (z) como el identificador de cada árbol cuaternario y los ejes (x, y) de ese (z) los datos del árbol cuaternario que generan los puntos para realizar la medición de convergencia.

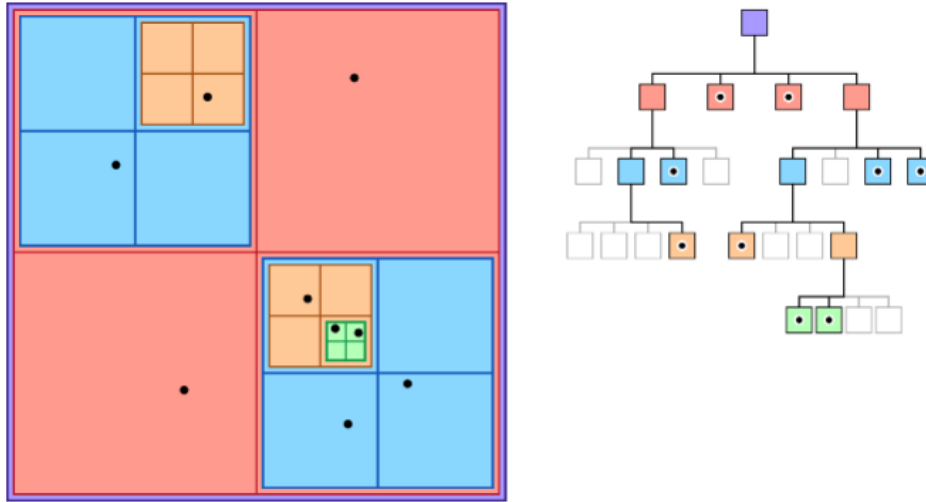


figura 20: Representación espacial (izquierda) y representación lógica (derecha) de un árbol cuaternario. (Apple)

3.3.2. Comparación de mediciones LiDAR.

El desafío relacionado a efectuar dichas medidas de desplazamientos convergentes utilizando nubes de puntos que fueron recolectadas utilizando un LiDAR es que, para efectuar el seguimiento de las deformaciones en el tiempo es necesario tener puntos de referencia y analizarlos en el tiempo. Para así determinar la progresión e intensidad de la deformación de las paredes en el macizo rocoso y del mismo modo plantear estrategias y medidas de prevención ante las mismas. La problemática recae en que cada medición de una sección en el tiempo (independiente a la obvia deformación) es diferente, en el sentido que tanto la cantidad de puntos medidos como la disposición y concentración de puntos, para una misma sección varia en el tiempo. Sujeto a las variaciones del ambiente y emisión de señal del mismo scanner LiDAR cuando realizo la medición. Por lo tanto, no se puede asumir un cálculo de desplazamiento convergente entre puntos a razón 1 a 1, sino que se debe realizar un proceso que entregue la correspondencia entre puntos

de la nube de puntos no deformada con los puntos de la nube deformada. Es por esto que se plantearan algoritmos para realizar este proceso.

3.3.2.1. Algoritmo Iterative Closest Point (ICP) y Nearest Neighbors Search (NNS).

El objetivo es lograr que las nubes de punto deformadas y no deformadas queden alineadas según correspondencia entre los puntos el uno contra el otro, para esto utilizaremos el algoritmo ICP (Besl & McKay 1992). El objetivo del algoritmo ICP es estimar una transformación rígida entre $p_i \in P$, un punto de la nube de puntos de la sección no deformada y $q_i \in Q$, un punto de la nube de la sección deformada. Usando, además el algoritmo de Nearest Neighbors Search y el cálculo de la distancia euclidiana. El algoritmo estima el punto más cercano entre p_i y q_i , ICP utiliza una función de error como se ve en la ecuación 1, para minimizar la suma de las distancias cuadradas.

Ecuación 1: Función de Error

$$E(R, t) = \min_{R, t} \sum_i ||p_i - (Rq_i + t)||^2$$

Una vez que las nubes de punto quedan alineadas, escaladas y la correspondencia entre los puntos del estado inicial (no deformados) al estado final (deformado) establecida (figura 21). Entonces se pueden comparar entre sí para ver el análisis de los desplazamientos convergentes.

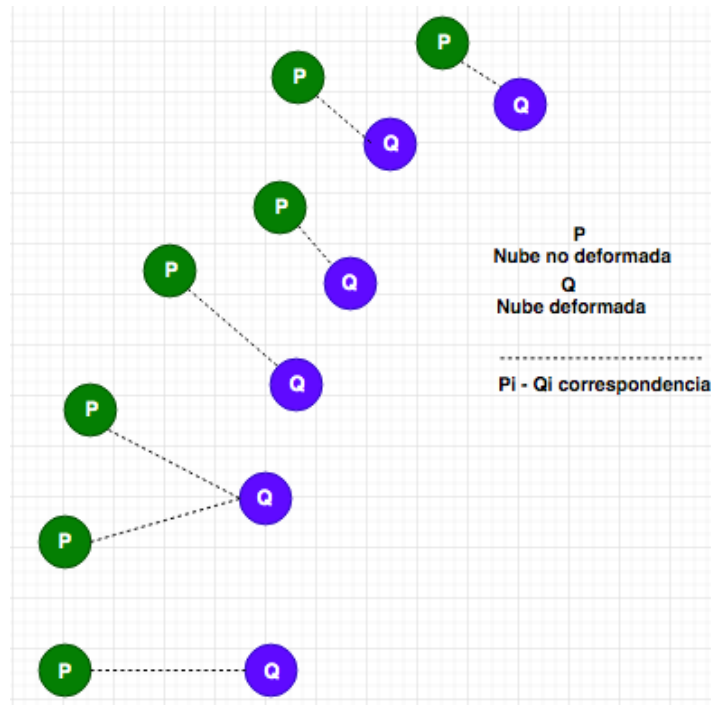


figura 21.: Correspondencia estimada entre nube de puntos (P) y nube de puntos (Q)

3.3.2.2. Cálculo de distancias

Efectuada la correspondencia anterior, se puede realizar un cálculo casi directo para conocer las métricas entre ambas nubes de puntos comparadas. El problema recae en que con 2 o más puntos de correspondencia, no se puede asumir como distancia real, a la distancia del más próximo entre los puntos de correspondencia. Por lo que, en este método, para cada punto de la nube comparada, se buscará el punto más cercano dentro de la nube de referencias y se calculará la distancia euclidiana, como se puede apreciar en la figura 22.

Utilizando como referencia la nube de punto del estado no deformado, se estima el punto más cercano espacialmente en la nube de puntos deformada, que para efectos de esta investigación será la distancia de Hausdorff (Hossain 2011). Y si ecuación presentada a continuación.

Ecuación 2: Distancia de Hausdorff

$$H(A, B) = \max_{a \in A} \{ \min_{b \in B} \{ d(a, b) \} \}$$

Que se define, como donde a y b son puntos de los conjuntos A y B respectivamente, y $d(a, b)$ es cualquier métrica entre estos puntos. Esto proporciona una medida de desviación por punto en el caso de las deformaciones en las paredes del macizo. Estos métodos propuestos dependen de las variaciones de densidad de los puntos entre los conjuntos de datos 3D. Esto se puede observar en la figura 22 donde dado el conjunto de datos y un punto perteneciente a la sección no deformada se puede notar tanto la distancia al nodo más cercano encontrado en el algoritmo NNS, como a su distancia calculada por Hausdorff.

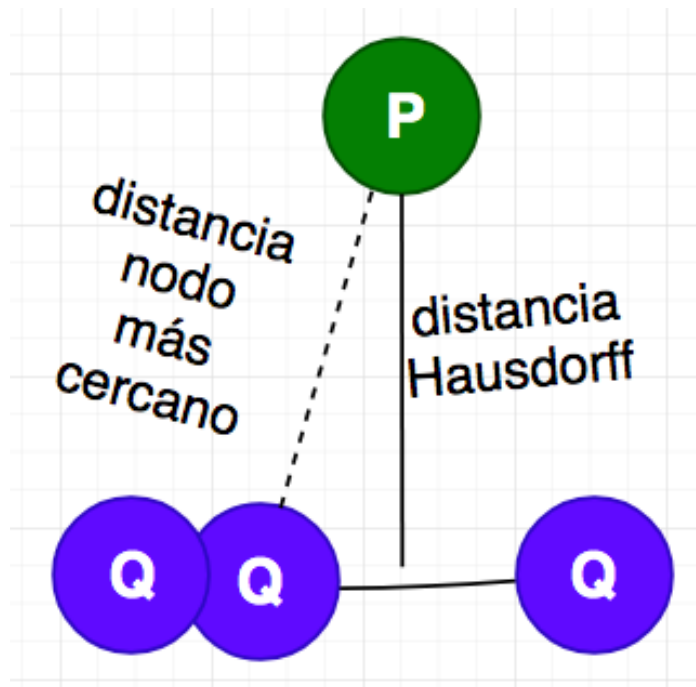


figura 22: Distancia medida respecto a nodo más cercano según algoritmo NNS y distancia calculada aplicando la ecuación de Hausdorff.

3.3.3. Proceso para el cálculo de distancias entre secciones medidas en el tiempo.

Entonces se establecen los pasos para realizar el proceso del cálculo de las convergencias entre los puntos de una excavación entre dos periodos representado en la figura 23, donde se define como estado no deformado (estado 0), al estado anterior a la última medición, y estado deformado a los puntos que representan al estado presente de la excavación (estado 1). Por lo tanto, al realizarse una tercera medición el estado deformado anterior debe ser considerado como estado no deformado y así progresivamente.

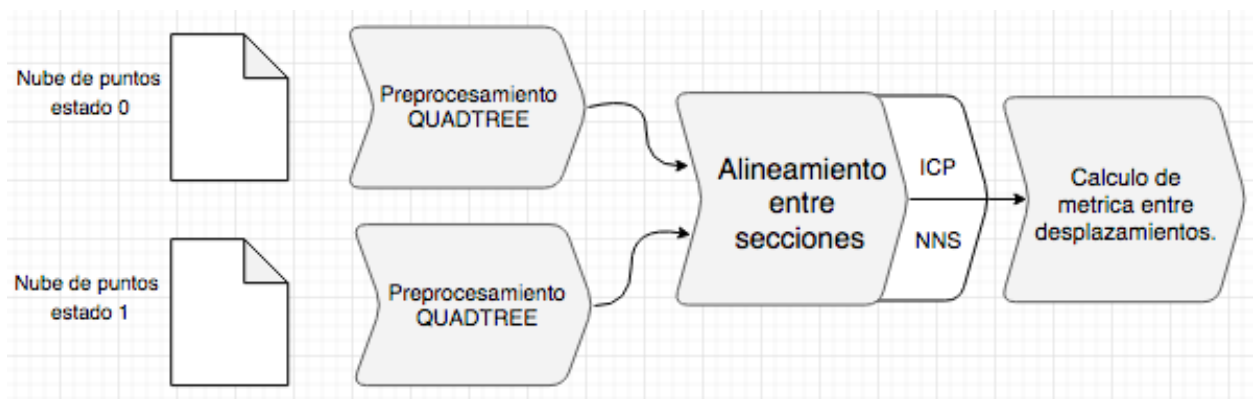


figura 23: Flujo del proceso para el cálculo de distancias entre secciones medidas en el tiempo.

Capítulo 4. Pruebas y Resultados.

En el capítulo se presentarán las pruebas realizadas al software de visualización de puntos, y se evaluará el funcionamiento y rendimiento del mismo, respecto a intentar renderizar dos archivos de nubes de puntos, y sus variaciones de densidad, junto también con medir un archivo perteneciente a una única sección, de forma de observar si existe un rendimiento exponencial importante, respecto a la cantidad de puntos y tiempos de procesamiento y renderizado del modelo tridimensional.

4.1. Equipo de Pruebas.

Con motivo de la exigencia de hardware exigida por la aplicación y para observar el mayor rendimiento posible que puede alcanzar la aplicación es que se realizaron las pruebas en un equipo de escritorio que posee las características que se presentan en la tabla a continuación.

Tabla 3: Especificaciones técnicas equipo de pruebas.

Procesador	2 x 3.33 GHZ Hexa-Core Intel Xeon X5680
Memoria RAM	64 GB DDR3 1333 ECC
Memoria de Video	Nvidia GTX 980 4GB GDDR5
Disco Duro	SSD 256GB [(R)562.5 MB/s (W)521.2 MB/s]
Sistema Operativo	MacOSX Mojave / Windows 10

4.2. Carga de Archivo.

Previo a iniciar el proceso de renderizado es necesario cargar un archivo de nube de puntos a la aplicación. Se implementaron tres métodos para efectuar esta operación, los cuales son seleccionar una nube almacenada dentro de la aplicación (figura 24), cargar un archivo de nube de puntos directamente desde el disco duro, explorando en las carpetas del sistema operativo del usuario (figura 25). Y por último utilizando el método de drag&drop, lo que permite al usuario seleccionar un archivo de nube de puntos y arrastrarlo en la aplicación para ejecutarlo (figura 26).

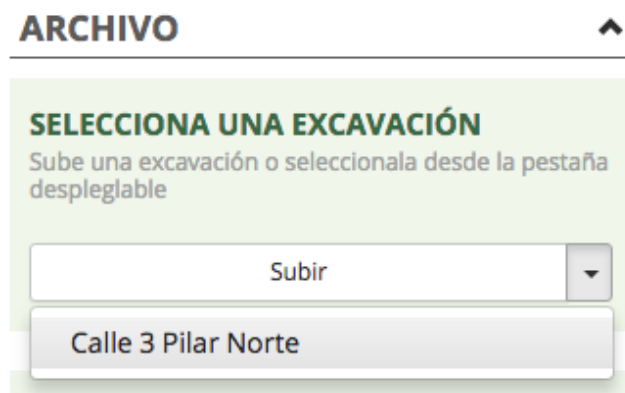


figura 24: Cargar nube de puntos almacenada dentro de la aplicación.



figura 25: Cargar Nube de puntos directamente desde disco duro.



figura 26: Cargar nube de puntos utilizando el método de drag&drop.

La selección de los distintos métodos en si son indiferentes y no representan en estos momentos impacto alguno en los tiempos de carga y ejecución. Ya que en estas pruebas la lectura del archivo siempre se realizó dentro del mismo equipo. Por lo tanto, el factor de carga corre por parte de la velocidad de lectura/escritura del disco duro.

4.3. Tiempo de Carga.

La aplicación al ser acelerada gráficamente tiene una dependencia directa a las prestaciones del equipo en que se ejecuta, por lo tanto y respecto a la gran cantidad de datos a renderizar, equipos con características más limitadas pueden provocar una carga más lenta o una vez cargado genere latencia en el movimiento de la figura. Es por esto que aprovechando la característica de árbol

(MNO), el cual, si bien no permite establecer una distancia modificable entre los nodos a renderizar, puede disminuir la densidad de datos, al no mostrar solo algunos puntos de nodos más densamente poblados, dicha acción es el equivalente a disminuir la resolución de la imagen renderizada obtenida.

Se realizaron pruebas de rendimiento para determinar si al aplicar dicha funcionalidad mejoraban los tiempos de carga. La siguiente tabla 4 muestra los archivos que se analizaron y la tabla 5 los resultados obtenidos al calcular los tiempos de ejecución entre las variaciones de densidad.

Tabla 4: Archivos de nubes de punto a analizar.

ID	Nube de Puntos	Cantidad Total de Puntos
1	20181212-Calle 3 Pilar Norte 000000	22.424.457
2	20181212-Calle 3 Pilar Norte 000001	24.340.000
3	Seccion4	423

Tabla 5: Resultados de rendimiento calculado.

ID	Densidad	Cantidad de Puntos	Tiempo (s)
1	100%	22.424.457	18.04
1	50%	11.212.229	14.44
1	30%	7.474.819	9.76
1	25%	5.606.115	8.23
1	11%	2.491.607	5.93
2	100%	24.340.000	17.69
2	50%	12.170.000	13.14
2	30%	8.113.334	10.39
2	25%	6.085.000	9.46
2	11%	2.704.445	7.01
3	100%	423	0.009

Por lo tanto, como se puede apreciar si se muestra un cambio importante en los tiempos de carga, al disminuir la densidad de puntos, cabe mencionar que todos estos tiempos está considerado el tiempo de procesamiento de los puntos desde el ingreso al árbol octal, hasta el renderizado, creación de escenario y posición de cámara, para el despliegue en pantalla. Y según lo observado se puede notar una leve diferencia entre la nube ID (2) vs ID (1) vistas en la tabla 5, en relación a una mayor velocidad de carga, considerando que posee más puntos en su composición. Se concluye que esa diferencia puede deberse a los procesos lectura del árbol, siendo posiblemente ID (1) con una estructura jerárquica más compleja que ID (2), a pesar de tener menor cantidad de puntos.

4.4. Resolución Visual.

A continuación, se muestra un renderizado visual para realizar una comparativa de las resoluciones entre el 10% y el 100% de los puntos. Figuras 27 y 28 respectivamente. Para apreciar si la pérdida de resolución o detalle perjudica de manera importante la visual del túnel.

Luego se aumentará el tamaño de los puntos para observar si se detectan puntos de oclusión también entre 10% y 100% de densidad de puntos. Figura 29 y 30 respectivamente.

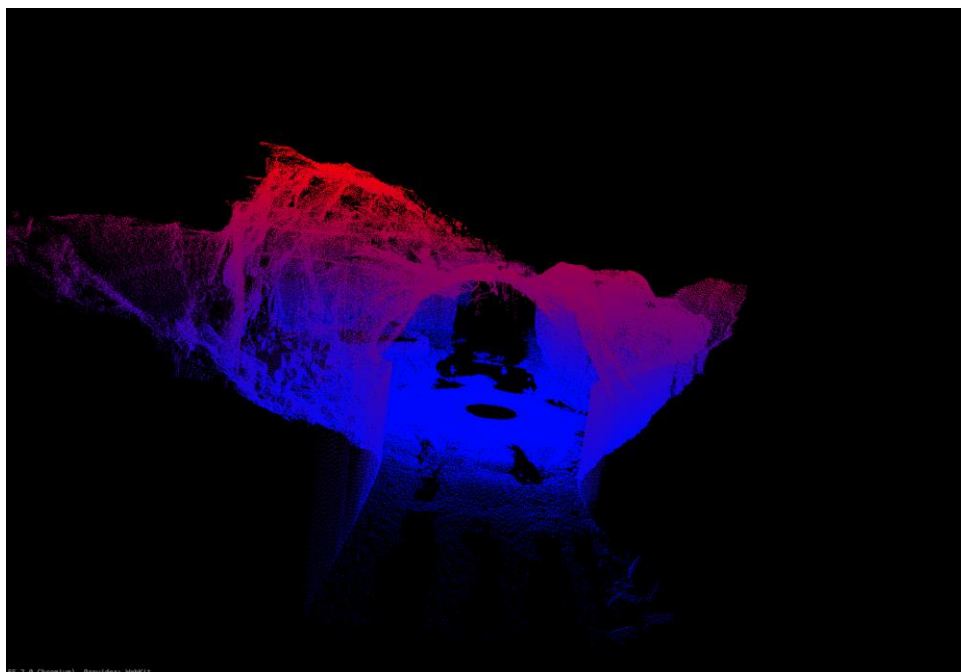


figura 27: Renderizado 3D del túnel al 10% de densidad de datos.

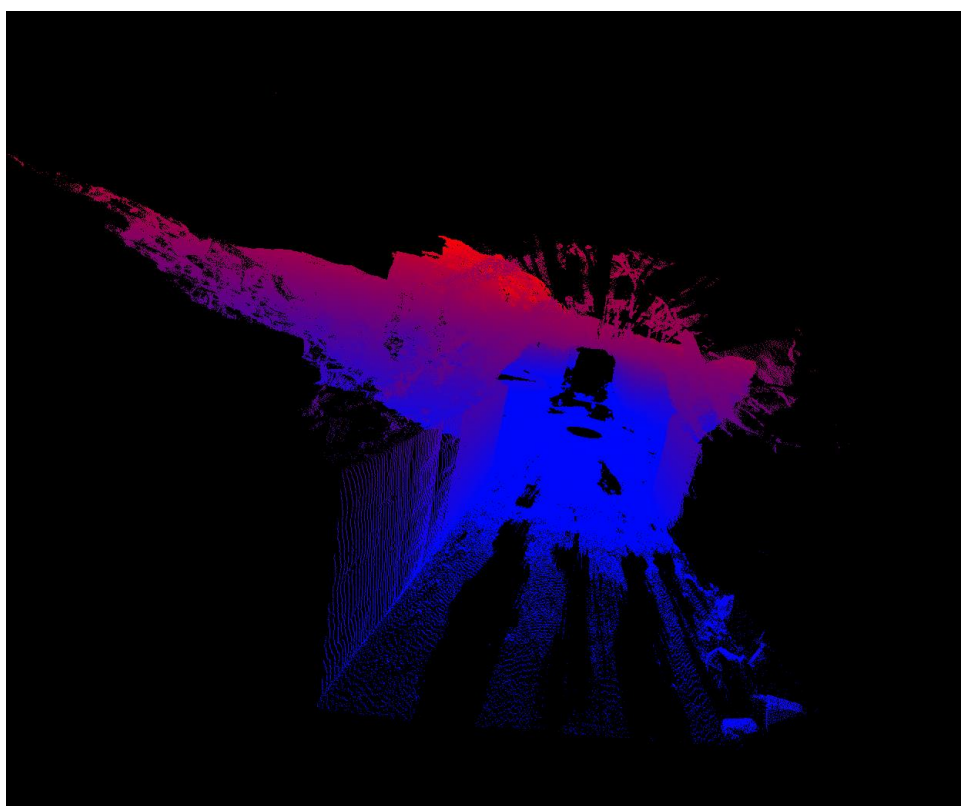


figura 28: Renderizado 3D del túnel al 100% de densidad de datos.

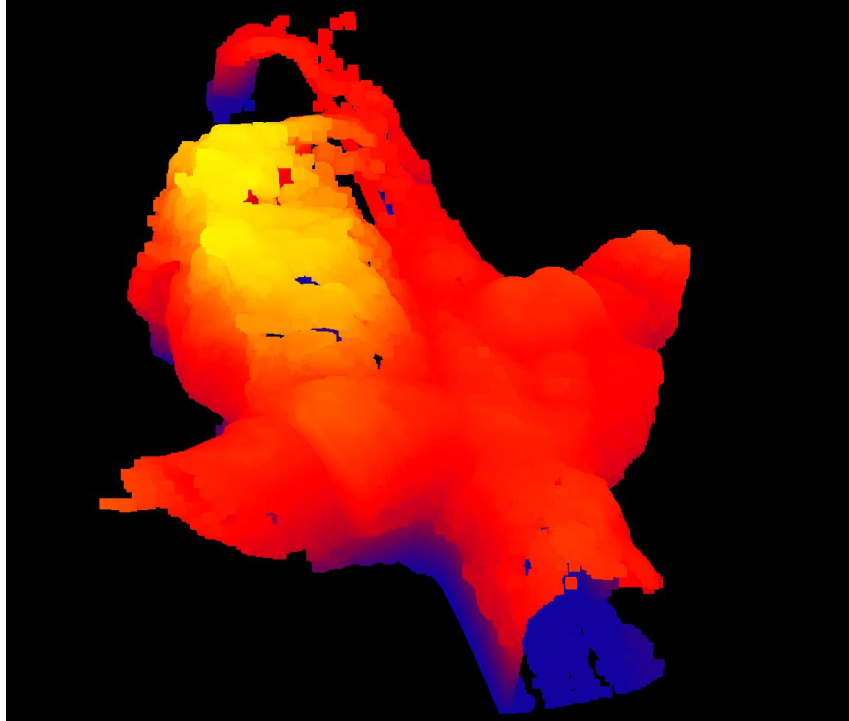


figura 29: Renderizado 3D con tamaño de puntos al máximo. (10% densidad).

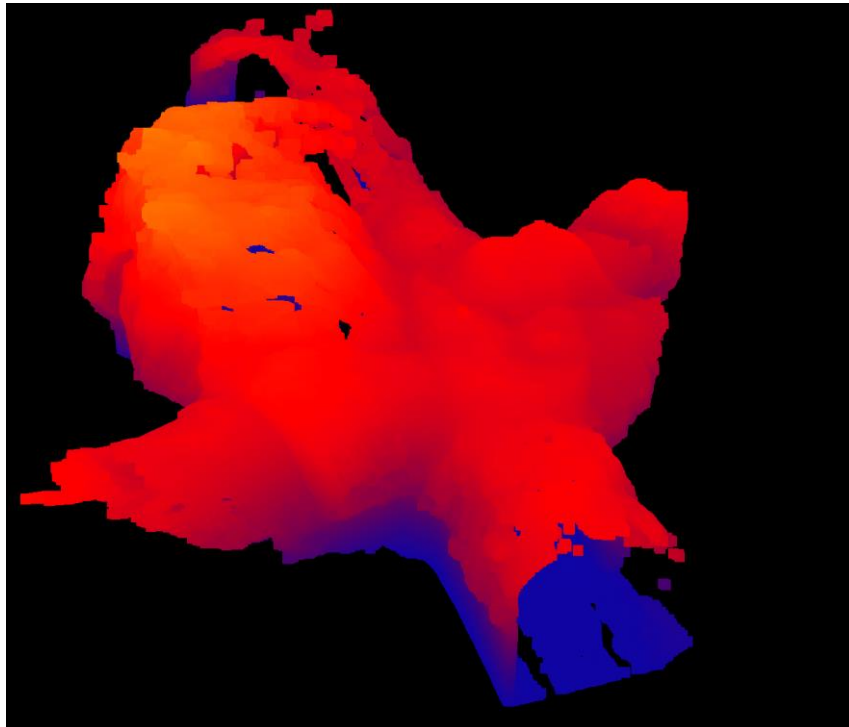


figura 30: Renderizado 3D con tamaño de puntos al máximo. (100% densidad).

En relación a los modelos observados entre una misma figura a distintas resoluciones se debe comentar que efectivamente se aprecia una pérdida de calidad y separación de puntos a densidades menores, pero no es un factor que dificulte la visión y solo es notorio al hacer un acercamiento al interior de la figura. Al aumentar el tamaño de los puntos esa diferencia no se nota como se pudo apreciar en las figuras 29 y 30.

Respecto a las oclusiones a términos generales, disminuir la cantidad de punto no las genera, a una escala notoria, como se comentó, solo se nota una mayor separación de puntos. Y las oclusiones existentes, están persistentes en ambas resoluciones producto de la medición por láser LiDAR.

4.5. Coloración.

Como se ha podido apreciar en las figuras 26,27,28,29. La coloración de los modelos se encuentra en una escala gradiente, respecto a la altura del punto. Pero también se implementó la coloración RGB que representa al índice de la roca. A continuación, en la figura 31 se puede apreciar dicha coloración, en una sección del túnel, respetando los valores de rango presentados en la tabla 2.

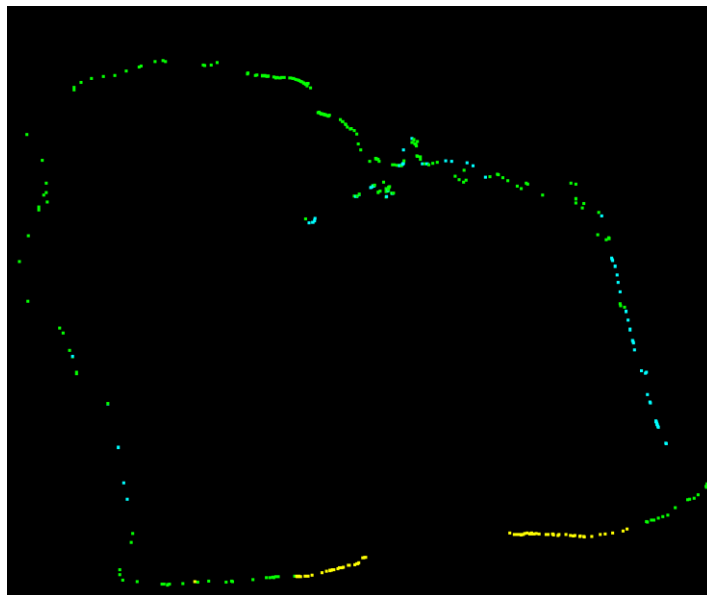


figura 31: Coloración RGB de sección según tabla de rangos de índice.

4.6. Alineamiento en árbol cuaternario.

Este proceso de la investigación no se aceleró gráficamente, y se realizó en RStudio. En la figura 32 se realizó una representación espacial de árbol cuaternario. Para observar la distribución de los cuadrantes al poblar de puntos la estructura de datos.

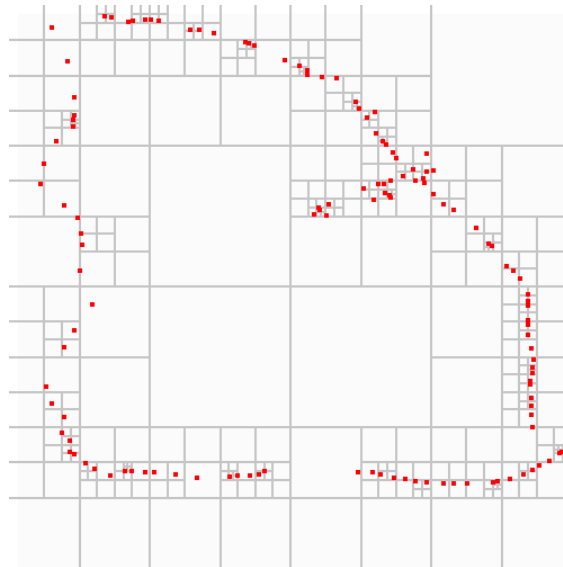


figura 32: Representación espacial de árbol cuaternario.

También se realizó un alineamiento entre secciones aplicando los algoritmos de ICP y NNS vistos en el capítulo anterior a modo de alinear y asociar puntos de un estado 0 a sus equivalentes en una medición de la sección posterior. Se debe mencionar que al momento de realizar estas mediciones no se contaba con una nube de puntos de una época posterior a la que ya se contaba, por lo que se decidió utilizar otra sección de la nube del mismo periodo, para realizar un alineamiento entre ellas

Por lo que se extrajeron de la nube ambas secciones y se aplicó el algoritmo ICP y NNS entre ellas y el resultado se puede apreciar en la figura 33. En la figura se puede observar el alineamiento de dos secciones, (772.955 y 772.940), que como se mencionó en el subcapítulo 3.3.1 esos valores representan al eje (z) de los datos medidos, que en el espacio 2D representan al identificador de cada árbol cuaternario. De acuerdo a la figura, se puede observar dos secciones con el identificador ya mencionado, alineadas respecto a sus puntos de concordancia entre la figura.

La distancia original entre ellas era mínima para simular la medición de una misma sección, sin mucha variación por deformación, ya que, a efectos reales, esta medición se aplica a dos secciones de igual identificador, es decir que sean de un mismo (z) pero en distintos espacios de tiempo.

El alineamiento es importante porque si bien un LiDAR realiza la misma medición del mismo objeto, el punto de origen es el centro del LiDAR, por lo tanto, pequeños desplazamientos de su posición original al realizar mediciones, las medidas de desplazamiento del túnel no serían correctas. Es ese el motivo por el que se deben alinear las mediciones antes de calcular movimientos convergentes.

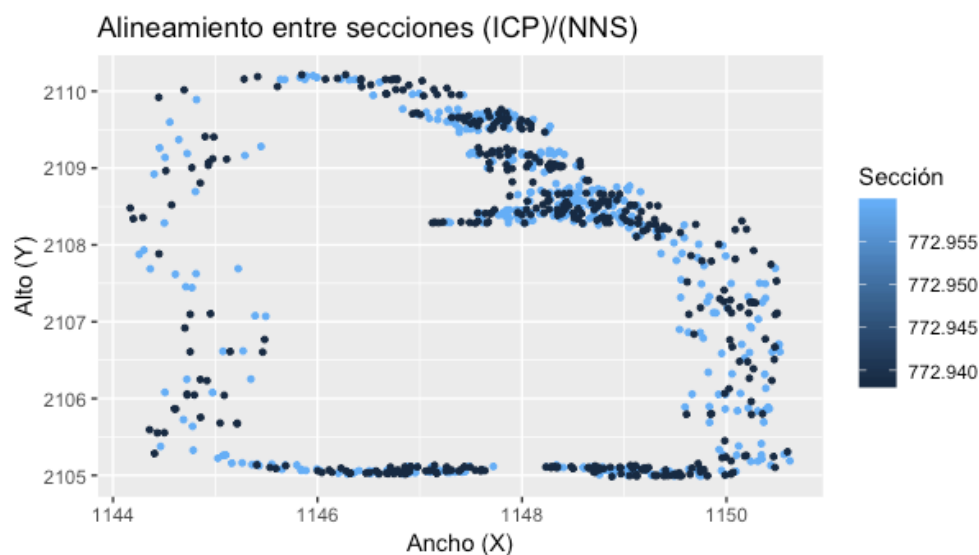


figura 33: Alineamiento de secciones mediante la ejecución de algoritmo ICP y NNS.

Capítulo 5. Conclusiones y Recomendaciones.

En relación a los resultados obtenidos respecto a las decisiones, diseño y desarrollo para la realización del software que renderice nubes de puntos masivas fueron las adecuadas, ya que se logró presentar modelos tridimensionales concordantes a los puntos recibidos, además de potenciar la optimización de los procesos de cálculo aplicando aceleración gráfica, manteniendo toda la operación bajo el mismo enfoque tecnológico. Recalcando la velocidad de renderizado lo que demuestra el alto rendimiento alcanzado.

La aplicación de Electron para obtener una aplicación multiplataforma, nos permitió generar los archivos empaquetados y ejecutables para las distribuciones de sistemas Windows, Linux y Mac. Sin la necesidad de instalar previa ni de ninguna librería externa para su funcionamiento. Tampoco se observaron variaciones significativas en las pruebas de rendimiento. Aunque cabe mencionar que las pruebas solamente se realizaron en ambientes Windows y Mac, se espera que la aplicación en Linux no presente dicha variación significativa, pensando en que su arquitectura está basada en UNIX, al igual que Mac.

La problemática que se presenta para ser abordada en una siguiente etapa, es en seguir investigando en soluciones de apoyo a medidas de convergencia, mediante la validación simultánea con cálculos tradicionales de la medida, de esta forma establecer si se presenta una tasa de error o precisión que pueda ser considerable para su implementación en ambientes mineros.

Referencias Bibliográficas.

- Apple., (2017) *GKQuadtree / Apple Developer Documentation*, Available at: <https://developer.apple.com/documentation/gameplaykit/gkquadtree> [Accessed 29 mar. 2019]
- Besl, P. J., & McKay, N. D. (1992). *Method for Registration of 3-D Shapes*. In, 1611:586-606. doi:10.1117/12.915264.
- Cabezas, R. E., & Vallejos, J.A. (2019) *A Criterion for the Extent of Damage Around Excavated Brittle Hard Rock Masses in Terms of its Convergence Over Time*. ISRM.
- Codelco, (2016). *División El Teniente*. | *Corporación Nacional del Cobre*. [online] Available at: https://www.codelco.com/descripcion-de-division-el-teniente/prontus_codelco/2016-02-25/155825.html [Accessed 03 feb. 2019]
- Codelco. (2017). *Proyectos e Inversiones Chuquicamata*. | *Corporación Nacional del Cobre* [online] Available at: https://www.codelco.com/proyectos-e-inversiones/prontus_codelco/2016-03-30/123937.html [Accessed 03 feb. 2019]
- Electron, (2018). *Electron Official Documentation*. Available at: <https://github.com/electron/electron/tree/master/docs> [Accessed dic. 2018]
- Finkel, R.A., & Bentley, J.L. (1974). *Quadtrees a Data Structure for Retrieval on Composite Keys*. Acta Informática, 4:1-9.
- Haugland, H.H., (2010), *Terrestrial LiDAR in Tunnels Under Construction*. A study of potential use for engineering geological and operational applications, and work-flow design for data acquisition and processing. (Master thesis in Geosciences). University of Oslo. Norway.
- Hossain, M. Julius, M. Dewan, A.A., Ahn, K. & Oksam, C., (2011), *A Linear Time Algorithm of Computing Hausdorff Distance for Content-Based Image Analysis*, Circuits, Systems and Signal Processing 31 (1): 389-99.
- Itasca., (2019). *Fish* | *Itasca S.A.* Available at: <https://www.itasca.cl/fish.php> [Accessed 03 feb. 2019]
- Jasim, M. (2017). *Building Cross-Platform Desktop Applications with Electron*. Packt Publishing Ltd.

- Kemeny, J., & Turner, K. (2008), *Ground-Based LiDAR: Rock Slope Mapping and Assessment*, Report FHWA-CFL/TD-08-006, Federal Highway Administration Central Federal Lands Highway Division, Lakewood, CO.
- Khronos. (2013). *The Standard for Embedded Accelerated 3D Graphics / Khronos.org*. Available at: <https://www.khronos.org/opengles/>
- Lato, M., & Diederichs, M. (2010), *Bias Correction for View-Limited for LiDAR Scanning of Rock Outcrops for Structural Characterization*, Rock Mechanics and Rock Engineering. pp. 616-619.
- Litayem, N., Bhrawna, D., & Sadia, R. (2015). *Review of Cross-Platform for Mobile Learning Application Development*, International Journal of Advanced Computer Science and Applications 6. IEEE Software 30, no. 1.
- Nvidia. (2019). *GPU Applications Catalog | Nvidia.com* [online] Available at: <https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/catalog/> [Accessed 27 mar. 2019].
- OpenGL. (2018). *OpenGL Shading Language / Khronos.org*. Available at: <http://www.opengl.org/documentation/glsl/> [Accessed nov. 2018]
- Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Elsevier.
- Scheiblauer, C., (2014), *Interaction with Gigantic Point Clouds. (PhD thesis)*. Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria.
- ThreeJS (2010) *ThreeJS JavaScript 3D library Repository / Github.com*. Available at: <https://github.com/mrdoob/three.js> [Accessed nov. 2018]
- ThreeJS Doc (2010) *ThreeJS Documentation / ThreeJS.org*. Available at: <https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene> [Accessed nov. 2018]
- WebGLStats. (2014). *WebGL Support | WebGLStats.com*. Available at: <http://webglstats.com/webgl> [Accessed 29 mar. 2019]
- Wu, R., Zhang, B., & Hsu, M. (2009), *Clustering Billions of Data Points Using GPUs*, HP Labs, Palo Alto, CA.