Universidad de Oviedo


# ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

## MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

### ÁREA DE INFORMÁTICA

# VISUALIZACIÓN DE NUBES DE PUNTOS DE ALTA DENSIDAD GENERADAS POR SENSORES 3D Y CÁMARAS LINEALES

**D. GALLEGO MATEOS, Iván**
**TUTOR: D. USAMENTIAGA FERNÁNDEZ, Rubén**

**FECHA: JULIO 2021**

Universidad de Oviedo


# DECLARACIÓN DE ORIGINALIDAD

D./Dña. Iván Gallego Mateos con DNI como alumno/a del Máster Universitario en Ingeniería Informática de la Universidad de Oviedo, DECLARO que el Trabajo Fin de Máster titulado "**VISUALIZACIÓN DE NUBES DE PUNTOS DE ALTA DENSIDAD GENERADAS POR SENSORES 3D Y CÁMARAS LINEALES**" es de mi autoría, es original y las fuentes bibliográficas utilizadas han sido debidamente citadas.

En Gijón, a 5 de Julio de 2021

Firmado:

Tabla de contenidos

# Índice de ilustraciones

## Índice de tablas

# 1 Introducción y motivación

Una nube de puntos es una estructura de datos que representa un conjunto de coordenadas 3D en el espacio. Cada punto, en su representación más básica, queda determinado por las coordenadas x, y, z que lo sitúan en el espacio, además de por información adicional relativa a su color o intensidad.

Una de las fuentes más comunes de nubes de puntos son los procesos de escaneo 3D, que permiten representar de manera veraz la superficie de un objeto o área determinada. Entre los campos de estudio que se sirven de representaciones de nubes de puntos, se podría mencionar el de la inspección de calidad industrial, la topografía o el proceso de animado 3D.

Aunque las nubes de puntos pueden ser visualizadas y analizadas a partir de los datos en bruto resultado de los procesos de escaneo 3D, es bastante frecuente su pre-procesamiento y transformación en una malla triangular mediante técnicas de reconstrucción de superficies. Estas mallas están formadas por un conjunto de triángulos con ejes comunes y representan la misma geometría que la nube de puntos inicial.

Un sistema industrial de escaneo 3D puede generar cientos de miles de medidas por segundo, resultando en conjuntos de datos de gran tamaño (GB de magnitud). Esto introduce ciertas dificultades que serán tratadas a lo largo de este estudio, y que se resumen en las siguientes:

- Necesidad de determinar un formato de representación de las nubes de puntos óptimo, que minimice el espacio en disco y facilite su lectura de manera eficiente.
- Todo procesamiento que sea necesario aplicar a las nubes de puntos resultará en largos tiempos de ejecución debido a su magnitud. Por ello, se deberá seleccionar bibliotecas especializadas que implementen las operaciones requeridas favoreciendo el rendimiento.
- La visualización de nubes de puntos de gran densidad requerirá de bibliotecas optimizadas para operar en conjuntos de datos de gran tamaño, además de depender de un motor de renderizado 3D para las tareas de visualización de bajo nivel.

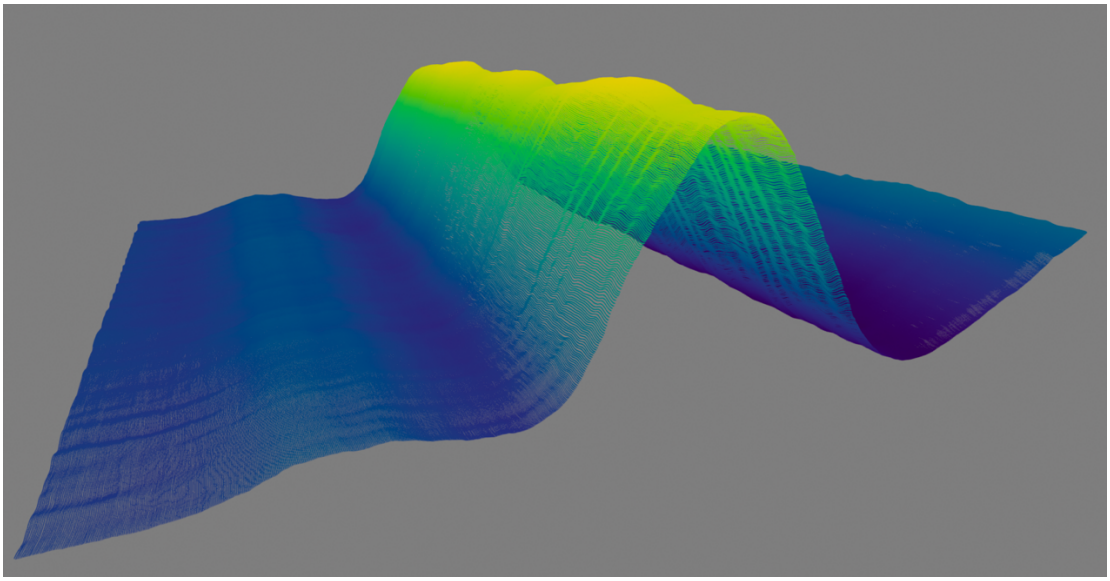


*Ilustración 1: Nube de puntos*

Resumidamente, en el presente proyecto se ha realizado un estudio pormenorizado del ecosistema actual de alternativas software para el procesamiento y visualización de geometrías 3D, así como de aspectos adyacentes como los formatos de representación de geometrías 3D. También se ha introducido una herramienta de escritorio de alcance reducido que da soporte a la aplicación de transformaciones y a la visualización de geometrías 3D diversas.

La presente memoria se estructura de la siguiente manera:

En primer lugar, se incluye un estudio para determinar el formato de representación de geometrías 3D más adecuado para las tareas de visualización y procesamiento que se realizarán posteriormente. Se hará un repaso detallado de los formatos más comunes, incluyendo una fase de pruebas para evaluar la eficiencia en el almacenamiento de cada uno de ellos.

Adicionalmente, se realizará una revisión de las principales bibliotecas de procesamiento de geometrías 3D de código abierto existentes, evaluando sus características principales. El objetivo de este análisis es doble: por una parte, seleccionar el conjunto de operaciones y transformaciones de utilidad comunes a las bibliotecas; por otro, determinar la biblioteca de procesamiento 3D alrededor de la que construir una aplicación de escritorio de elaboración propia.

El análisis anterior vendrá complementado con un estudio del rendimiento de las bibliotecas evaluadas, lo que permitirá determinar si las bibliotecas cumplen con los requisitos de procesamiento de geometrías esperados.

Posteriormente, se evaluarán las distintas alternativas, tanto de código abierto como comerciales, en el ámbito de aplicaciones de escritorio para la visualización y manejo de geometrías 3D de grandes dimensiones. Esto permitirá determinar la funcionalidad común más relevante que implementar en la herramienta desarrollada.

Finalmente, se presenta una aproximación a una aplicación de escritorio dedicada a la visualización y procesamiento de geometrías 3D, estructurada alrededor de la biblioteca de procesamiento 3D seleccionada anteriormente. Esta herramienta da soporte a un conjunto de escenarios de uso comunes en el ámbito del procesamiento de geometrías 3D, como la reconstrucción de superficies en nubes de puntos, la toma de medidas o la visualización avanzada.

La herramienta implementa un conjunto de operaciones básicas para el manejo y visualización de geometrías 3D, además de incluir operaciones de transformación de geometrías avanzadas accesibles y configurables desde la interfaz gráfica de la aplicación. La herramienta también proporciona acceso a las propiedades de renderizado de la geometría representada mediante elementos de interfaz dedicados, lo que facilita la creación de escenarios de visualización avanzada.

La herramienta ha sido diseñada para facilitar la incorporación de rutinas de transformación de geometrías adicionales, buscando reducir al mínimo los conocimientos específicos de las particularidades de la aplicación por parte del usuario programador, como la distribución y acceso a los elementos de interfaz gráfica.

La información relativa a la herramienta incluida en la memoria se reduce a un acercamiento a la motivación de la misma, acompañada de un catálogo de requisitos detallados asociados al sistema y un desglose de alto nivel de la funcionalidad implementada, con capturas de la

herramienta final. Para una referencia completa de la funcionalidad incluida desde la perspectiva de un usuario final de la aplicación, referirse al documento adjunto **MANUAL DE USUARIO**. Alternativamente, el documento adjunto **MANUAL DEL PROGRAMADOR** recoge aspectos de diseño de la herramienta como los principales diagramas de clases, de secuencia y de actividad, acompañado de detalles de implementación de bajo nivel. También se exponen los mecanismos implementados para facilitar la extensión de la herramienta añadiendo rutinas de transformación de geometrías adicionales.

Se incluye también un documento adjunto **PRUEBAS** que expone un catálogo de pruebas manuales destinado a corroborar que la herramienta implementada cumple todos los requisitos funcionales definidos en la fase de diseño de la misma. Las pruebas manuales van acompañadas de un conjunto de pruebas unitarias destinadas a evaluar el correcto funcionamiento de aspectos de implementación de la herramienta.

Finalmente, se presenta un apartado que recoge ciertas ideas y mejoras surgidas durante el proceso de desarrollo de la herramienta y que podrían ser incorporadas en futuras iteraciones, seguido de un apartado de conclusiones que resume las tareas realizadas en la totalidad del proyecto.

En el documento adjunto **PLANIFICACIÓN Y PRESUPUESTO** se incluye un desglose de las fases y tareas a realizar durante el marco temporal de ejecución del proyecto. Este documento incluye una planificación temporal estimada junto a un presupuesto detallado para la realización del proyecto.

Los documentos mencionados son acompañados del código fuente de la herramienta desarrollada durante la ejecución del proyecto en el fichero **CODIGO.zip**. Los pasos para su instalación y ejecución son detallados en el documento MANUAL DEL PROGRAMADOR. Alternativamente, se proporciona una distribución de la herramienta mediante un fichero ejecutable, contenido en el fichero **HERRAMIENTA.zip**, que elimina el proceso de instalación en sistemas Windows 10. Una vez más, los detalles para la generación de la distribución ejecutable se encuentran documentados en el MANUAL DEL PROGRAMADOR.

# 2 Formatos de datos de geometrías 3D

En el presente apartado, se evalúan algunos de los principales formatos de representación de geometrías 3D (nubes de puntos y mallas), exponiendo sus principales características. El estudio resultante permitirá determinar el formato de datos más apropiado para la representación de las geometrías que serán utilizadas en el proyecto.

Por norma general, cualquier formato de fichero para la representación de estructuras 3D será válido tanto para la representación de nubes de puntos como de mallas, puesto que cualquier estructura está formada por un conjunto de polígonos identificados a su vez por los vértices que los conforman. De este modo, una nube de puntos podría ser considerada una estructura 3D que solamente contiene vértices, no siendo necesario almacenar información sobre las caras que la conforman. En la evaluación de los distintos formatos de datos, se favorecerá a aquellos que permitan representar tanto nubes de puntos como mallas triangulares, puesto que se espera que la herramienta desarrollada al final del proyecto permita visualizar ambos tipos de geometría.

El principal aspecto diferenciador entre los distintos formatos es la representación base que emplean, que puede ser binaria o ASCII. Resulta evidente que los formatos basados en texto plano generan ficheros de tamaño mucho mayor que los formatos binarios. Alternativamente, los formatos binarios pueden ser manejados de manera más eficiente en operaciones de lectura y escritura a la hora de tratar geometrías de gran tamaño.

De esta manera, los formatos basados en texto plano quedarían relegados a la representación de geometrías de menor tamaño, en los que pudiera resultar de utilidad la edición o lectura de los ficheros de manera directa, sin depender de herramientas alternativas.

## 2.1 PTS

El formato de fichero PTS permite representar nubes de puntos generadas por escáneres LIDAR almacenando únicamente las características básicas de cada punto: coordenadas espaciales, color e intensidad. Puesto que sólo se almacenan las coordenadas de cada punto, el formato PTS no permite representar mallas triangulares.

Siendo un formato de texto plano, su principal ventaja es su legibilidad, puesto que puede ser abierto e interpretado en cualquier editor de texto. No obstante, este aspecto resulta a su vez en ficheros de gran tamaño cuando se trata de representar una cantidad de puntos elevada. La necesidad de interpretación de cada elemento en texto plano dificulta a su vez el procesamiento posterior de la nube de puntos por otras herramientas.

El formato de fichero PTS es comparable a otros formatos de representación de geometrías 3D en texto plano, como el formato XYZRGB, cuya especificación se limita a incluir los valores de coordenadas y color de los puntos de una nube en líneas independientes.

### 2.1.1 Especificación

Con respecto a la estructura de los ficheros, el formato PTS incluye, en la primera línea, el número total de puntos representados en el fichero. Las líneas restantes representan la información de un punto en el siguiente orden: coordenadas x, y, z, intensidad de la señal del escáner y colores RGB representados como un byte (0-255).

El valor de la intensidad se corresponde con la estimación de la radiación reflejada por la superficie representada, tomándose como 0 un retorno de señal pobre y 255 como muy robusta.

Los valores de la intensidad son a menudo dependientes del fabricante de cada escáner, pudiendo existir diferencias entre vendedores.

El siguiente fichero de ejemplo representa una nube de puntos de 4 puntos en formato PTS:

```
4
0 0 0 220 255 0 0
10 0 0 220 255 0 0

0 10 0 223 255 0 0
10 10 0 215 255 0 0
```

## 2.2 PLY

El formato de fichero PLY (*Polygon File Format*) fue desarrollado en la Universidad de Stanford con el objetivo de aportar versatilidad a formatos previos para la representación de superficies 3D, que se limitaban a representar exclusivamente las dimensiones y características básicas de una superficie (vértices y caras).

Por el contrario, el formato PLY, además de definir estructuras primarias como vértices y caras, permite definir dimensiones adicionales a voluntad del usuario. Alternativamente, también ofrece la posibilidad de asociar propiedades o valores a cada una de las dimensiones definidas (por ejemplo, es posible asociar valores RGB a cada vértice).

La posibilidad de representar propiedades arbitrarias asociadas a los elementos principales de la geometría 3D de un objeto permite utilizar bibliotecas especializadas que sepan interpretar estas propiedades sin invalidar la interpretación del fichero por aplicaciones más básicas que solo sean capaces de interpretar ejes y caras.

Por su naturaleza, permite representar tanto geometrías de nubes de puntos como mallas poligonales.

### 2.2.1 Especificación

Los ficheros de datos en formato PLY constan de dos fragmentos principales:

- **Cabecera**: Representación en texto plano de metadatos del fichero, como por ejemplo el sub-formato (binario o ASCII), la definición de las dimensiones representadas, el tipo de datos y el número de elementos de cada una de ellas.
- **Cuerpo**: Datos en bruto de cada uno de los elementos definidos en la cabecera, junto con las propiedades asociadas a los mismos.

La cabecera del fichero, representada en texto plano, está formada por un conjunto de líneas que representan los metadatos necesarios para interpretar el resto del fichero. La cabecera proporciona una descripción de cada tipo de elemento incluido en el fichero, su nombre y el número de elementos de este tipo que aparecerán, junto con propiedades asociadas al mismo. La cabecera también define si el fichero es binario o de texto plano.

A continuación, se proporciona una cabecera de ejemplo que representa una nube de puntos, asociando unas coordenadas x, y, z a un conjunto de vértices:

```
ply

format ascii 1.0
```

```
element vertex 10

property float x

property float y

property float z

end_header

...
```

La cabecera anterior define un elemento vértice del que se presentarán 10 elementos, cada uno con 3 propiedades asociadas, x, y, z representadas con tipo flotante. Los caracteres *ply* de la primera línea siempre deben aparecer al comienzo del fichero, así como la clave "*format*", que define el formato y la versión del fichero.

A continuación de la cabecera, se incluyen los datos en bruto que representan cada elemento. Las propiedades incluidas tras la línea de definición de un elemento definen el orden en que las propiedades serán representadas, además del tipo de datos de cada uno de ellos. El formato acepta dos tipos de datos para cada propiedad: escalar o lista. Los tipos escalares aceptados son los siguientes:

| ***Nombre*** | **Tipo de datos** | **Bytes** |
|---|---|---|
| *char* | Carácter | 1 |
| *uchar* | Carácter sin signo | 1 |
| *short* | Entero 2 bytes | 2 |
| *ushort* | Entero 2 bytes sin signo | 2 |
| *int* | Entero | 4 |
| *uint* | Entero sin signo | 4 |
| *float* | Flotante precisión simple | 4 |
| *double* | Flotante precisión doble | 8 |

*Tabla 1: Tipos de datos PLY*

Existe una definición de propiedad alternativa que permite definir un tipo de datos con un número de elementos variable. El siguiente ejemplo muestra cómo es posible definir una superficie como una lista que contiene un número predefinido de vértices.

```
property list uchar int vertex_index
```

La propiedad `vertex_index` de tipo lista contiene en primer lugar un carácter que determina el número de valores que contiene, seguido de una lista de enteros que en este caso representan el índice de los vértices.

### 2.2.2 Ejemplo

El siguiente fichero de ejemplo representa un cubo en formato *.ply*. En primer lugar, se definen los elementos que lo componen (8 vértices y 6 caras). Cada vértice está representado por sus coordenadas x, y, z y cada cara por una lista con los identificadores de los vértices que la conforman.

```
ply
format ascii 1.0
```

```
element vertex 8
property float x
property float y
property float z
element face 6
property list uchar int vertex_indices
end_header
-1 -1 -1
1 -1 -1
1 1 -1
-1 1 -1
-1 -1 1
1 -1 1
1 1 1
-1 1 1
4 0 1 2 3
4 5 4 7 6
4 6 2 1 5
4 3 7 4 0
4 7 3 2 6
4 5 1 0 4
```

La Ilustración 2 muestra el cubo representado en el fichero *.ply* anterior.

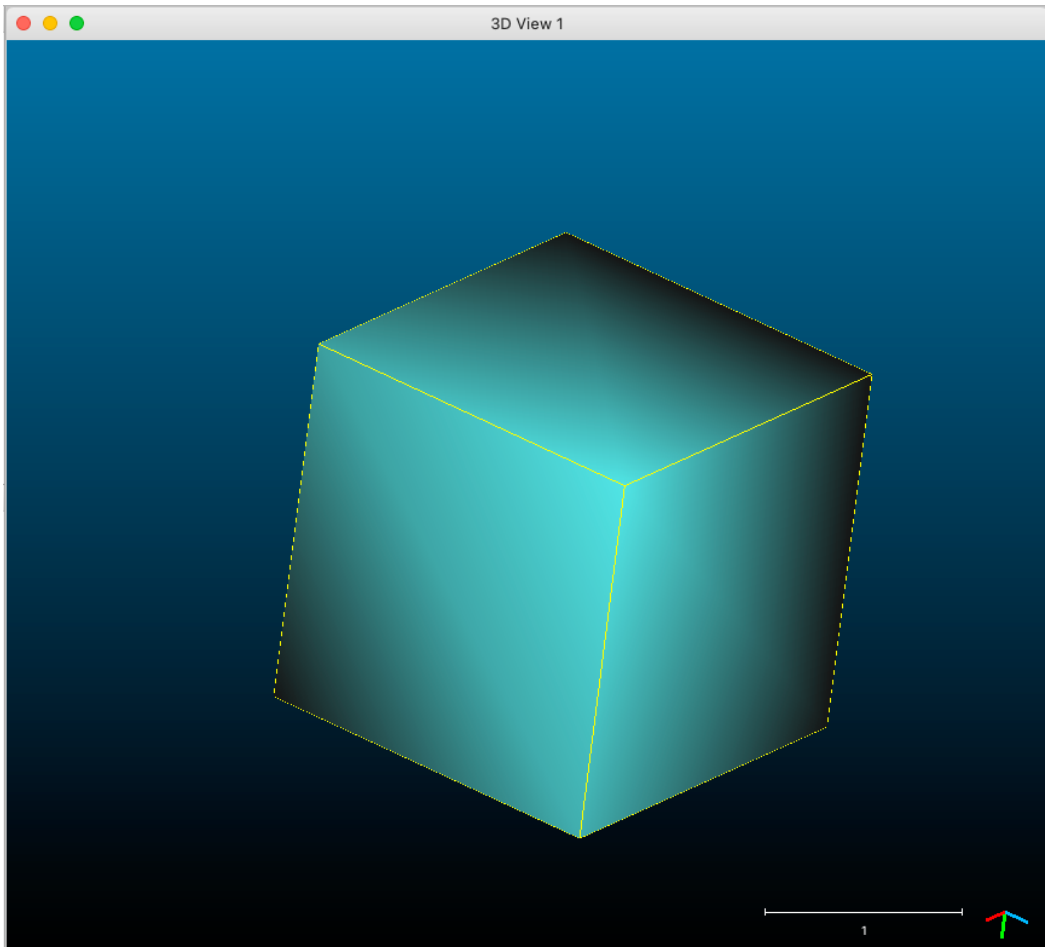



*Ilustración 2: Cubo .ply*

### 2.3 PCD

El formato PCD para la representación de geometrías 3D fue ideado para operar en el marco de la biblioteca PCL (*Point Cloud Library*), aunque resuelve a su vez ciertas ineficiencias existentes en otros formatos para la representación de datos de N dimensiones. Como principal particularidad, destaca su eficiencia en la representación de nubes de datos estructuradas, además de permitir definir nubes de puntos cuyas dimensiones principales tienen a su vez múltiples componentes.

Las nubes de puntos estructuradas presentan una estructura que asemeja al de una imagen (o matriz) organizada, en el que los datos se dividen en filas y columnas. Como ejemplo de nube de puntos organizada se podría considerar los datos generados por cámaras estéreo 3D. La ventaja de representar la nube de puntos de manera estructurada es que permite conocer la relación entre puntos adyacentes (ej. píxeles en una imagen), lo que mejora la eficiencia de cierto tipo de algoritmos de computación.

#### 2.3.1 Especificación

Cada fichero PCD contiene una cabecera que declara un conjunto de propiedades o metadatos de la nube de puntos almacenada en el fichero. Esta cabecera se define siempre en ASCII, independientemente de que los datos de los puntos se representen en ASCII o binario. A

continuación de la cabecera se encuentra la información de los puntos 3D. Cada uno de los campos de la cabecera, además de la propia información de cada punto, se define en líneas independientes.

La versión más reciente del formato PCD (*PCD_V7*), define los siguientes campos en la cabecera:

- **VERSION**: Versión del fichero PCD.
- **FIELDS**: Nombre de cada dimensión asociada a un punto.
- **SIZE**: Tamaño de cada dimensión en bytes.
- **TYPE**: Especifica el tipo de cada dimensión como un carácter. Los tipos soportados son *I* para dimensiones enteras positivas, *U* para las dimensiones enteras sin signo y *F* para tipos flotantes.
- **COUNT**: Número de elementos presente para cada dimensión. Esto permite definir dimensiones con múltiples valores que serán almacenados en posiciones de memoria contiguas.
- **WIDTH**: Ancho de la nube de puntos. Para nubes de puntos no estructuradas el ancho representa el número total de puntos de la nube, y será igual al campo *POINTS* definido posteriormente. En nubes de datos organizadas este campo determina el número de puntos que corresponden a una fila.
- **HEIGTH**: Altura en número de puntos de una nube de puntos. En nubes de datos organizadas determina el número de filas del conjunto de datos. En nubes de datos no organizadas se fija a 1.
- **VIEWPOINT**: Angulo de adquisición de los puntos como una combinación de la traslación (tx, ty, tz) y el cuaternión (qw, qx, qy, qz)
- **POINTS**: Número total de puntos de la nube.
- **DATA**: Formato de almacenamiento de los puntos de la nube. Los formatos válidos son "*ascii*" y "*binary*"

### 2.3.2 Ejemplo

El siguiente fichero de ejemplo representa los vértices de un cubo en formato .pcd:

```
VERSION 0.7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 8
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 8
DATA ascii
-1 -1 -1
1 -1 -1
1 1 -1
-1 1 -1
-1 -1 1
1 -1 1
1 1 1
```

```
-1 1 1
```

## 2.4 LAS

El formato de fichero LAS fue originado como medio de intercambio libre para nubes de puntos generadas por escáneres LIDAR. A raíz de este se generó el formato LAZ, una variante que introduce compresión sin pérdida de información del formato LAS. Existen diferentes versiones de la especificación LAS que fueron apareciendo a lo largo de los años para satisfacer las necesidades producidas por nuevos tipos de escáner. Cada una de estas versiones introducía a su vez nuevos formatos de representación de cada punto de la nube.

El formato LAS es, en muchos casos, el formato más adecuado para la representación de datos generados por escáneres LIDAR, puesto que permite almacenar de manera eficiente múltiples datos generados en el proceso de escaneo de cada punto perteneciente a la nube, como el número de retorno o el ángulo de escaneo.

No obstante, puesto que todos los campos son obligatorios para cada punto representado en la nube, su utilización resulta ineficiente en aquellos casos en que no se disponga de todos estos datos, o sólo se pretenda representar información espacial 3D básica de una nube de puntos.

### 2.4.1 Especificación

Los ficheros LAS se organizan en 3 segmentos principales:

- **Cabecera**: Contiene información sobre los datos almacenados, como la versión del formato LAS del fichero y las dimensiones y tipos de los datos almacenados para cada punto.
- **Registros de longitud variable (VLR, *Variable Length Record*)**: Permite almacenar información adicional asociada a cada punto de la nube, como por ejemplo el sistema de referencia espacial utilizado o dimensiones adicionales.
- **Registro de puntos (*Point Records*)**: Incluye la información de los puntos que componen la nube, presente de manera secuencial. Todos los puntos son representados con el mismo formato (especificado en la cabecera), lo que determina las dimensiones y campos almacenados en el fichero.

La siguiente tabla especifica todos los campos que se definen para cada punto aplicando el formato LAS 2, con un tamaño total por punto de 26 bytes. Formatos posteriores añaden dimensiones adicionales o incrementan el tamaño de los campos existentes para representar valores de mayor magnitud.

| Campo | Formato | Tamaño | Obligatorio |
|---|---|---|---|
| x | long | 4 bytes | * |
| y | long | 4 bytes | * |
| z | long | 4 bytes | * |
| Intensidad | unsigned short | 2 bytes | |
| Número de retorno | unsigned short | 3 bits | * |
| Retornos por pulso | 3 bits | 3 bits | * |
| Flag dirección del escáner | 1bit | 1 bit | * |
| Edge of flight | 1 bit | 1 bit | * |
| Clasificación | unsigned char | 1 byte | * |

| | | | |
|---|---|---|---|
| Ángulo escáner | char | 1 byte | * |
| Datos usuario | unsigned char | 1 byte | |
| Id fuente | unsigned short | 2 bytes | * |
| R | unsigned short | 2 bytes | * |
| G | unsigned short | 2 bytes | * |
| B | unsigned short | 2 bytes | * |

*Tabla 2: Campos LAS*

## 2.5 Comparativa formatos de fichero

Una vez analizados los principales formatos de representación de geometrías 3D, se procede a realizar una comparativa de los mismos, atendiendo al tamaño de fichero generado por cada uno de ellos a la hora de representar una nube de puntos dada.

### 2.5.1 Metodología

Con el objetivo de definir un procedimiento de comparativa replicable, se decide implementar una rutina para la creación de un conjunto de datos sintético con número de puntos parametrizable. La implementación del script se realizó en Python haciendo uso de las bibliotecas NumPy y Open3D.

El script implementado genera, de manera dinámica, una matriz tridimensional con las coordenadas x, y, z de cada uno de los puntos que conformará la nube de puntos. La matriz generada representa un cuadrado en un espacio 2D a la que posteriormente se le aplica una transformación que desplaza la mitad de la matriz en el eje z, añadiendo la tercera dimensión. Los colores RGB de cada uno de los puntos son generados, a su vez, de manera aleatoria.

Una vez generada la matriz tridimensional, se procede a construir la nube de puntos haciendo uso de la biblioteca de Open3D, que ofrece integración directa con los arrays multidimensionales generados por NumPy. Inicializar la nube de puntos mediante este método facilita la visualización de esta, además de dar acceso a funciones de utilidad para la escritura de nubes de puntos de Open3D. De esta manera, es posible representar la misma nube de puntos en los siguientes formatos de manera automática:

- PLY binario.
- PLY texto plano.
- PCD binario.
- PCD texto plano.
- PTS.
- XYZRGB

A continuación, se presenta una captura de la visualización de la nube de puntos generada. Los contenidos del script en su totalidad serán incluidos en el entregable de código final del proyecto.

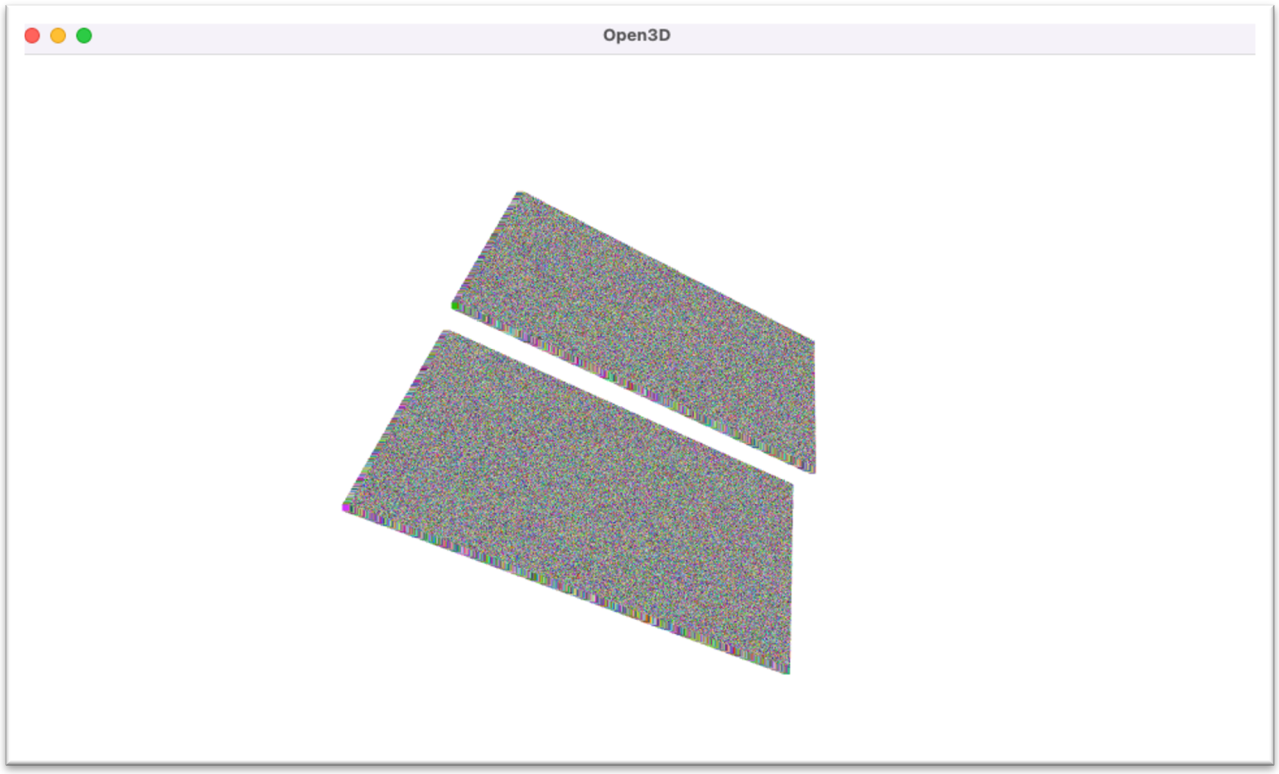
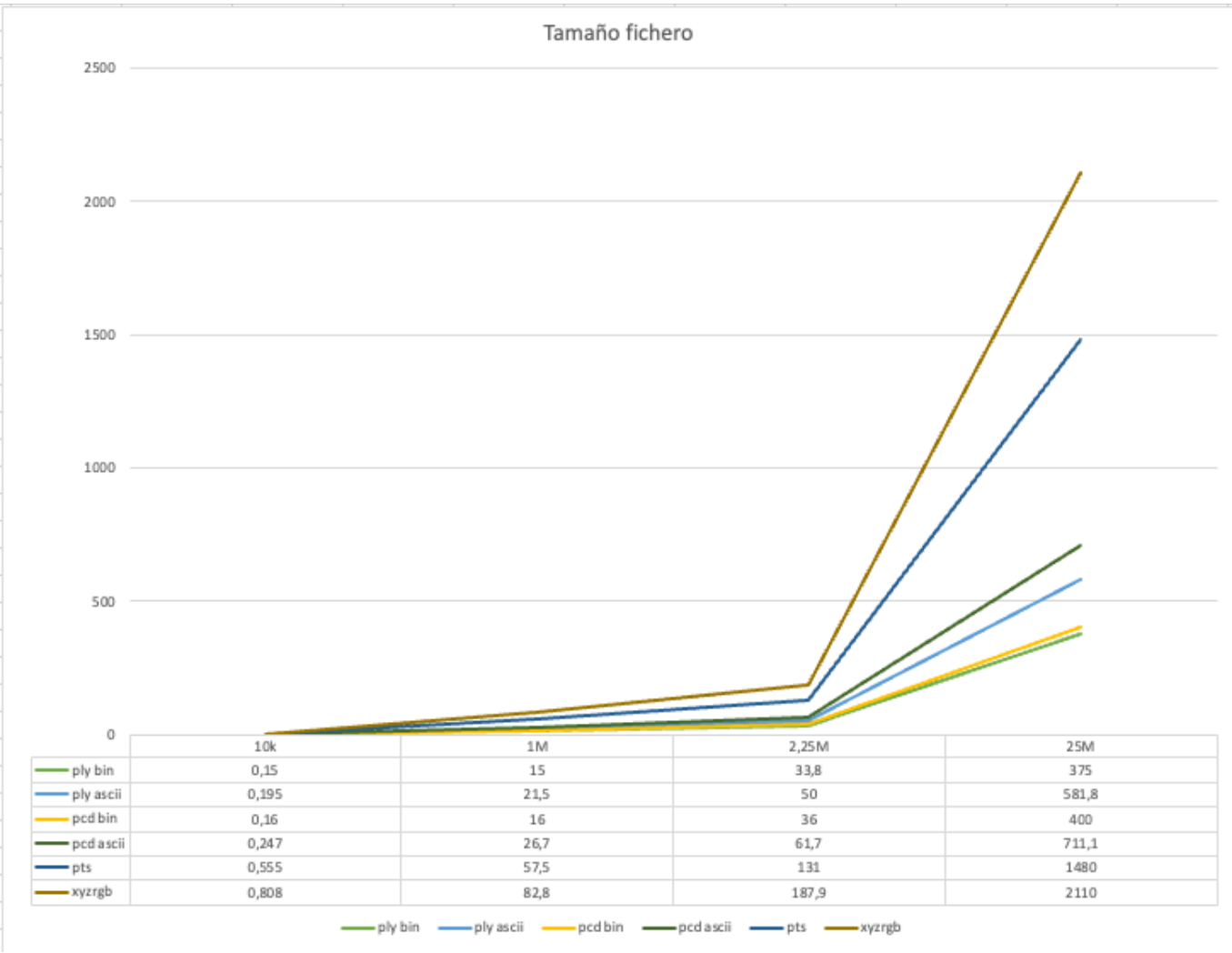



*Ilustración 3: Nube de puntos sintética*

### 2.5.2 Evaluación de resultados

La siguiente gráfica representa los resultados de la comparativa de todos los ficheros generados:




| | 10k | 1M | 2,25M | 25M |
|---|---|---|---|---|
| ply bin | 0,15 | 15 | 33,8 | 375 |
| ply ascii | 0,195 | 21,5 | 50 | 581,8 |
| pcd bin | 0,16 | 16 | 36 | 400 |
| pcd ascii | 0,247 | 26,7 | 61,7 | 711,1 |
| pts | 0,555 | 57,5 | 131 | 1480 |
| xyzrgb | 0,808 | 82,8 | 187,9 | 2110 |

*Ilustración 4: Evolución tamaño de fichero nube de puntos representados en MB*

A partir de los resultados obtenidos se percibe que el tamaño de fichero generado con el formato PLY en su representación binaria era mucho mayor del esperado, siendo en muchos casos similar al de su representación en texto plano. Se llega a la conclusión de que la función de escritura de nubes de puntos de la biblioteca Open3D representa cada una de las coordenadas x, y, z de los puntos con tipo flotante de 64 bits, en lugar de flotante de 32 bits como debería.

Para corregir este problema, se decidió importar la nube de puntos en el software de visualización de geometrías 3D *CloudCompare* y exportarla de nuevo en formato PLY binario, lo que genera los valores finales representados.

Como era de esperar, los formatos de representación de datos en texto plano generan ficheros de tamaño mucho mayor que su contrapartida binaria. Estos formatos quedan relegados, de esta manera, a la representación de nubes de puntos de tamaño reducido, en las que pueda ser necesario una interpretación directa del contenido del fichero.

Con respecto a los formatos de representación de nubes de puntos en binario, podemos comprobar que generan ficheros de tamaño similar. Esto se debe a que, a bajo nivel, ambos

formatos representan las coordenadas de cada punto con el mismo tipo. A continuación, se presentan las cabeceras de los ficheros PLY y PCD generados, donde se puede comprobar el tipo de datos utilizado en la representación de cada variable.

```
ply

format binary_little_endian 1.0

element vertex 10000

property float x

property float y

property float z

property uchar red

property uchar green

property uchar blue

end_header

…
```

```
# .PCD v0.7 - Point Cloud Data file format

VERSION 0.7

FIELDS x y z rgb

SIZE 4 4 4 4

TYPE F F F F

COUNT 1 1 1 1

WIDTH 10000

HEIGHT 1

VIEWPOINT 0 0 0 1 0 0 0

POINTS 10000

DATA binary

…
```

Es posible comprobar como, en el caso del fichero con formato PLY, las coordenadas x, y, z son representadas como flotantes de 32 bits, mientras que los valores RGB se representan como caracteres de 1 byte. En el caso del fichero con formato PCD, tanto las coordenadas x, y, z como los valores RGB son representados como flotantes de 32 bits, siendo los valores RGB agrupados en una única variable.

Por completitud, se presentan los resultados de la comparativa en la Tabla 3

| Número de puntos | Formato fichero | Subformato | Tamaño |
|---|---|---|---|
| **10.000** | PLY | Binario | 150 KB |
| **10.000** | PLY | Texto | 195 KB |
| **10.000** | PCD | Binario | 160 KB |
| **10.000** | PCD | Texto | 247 KB |
| **10.000** | PTS | - | 555 KB |
| **10.000** | XYZRGB | - | 808 KB |
| **1 M** | PLY | Binario | 15 MB |
| **1 M** | PLY | Texto | 21.5 MB |
| **1 M** | PCD | Binario | 16 MB |
| **1 M** | PCD | Texto | 26.7 MB |
| **1 M** | PTS | - | 57.5 MB |
| **1 M** | XYZRGB | - | 82.8 MB |
| **2.25 M** | PLY | Binario | 33.8 MB |
| **2.25 M** | PLY | Texto | 50 MB |
| **2.25 M** | PCD | Binario | 36 MB |
| **2.25 M** | PCD | Texto | 61.7 MB |
| **2.25 M** | PTS | - | 131 MB |
| **2.25 M** | XYZRGB | - | 187.9 MB |
| **25 M** | PLY | Binario | 375 MB |
| **25 M** | PLY | Texto | 581.8 MB |
| **25 M** | PCD | Binario | 400 MB |
| **25 M** | PCD | Texto | 711.1 MB |
| **25 M** | PTS | - | 1.48 GB |
| **25 M** | XYZRGB | - | 2.11 GB |

*Tabla 3: Tamaños de fichero*

Por los resultados obtenidos, se considera que tanto el formato PLY como el formato PCD, en sus representaciones binarias, pueden ser los indicados para la representación de nubes de puntos requeridas por el presente proyecto. Puesto que también se considera la posibilidad de que la herramienta desarrollada permita importar mallas triangulares, se favorecerá la utilización del formato .ply, ya que permite representar ambos tipos de geometría.

## 2.6 Geometrías 3D de referencia

Por completitud, se presentan a continuación una serie de capturas y características generales de las geometrías 3D que serán utilizadas a lo largo del proyecto. Todas ellas fueron generadas en el formato de representación PLY, según las conclusiones extraídas de los apartados anteriores. Todas las geometrías utilizadas serán distribuidas como referencia junto al código fuente de la herramienta desarrollada, lo que permitirá replicar los escenarios de uso expuestos y evaluar la herramienta.

En primer lugar, se presenta una serie de geometrías generadas por cámaras de profundidad y escáneres industriales. La Ilustración 5 representa una superficie como una nube de puntos que cuenta con 986.472 puntos. Por el contrario, las capturas presentadas en la Ilustración 6, Ilustración 7 e Ilustración 8 representan diferentes tipos de carril como mallas triangulares.

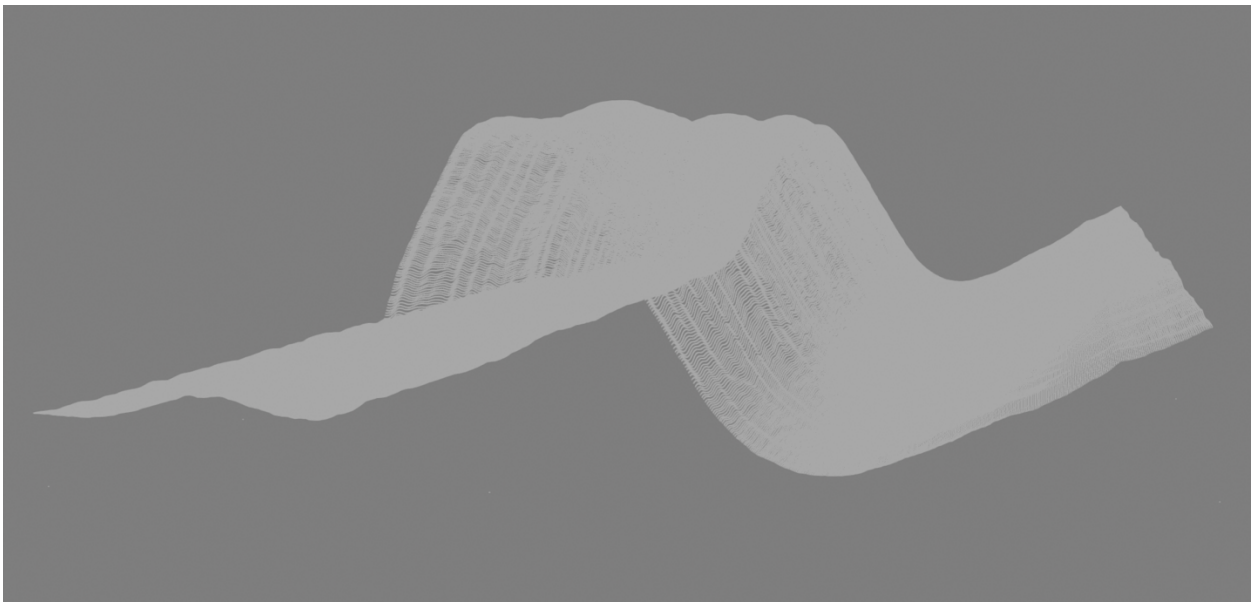


*Ilustración 5: Nube de puntos cloud.ply*

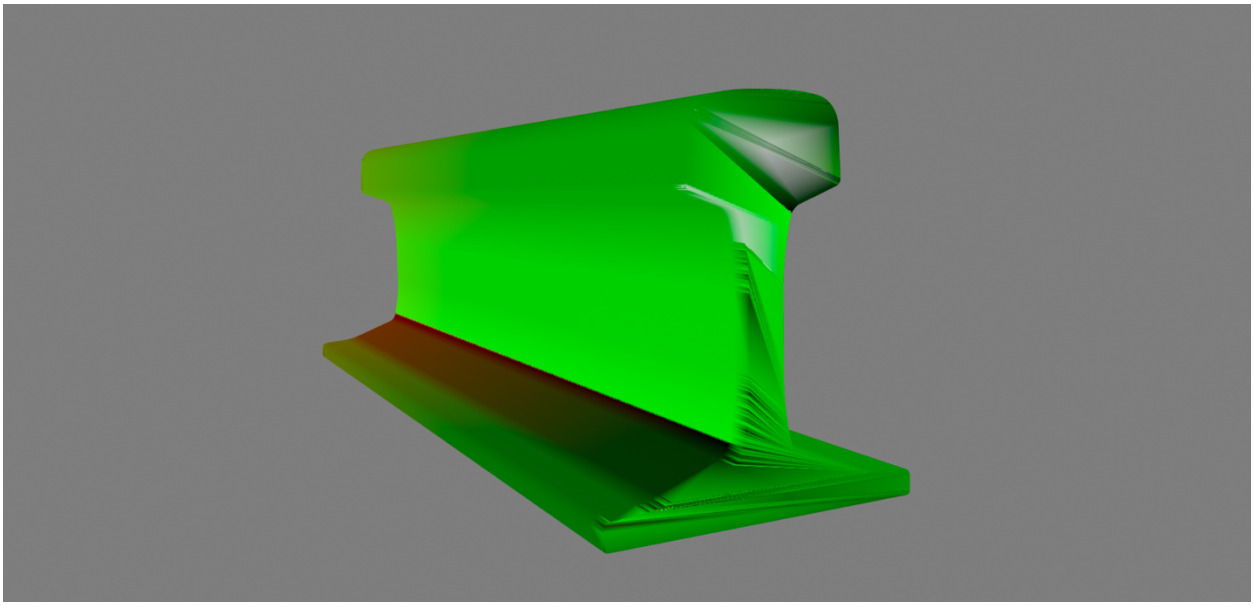


*Ilustración 6: Malla CAR1.ply*

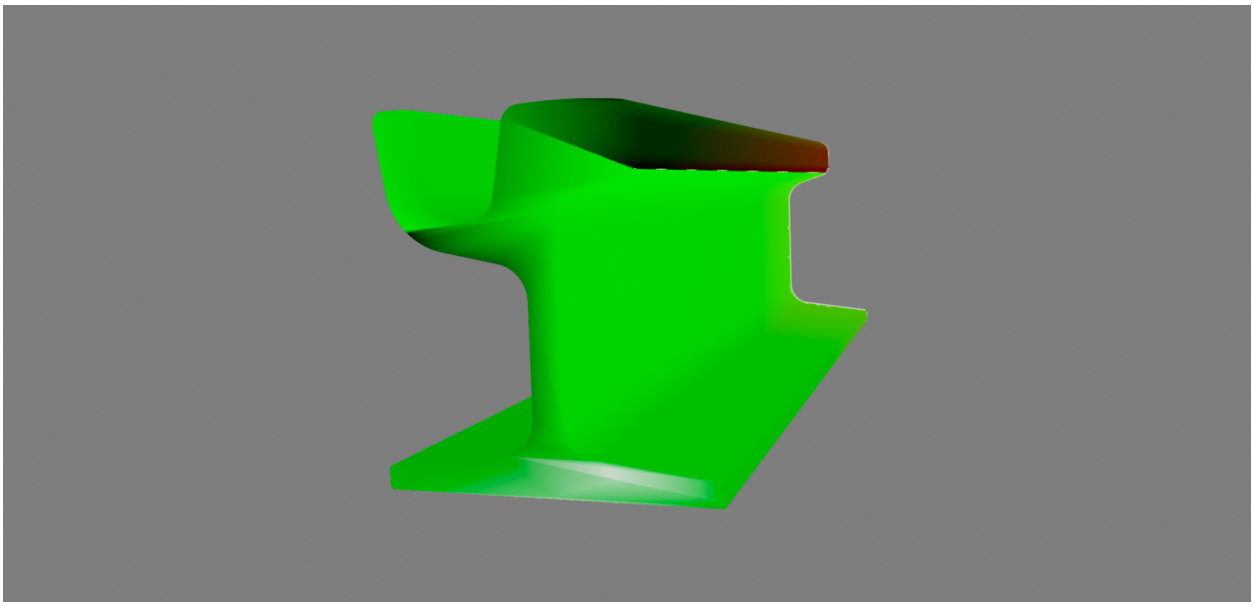


*Ilustración 7: Malla CAR2.ply*

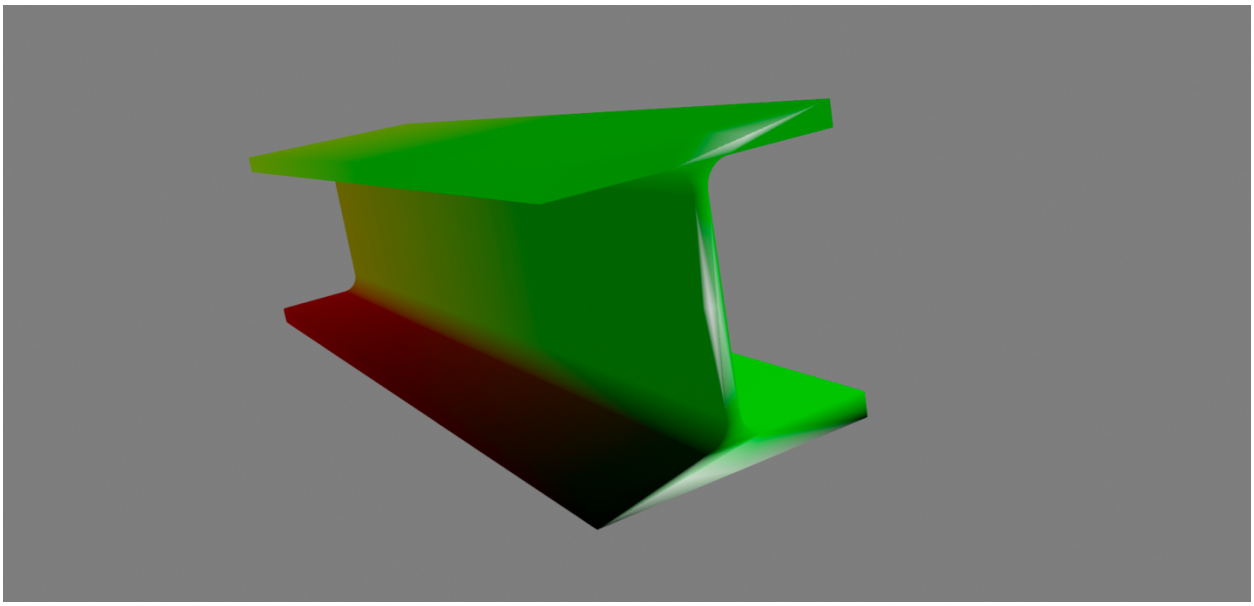


*Ilustración 8: Malla CAR3.ply*

Por último, se incluye una geometría de ejemplo, extraída de la biblioteca de procesamiento 3D Open3D, que representa una habitación como una nube de puntos generada por un escáner 3D, lo que permite demostrar ciertas funcionalidades de las bibliotecas de procesamiento de nubes de puntos, como los algoritmos de *clustering*, que serán tratados más adelante. Esta geometría es presentada en la captura de la Ilustración 9.

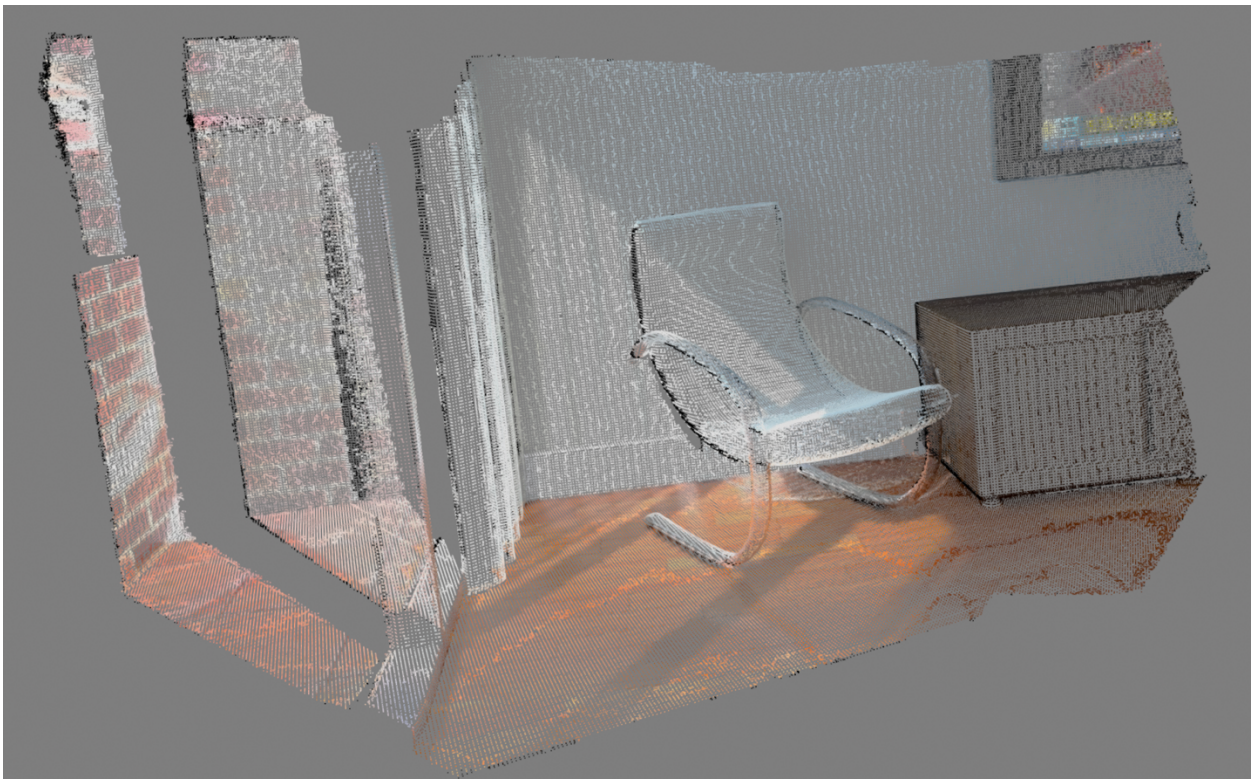


*Ilustración 9: Nube de puntos room.ply*

La Tabla 4: Propiedades geometrías 3D muestra las principales características de cada una de las geometrías utilizadas, destacando el tamaño de estas.

| Nombre | Tipo | Num. vértices | Num triángulos | Tamaño fichero |
|---|---|---|---|---|
| **cloud** | Nube de puntos | 986.472 | - | 27,6 MB |
| **room** | Nube de puntos | 196.133 | - | 6,1 MB |
| **CAR1** | Malla triangular | 31.925 | 59.264 | 1,3 MB |
| **CAR2** | Malla triangular | 28.713 | 53.962 | 1,2 MB |
| **CAR3** | Malla triangular | 9.597 | 17.226 | 378 KB |

*Tabla 4: Propiedades geometrías 3D*

# 3 Bibliotecas de procesamiento de geometrías 3D

El presente apartado realiza un análisis de las principales bibliotecas software de código abierto dedicadas al procesamiento de geometrías de nubes de puntos. Todas ellas presentan una interfaz exclusivamente programática que permite implementar rutinas de transformación de geometrías 3D.

Todas las bibliotecas evaluadas cuentan a su vez con módulos de renderizado para la representación de las geometrías tratadas, aunque no necesariamente proporcionan módulos para la creación de interfaces gráficas de usuario. De este modo, si se quisiera desarrollar una aplicación de escritorio con interfaz gráfica que hiciera uso de estas bibliotecas de

procesamiento, sería necesario realizar un proceso de integración con un *framework* dedicado para la creación de interfaces, como Qt.

En un apartado posterior (Sección 4), se evaluará un conjunto de aplicaciones de escritorio o navegador que sí proporcionan una interfaz gráfica de usuario como punto de entrada a sus funciones de transformación y visualización de geometrías.

Con el objetivo de realizar una primera selección de las bibliotecas más relevantes, se evaluó la actividad de cada una como iniciativa de código abierto, presentando los resultados en la Tabla 5. La tabla mencionada resume la relevancia de cada una de las bibliotecas, evaluada con respecto a la tracción y contribuciones de cada una de ellas en Github. Por referencia, los datos fueron tomados en Enero de 2021 y pueden haber cambiado en el desarrollo del proyecto.

| Biblioteca | Lenguaje | Repositorio código fuente | Menciones | Última contribucion |
|---|---|---|---|---|
| Open3D | Bindings Python, backend C++ | https://github.com/intel-isl/Open3D/ | 3900 | Jan 22, 2021 |
| PyVista | Bindings Python, backend C++ | https://github.com/pyvista/pyvista | 664 | Jan 23, 2021 |
| Pyntcloud | Python | https://github.com/daavoo/pyntcloud | 884 | Nov 19, 2020 |
| PCL | C++ | https://github.com/PointCloudLibrary/pcl | 5900 | Jan 22, 2021 |
| Cilantro | C++ | https://github.com/kzampog/cilantro | 566 | 29 Sep, 2020 |

*Tabla 5: Menciones bibliotecas*

Los siguientes apartados presentan una breve descripción de las bibliotecas de procesamiento de nubes de puntos evaluadas, exponiendo sus principales características, puntos fuertes y desventajas. Este análisis inicial permitirá determinar la biblioteca más adecuada para el desarrollo de la herramienta de procesamiento de geometrías 3D planteada como objetivo de este proyecto. También permitirá identificar funcionalidad común a las bibliotecas que puede ser integrada en la herramienta desarrollada.

En todos los casos presentados a continuación, las descripciones de las características y funcionalidad de cada biblioteca fueron extraídas directamente de la documentación y de las APIs de referencia de cada biblioteca. Algunas de ellas fueron evaluadas en mayor profundidad en el análisis de rendimiento presentado en un apartado posterior.

### 3.1 Open3D

Open3D es una biblioteca de código abierto que facilita el desarrollo de software relacionado con el procesamiento de geometrías 3D, creada por el equipo de *Intel Intelligent Systems Lab* en 2018. Presenta como principales fortalezas la eficiencia en el procesamiento de geometrías 3D,

su diseño orientado a la paralelización y la facilidad de uso, reduciendo al mínimo las dependencias de bibliotecas complejas.

*Ilustración 10: Logo Open3D*

La principal ventaja de Open3D es su versatilidad y simplicidad. Como biblioteca de procesamiento de geometrías 3D, facilita tanto la implementación de rutinas de procesamiento estáticas como el desarrollo de aplicaciones complejas. Dispone de módulos para la creación de interfaces gráficas y para el renderizado de geometrías 3D, lo que posibilita la creación de aplicaciones de escritorio. Ha sido implementada en C++, aunque incluye *bindings* para Python en todos sus componentes, lo que facilita el acceso a su ecosistema de bibliotecas, posibilitando casos de uso como el *machine learning* aplicado a modelos de datos 3D. La Ilustración 11 muestra un ejemplo de aplicación de visualización creada exclusivamente con esta biblioteca.

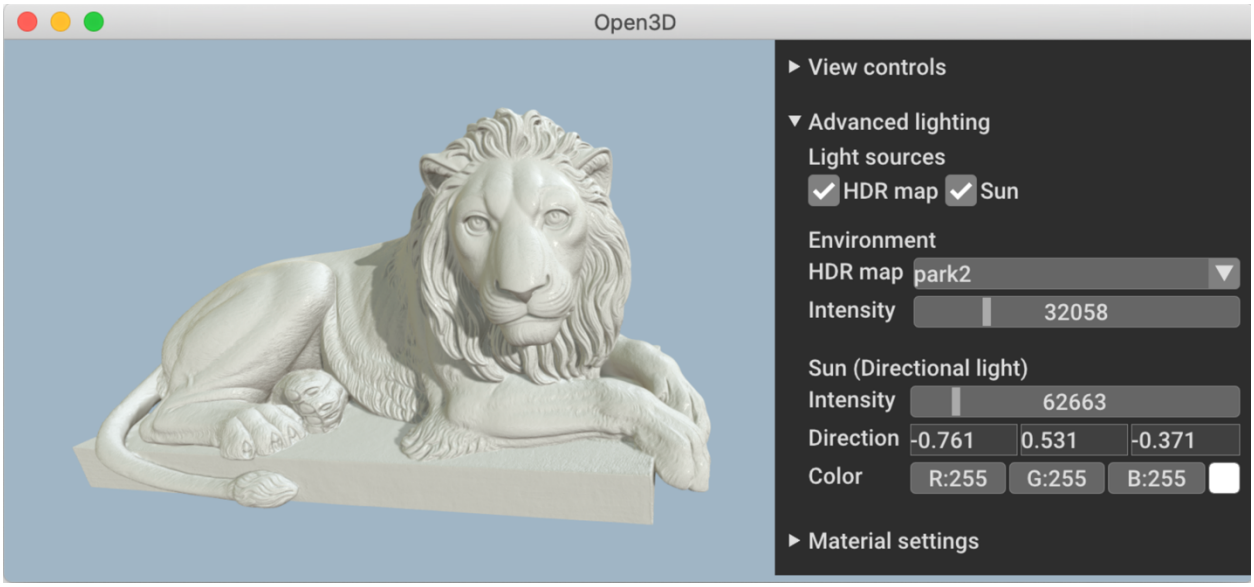



*Ilustración 11: Ejemplo visualización Open3D*

Es un proyecto joven y en continuo desarrollo con gran apoyo de la comunidad de código abierto, contando con alrededor de 3900 menciones en Github. Como contrapartida, la versión más reciente de la biblioteca (0.13.0), aún siendo robusta, se encuentra lejos de ser una versión definitiva. Esto es particularmente notable en sus módulos de visualización y desarrollo de interfaces gráficas de usuario, en constante cambio.

### 3.1.1 Características principales

Open3D implementa estructuras de datos propias para la representación de nubes de puntos, mallas triangulares e imágenes de profundidad RGBD. Cada una de ellas tiene asociado un conjunto de algoritmos de procesamiento básicos que facilitan su lectura, transformación, visualización y manejo. También implementa módulos de mayor complejidad, como los dedicados al aprendizaje automático en entornos 3D.

**Geometrías soportadas**

En el caso de las estructuras de nubes de puntos, Open3D ofrece operaciones de utilidad para el importado y exportado de ficheros en los siguientes formatos:

- **XYZ**: Formato de texto plano en que cada línea representa las coordenadas x, y, z de cada punto separadas por comas.
- **XYZN**: Formato de texto plano en que cada línea representa las coordenadas x, y, z y las normales nx, ny, nz de cada punto separadas por comas.
- **XYZRGB**: Formato de texto plano en que cada línea representa las coordenadas x, y, z de cada punto, seguido de los colores RGB representados en el rango [0, 1].
- **PTS:** Formato de texto plano para la representación de nubes de puntos. Tratado en detalle en la sección 2.1.
- **PLY:** Formato orientado a la representación de geometrías 3d complejas. Tratado en detalle en la sección 2.2.
- **PCD:** Formato orientado a la representación de nubes de puntos complejas. Tratado en detalle en la sección 2.3.

En el caso de las geometrías 3D de mallas triangulares, Open3D permite exportar e importar ficheros en los formatos *ply*, *stl* y *obj,* entre otros.

Con respecto a las imágenes RGBD, Open3D ofrece funcionalidad para su creación a partir de dos imágenes independientes que representen el color y la profundidad, registradas por la misma cámara con la misma resolución. Una vez generada la imagen RGBD, es posible transformarla en una nube de puntos tradicional a partir de una configuración de cámara predefinida, con una función de utilidad de Open3D.

**Motor de renderizado**

Como *backend* dedicado al renderizado de escenas 3D en tiempo real, Open3D hace uso de Filament, el motor de renderizado basado en físicas (*Phisically Based Rendering*, *PBR*) desarrollado por Google. Open3D implementa un módulo, denominado *rendering,* que permite acceder a características de renderizado de Filament, lo que posibilita la creación de aplicaciones con características de visualización avanzada. Filament es compatible con Android, WebGL, Windows, macOS y Linux e implementa múltiples funcionalidades de renderizado realista de escenas.

*Ilustración 12: Ejemplo Filament*

Los métodos de renderizado basado en físicas (*PBR*) permiten una representación realista de los materiales de una geometría y la manera en que estos interactúan con la luz de la escena 3D. La Ilustración 12 muestra un ejemplo de las posibilidades del motor Filament en el renderizado de escenas complejas, extraído de la documentación de Filament.

Open3D ofrece una interfaz de renderizado en la que es posible interactuar con el modelo de materiales de Filament, lo que permite modificar en tiempo real las propiedades físicas del material asociado a una geometría dada o las características de iluminación de la escena 3D.

En versiones recientes de la biblioteca, Open3D incluye una herramienta de visualización propia, denominada O3DViewer, que hace uso del motor de renderizado de Filament e incorpora parte de la funcionalidad de la biblioteca, como soporte para animaciones de cámara, importado de geometrías y selección de elementos de una geometría. Esta herramienta se encuentra actualmente en estado de *beta* y aunque portable, sólo está disponible en C++, no estando toda su funcionalidad disponible en Python.

La versión más reciente de la biblioteca Open3D (0.13, Junio 2021) incluye un modo experimental de renderizado web, lo que facilita la visualización de geometrías en navegadores web, realizando las tareas de procesamiento de geometrías en un servidor remoto.

**Aprendizaje automático**

Open3D también ha incorporado recientemente (Octubre 2020, versión 0.11) un módulo de aprendizaje automático aplicado a geometrías 3D, denominado *Open3D-ML,* con soporte para las bibliotecas Pytorch y Tensorflow. Este módulo incluye un conjunto de algoritmos, modelos y

operadores 3D que facilitan la creación de aplicaciones que permiten aplicar redes neuronales a problemas relacionados con geometrías 3D.

En particular, este módulo soporta flujos de segmentación semántica y de detección de objetos en entornos 3D, lo que permite identificar objetos en una escena 3D en tiempo real. Otros casos de uso complejos soportados por este módulo serían los siguientes:

- **Reconstrucción de entornos 3D en tiempo real**: Permite realizar una reconstrucción 3D de una escena a partir de imágenes RGBD tomadas en tiempo real. Cuenta con soporte para GPU y CPU.
- **Alineado de nubes de puntos en tiempo real (*Point cloud registration*)**: Implementación del algoritmo ICP (*Iterative Closest Point*) para la alineación de nubes de puntos en tiempo real, con soporte para CPU y GPU. Estos algoritmos permiten, por ejemplo, unificar las medidas tomadas por escáneres LIDAR móviles.

El módulo Open3D-ML está acompañado de una herramienta de visualización desarrollada mediante el módulo de interfaz de Open3D que permite inspeccionar los conjuntos de datos y las predicciones realizadas por los modelos de aprendizaje automático. La Ilustración 13 muestra un ejemplo de la interfaz del módulo de aprendizaje automático, extraída de la documentación de la biblioteca.

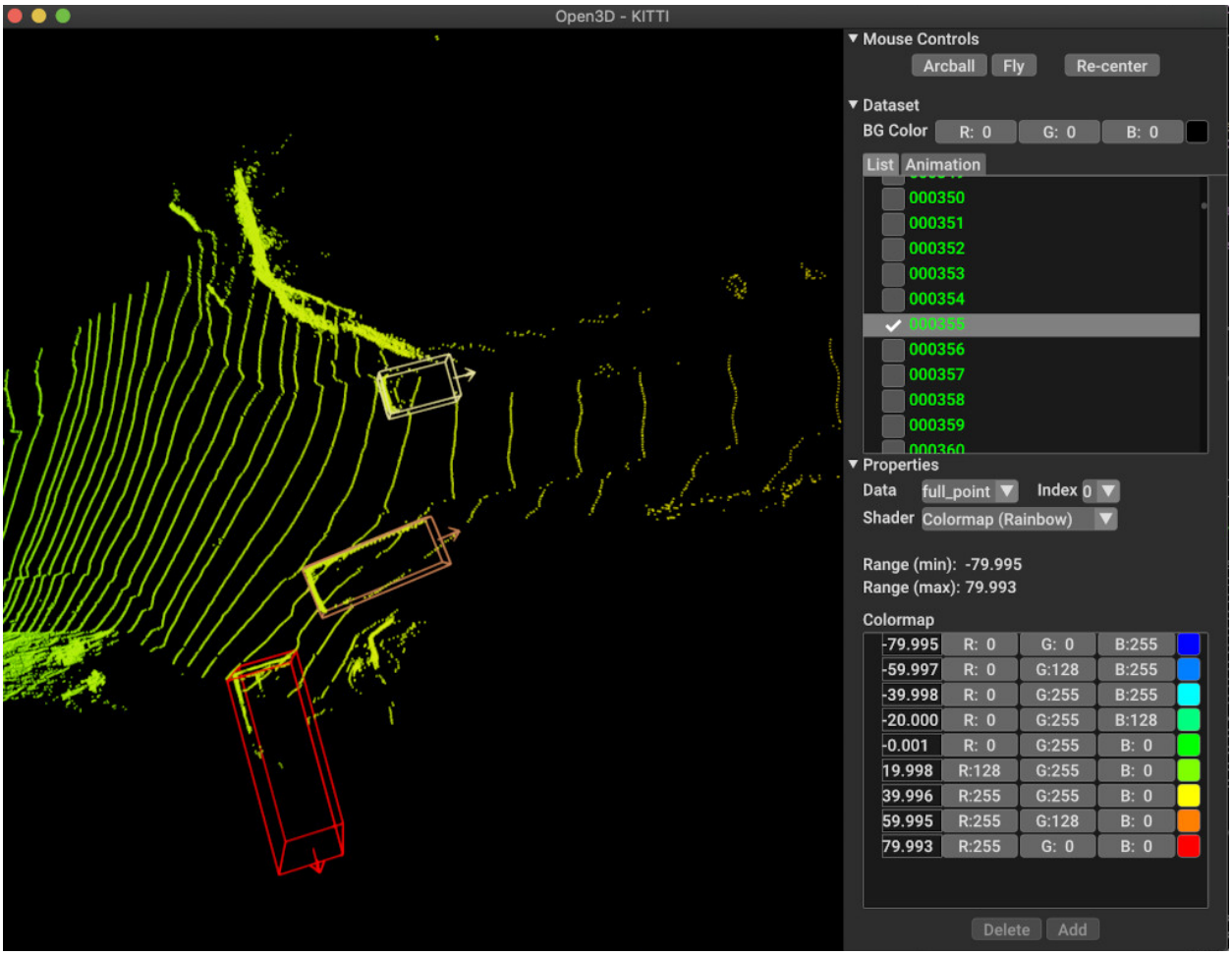



*Ilustración 13: Open3D-ML*

**Operaciones de geometrías**

Open3D implementa un conjunto de operaciones de transformación y procesamiento básicas para cada una de las geometrías soportadas por la biblioteca.

La siguiente lista, no exhaustiva, menciona algunas de las transformaciones soportadas en estructuras de nubes de puntos:

- **Filtrado**: Incluye técnicas que permiten reducir la densidad de las nubes de puntos, como técnica de pre-procesamiento antes de aplicar transformaciones más complejas.
- **Estimación de normales**: Permite estimar las normales de cada punto de una nube a partir de un conjunto de puntos cercanos.
- **Cortar geometría**: Permite seleccionar subconjuntos de una nube de puntos a partir de un polígono predefinido.
- **Coloreado**: Permite modificar los colores de un subconjunto de puntos de la nube.
- **Cálculo de distancias entre nubes de puntos**.
- **Clustering**: Permite agrupar conjuntos de puntos en base a sus coordenadas y cercanía con otros puntos cercanos.
- **Segmentación de planos:** Permite determinar de manera iterativa el plano con mayor soporte en una geometría. En la práctica este es el plano con mayor cantidad de puntos de la geometría a una distancia reducida del mismo.
- **Eliminación de puntos ocultos:** Permite estimar la visibilidad de los puntos de una nube a partir de una perspectiva de cámara. Esto permite eliminar los puntos de la parte posterior de una nube que ofuscan la visibilidad de la geometría.

A continuación, se presenta alguna de las transformaciones soportadas en el manejo de mallas triangulares:

- **Cortar geometría**: Permite seleccionar un subconjunto de la malla.
- **Coloreado**: Permite modificar el color de los polígonos de la malla.
- **Suavizado**: Permite suavizar las superficies de mallas triangulares mediante diversas técnicas de filtrado. Estos métodos permiten eliminar el ruido y complejidad de la malla.
- **Muestreo**: Permite realizar un muestreo de una malla triangular, generando una nube de puntos a partir de un subconjunto de los vértices que la conforman.
- **Subdivisión:** Permite aumentar la complejidad de una malla triangular dividiendo cada triángulo en múltiples triángulos. De esta manera se mantiene la superficie y área de la geometría, aumentando el número de vértices y triángulos.
- **Simplificación:** Incluye varias técnicas para simplificar una malla triangular, reduciendo el número de vértices y triángulos de esta conservando la estructura inicial dentro de lo posible. Las siguientes imágenes extraídas de la documentación de Open3D presentan un ejemplo de una malla transformada con una operación de simplificación.

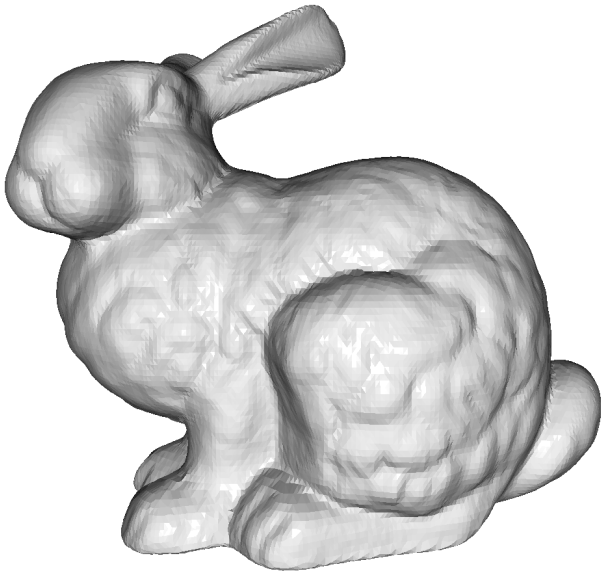


*Ilustración 14: Malla previa al simplificado*

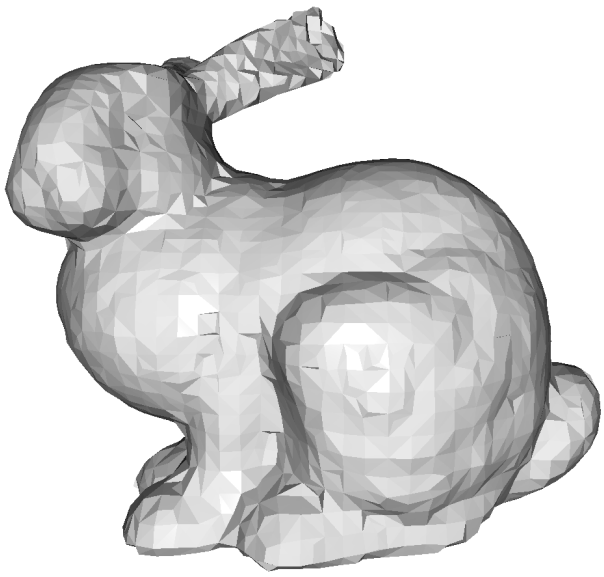


*Ilustración 15: Malla tras el simplificado*

**Módulo de interfaz**

En su actualización a la versión 0.10 en Mayo de 2020, los creadores de Open3D introdujeron en la biblioteca un nuevo módulo para la creación de interfaces gráficas. Este módulo facilita la creación de aplicaciones de escritorio a medida para la visualización y procesamiento de geometrías 3D.

Este módulo de interfaz, denominado simplemente *gui*, está basado en el proyecto *Dear ImGUI*, principalmente debido a su tamaño compacto y reducido conjunto de dependencias. *ImGui* es una biblioteca implementada en C++ para el desarrollo de interfaces gráficas de usuario. Es una biblioteca auto contenida, portable y que optimiza el rendimiento. Está orientada al desarrollo de herramientas de depuración o visualización de bajo nivel, aunque también puede ser utilizada para construir interfaces de usuario simples. Como ejemplos comunes de uso de *ImGui*, se destacan herramientas de motores de renderizado de videojuegos o aplicaciones 3D en tiempo real.

El módulo GUI de Open3Dfue implementado en C++, al igual que el resto de la biblioteca, y ofrece *bindings* en Python que simplifican su utilización.

Como ejemplo de las posibilidades que introduce un módulo de este tipo, Open3D ofrece una aplicación de escritorio dedicada, implementada en C++, denominada *Open3D Visualizer.* Esta aplicación permite configurar las propiedades de visualización básicas del motor de renderizado Filament.

A continuación, se mencionan los elementos de interfaz que es posible crear mediante el módulo de interfaz *gui* de Open3D. Estos son un subconjunto de los elementos principales proporcionados por la biblioteca base *ImGui,* junto una serie de elementos adicionales introducidos por Open3D. Cada elemento será identificado por la clase que los representa dentro del modulo *gui*:

- **Application**: Representa la aplicación gráfica en su totalidad. Contiene el menú de la aplicación, las ventanas creadas y todos los elementos de interfaz.
- **Button**: Botones simples interactivos.
- **Combobox**: Lista desplegable en la que es posible seleccionar un único elemento.
- **Checkbox**: Casilla de verificación.
- **CollapsableVert:** Elemento de interfaz desplegable que permite agrupar otros elementos de interfaz. Tiene una etiqueta asociada que representa el grupo de elementos.
- **ColorEdit:** Selector de color.
- **Dialog**: Ventana de diálogo adicional (*pop up*) a la ventana principal. Permite mostrar mensajes al usuario.
- **FileDialog**: Ventana de diálogo que permite seleccionar un fichero del equipo del usuario, mostrando un árbol de directorios.
- **Horiz**: Permite hacer una distribución de otros elementos de interfaz de manera horizontal. Como ejemplo, considerar el menú superior de una aplicación.
- **Vert**: Permite hacer una distribución de otros elementos de interfaz de manera vertical. Como ejemplo, considerar un menú que ocupe todo el lateral de una aplicación.
- **ImageLabel**: Representa una imagen 2D.
- **Label**: Representa una etiqueta de texto en 2D.
- **Label3D**: Permite representar texto en una escena 3D.

- **Menu**: Permite crear menús con un conjunto de submenús desplegables con una serie de elementos interactivos.
- **NumberEdit**: Campo de texto que permite al usuario introducir un valor numérico.
- **TextEdit**: Campo de texto que permite al usuario introducir una cadena de texto.
- **ProgressBar**: Barra de progreso.
- **SceneWidget**: Permite representar contenido 3D en una escena principal.
- **Slider**: Permite seleccionar un valor numérico.
- **TabControl**: Permite agrupar otros elementos de interfaz en pestañas seleccionables.

Todos los elementos de interfaz previos permiten asociar funciones predefinidas a eventos de interacción con los mismos. Por ejemplo, la función a ejecutar cuando un usuario hace click en un botón o cuando se cambia el elemento seleccionado en una lista desplegables. Este módulo también facilita el mapeo de eventos de usuario a comportamientos, como por ejemplo la asignación de funciones a combinaciones de teclas de teclado.

La Ilustración 16 muestra un ejemplo de interfaz que es posible crear mediante el módulo *gui*, extraído de los ejemplos de la biblioteca. Esta interfaz muestra gran parte de los componentes mencionados en la lista previa.


*Ilustración 16: Ejemplo elementos interfaz Open3D*

## 3.2 PyVista

PyVista es una biblioteca implementada en Python que simplifica la utilización de VTK, el toolkit de visualización 3D desarrollado por Kitware para la visualización y procesamiento de datos científicos. VTK (*Visualization Toolkit*) es una plataforma de procesamiento y renderizado de gráficos 3D compleja y potente que lleva en desarrollo desde 1993. Siendo VTK una plataforma de tan largo recorrido, presenta ciertas dificultades e ineficiencias que bibliotecas como PyVista intentan solventar.

El propio framework de VTK cuenta con *bindings* dedicados en Python, lo que permite mantener el rendimiento nativo de C++ junto a las características dinámicas y compatibilidad con el ecosistema de bibliotecas científicas de Python. No obstante, su utilización es compleja puesto que es un *wrapper* directo a las funciones de la API en C++ que no se adapta a las características dinámicas de Python. PyVista pretende simplificar el proceso de creación de mallas y de visualización de geometrías, sin perder características funcionales del framework VTK.

Siendo PyVista una biblioteca que facilita la utilización de VTK, cuenta con las dependencias complejas de la misma. Como biblioteca de código abierto cuenta con un destacable apoyo de la comunidad (664 menciones) y presenta múltiples contribuciones recientes.

### 3.2.1 Características principales

PyVista permite representar tanto nubes de puntos como mallas poligonales en una estructura propia denominada malla (*mesh*). Esta estructura de malla representa la geometría de una superficie determinada en un entorno 3D. PyVista no diferencia entre mallas 2D, como por ejemplo una malla triangular, y mallas 3D, como las mallas formadas por vóxeles, lo que resulta transparente para el usuario.

PyVista define un conjunto de estructuras adicionales asociadas a una malla:

- **Nodos** (*Nodes*): Los nodos representan los vértices de una malla. Definen las coordenadas x, y, z de la geometría.
- **Células** (*Cells*): Una célula está definida por un conjunto de nodos interconectados, formando una geometría determinada. Por ejemplo, las células en una malla triangular estarían formadas por conjuntos de 3 nodos interconectados por líneas, lo que formaría un triangulo en 2 dimensiones. En el caso de una malla formada por vóxeles, las células representarían cubos entre 8 nodos de la malla.
- **Atributos** (*Attributes*): Los atributos son valores que pueden ser asociados nodos o células concretas de una malla.

Para representar una nube de puntos mediante PyVista, se definiría una malla en la que sólo se definen nodos, con células de una dimensión que representan al propio nodo. La Ilustración 17 muestra un ejemplo de una malla de células triangulares representada mediante PyVista.


*Ilustración 17: Malla triangular PyVista*

A continuación, se menciona brevemente algunos de los procesos de transformación disponibles en PyVista. Estos proporcionan acceso directo a la extensa colección de filtros de transformación implementados en VTK:

- Cortado y transformación de mallas.
- Cálculo de características volumétricas y de área.
- Simplificación de mallas mediante división de células.
- Proyección de una malla a un plano.
- Cálculo de distancias entre mallas.
- Suavizado de superficies.
- Subdivisión de células triangulares.

Para una referencia completa de los filtros proporcionados en PyVista, consultar la documentación de su API (https://docs.pyvista.org/core/filters.html).

También cuenta con las siguientes características de visualización y renderizado para la representación de mallas:

- Asignación de gradientes de color a valores escalares de la malla.
- Visualización de ejes de la malla.

- Renderizado de imágenes de profundidad.
- Etiquetado de puntos o nodos de una malla.
- Animaciones de cámara alrededor de una malla.
- Control de iluminación de la escena 3D con un conjunto de perfiles predeterminados. La Ilustración 18 muestra un ejemplo de superficie representada con características de iluminación habilitadas.

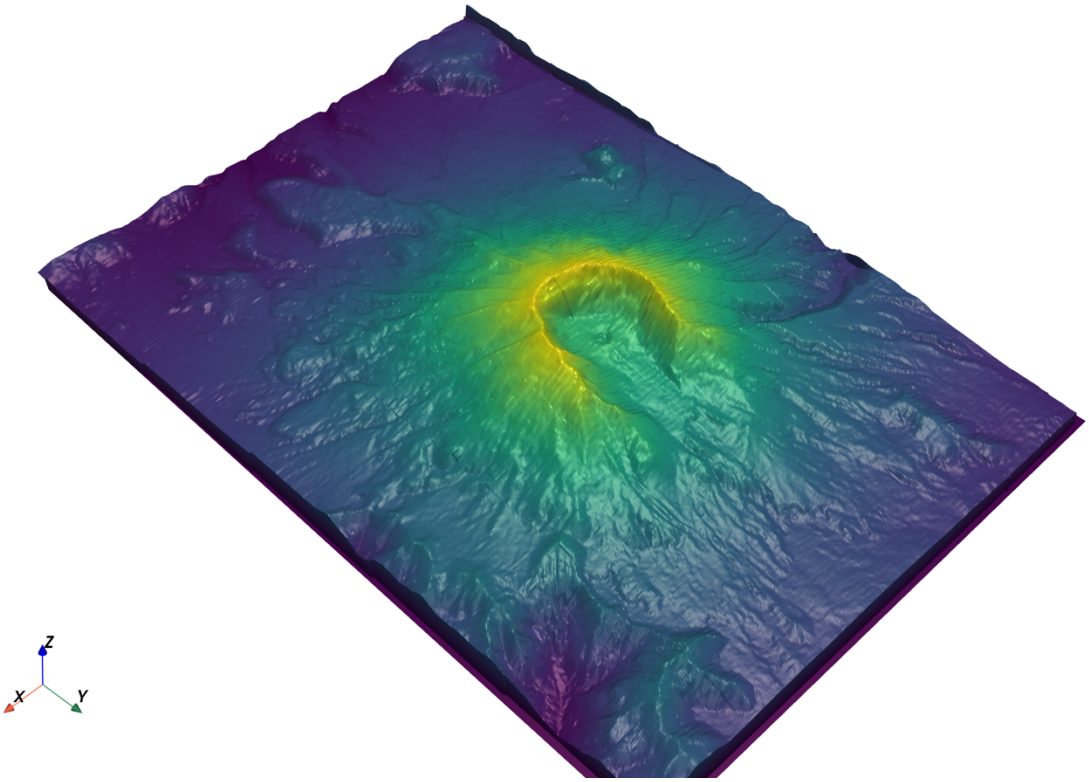



*Ilustración 18: Superficie iluminada PyVista*

## 3.3 pyntcloud

Pyntcloud es una biblioteca implementada en Python para la exploración interactiva y el procesamiento básico de nubes de puntos, con énfasis en la integración con el ecosistema de análisis de datos de Python, como por ejemplo las bibliotecas *pandas* o *NumPy*.

Implementa operaciones básicas como la lectura de ficheros de nubes de puntos en múltiples formatos, la adición de nuevos campos escalares a puntos de una nube o el filtrado de nubes de puntos. Adicionalmente, posee integración con otras bibliotecas de procesamiento 3D más robustas como Open3D o PyVista.

Como biblioteca de código abierto, cuenta con 884 menciones en Github. Es desarrollada principalmente por una única persona y no parece especialmente activa en la actualidad (última contribución en Noviembre 2020).

### 3.3.1 Características principales

La biblioteca permite importar estructuras tanto de nubes de puntos como de mallas triangulares, pero sus operaciones de transformación operan únicamente en nubes de puntos. Por otra parte, Pyntcloud no proporciona un módulo de renderizado propio, siendo necesario realizar la visualización a partir de bibliotecas externas, como Open3D, PyVista o Three.js.

De entre los formatos de geometrías soportados para el importado de geometrías, destacan los siguientes:

- LAS
- OBJ
- PCD
- PLY
- ASCII: Soporta múltiples formatos de texto plano, como PTS, XYZ o CSV.

A continuación, se mencionan algunas de las operaciones de transformación y visualización implementadas por la biblioteca:

- **Asignar valores escalares**: Permite asociar un conjunto de valores arbitrario al conjunto de puntos que conforman una nube.
- **Mapeo de colores**: Permite modificar los colores de una nube de puntos a partir de los valores de un campo escalar predeterminado asociado a un punto. Por ejemplo, permite asignar un gradiente de color a valores crecientes de un determinado eje.
- **Conversión de mallas:** Transformación de una malla triangular en nube de puntos a partir de un muestreo aleatorio.
- **Cálculo de normales:** Estimación de las normales de cada punto de una nube a partir de los puntos más cercanos a los mismos.
- **Voxelización:** Ofrece múltiples técnicas para la voxelización de una nube de puntos. Este proceso genera una serie de polígonos que contienen un conjunto de puntos de la nube, formando una superficie irregular a partir de la nube de puntos original.
- **Representación de líneas:** Permite crear líneas de múltiples segmentos en un entorno 3D. Estas líneas están definidas por una serie de puntos 3D que las conforman, junto al color con el que serán representadas.

## 3.4 PCL

Point Cloud Library (PCL) es una biblioteca implementada en C++ para el procesamiento de nubes de puntos y geometrías 3D. Lanzada inicialmente en 2010, es una de las bibliotecas más utilizadas para todo tipo de flujos de procesamiento de conjuntos de datos 3D, con funcionalidad diversa para la segmentación, filtrado y visualización de nubes de puntos.

Por contrapartida, su extensa funcionalidad dificulta su utilización, y cuenta con un conjunto de dependencias complejo. Como biblioteca de código abierto, cuenta con gran apoyo de la comunidad, con 5900 menciones en Github, 400 contribuidores y múltiples actualizaciones recientes.

*Ilustración 19: Logo PCL*

### 3.4.1 Características principales

A continuación, se menciona un conjunto de las características principales de la biblioteca PCL. Debido a la extensión de la funcionalidad ofrecida, se mencionarán las más relevantes para los objetivos de este proyecto. Para un análisis exhaustivo, referirse a la API de referencia y documentación de PCL.

**Filtros**

El módulo de filtros de PCL incluye múltiples técnicas para la eliminación de valores anómalos y ruido en nubes de puntos. La mayoría de estos valores anómalos pueden ser identificados y filtrados realizando un análisis estadístico de las distancias entre puntos de la nube con respecto a otros puntos cercanos. Algunos de los filtros incluidos son los siguientes:

- **Filtro *PassThrough*:** Filtro simple que elimina todos aquellos puntos cuyas coordenadas están dentro o fuera de un rango definido por el usuario en un eje dado.
- **Filtro estadístico**: Elimina todos los puntos de la nube cuya distancia mínima a un conjunto de puntos cercana es mayor a la media de distancia entre puntos de la nube.
- **Filtro por radio**: Elimina todos los puntos de una nube que no tienen un cierto número de puntos cercanos en un radio determinado.
- **Downsampling**: Permite simplificar una nube de puntos a partir de un grid de vóxeles. Esta técnica construye un grid de cubos de un determinado tamaño. Los puntos de la nube dentro de cada cubo son simplificados como un único punto resultante de la media de coordenadas de los puntos.

**Propiedades**

En la biblioteca PCL, las propiedades (*features*) de una nube de puntos son una serie de estimaciones y cálculos realizados sobre un punto de la nube a partir de sus elementos cercanos. Las propiedades más comunes de un punto son su normal y curvatura, calculadas a partir de sus puntos vecinos. PCL introduce algunas propiedades más avanzadas orientadas a optimizar otros algoritmos como la detección de objetos o estimación de poses, implementados en otros módulos de la biblioteca.

**Puntos de interés**

El módulo de puntos de interés (*keypoints*) de PCL implementa un conjunto de algoritmos para la detección de puntos de interés en una nube de puntos o imagen dada. Los puntos de interés son un subconjunto muy reducido de la nube de puntos original y representan los aspectos más distintivos de la nube. De entre los métodos implementados, destaca el algoritmo NARF (*Normal Aligned Radial Feature*). Este método determina puntos de interés de una geometría detectando cambios bruscos en la geometría, además de tener en cuenta los bordes de cada objeto.

**Alineado**

Los mecanismos de alineado o registro de nubes de puntos (*Geometric registration*) permiten combinar múltiples nubes de puntos tomadas desde diferentes perspectivas en una nube de puntos global consistente que representa una escena completa. El principio básico de funcionamiento de estos algoritmos es el de identificar los puntos comunes de unión entre las nubes de puntos y buscar la transformación a aplicar en una de las nubes para minimizar la distancia entre los puntos de ambas. El proceso de alineado es repetido un numero determinado de veces con diferentes configuraciones de rotación y traslación buscando minimizar el error de alineado.

PCL implementa múltiples algoritmos de alineado o registro de nubes de puntos, de entre los que se mencionan los siguientes:

- **Iterative Closest Point** (ICP): Compara dos nubes de puntos, determinando si ambas representan la misma perspectiva calculando la transformación necesaria para alinearlas, tratando de minimizar la distancia entre puntos de la nube.
- **Normal Distributions Transform**: Calcula la transformación necesaria para alinear nubes de puntos de gran tamaño aplicando un algoritmo de optimización.

La Ilustración 20 y la Ilustración 21 muestran como es posible alinear un conjunto de nubes de puntos tomadas desde perspectivas independientes formando una nube de puntos uniforme que representa la totalidad de la escena.

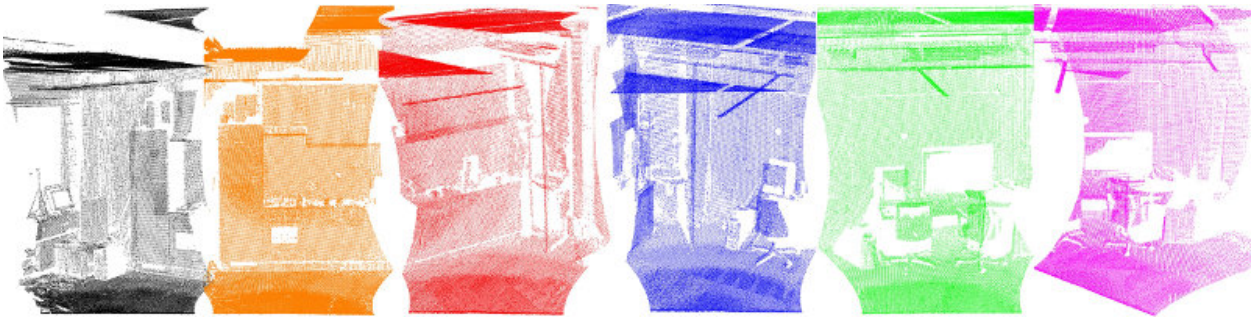


*Ilustración 20: Nubes de puntos independientes*

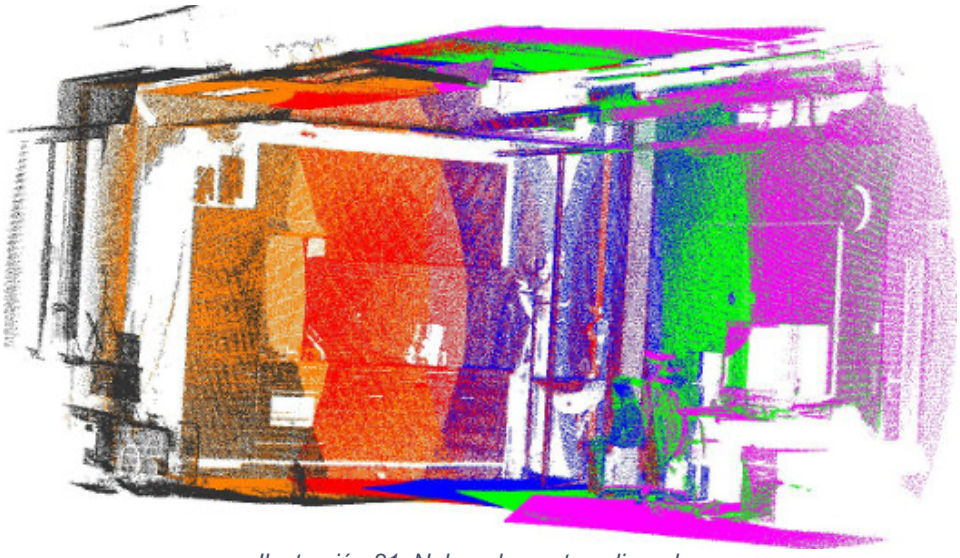


*Ilustración 21: Nubes de puntos alineadas*

**Segmentación**

PCL incluye un módulo de segmentación que incluye múltiples mecanismos para agrupar puntos de una nube siguiendo un conjunto de estrategias. Esto permite dividir una nube de puntos en sus componentes principales. A continuación, se menciona alguno de los métodos de segmentación incluidos:

- **Segmentación de planos**: Determina los puntos de una nube que dan soporte a un plano determinado. En la práctica, esto permite seleccionar los puntos de la nube que se encuentran en la proximidad de un plano.
- **Segmentación de cilindros:** Proceso similar al de segmentación de planos, aplicado en este caso a geometrías cilíndricas.
- **Segmentación de región creciente** (*Region growing segmentation*)**:** Permite agrupar puntos de una nube a partir de la supuesta uniformidad de la superficie que representan. Este método permite combinar conjuntos de puntos de una nube partiendo de los ángulos formados por las normales de los puntos evaluados.
- **Segmentación por color:** Es un proceso similar al de región creciente en que se combinan regiones de una nube con colores similares.

PCL incluye a su vez una serie de algoritmos basados en RANSAC (*Random Sample Consensus*) para identificar, de manera iterativa, múltiples geometrías comunes en una nube de puntos dada. Entre las geometrías soportadas se incluyen planos, cilindros, líneas y esferas. La Ilustración 22 muestra un ejemplo de como es posible automatizar la detección de distintos tipos de geometrías aplicando técnicas RANSAC.

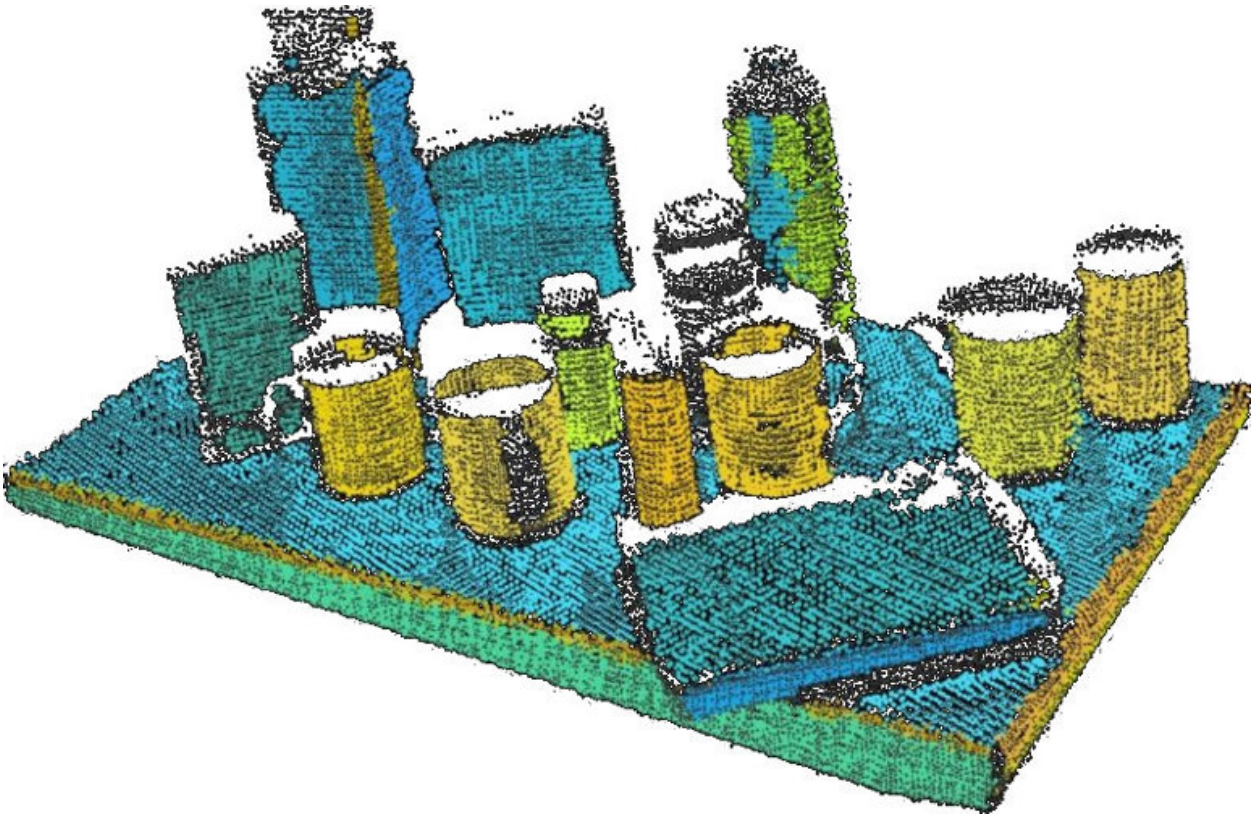


*Ilustración 22: PCL RANSAC*

**Reconstrucción de superficies**

PCL incluye un módulo para la reconstrucción de superficies de una nube de puntos. Posibilita la creación de mallas detalladas o la simplificación de una nube de puntos a partir de la mínima geometría convexa que contiene todos los puntos (*hull*). Incluye, entre otros, las siguientes funcionalidades:

- **Suavizado de nube de puntos**: Permite homogeneizar pequeñas diferencias en las coordenadas de una nube de puntos generadas por errores de medición.
- **Geometrías poligonales**: Construye una malla triangular que contiene en su interior todos los puntos de la nube. Sirve para obtener una superficie simplificada de una nube de puntos y para obtener los márgenes o límites de esta.
- **Superficie por triangulación**: Construye una malla a partir de una nube de puntos en un proceso iterativo que construye triángulos entre puntos cercanos de una nube. Es el proceso de reconstrucción más rápido con el que cuenta la biblioteca.

**Visualización**

PCL implementa un módulo de visualización destinado a la representación de resultados de algoritmos de procesamiento de nubes de puntos. Toda la funcionalidad de visualización hace uso del framework de renderizado VTK. Incluye, entre otras, las siguientes opciones de visualización:

- **Propiedades visualización**: Permite modificar múltiples propiedades de renderizado de cada punto de la nube, como el tamaño de puntos, opacidad o color.
- **Formas geométricas**: Permite representar formas geométricas simples en una escena 3D, como cilindros, esferas o polígonos.
- **Gráficas 2D**: Permite representar histogramas 2D asociados a un valor escalar de la nube de puntos.
- **Imágenes RGBD**: Visualización de imágenes de profundidad RGBD

## 3.5 Cilantro

Cilantro es una biblioteca de código abierto implementada en C++ para el procesamiento de nubes de puntos y geometrías 3D. Ofrece un gran rendimiento en la aplicación de transformaciones de bajo nivel a nubes de puntos, además de implementar funcionalidad avanzada para la segmentación, clustering o alineación de nubes de puntos.

Incluye módulos de visualización de estructuras 3D utilizando la biblioteca Pangolin, dedicada al renderizado 3D con OpenGL como motor de renderizado. Como biblioteca de código abierto no parece especialmente activa en la actualidad, con 9 contribuyentes, aunque cuenta con un gran número de menciones en Github (566)

### 3.5.1 Características principales

A continuación, se resumen las características principales proporcionadas por la biblioteca, recogidas de su documentación oficial:

- Estimación de normales y curvatura de una nube de puntos.
- Construcción de nubes de puntos a partir de imágenes de profundidad RGBD.
- Lectura y escritura de nubes de puntos en formato PLY.
- Múltiples técnicas de *clustering* avanzado.
- Alineado de nubes: Presenta varios algoritmos de registro geométrico de nubes de puntos a partir de técnicas iterativas (*Geometric registration*). Estas técnicas permiten reconstruir escenas 3D a partir de cámaras RGBD móviles, aplicando un proceso de alineación de las nubes de puntos capturadas.
- Cálculo de la mínima geometría que contiene a todos los puntos de la nube.
- Análisis de componentes principales.

La Ilustración 23 muestra un ejemplo de un flujo de registro geométrico (*Geometric registration*) en el que se reconstruye una nube de puntos de una escena.

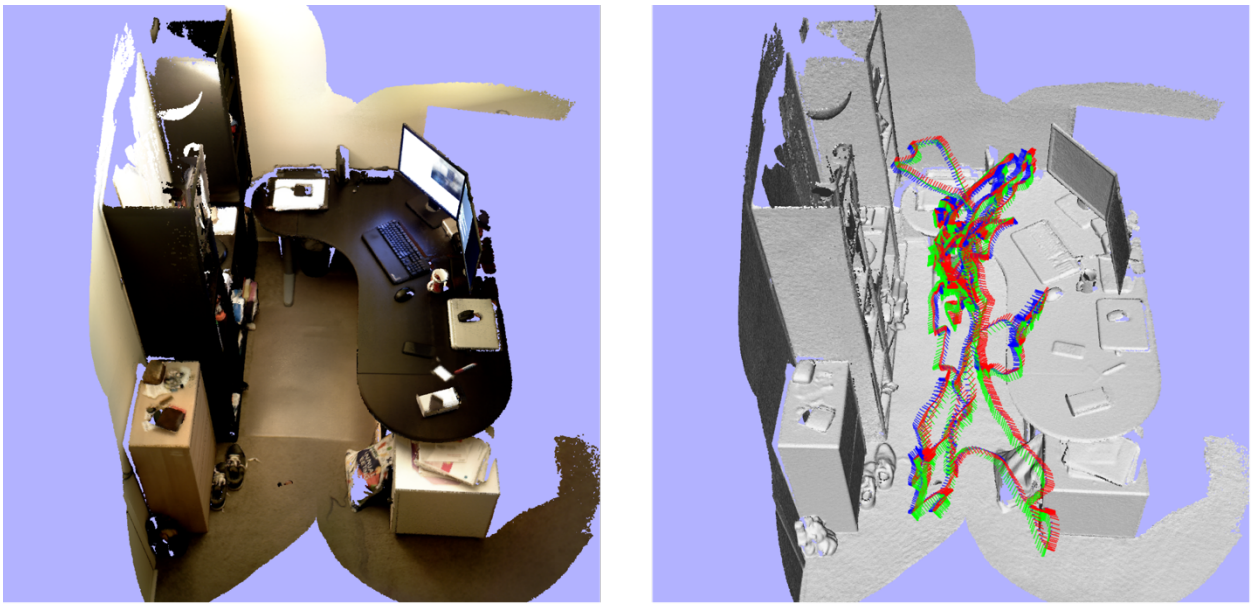


*Ilustración 23: Cilantro alineado*

## 3.6 Evaluación bibliotecas

Una vez realizado un análisis funcional de las características de cada biblioteca, se procede a determinar cual de ellas es la más adecuada para el desarrollo de una herramienta dedicada al procesamiento y visualización de geometrías 3D. Esta herramienta será presentada como una aplicación de escritorio con interfaz gráfica de usuario e implementará un conjunto de funciones básicas de transformación de nubes de puntos, lo que permite acotar la selección de bibliotecas.

Por lo tanto, se decide evaluar las bibliotecas seleccionadas teniendo en cuenta a las siguientes consideraciones:

- **Rendimiento**: Se valorará la fluidez de cada biblioteca en el manejo y procesamiento de geometrías 3D, así como los tiempos de carga.
- **Integración**: La biblioteca debe facilitar la integración con otras bibliotecas del ámbito científico, para simplificar la implementación de funciones de transformación de geometrías no incluidas en la biblioteca.
- **Renderizado**: La biblioteca debe proporcionar acceso a un motor de renderizado dedicado a la visualización de geometrías o facilitar integración con otras bibliotecas que lo permitan.
- **Interfaz gráfica:** Se valorará que la biblioteca incorpore módulos para la creación de interfaces gráficas de usuario. Si no fuera el caso, sería necesario implementar la interfaz gráfica mediante un framework dedicado como Qt.
- **Actividad**: Se consultará la actividad de cada biblioteca de código abierto, así como la frecuencia de actualización de esta y su estado actual.
- **Operaciones**: Se evaluará en cada caso las funciones de transformación de geometrías proporcionadas por cada biblioteca, lo que simplificará su inclusión en la herramienta desarrollada.

### 3.6.1 Consideraciones iniciales

A partir de los aspectos mencionados y las características básicas esperadas de la herramienta a implementar, se realizará un primer filtrado de bibliotecas. Posteriormente, se realizará un

análisis más detallado de los aspectos que no pueden ser valorados directamente, como el rendimiento en la carga y manejo de geometrías de cada biblioteca.

**Motor de renderizado**

En este apartado se valora la funcionalidad proporcionada por cada biblioteca para el renderizado de nubes de puntos y estructuras 3D, o la posibilidad de su integración con bibliotecas externas que lo permitan.

La siguiente lista muestra las opciones de renderizado de cada una de las bibliotecas evaluadas:

- **Pyntcloud**: No proporciona integración con ningún motor de renderizado, siendo necesario relegar la visualización de geometrías a bibliotecas externas. La documentación de pyntcloud menciona múltiples *backends* disponibles para la realización de esta tarea, como por ejemplo PyVista, Open3D o Three.js.
- **PyVista**: Siendo PyVista una biblioteca que facilita acceso al framework de visualización VTK, hace uso de su motor de renderizado basado en OpenGL.
- **Open3D**: La API de renderizado implementada en Open3D ofrece acceso al motor de renderizado Filament.
- **Cilantro**: No implementa módulos de renderizado dedicados, aunque facilita la integración con la biblioteca Pangolin que usa OpenGL como backend.
- **PCL**: La biblioteca PCL utiliza el framework de visualización VTK y su motor de renderizado basado en OpenGL.

Como se puede comprobar, todas las bibliotecas ofrecen opciones para el renderizado de geometrías. Tanto PyVista como PCL se integran con el framework de visualización VTK para implementar sus opciones de renderizado, mientras que Open3D hace uso de Filament.

**Ecosistema bibliotecas**

En este apartado se valora la capacidad de integración de cada biblioteca con otras bibliotecas de ámbito científico. Esto será de especial relevancia si en la elaboración de la herramienta de visualización es necesario implementar cierto tipo de procesamiento que no sea soportado por la propia biblioteca. Este aspecto queda determinado en gran medida por el lenguaje en que está implementada cada biblioteca.

La mayoría de las bibliotecas mencionadas anteriormente presentan una interfaz en Python a un backend implementado en C++. Esto proporciona la eficiencia y optimización en el procesamiento de nubes de puntos aportada por C++ junto con el acceso al ecosistema científico asociado a Python. Entre otras, se tendría acceso a NumPy, alternativa de utilidad en caso de requerir implementar nuevas rutinas de procesamiento, puesto que cuenta con un backend C++ que facilitaría el procesamiento de las estructuras de nubes de puntos de manera eficiente. Adicionalmente, alguna de las bibliotecas mencionadas, como Open3D, posibilitan la conversión inmediata de sus estructuras dedicadas a arrays multidimensionales de NumPy.

Por ello, se favorecerá la selección de aquellas bibliotecas implementadas en C++ que presenten una interfaz en Python, que en este caso son PyVista, Open3D y pyntcloud.

**Interfaz gráfica de usuario**

En el presente apartado se valorarán las alternativas existentes para el desarrollo de aplicaciones de escritorio que ofrezcan acceso a la biblioteca de procesamiento 3D mediante interacción con elementos de interfaz gráfica de usuario.

De entre todas las bibliotecas evaluadas, solo Open3D proporciona un módulo dedicado a la creación de interfaces gráficas de usuario. Este módulo, evaluado en detalle en la sección 3.1.1, permite configurar elementos gráficos como menús, desplegables y botones interactivos, mediante una interfaz simplificada a la biblioteca de creación de interfaces gráficas de usuario *ImGui*.

Este módulo, aunque presente funcionalidad básica, permite desarrollar aplicaciones de escritorio completas desde la propia biblioteca, ofreciendo una interfaz gráfica de usuario a funciones de procesamiento de geometrías 3D.

En el caso del resto de bibliotecas, sería posible desarrollar una aplicación de escritorio integrando la biblioteca con algún framework dedicado para la creación de interfaces gráficas de usuario, como PyQt. Este proceso complicaría la implementación, además de aumentar la complejidad de las dependencias de la herramienta.

**Operaciones en geometrías 3D**

El hecho de que la biblioteca seleccionada presente una oferta diversa de operaciones de transformación y manejo de geometrías facilitará su inclusión en la herramienta desarrollada. Si la biblioteca no incluyera una cierta transformación requerida en la herramienta, sería necesario implementarla o dar acceso a la misma a través de otra biblioteca incluida como dependencia.

En todo caso, las operaciones de transformación implementadas por cada biblioteca serán tomadas como referencia a la hora de definir los requisitos finales de la herramienta desarrollada.

En los apartados dedicados a las características funcionales de cada biblioteca presentados en la sección 3, se evaluaron las principales funciones de procesamiento y transformaciones de geometrías 3D proporcionadas por cada una de ellas. De ellas se extraen las siguientes consideraciones:

- **PCL**: Posiblemente sea la biblioteca con más opciones para el procesamiento de nubes de puntos, soportando múltiples flujos de transformación de geometrías 3D, tanto básicos como avanzados. Sus operaciones se limitan principalmente a las transformaciones en estructuras de nubes de puntos.
- **Open3D**: Implementa un conjunto de operaciones básicas y comunes en flujos de procesamiento de geometrías generales (eliminación de valores anómalos, filtrado…). También implementa algunas operaciones avanzadas como la reconstrucción de superficies de una nube de puntos o la segmentación de planos. Da soporte al procesamiento y visualización tanto de nubes de puntos como de mallas triangulares.
- **PyVista**: El framework de visualización VTK al que proporciona acceso PyVista es uno de los más utilizados en el ámbito científico para la representación y transformación de geometrías 3D. Implementa un conjunto de funciones de transformación diverso, donde destacan principalmente aquellas orientadas a la visualización de geometrías. Incluye transformaciones tanto para nubes de puntos como para mallas poligonales.

- **Cilantro**: Cuenta con un conjunto de transformaciones de nubes de puntos reducido pero complejo, centrado en algoritmos avanzados para el *clustering* y el alineado de nubes de puntos. Todas sus operaciones de transformación se limitan a las estructuras de nubes de puntos.
- **Pyntcloud**: Presenta un conjunto de transformaciones reducido, donde destaca el soporte para múltiples formatos de representación de nubes de puntos.

**Conclusiones**

Los aspectos evaluados anteriormente permiten descartar las siguientes bibliotecas, por las razones mencionadas a continuación:

**PCL**: Está dedicada principalmente al procesamiento de geometrías de nubes de puntos y sólo proporciona interfaz C++. Puesto que presenta un catálogo de transformaciones de geometrías diverso, se tomará como referencia a la hora de implementar la herramienta planteada.

**Cilantro**: Dedicada exclusivamente al procesamiento de nubes de puntos. El conjunto de operaciones que implementa es reducido pero muy especializado, lo que va en contra de los objetivos de la herramienta planteada, que sería soportar flujos de procesamiento básicos de geometrías 3D.

Alternativamente, ambas presentan una interfaz exclusivamente en C++, lo que podría dificultar ciertos aspectos de la implementación, como el desarrollo de interfaces gráficas de usuario.

De este modo, se reducen las bibliotecas a evaluar a las siguientes:

- Open3D
- PyVista
- Pyntcloud

Las bibliotecas seleccionadas son meramente interfaces a bibliotecas C++, por lo que se mantiene la eficiencia en el procesamiento y facilita la integración con otras bibliotecas del ecosistema científico de Python, como pandas o NumPy, sin sacrificar el rendimiento en el procesamiento de geometrías 3D.

### 3.6.2 Rendimiento carga

A continuación, se resume la experiencia de utilización básica de las bibliotecas mencionadas anteriormente. El objetivo de este experimento es determinar la estabilidad de las bibliotecas en la carga y visualización de nubes de puntos de distintos tamaños. Este experimento no pretende ser exhaustivo, teniendo como objetivo determinar si la biblioteca se adapta a las necesidades de la herramienta de visualización que se realizará posteriormente. También servirá en ciertos casos para evaluar la usabilidad de estas, tanto en su instalación como en la utilización a la hora de implementar una tarea de procesamiento simple.

Con el objetivo de evaluar el rendimiento de cada biblioteca en la carga de nubes de puntos de distintos tamaños, se decidió implementar una rutina a medida. Esta rutina se presenta como un script desarrollado en Python, incluido junto al código fuente del proyecto, que mide los tiempos de ejecución de cada una de las bibliotecas evaluadas (PyVista, Open3D y PyNtCloud). Se instrumentalizó la carga de múltiples ficheros de nubes de puntos de distintos tamaños y formatos, generados en el apartado 2.5 dedicado al análisis de diferentes formatos de visualización de nubes de puntos.

Para la realización de las pruebas de rendimiento definidas, se utilizó un equipo con las siguientes características:

- Equipo: MacBook Pro M1 (2020)
- RAM: 16 GB
- Procesador: Chip M1

Los datos presentados en la Tabla 6 se corresponden con los resultados obtenidos al cargar cada uno de los ficheros definidos mediante las bibliotecas evaluadas. Para que los resultados fueran más representativos, cada función se ejecutó múltiples veces, presentando para cada caso la media y la desviación típica de todos los valores obtenidos.

En esta tabla se puede comprobar la gran diferencia en el formato de representación utilizado, siendo perceptiblemente mayor el tiempo de carga de nubes de puntos representadas en texto plano. Destaca el caso de los elevados tiempos de carga de la biblioteca PyNtCloud para el formato PLY de texto. En la práctica esta diferencia se deberá a las rutinas de parseo de texto plano que implementa cada biblioteca. En el caso de los ficheros de mayor número de puntos representados en modo texto, se excluyó la experimentación, debido a los largos tiempos de ejecución de todas las repeticiones, lo que descartaría su utilización de manera definitiva.

| Biblioteca | Número de puntos | Formato | Tiempo de carga medio (s) | Desviación típica (s) | Repeticiones |
|---|---|---|---|---|---|
| Open3D | 10.000 | PLY Binario | 0.00115 | 0.000073 | 100 |
| PyVista | 10.000 | PLY Binario | 0.0028 | 0.0003 | 100 |
| PyNtCloud | 10.000 | PLY Binario | 0.00145 | 0.00067 | 100 |
| Open3D | 10.000 | PLY ASCII | 0.00445 | 0.00013 | 100 |
| PyVista | 10.000 | PLY ASCII | 0.00365 | 0.000047 | 100 |
| PyNtCloud | 10.000 | PLY ASCII | 0.04159 | 0.01003 | 100 |
| Open3D | 1M | PLY Binario | 0.10315 | 0.00228 | 100 |
| PyVista | 1M | PLY Binario | 0.26931 | 0.00167 | 100 |
| PyNtCloud | 1M | PLY Binario | 0.00945 | 0.00037 | 100 |
| Open3D | 1M | PLY ASCII | 0.45422 | 0.00269 | 100 |
| PyVista | 1M | PLY ASCII | 0.36911 | 0.00240 | 100 |
| PyNtCloud | 1M | PLY ASCII | - | | - |
| Open3D | 2.25M | PLY Binario | 0.2158 | 0.0061 | 50 |
| PyVista | 2.25M | PLY Binario | 0.6516 | 0.0049 | 50 |
| PyNtCloud | 2.25M | PLY Binario | 0.0184 | 0.0019 | 50 |
| Open3D | 2.25M | PLY ASCII | - | - | - |
| PyVista | 2.25M | PLY ASCII | - | - | - |
| PyNtCloud | 2.25M | PLY ASCII | - | - | - |
| Open3D | 25 M | PLY Binario | 2.855 | 0.0165 | 20 |
| PyVista | 25 M | PLY Binario | 7.5939 | 0.0301 | 20 |
| PyNtCloud | 25 M | PLY Binario | 0.4552 | 0.0065 | 20 |
| Open3D | 25 M | PLY ASCII | - | - | - |

| | | | | | |
|---|---|---|---|---|---|
| PyVista | 25 M | PLY ASCII | - | - | - |
| PyNtCloud | 25 M | PLY ASCII | - | - | - |

*Tabla 6: Resultados carga fichero*

La gráfica de la Ilustración 24 muestra los tiempos de carga medios para los ficheros de todos los tamaños en el formato PLY binario.

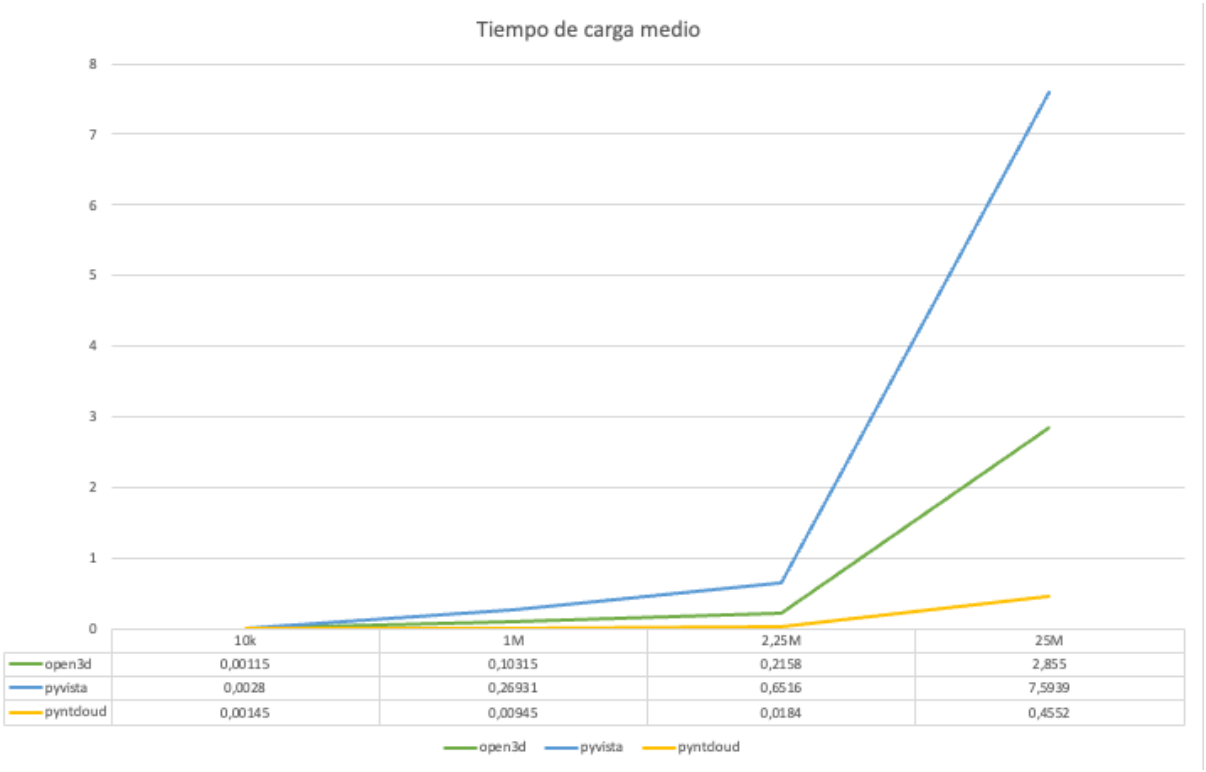



*Ilustración 24: Tiempo de carga medio*

Es posible comprobar como en el caso de los formatos binarios, la biblioteca PyNtCloud genera tiempos de carga de un orden inferior al resto de bibliotecas, lo que es más evidente para los ficheros de mayor tamaño. No obstante, se considera que cualquiera de las bibliotecas evaluadas podría ser adecuada para el objetivo del presente proyecto, generando todas ellas tiempos de carga razonables. Todas ellas basan la implementación de sus estructuras de nubes de puntos en arrays multidimensionales de NumPy, optimizados en C++. Se concluye que las diferencias en rendimiento que puedan existir entre cada una de las bibliotecas quedarían demarcadas por la eficiencia de las funciones de procesamiento de nubes de puntos que cada una implementa.

Por otra parte, se demuestra que cualquiera de las bibliotecas evaluadas presenta un rendimiento suficiente para importar las geometrías de referencia definidas en la sección 2.6. Como se puede comprobar en la sección mencionada, estas geometrías tienen en todos los casos un tamaño menor de 1 millón de vértices. Las pruebas de rendimiento realizadas presentaron un rendimiento aceptable para geometrías de varios millones de puntos, por lo que se consideran válidas.

### 3.6.3 Rendimiento renderizado

El presente apartado expone un conjunto de pruebas simples para evaluar el rendimiento en el renderizado de geometrías de cada una de las bibliotecas evaluadas. Se decide reducir las bibliotecas seleccionadas a Open3D y PyVista, puesto que hacen uso de los dos motores de renderizado principales utilizados por el resto de las bibliotecas (Filament y VTK, respectivamente). Se decide descartar Pyntcloud puesto que no cuenta con un módulo de renderizado propio, dependiendo de bibliotecas externas como Open3D para realizar esta tarea.

Para el desarrollo de las pruebas de rendimiento, se decide implementar una rutina básica de carga y visualización de una nube de puntos en cada una de las bibliotecas evaluadas. En cada caso, la nube es representada en una escena simple que permite al usuario rotar la geometría y desplazarse en el entorno 3D. El script será incluido para futura referencia junto al código fuente del proyecto.

Las pruebas de rendimiento fueron realizadas partiendo de los ficheros de nubes de puntos generados en la sección 2.5.1. Los parámetros principales del script determinan el fichero de nube de puntos a importar y la biblioteca con la que representar la nube de puntos. Una vez representada la nube de puntos en la escena principal, se procede a interactuar con la escena rotando la nube de puntos de manera manual, anotando la tasa de refresco (FPS) obtenida mediante la aplicación *OpenGL Profiler* de MacOS. Puesto que ambos motores de renderizado utilizan OpenGL como backend, es posible aplicar la misma metodología para ambas bibliotecas.

Las siguientes imágenes muestran un ejemplo en el que se visualiza una nube de puntos (Ilustración 25) y se evalúan los FPS, conectando *OpenGL Profiler* a la aplicación Python en la que evaluar el rendimiento (Ilustración 26).

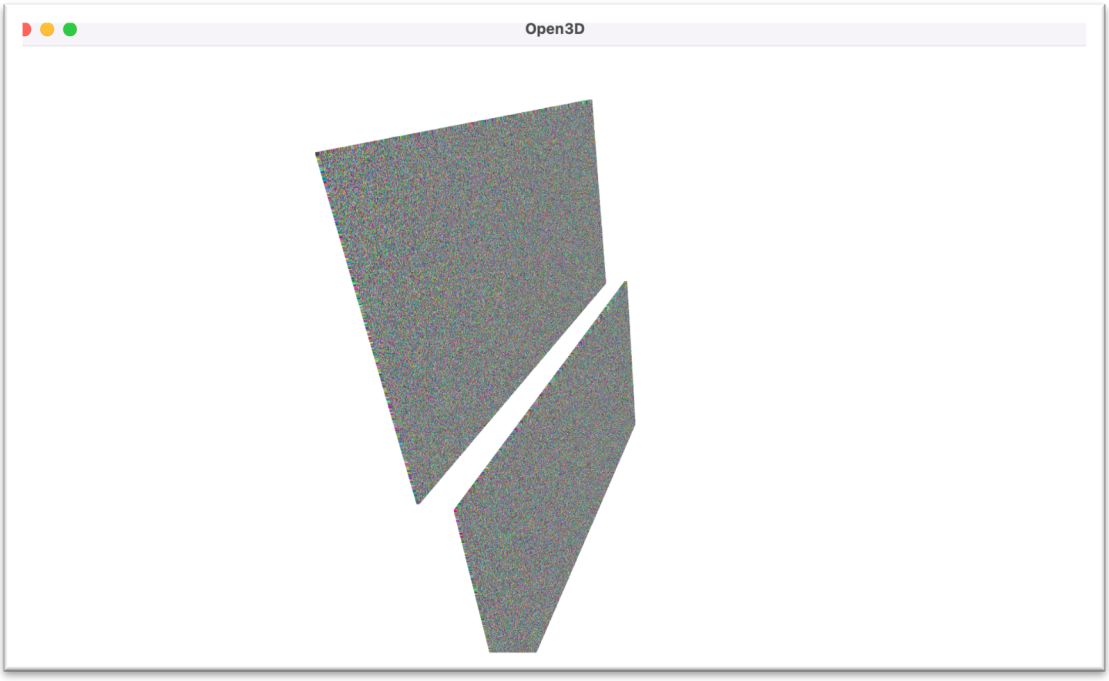



*Ilustración 25: Visualización nube de puntos*

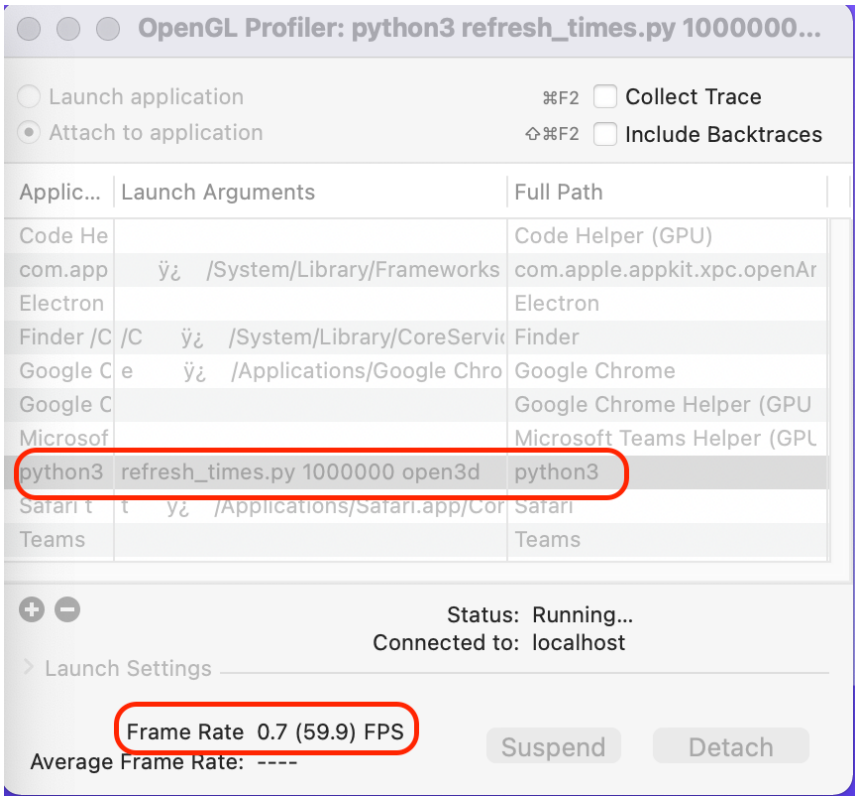



*Ilustración 26: OpenGL Profiler*

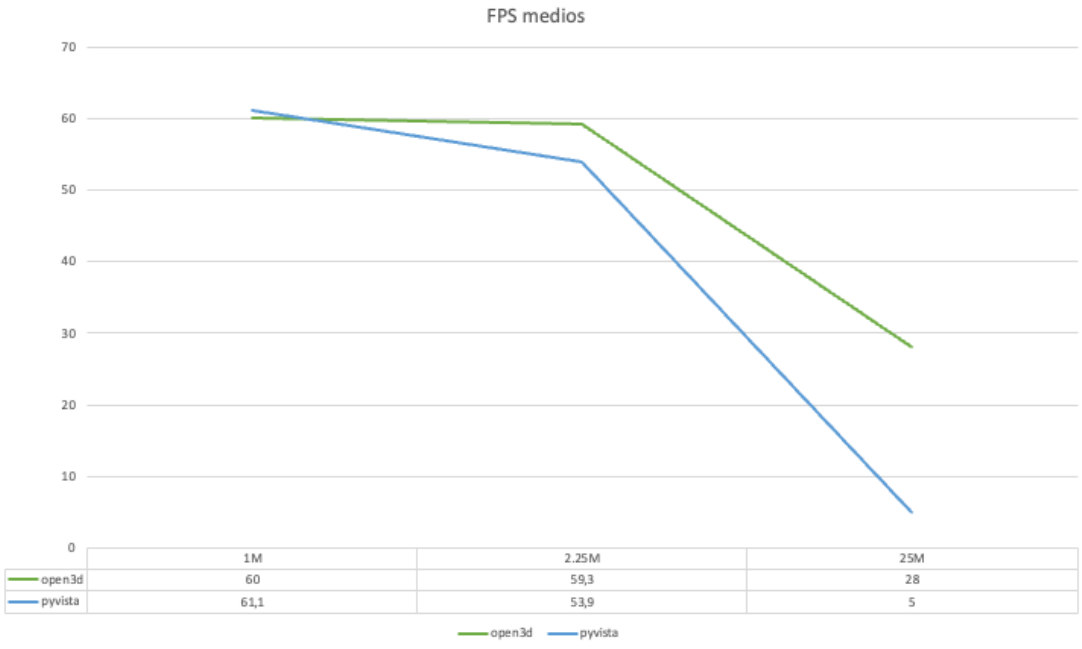



*Ilustración 27: Comparativa FPS*

El grafico presentado en la Ilustración 27 demuestra como, de manera evidente, los ficheros de mayor tamaño (25 millones de puntos) tienen gran impacto en la fluidez de ambas herramientas. Cabe destacar que en el caso de Open3D, aunque la herramienta de *profiling* genere medidas de alrededor de 28 FPS, la pérdida de fluidez es notable, con múltiples saltos bruscos en el renderizado de la geometría.

Con respecto a Open3D, en ningún caso la herramienta deja de ser usable, siendo posible interactuar con la nube de puntos de manera fluida en todos los tamaños. No obstante, probablemente deje de ser recomendable su uso en geometrías de tamaño mucho mayor.

En el caso de PyVista, aún respondiendo de manera aceptable, la pérdida de fluidez comenzó a ser notable en los ficheros de 2.25 millones de puntos, siendo difícilmente usable en las nubes de puntos de 25 millones de puntos.

### 3.6.4 Conclusiones

Teniendo en cuenta los aspectos evaluados en los apartados anteriores y las características básicas de cada una de las bibliotecas evaluadas, se decide utilizar **Open3D** como biblioteca principal para la implementación de la herramienta de visualización y procesamiento de nubes de puntos.

A continuación, se enumeran las razones principales tras esta decisión:

**Módulo GUI**: Open3D incluye un módulo dedicado a la creación de interfaces gráficas de usuario embebido en la propia biblioteca, lo que permite crear elementos de interfaz diversos como menús, elementos desplegables o botones interactivos. Este aspecto eliminará la complejidad asociada a la integración con un framework de creación de interfaces y no introducirá dependencias adicionales a la herramienta.

Se considera que las características del módulo de interfaz de Open3D son suficientes para los requisitos del proyecto, ya que permitirá crear una aplicación de escritorio que facilitará la aplicación de rutinas de procesamiento y visualización de nubes de puntos a usuarios sin conocimientos avanzados de programación.

**Operaciones de transformación**: Open3D implementa un conjunto de operaciones de transformación variado, que posibilita múltiples flujos de procesamiento de geometrías 3D, tanto básicos como avanzados. Como particularidad, da soporte tanto a nubes de puntos como a mallas poligonales, uno de los requisitos de la herramienta a desarrollar.

**Dependencias**: La biblioteca tiene un conjunto de dependencias reducido, lo que facilitará tanto la instalación como la distribución de la herramienta desarrollada en forma de un fichero ejecutable.

**Relevancia**: Atendiendo a su contribución al ecosistema de código abierto, Open3D es la biblioteca de procesamiento de nubes de puntos más activa en la actualidad, teniendo en cuenta su corta vida. Se publican nuevas versiones de la biblioteca de manera regular, siendo la más reciente de Junio de 2021, y cada una introduce funcionalidad relevante. A pesar de encontrarse lejos de su versión 1.0, implementa toda la funcionalidad necesaria para la elaboración de la herramienta definida.

**Ecosistema**: Open3D es una biblioteca en C++ con una interfaz completa implementada en Python. Esto aúna la eficiencia en el procesamiento de geometrías con la facilidad de integración

con las bibliotecas del ecosistema científico de Python. La utilización de Python como principal lenguaje de programación permitirá hacer uso de aspectos dinámicos que simplifiquen la implementación de funcionalidad de la herramienta.

**Rendimiento**: Las pruebas de rendimiento realizadas demuestran que el rendimiento del motor de renderizado Filament al que Open3D proporciona acceso es más que suficiente para las necesidades de la herramienta planteada, siendo posible visualizar geometrías del orden de millones de puntos.

# 4 Software de visualización de estructuras 3D

En la presente sección se evalúa un conjunto de herramientas de visualización de nubes de puntos y sus principales características. Al contrario que las bibliotecas analizadas anteriormente, estas herramientas ofrecen una experiencia de usuario predefinida y no están orientadas a la extensión y al desarrollo de software de procesamiento de nubes de puntos.

El análisis de la funcionalidad ofrecida por múltiples herramientas dedicadas a la visualización de nubes de puntos permitirá determinar la funcionalidad común en las mismas que integrar posteriormente en la herramienta de visualización propia que se desarrollará posteriormente.

En todos los casos, la información presentada fue extraída exclusivamente de la documentación oficial y APIs de referencia de cada herramienta. En el caso de Potree y Mountains, por considerarlas de especial relevancia, se realizó un análisis detallado manual a partir de las versiones de prueba que proporcionan.

## 4.1 Potree

Potree es un motor de renderizado de nubes de puntos de gran densidad basado en WebGL y ejecutable en un navegador web. Es una herramienta de código abierto basada en la biblioteca de renderizado 3D Three.js y permite la visualización de nubes de puntos del orden de millones de puntos.

Representa las nubes de puntos con una estructura de datos propia que busca reducir la tasa de transferencia de datos por red, posibilitando el renderizado de grandes conjuntos de datos en navegadores web. Como resumen de su funcionamiento, utiliza una estructura de datos jerárquica que almacena subconjuntos de la nube de puntos inicial a distintas resoluciones. Esta estructura permite determinar de manera eficiente los puntos que se encuentran fuera de la perspectiva del usuario, así como renderizar con menor fidelidad las secciones más distantes a la perspectiva original.

La Ilustración 28 muestra un ejemplo de visualización con Potree de un conjunto de datos del orden de millones de puntos.

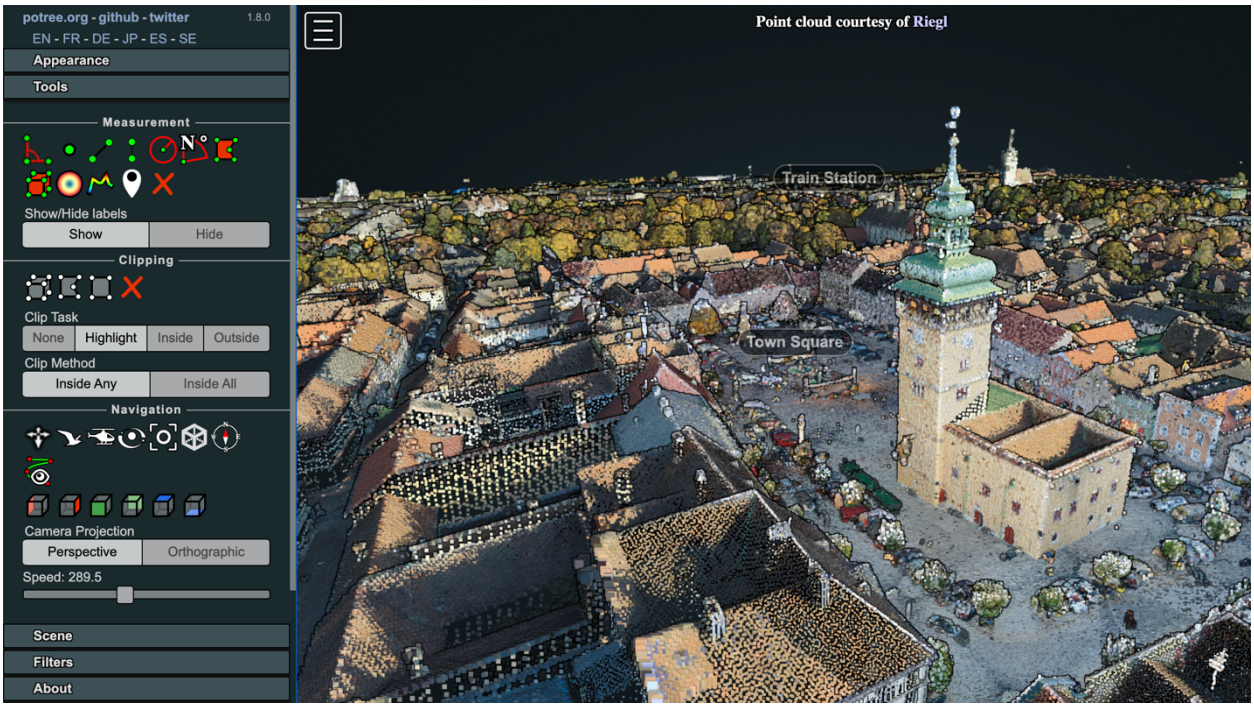



*Ilustración 28: Ejemplo visualización Potree*

La principal ventaja que aporta, además de la eficiencia en el renderizado de grandes nubes de puntos, es su portabilidad y extensibilidad. Al contrario que otras bibliotecas o herramientas, permite una representación eficiente de datos a través de red, que no requiere de una copia física del conjunto de datos en el equipo local del usuario, cuyo tamaño puede sobrepasar las decenas de GBs. Siendo una herramienta orientada a la web, puede hacer uso de su ecosistema y ser integrada en webs convencionales. La contrapartida a este aspecto sería la dificultad de integración con herramientas de procesamiento de nubes de puntos no basadas en web. Alternativamente, siendo una herramienta orientada exclusivamente al renderizado y visualización, no implementa algoritmos de procesamiento complejos.

Como proyecto de código abierto cabe destacar la actividad y apoyo de la comunidad al proyecto que encontrándose en actual desarrollo en 2021 cuenta con más de 2100 menciones en GitHub.

### 4.1.1 Funcionalidad principal

Además del propio motor de renderizado, Potree incluye un visor web de nubes de puntos que implementa un conjunto de funciones de utilidad como las descritas a continuación:

#### *4.1.1.1 Herramientas de navegación*

Potree implementa varios modos de navegación que pueden resultar de utilidad al usuario, de entre los que destacan los siguientes:

- **Controles orbitales**: Permiten desplazarse por la escena orbitando alrededor de un punto predefinido. En este modo de navegación la propia nube de puntos se mantiene en la misma posición, siendo la cámara la que se desplaza alrededor de la nube de puntos. El uso de la rueda del ratón permite cambiar la distancia al punto de rotación y el uso de doble click realiza un zoom hacia el punto seleccionado.

- **Controles en primera persona**: Permiten el desplazamiento en primera persona, sobrevolando la nube de puntos. Los controles principales con W, A, S y D facilitan el desplazamiento de la cámara en 4 direcciones. Mediante el click izquierdo es posible girar la cámara hacia cualquier dirección, mientras que el click derecho permite hacer un paneo completo sobre la nube de puntos, trasladando la cámara y la nube de puntos en la misma dirección.
- **Controles terrestres**: En este modo de navegación, el movimiento de ratón con click izquierdo pulsado permite desplazar la nube de puntos en el plano. El click derecho permite rotar la cámara alrededor del punto seleccionado, mientras que la rueda del ratón permite hacer zoom hacia el punto seleccionado.

#### *4.1.1.2 Selección y filtrado (clipping)*

La creación de *clip-boxes* permite al usuario destacar un área de interés dentro de la nube de puntos a visualizar, mediante el resaltado de todos los puntos dentro del área definida por el usuario o el filtrado de todos los puntos que se encuentren fuera de la misma. Esto resulta de especial utilidad en nubes de puntos de gran densidad en las que parte de la nube de puntos dificulte la visualización de otras partes de esta. Por ejemplo, sería posible ocultar todos los puntos pertenecientes a la parte superior de un edificio para poder ver el interior de este sin recurrir a la navegación.

Esta funcionalidad es implementada en Potree mediante el uso de *shaders* que destacan o descartan vértices en función de si se encuentran dentro del área definida por el usuario. Adicionalmente, se proporciona funcionalidad que representa todos los puntos en el interior o exterior de la zona seleccionada.

La Ilustración 30 y la Ilustración 29 muestran como es posible resaltar y filtrar elementos de una nube de puntos mediante las herramientas de clip boxes de Potree.

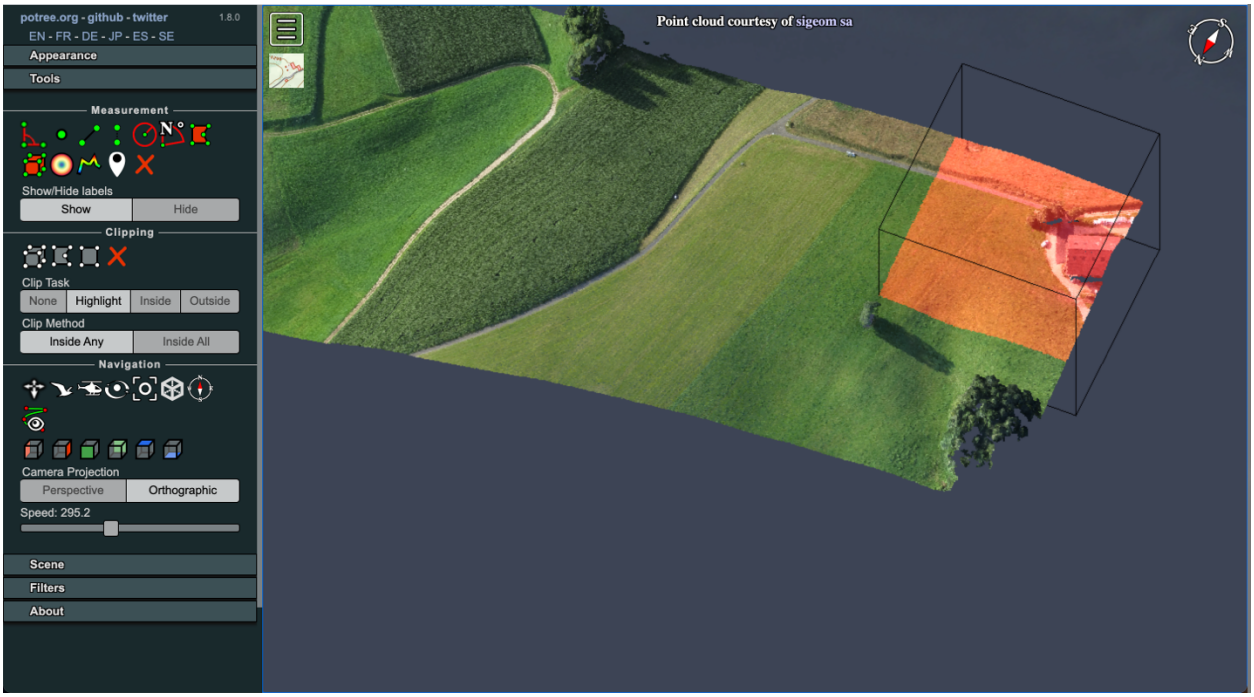



*Ilustración 29: Resaltado Clip Boxes*

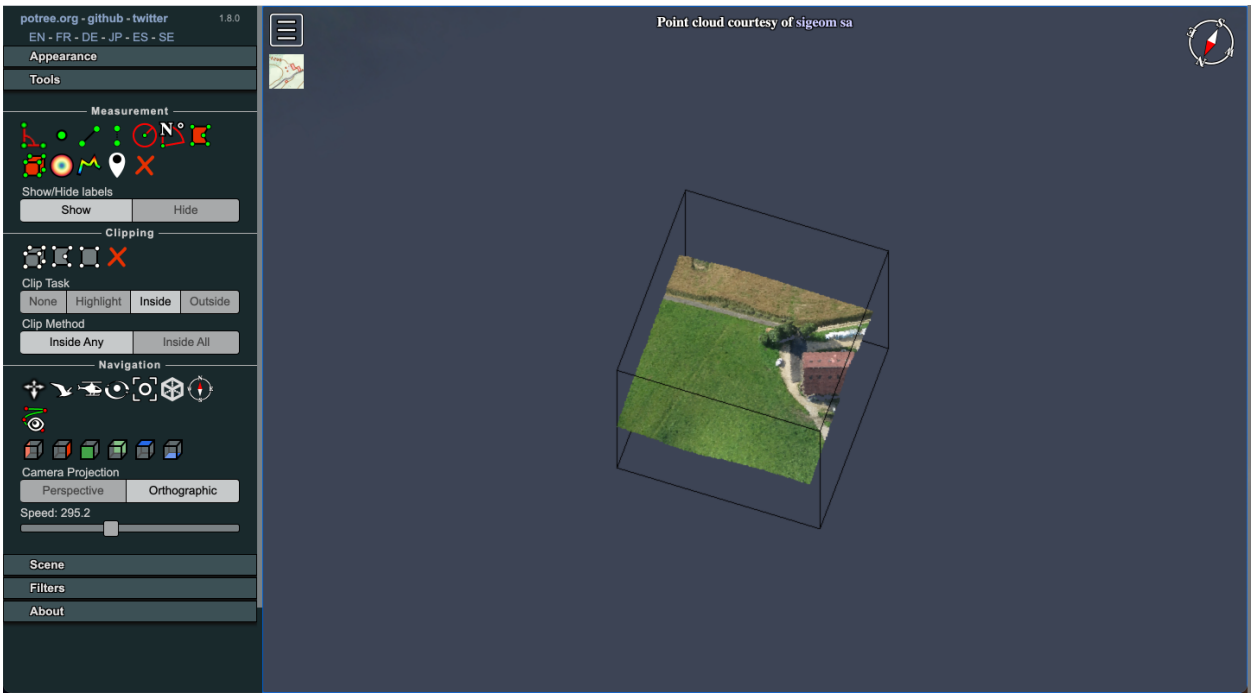



*Ilustración 30: Filtrado clip boxes*

#### 4.1.1.3 Medidas

Potree ofrece un conjunto de herramientas para el cálculo de medidas de, entre otros, distancias, áreas y ángulos en nubes de puntos. Esta funcionalidad permite al usuario definir los parámetros de cada medida (longitud, ángulo) sobre la propia nube de puntos, utilizando el ratón para seleccionar puntos.

Los resultados de cada medida son representados en etiquetas flotantes sobre la nube de puntos. La Ilustración 31 muestra un ejemplo de medidas realizadas con algunas de las herramientas de medición ofrecidas por Potree.

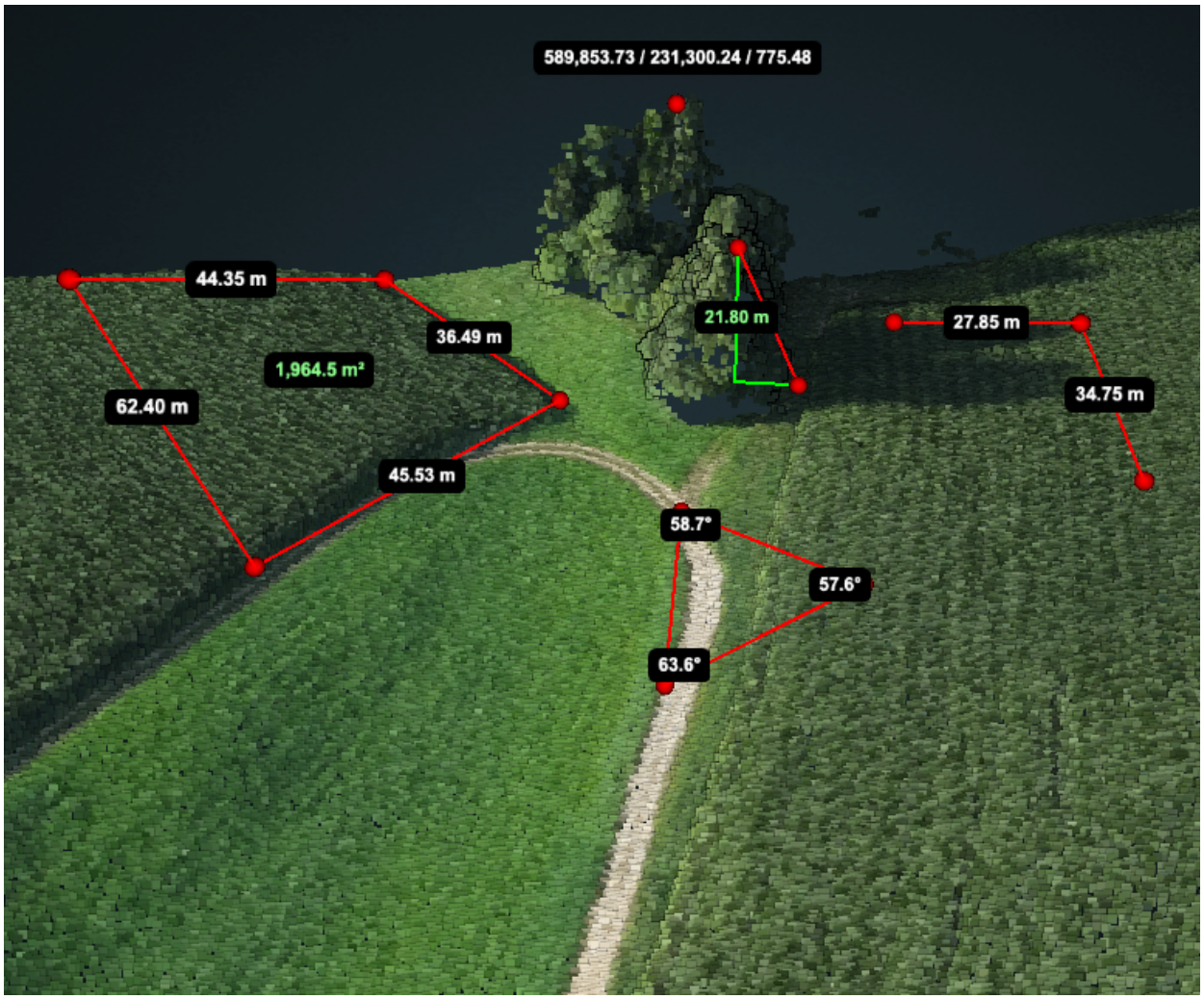



*Ilustración 31: Medidas Potree*

#### *4.1.1.4 Clasificación*

Permite el filtrado de puntos a partir de características preestablecidas de la nube de puntos. Por norma general, permite identificar si un punto pertenece a clases como vegetación o estructuras, aunque se podría seguir otra metodología de asignación de clases. En la práctica, en Potree la clase de cada punto se representa como un número entero predefinido, que posteriormente se mapea con un color con el que se representa. La Ilustración 32 muestra un ejemplo de clasificación de una nube de puntos con múltiples clases predefinidas. Como se puede comprobar, es posible ocultar o mostrar cada una de las clases, además de cambiar el color con el que son representadas.

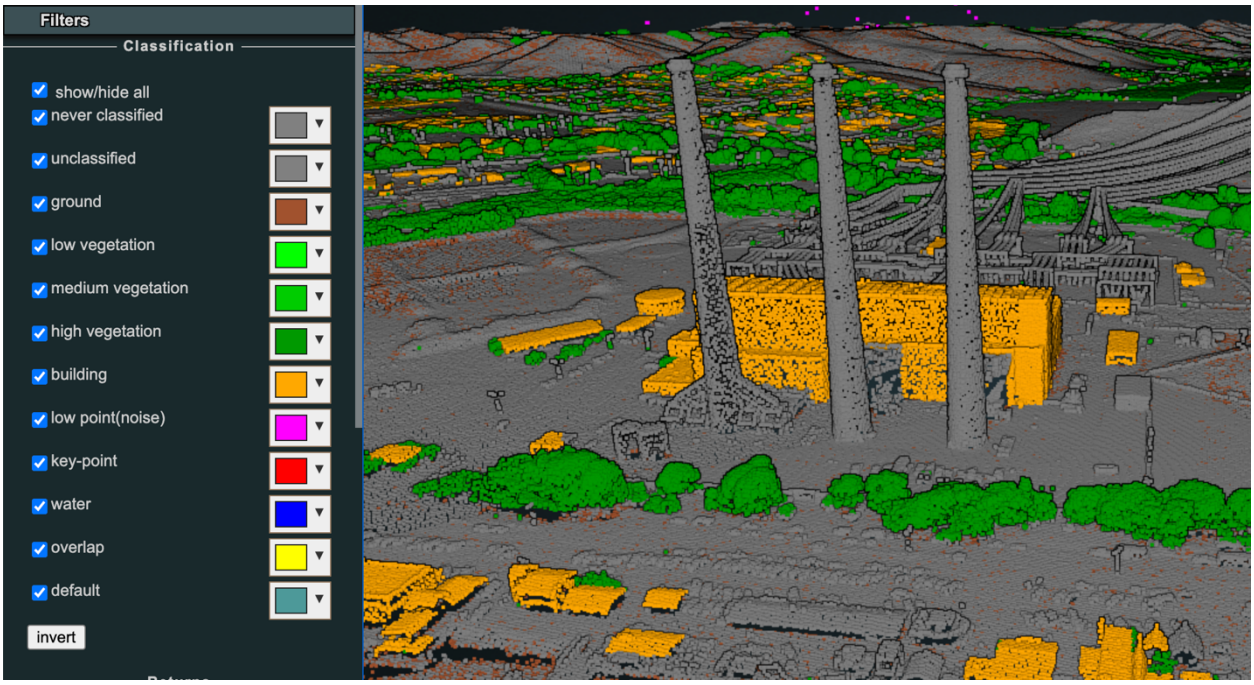



*Ilustración 32: Clasificación Potree*

#### 4.1.1.5 Anotaciones

Las anotaciones permiten asociar etiquetas de texto en determinados puntos de interés de un modelo 3D. Adicionalmente, Potree implementa herramientas de navegación que sitúan la cámara del usuario en posiciones asociadas a cada anotación de manera automática. Siendo Potree una herramienta basada en la web, permite definir anotaciones como HTML estático, lo que facilita la inserción de texto, imágenes o videos en las mismas.

Las anotaciones pueden a su vez activar funcionalidad que interactúe con la propia nube de puntos o con el motor de renderizado. La Ilustración 33 muestra varios ejemplos de anotaciones en Potree. Una de ellas permite, en particular, cambiar el gradiente de color a aplicar a la elevación de la nube de puntos representada.

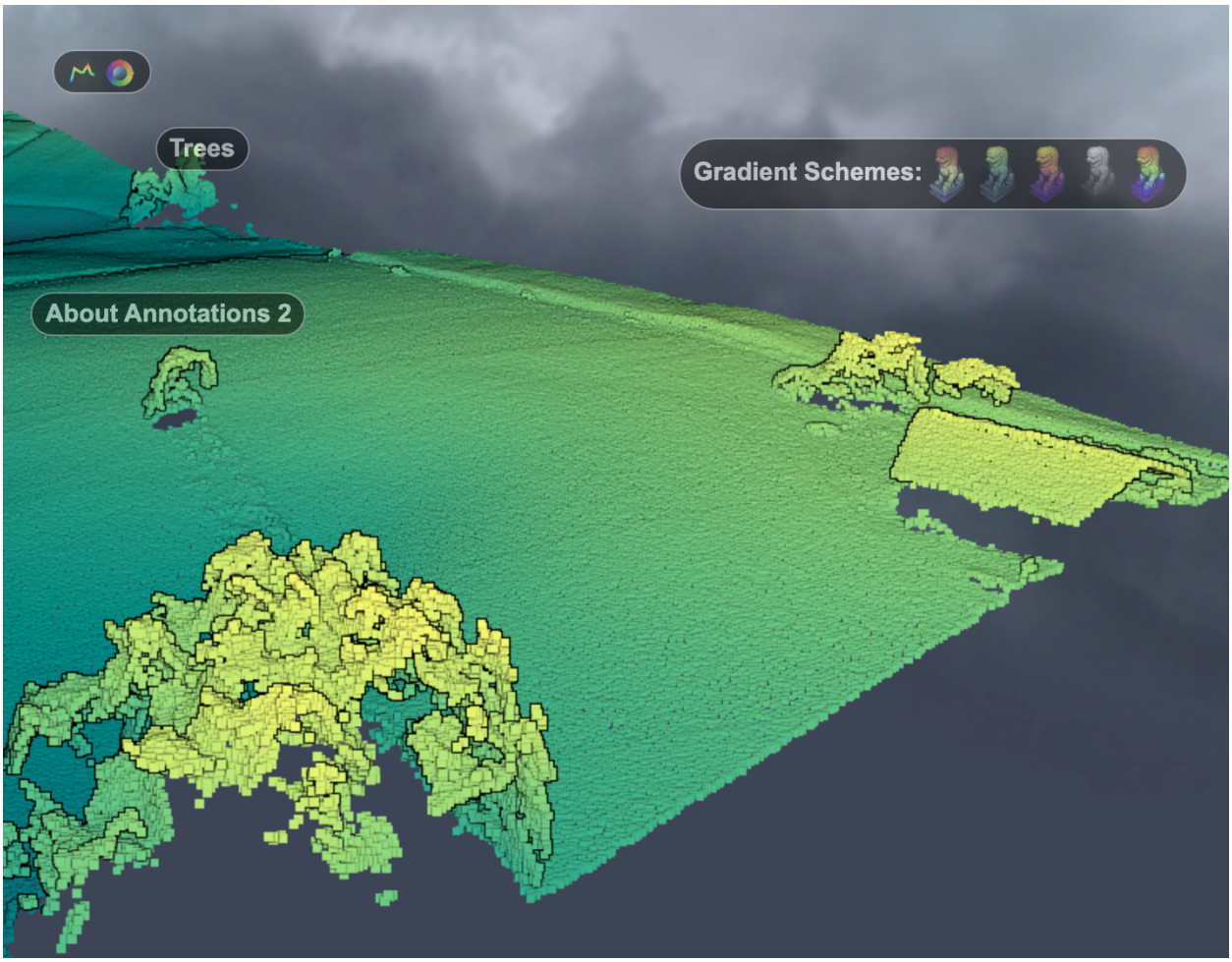



*Ilustración 33: Ejemplos anotaciones*

## 4.2 ParaView

ParaView es una herramienta multiplataforma de código abierto para el análisis y visualización de datos científicos. Fue desarrollada con la intención de analizar conjuntos de datos de gran tamaño de manera distribuida, siendo a la vez posible su utilización en ordenadores personales de especificaciones técnicas más reducidas para la visualización de conjuntos de datos de menor tamaño.

ParaView está orientada principalmente hacia las áreas científicas que hacen uso extensivo de técnicas de elementos finitos, aplicadas al cálculo de resultados en estructuras de 3 dimensiones. No obstante, fue diseñada de manera flexible y facilita la visualización de estructuras de datos 3D de manera genérica.

La Ilustración 34 muestra un ejemplo de posibles visualizaciones realizadas con ParaView, extraída de su galería de ejemplos (https://www.paraview.org/gallery/).

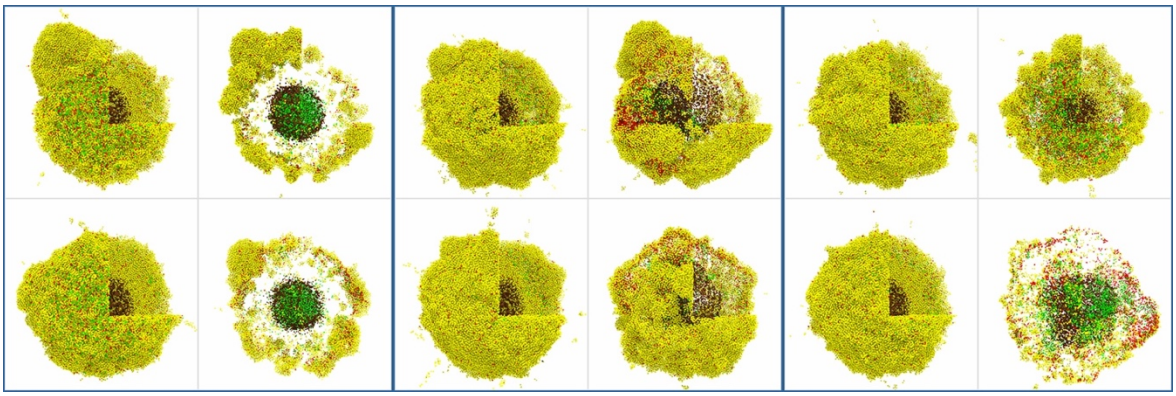


*Ilustración 34: Visualización ParaView*

De entre los principales dominios científicos a los que se aplica la herramienta, destacan los siguientes:

- **Climatología**: Análisis y visualización de datos meteorológicos para la realización de predicciones climáticas, desplazamiento de nubes…
- **Simulaciones fluidodinámicas (CFD)**: Visualización de mallas generadas por software específico de CFD.
- **Análisis de estructuras:** Visualización de mallas generadas por algoritmos de métodos de elementos finitos.

Además de los campos de estudio mencionados anteriormente, ParaView también ofrece módulos dedicados al análisis de nubes de puntos. Esta funcionalidad facilita la visualización interactiva y el procesamiento de datos de estructuras de nubes de puntos generados por múltiples fuentes de datos como escáneres LiDAR o cámaras de profundidad. Ofrece a su vez funcionalidad dedicada para el procesamiento de nubes de puntos, en la forma del *plugin* PCL, que proporciona los siguientes algoritmos básicos:

- Estimación de superficies.
- Clustering.
- Técnicas de filtrado y downsampling.
- Eliminación de outliers.
- Segmentación.

La Ilustración 34 muestra un ejemplo de visualización de nubes de puntos mediante ParaView.

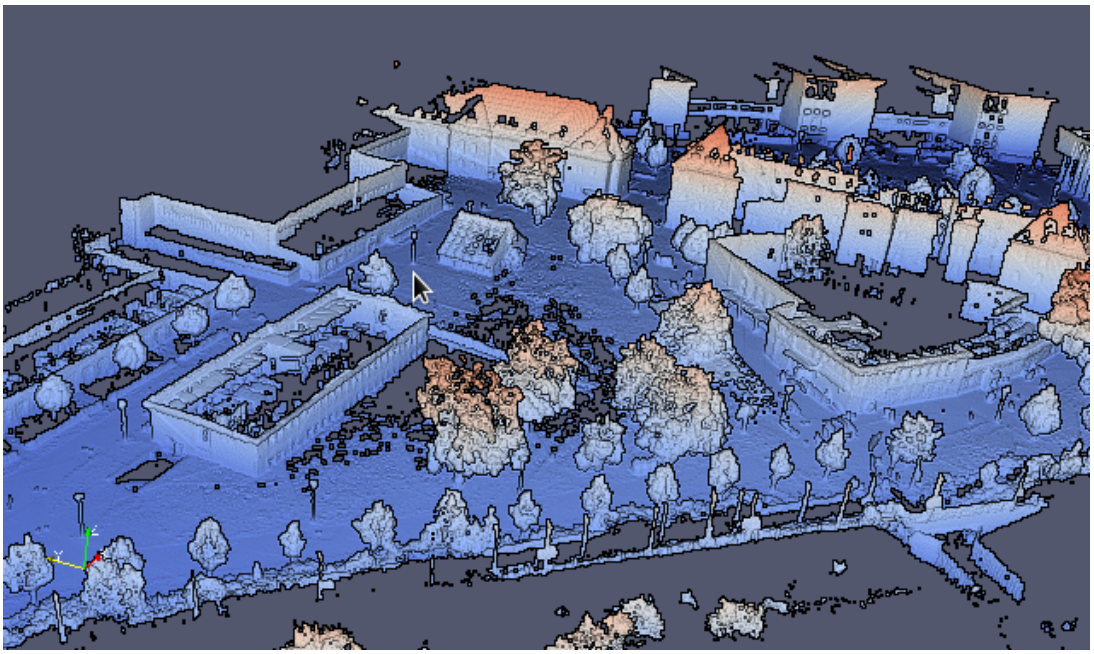


*Ilustración 35: Nube de puntos ParaView*

## 4.3 MeshLab

MeshLab es una herramienta de código abierto para el procesamiento, edición y visualización de mallas 3D triangulares. Es una herramienta de escritorio que implementa funcionalidad avanzada para la edición e inspección de nubes de puntos y modelos 3D generados por sensores y cámaras de profundidad. La Ilustración 37 muestra un ejemplo de posibles visualizaciones extraído de su documentación.

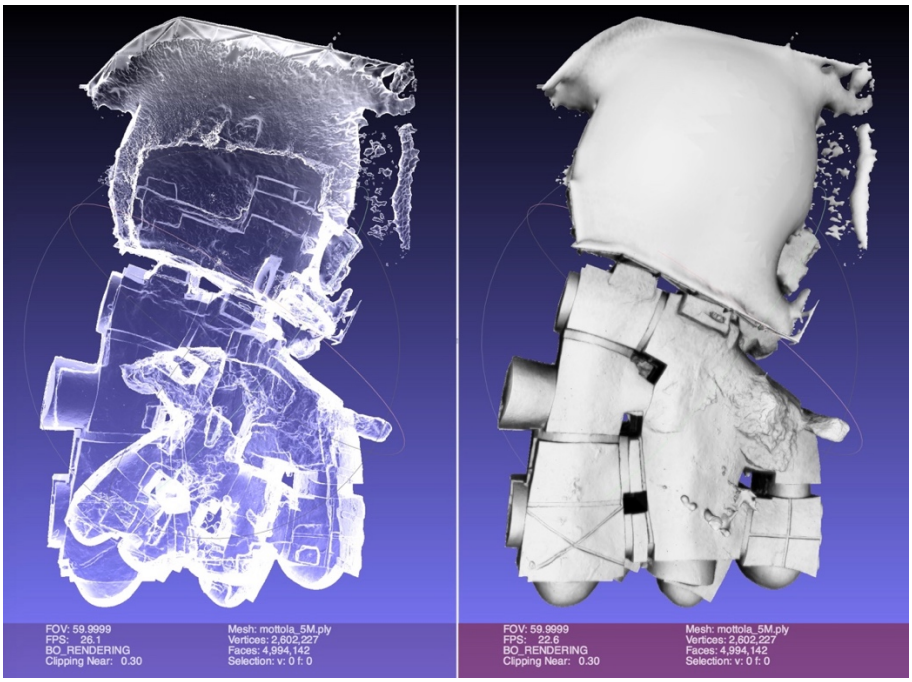



*Ilustración 36: Visualización MeshLab*

### 4.3.1 Funcionalidad principal

A continuación, se describen algunas de las funcionalidades implementadas por la herramienta MeshLab:

**Alineación de mallas**

La alineación de estructuras 3D, también conocida como *point cloud registration* o *scan matching*, es el proceso que permite automatizar la alineación de nubes de puntos mediante transformaciones espaciales. Este proceso permite combinar múltiples conjuntos de datos en un único modelo que englobe toda la información de la geometría.

MeshLab implementa funcionalidad que permite desplazar múltiples mallas en un sistema de referencia común, facilitando la alineación manual. Alternativamente, incluye un algoritmo de alineación basado en ICP (*Iterative Closest Point*) para automatizar el proceso. El proceso de alineación se puede aplicar tanto a nubes de puntos como a mallas 3D procedentes de escáneres y herramientas de captura 3D.

**Reconstrucción de superficies**

MeshLab ofrece varios algoritmos para la reconstrucción de mallas a partir de nubes de puntos discretas, de entre los que destaca la reconstrucción de Poisson. La Ilustración 37 muestra un ejemplo del proceso de reconstrucción de superficies proporcionado por la herramienta.

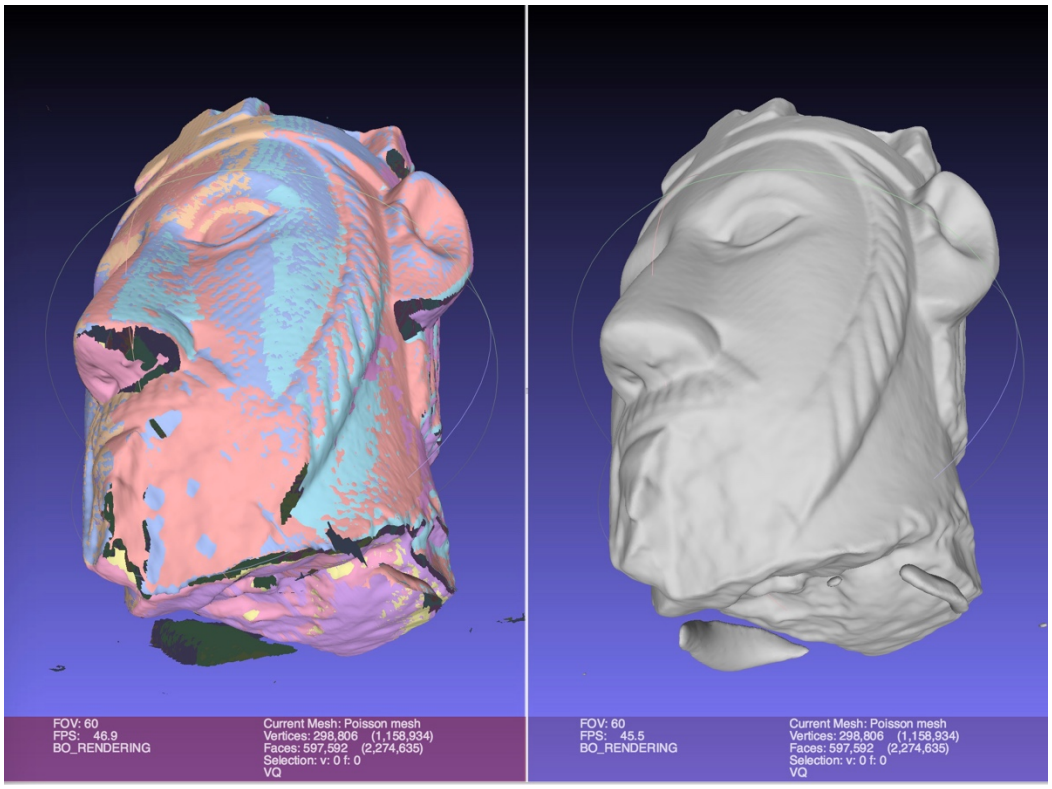



*Ilustración 37: MeshLab reconstrucción superficies*

**Asignación de color**

MeshLab ofrece funcionalidad para la alineación y mapeo de colores en modelos 3D a partir de imágenes 2D, lo que resulta de utilidad en aquellos conjuntos de datos que no contengan información de color. MeshLab proporciona varios métodos, automatizados y asistidos, para facilitar el mapeo de color tanto a nivel de vértices como de texturas o superficies.

**Filtrado y limpieza de modelos**

MeshLab implementa funcionalidad diversa para la limpieza y filtrado de *outliers* y elementos anómalos en modelos 3D y nubes de puntos, como por ejemplo la eliminación de vértices duplicados o caras superpuestas de una malla. El proceso de filtrado es en algunos casos automático y en otros interactivo, siendo necesaria la interacción por parte del usuario para realizar el proceso de selección.

**Simplificado de mallas**

Se incluyen funcionalidad diversa para la simplificación de modelos 3D, reduciendo la complexidad geométrica de los mismos sin perder la forma básica de los mismos. También es posible, por el contrario, aumentar el número de triángulos que conforma una malla o aumentar los puntos de una nube de puntos.

**Medidas**

MeshLab ofrece un conjunto de filtros automáticos para el cálculo de información básica de una estructura 3D, como el número de vértices, la curvatura o la distancia geométrica entre dos puntos.

**Coloreado**

MeshLab ofrece un conjunto de filtros predefinidos para manipular el color de los vértices y caras de un modelo 3D, como por ejemplo la saturación y el contraste. Adicionalmente, incluye otros filtros automáticos como el de la oclusión ambiental, y ofrece funcionalidad para el coloreado de propiedades escalares específicas de cada vértice o cara de un modelo 3D.

### 4.4 Mountains

Mountains es una herramienta software para el análisis de superficies desarrollado por la compañía Digital Surf. Ofrece un conjunto de módulos dedicados a diversos campos de la metrología 3D.

MountainsMap es un módulo dedicado al análisis de texturas de superficies 2D y 3D, diseñado para su uso junto a profilómetros láser 3D y otros instrumentos de medida de superficies. Ofrece las siguientes características:

- Visualización y corrección de superficies: Pre-procesamiento de datos mediante la eliminación de valores atípicos y ruido.
- Cálculo de parámetros de perfiles de superficie y rugosidad a partir de estándares ISO.
- Análisis de superficies: Cálculo de distancias, áreas y volúmenes en la geometría de superficie.
- Visualización 3D: Visualización de topografía de superficies 3D de alta calidad, visualización de perfiles de superficie e imágenes 2D.

- Análisis de contornos: detección de defectos en superficies mediante el análisis del perfil de curvatura de la superficie.

La presente sección proporciona una visión general de la herramienta Mountains y de sus principales características. La evaluación de la herramienta fue realizada a partir de la versión de prueba gratuita proporcionada por Digital Surf. En primer lugar, se repasan varios conceptos básicos asociados a la herramienta y se proporcionan ejemplos de las principales características de Mountains aplicadas a perfiles 2D y a superficies 3D.

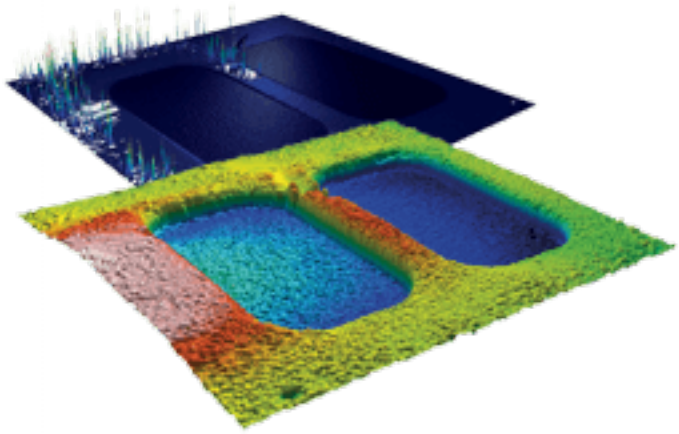


*Ilustración 38: Ejemplo visualización MountainsMap*

### 4.4.1 Conceptos básicos

A continuación, se repasa el significado de ciertos conceptos básicos asociados a la herramienta Mountains a los que se hará referencia en sucesivos apartados. Mountains implementa un modelo de uso basado en *documentos* interactivos que permite aplicar un conjunto de *operadores* a estructuras de datos denominadas *estudiables*. También se permite aplicar un conjunto de visualizaciones y análisis a los estudiables denominados *estudios*.

- **Estudiable** (*Studiable*): Un estudiable representa cualquier conjunto de datos que puede ser analizado en la herramienta Mountains, como por ejemplo datos de perfiles y superficies generados por profilómetros.
- **Estudio** (*Study*): Un estudio es una visualización concreta de un estudiable o de un análisis aplicado al mismo.
- **Operador** (*Operator*): Los operadores son herramientas o funciones que modifican un estudiable, produciendo nuevos estudiables.
- **Documento** (*Document*): Todos los estudios son organizados en un documento de texto interactivo que puede ser modificado, guardado y exportado.
- **Workflow**: El workflow representa todos los elementos que conforman un documento, mostrando la relación entre estudiables y estudios. Esto permite visualizar la cronología de transformaciones (operadores) aplicadas a un conjunto de datos (estudiable) y las distintas visualizaciones aplicadas al mismo (estudios.)

### 4.4.2 Operadores

Los operadores son funciones proporcionadas por Mountains para modificar un estudiable, generando uno o múltiples estudiables adicionales. Estos se encuentran agrupados por funcionalidad en la vista de operadores de Mountains, y son específicos para cada tipo de geometría.

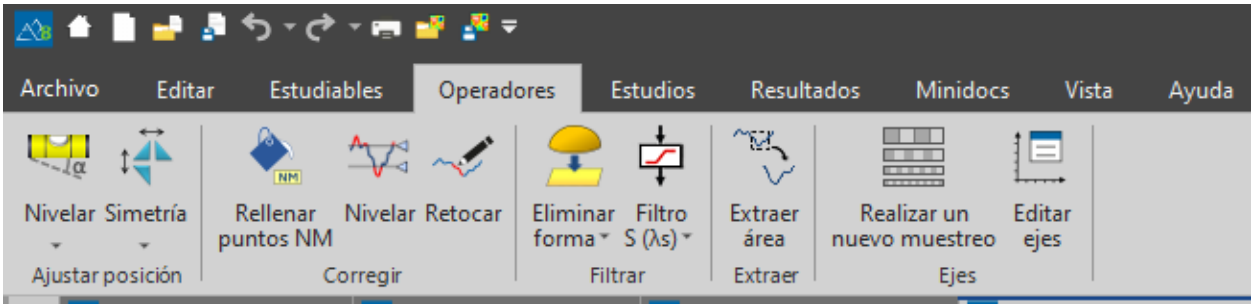



*Ilustración 39: Operadores Mountains*

Todos los operadores tienen asociado un menú de configuración propio que permite definir los parámetros principales que los gobiernan. Adicionalmente, este menú también proporciona una pre-visualización del nuevo estudiable que se generará una vez aplicado el operador al estudiable original.

#### *4.4.2.1 Operadores aplicados a perfiles*

A continuación, se describe un subconjunto de los operadores que es posible aplicar a un estudiable que representa un perfil 2D:

**Ajustar posición**

*Nivelar*: Eliminar la pendiente general de un perfil 2D. El menú de configuración del operador ofrece múltiples métodos de rectificación a aplicar (línea media cuadrática, zona mínima...), como queda recogido en la Ilustración 40.

*Simetría*: Ajustar simetría del perfil con respecto a los ejes x e y.

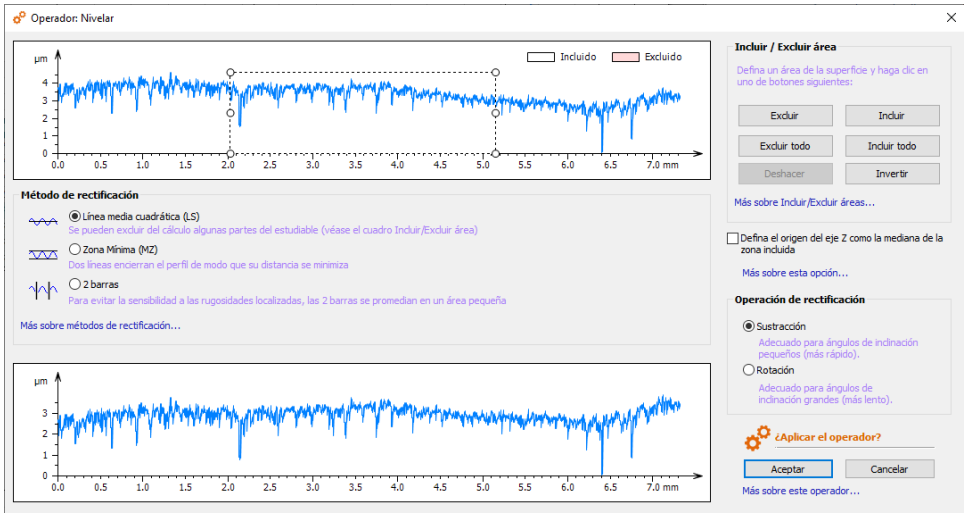



*Ilustración 40: Operador nivelación de perfiles*

**Corregir**

*Rellenar puntos No Medidos*: Rellenar puntos de datos no medidos por el profilómetro utilizando una forma suave calculada a partir de los puntos vecinos o de los puntos de altura máxima, mínima y media. La Ilustración 41 muestra las opciones de configuración del operador para el rellenado de puntos no medidos.

*Nivelar*: Nivelar para eliminar valores atípicos o para seleccionar una zona de interés en el eje Z.

*Retocar*: Retocar un área del perfil definida de manera manual para eliminar un defecto local.

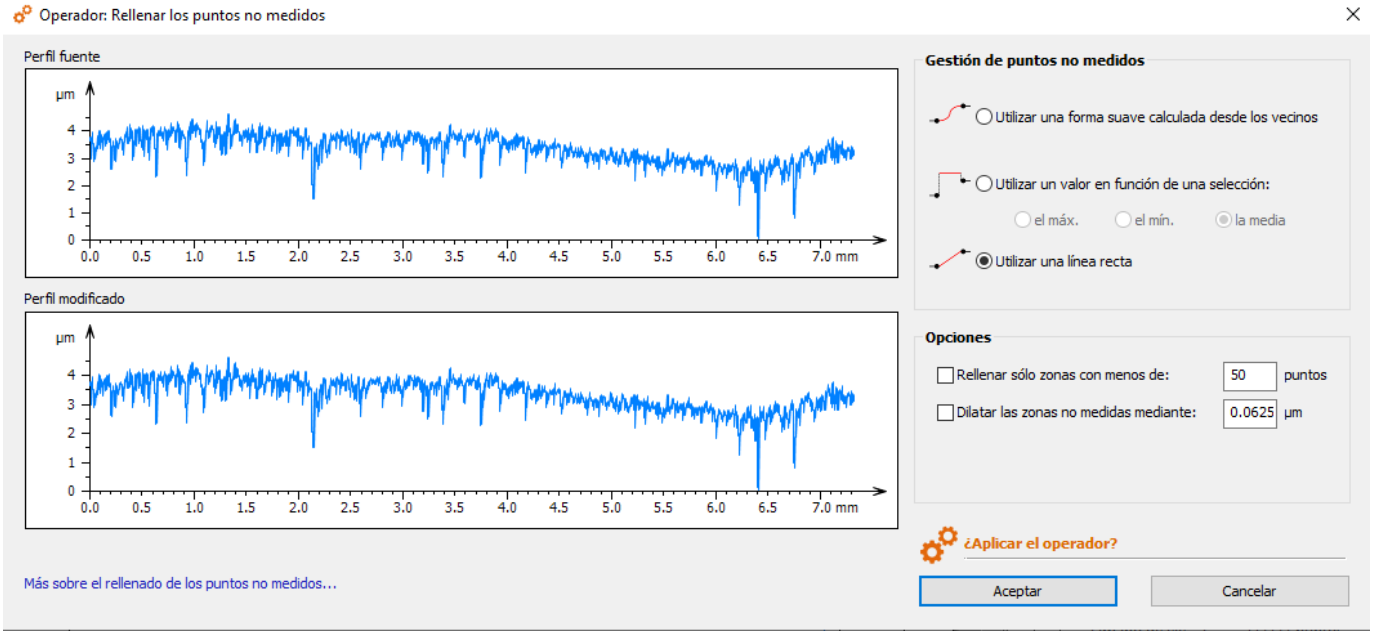



*Ilustración 41: Configuración operador puntos No Medidos*

**Filtrar**

*Eliminar forma*: Eliminar una forma circular o polinómica del perfil 2D.

*Filtro S*: Eliminar micro rugosidad mediante un filtro S.

*Filtro metrológico*: Separar los componentes de rugosidad (paso alto) y ondulación (paso bajo).

*Filtrar espectro*: Filtrar seleccionando las frecuencias que se van a excluir del espectro de frecuencia obtenido usando FTT (Transformación rápida de Fourier).

**Extraer**

*Extraer área*: Permite extraer una región de interés del perfil y utilizarlo como nuevo estudiable.

**Reunir**

*Unir dos perfiles*: Fusionar dos perfiles de forma manual o automática

#### 4.4.2.2 Operadores superficies

A continuación, se muestra un subconjunto de los operadores proporcionados por Mountains para la modificación de superficies 3D.

**Ajustar posición**

*Nivelar*: Eliminar la pendiente general de la superficie.

*Corrección línea por línea*: Corregir la superficie línea por línea para reducir los efectos de cambio de línea del profilómetro.

*Reflejar*: Simetría respecto a uno o varios ejes.

*Girar*: Girar el estudiable en cualquier ángulo de manera manual.

**Corregir**

*Filtro espacial*: Aplicar un filtro matriz para eliminar el ruido, para suavizar o para detectar formas. La Ilustración 42 muestra como es posible configurar el operador de filtro espacial, con una pre-visualización del estudiable generado.

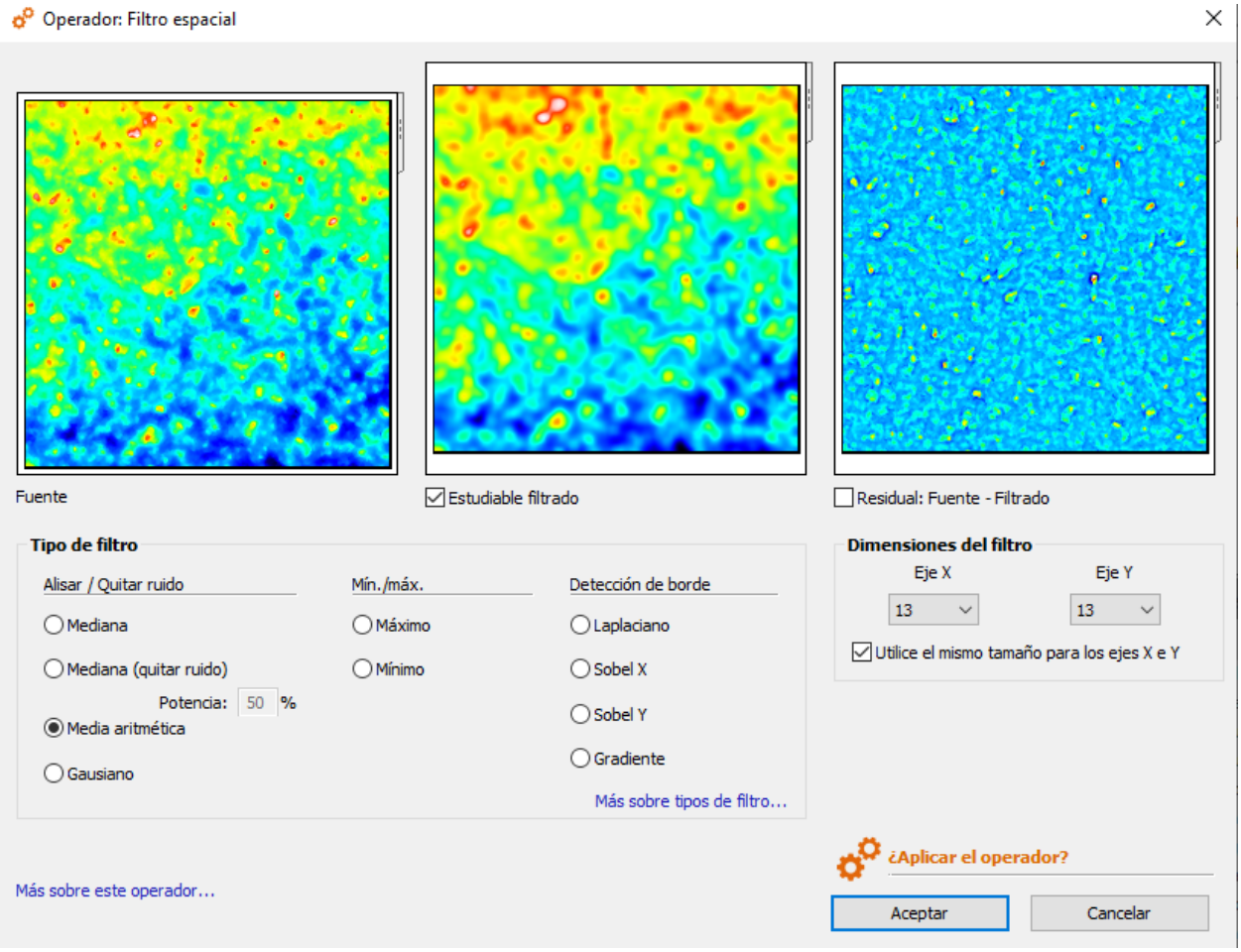



*Ilustración 42: Configuración operador filtro espacial*

*Rellenar puntos No Medidos*: Rellenar puntos de datos no medidos utilizando una forma suave calculada a partir de los puntos vecinos o de los puntos de altura máxima, mínima y media.

*Nivelar*: Nivelar para eliminar valores atípicos o para seleccionar una zona de interés en el eje Z.

*Retocar*: Retocar un área de la superficie definida de manera manual para eliminar un defecto local.

*Eliminar valores atípicos*: Eliminar los valores atípicos que a menudo están presentes en superficies medidas con microscopios ópticos. Permite eliminar tanto los valores atípicos de los bordes, así como los valores atípicos aislados. Los valores atípicos se convierten en puntos de datos No Medidos que pueden ser rellenados automáticamente mediante interpolación de valores vecinos.

**Filtrar**

*Eliminar forma*: Eliminar una forma polinómica, una esfera o un cilindro de la superficie 3D.

*Filtro S*: Eliminar micro rugosidad mediante un filtro S.

*Filtro metrológico*: Separar los componentes de rugosidad (paso alto) y ondulación (paso bajo).

**Extraer**

*Extraer área*: Permite extraer una región de interés de la superficie y utilizarla como nuevo estudiable.

*Extraer perfil*: Permite extraer una sección transversal de una superficie. El perfil extraído resultante se corresponde con la intersección de la superficie con un plano vertical perpendicular a la superficie. La Ilustración 43 muestra una captura de la herramienta de extracción de perfiles que permite al usuario definir gráficamente el perfil a extraer.

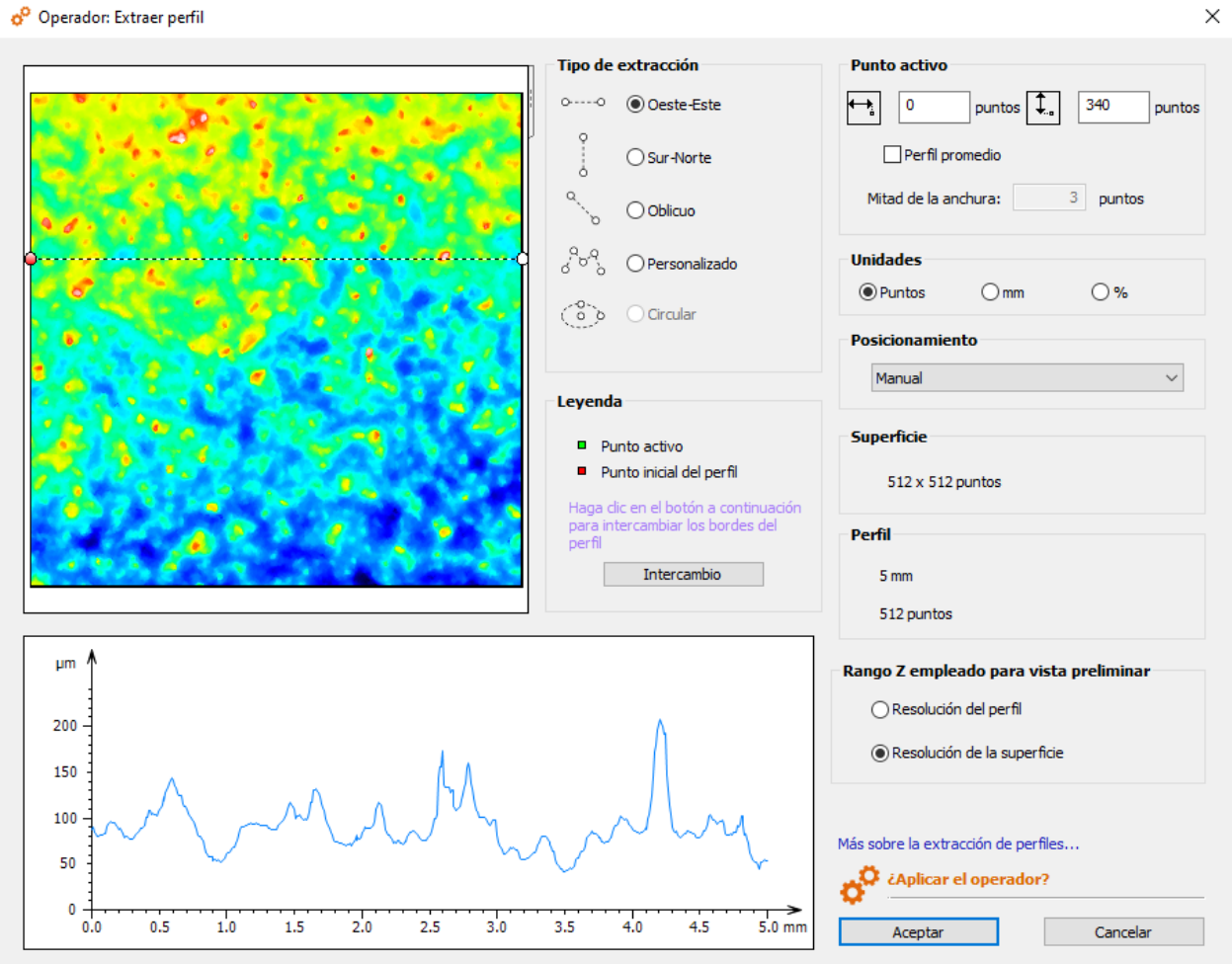



*Ilustración 43: Operador extracción de perfil*

## Convertir

*Serie de perfiles*: Permite convertir una superficie en una serie de perfiles usando parte de la superficie. El menú de configuración del operador permite definir múltiples aspectos de este, como el área del que extraer perfiles. Estos parámetros están representados en la Ilustración 44.

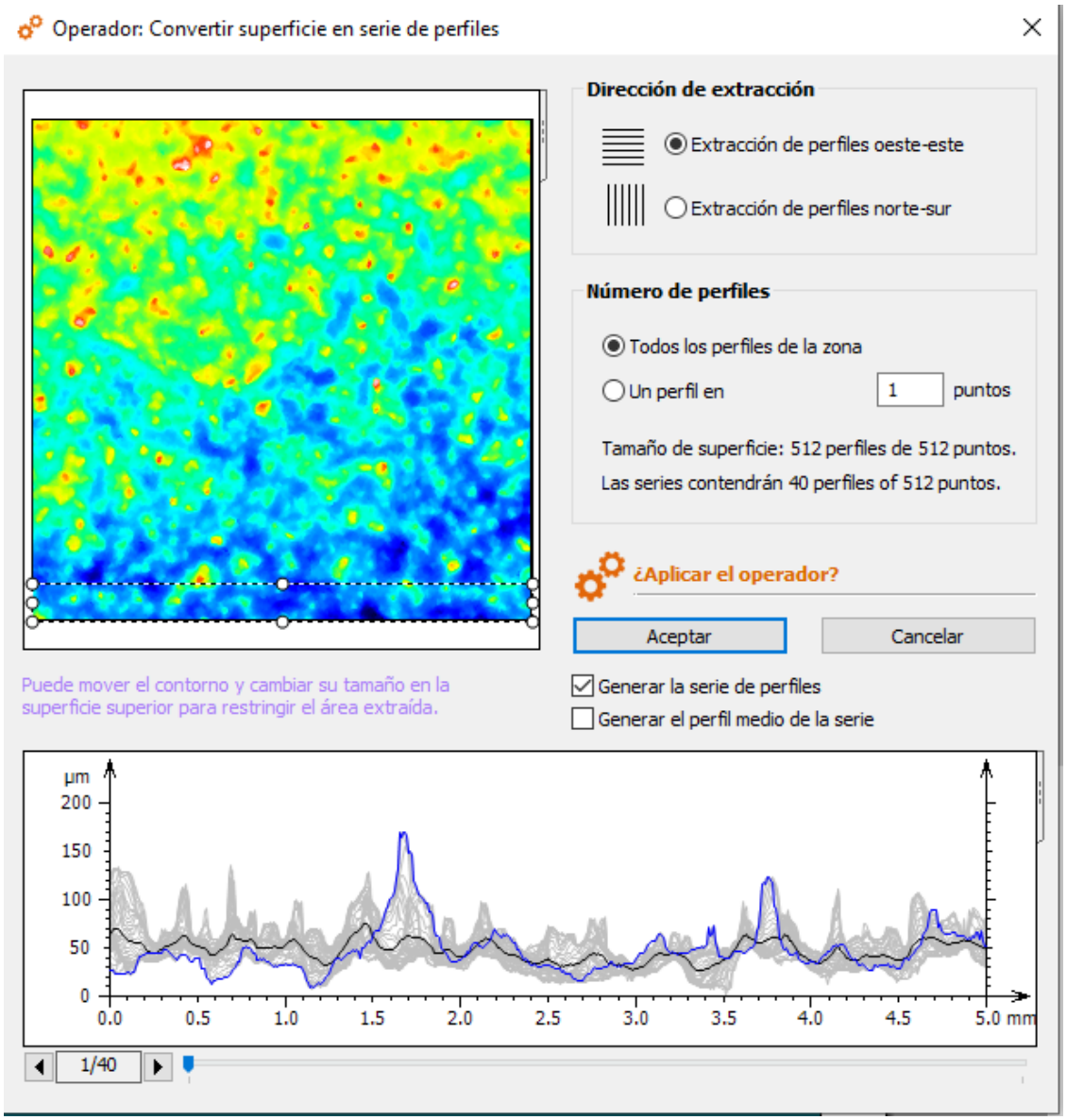



*Ilustración 44: Operador serie de perfiles*

### 4.4.3 Estudios

Como concepto principal de la herramienta Mountains, los estudios son diferentes tipos de análisis aplicados a conjuntos de datos. Al contrario que los operadores, los estudios no modifican los estudiables de partida, sino que permiten extraer información adicional de los mismos mediante técnicas de visualización y análisis.

Para aplicar un estudio a un estudiable mediante la herramienta Mountains, basta con seleccionar el estudiable en el documento y pulsar el estudio requerido en el menú superior. El estudio es insertado de manera automática en el documento a continuación del estudiable y ofrece menús contextuales que facilitan su configuración.

#### 4.4.3.1 *Estudios aplicados a perfiles 2D*

A continuación, se mencionan algunos de los estudios que es posible realizar sobre un estudiable asociado a un perfil 2D. La herramienta cuenta con múltiples estudios avanzados adicionales asociados al análisis estructural o de frecuencia de un perfil que no serán tratados.

**Visualización**

*Curva del perfil*: Visualización 2D de la forma del perfil.

*Perfiles filtrados*: Mostrar el perfil primario y los perfiles de rugosidad u ondulación del perfil actual.

*Envolventes morfológicas*: Visualizar la envolvente superior y la inferior.

La Ilustración 45 muestra una captura de un documento de Mountains en el que se aplican los diferentes estudios relacionados con la visualización a un perfil 2D.

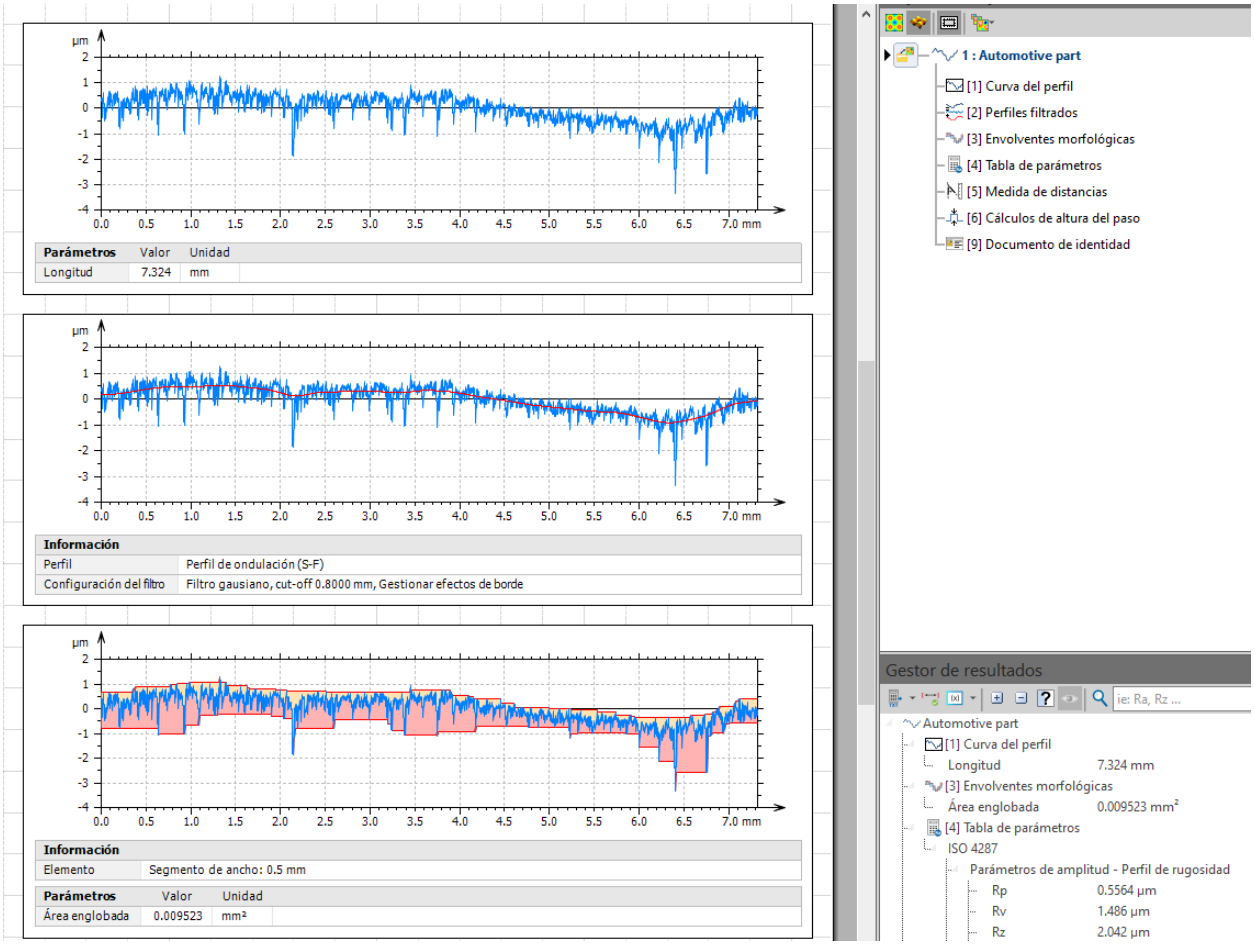



*Ilustración 45: Estudios de visualización de perfiles 2D*

**Parámetros**

*Tabla de parámetros*: Calcular la textura de la superficie o los parámetros de forma de acuerdo con las normas internacionales ISO.

**Geometría**

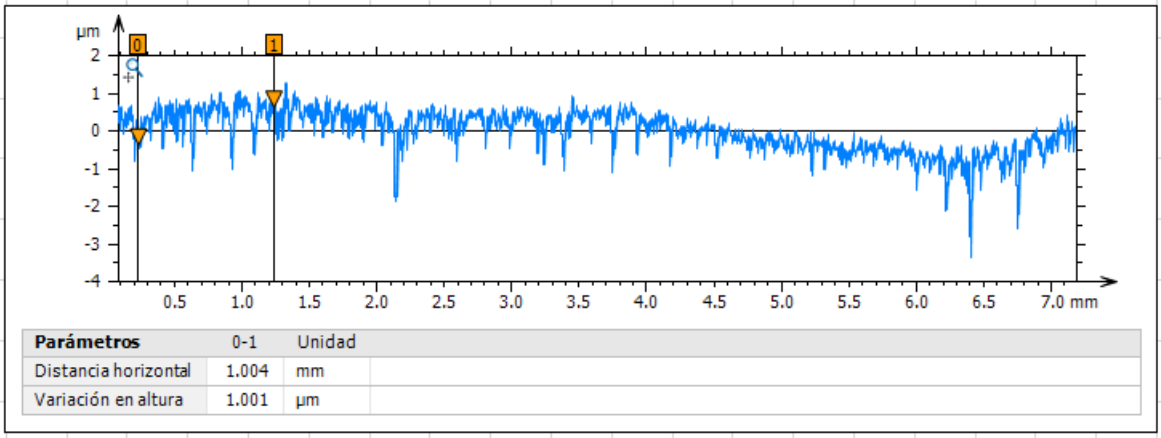
*Medida de distancias*: Mediciones múltiples en el perfil, como por ejemplo la medición de distancia, diferencia de altura, pendiente o ángulo.

*Altura del paso*: Cálculo de los parámetros de altura del paso, como la altura, profundidad o ancho.

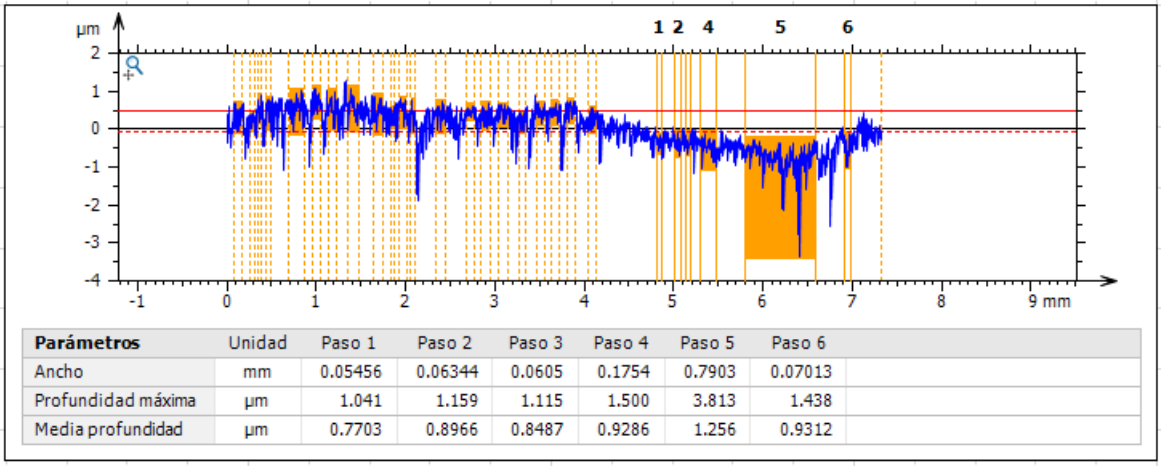
La Ilustración 46 muestra una captura de un documento de Mountains en el que se aplica un conjunto de estudios de visualización a un perfil 2D.

**Identidad**

*Documento de identidad*: Información general sobre el estudiable como nombre, tamaño y resolución.




| Parámetros | 0-1 | Unidad |
|---|---|---|
| Distancia horizontal | 1.004 | mm |
| Variación en altura | 1.001 | µm |




| Parámetros | Unidad | Paso 1 | Paso 2 | Paso 3 | Paso 4 | Paso 5 | Paso 6 |
|---|---|---|---|---|---|---|---|
| Ancho | mm | 0.05456 | 0.06344 | 0.0605 | 0.1754 | 0.7903 | 0.07013 |
| Profundidad máxima | µm | 1.041 | 1.159 | 1.115 | 1.500 | 3.813 | 1.438 |
| Media profundidad | µm | 0.7703 | 0.8966 | 0.8487 | 0.9286 | 1.256 | 0.9312 |

| ISO 4287 - Rugosidad (S-L) | | | |
|---|---|---|---|
| *F: Ninguno* | | | |
| *Filtro S (λs): Gausiano, 2.5 µm* | | | |
| *Filtro L (λc): Gausiano, 0.8 mm* | | | |
| *Longitud de evaluación: Todos los λc (9)* | | | |
| **Parámetros de amplitud** | | | |
| Rp | 0.5564 | µm | |
| Rv | 1.486 | µm | |
| Rz | 2.042 | µm | |
| Rc | 0.6834 | µm | *Sin promediado (valor individual)* |
| Rt | 3.192 | µm | |
| Ra | 0.2093 | µm | |
| Rq | 0.2945 | µm | |
| Rsk | -1.591 | | |
| Rku | 7.592 | | |
| **Parámetros de relación de material** | | | |
| Rmr | 86.12 | % | *c = 1 µm por debajo del pico más alto* |
| Rdc | 0.3815 | µm | *p = 20%, q = 80%* |

| Documento de identidad | | |
|---|---|---|
| Nombre: | Automotive part | |
| Ruta del archivo: | C:\Users\Ivan\AppD ...\Automotive part.pro | |
| Tipo de estudiable: | Perfil | |
| **Ejes:** | **X** | |
| Longitud: | 7.324 | mm |
| Tamaño: | 117177 | puntos |
| Espaciado: | 0.0625 | µm |
| Offset: | 0.3624 | mm |
| **Ejes:** | **Z** | |
| Tipo de capa: | Topografía | |
| Longitud: | 4.634 | µm |
| Mín: | 0.3915 | µm |
| Máx: | 5.025 | µm |
| Tamaño: | 4633546 | Dígitos |
| Espaciado: | 0.0010 | nm |

*Ilustración 46: Estudio medición manual de perfiles*

#### 4.4.3.2 Estudios superficies 3D

A continuación, se mencionan los principales estudios que es posible realizar en Mountains sobre un estudiable que representa una superficie 3D.

**Visualización**

Esta agrupación de estudios engloba todas las opciones de visualización proporcionadas por Mountains para la representación de superficies 3D:

*Vista pseudo-color*: Imagen en 2D coloreada para representar las alturas de los puntos de la superficie.

*Simulación de foto*: Simulación de una fotografía en 2D para obtener una representación natural de la superficie simulando efectos de luz.

*Líneas de contorno*: Gráfico en 2D que muestra las líneas de contorno de los puntos de superficie que están al mismo nivel de altura, dividiendo la superficie en secciones horizontales regulares.

*Vista en 3D*: Muestra de la topografía en una visualización en 3D interactiva y a tiempo real.

La Ilustración 47: Visualización de superficiesIlustración 47 Ilustración muestra las principales visualizaciones disponibles, aplicadas a una superficie 3D y agrupadas en un único documento.

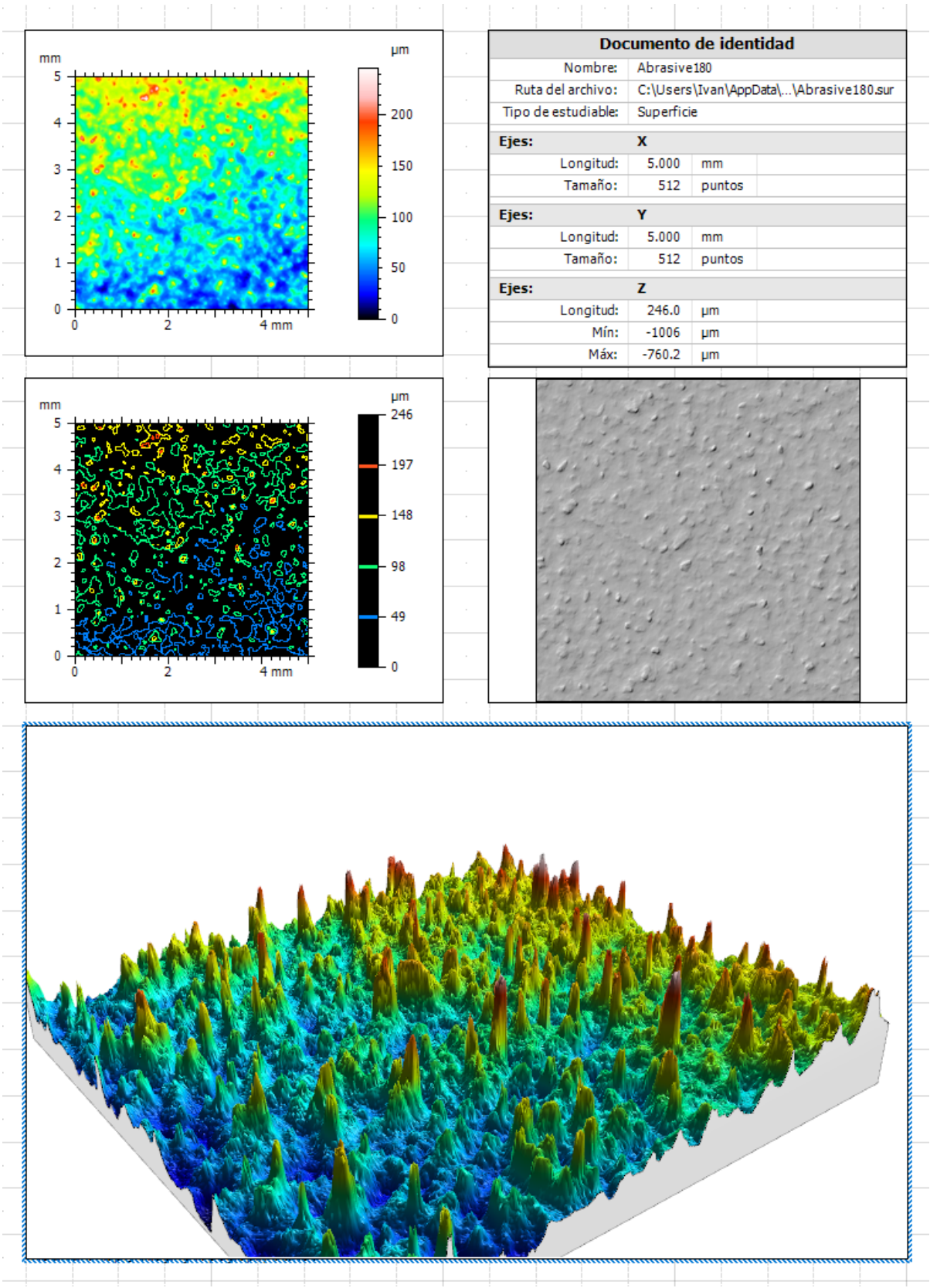



*Ilustración 47: Visualización de superficies*

**Parámetros**

*Tabla de parámetros*: Calcular la textura de la superficie o los parámetros de forma de acuerdo con las normas internacionales ISO.

**Geometría**

*Mediciones manuales*: Mediciones manuales múltiples en la superficie. Permite hacer mediciones de los siguientes aspectos:

- Coordenadas de un punto.
- Distancia entre dos puntos.
- Ángulo definido por tres puntos.
- Rectángulo: Parámetros de área a partir de una forma rectangular definida por el usuario.
- Mínimo y máximo: Mostrar etiquetas en los puntos mas altos/bajos de la superficie.

*Altura de paso*: Detección de dos o más planos. Los planos pueden ser detectados automáticamente o definidos manualmente.

- Altura del paso.
- Diferencia de ángulo entre dos planos.
- Orientación.
- Inclinación.

La Ilustración 48 muestra un ejemplo de documento de Mountains en el que se realizan un conjunto de mediciones manuales y automáticas. Estas mediciones son interactivas y es posible definir sus límites sobre la propia imagen 2D.

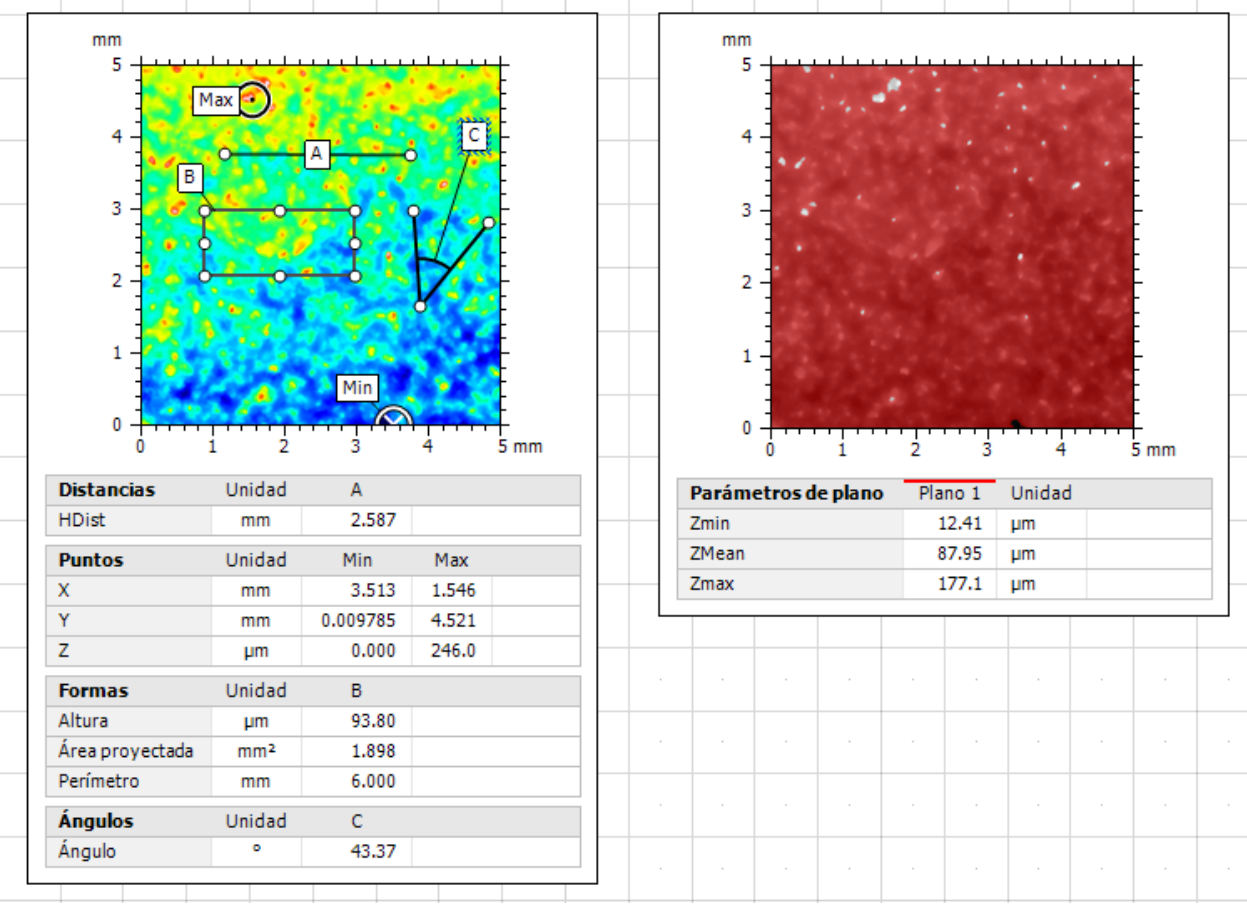



*Ilustración 48: Mediciones manuales de superficies*

### 4.4.1 Ejemplo completo de uso

A continuación, se proporciona un ejemplo de utilización básica de la herramienta Mountains que muestra sus características principales.

En este ejemplo se elabora un documento de Mountains que tiene como objetivo la visualización de una superficie 3D eliminando los valores atípicos (*outliers*) de la misma. Los valores atípicos son principalmente el resultado de medidas incorrectas por parte de los instrumentos de medida, y deben ser eliminados para que no interfieran en análisis posteriores. Son especialmente reconocibles por tener valores mucho más elevados o reducidos que el resto de las medidas del área circundante.

La Ilustración 49 muestra un documento con la superficie de partida. El documento incluye dos estudios simples para visualizar el estudiable inicial: la Vista pseudocolor y la Vista 3D. Los valores atípicos son perceptibles tanto en la vista 2D, destacados en color claro, como en la vista 3D por su valor en el eje Z.

Pulsando el operador "*Eliminar valores atípicos*" del menú superior, es posible aplicar el operador a la superficie de partida. La Ilustración 50 muestra como se configuró el operador para obtener la superficie final. Principalmente, se decidió eliminar todos los valores atípicos y sustituirlos realizando una interpolación. La imagen de la derecha ofrece una representación del resultado de aplicar el operador a la superficie de partida.

Finalmente, configuramos el documento para representar una vista 3D del estudiable resultante. El documento final es representado en la Ilustración 51.

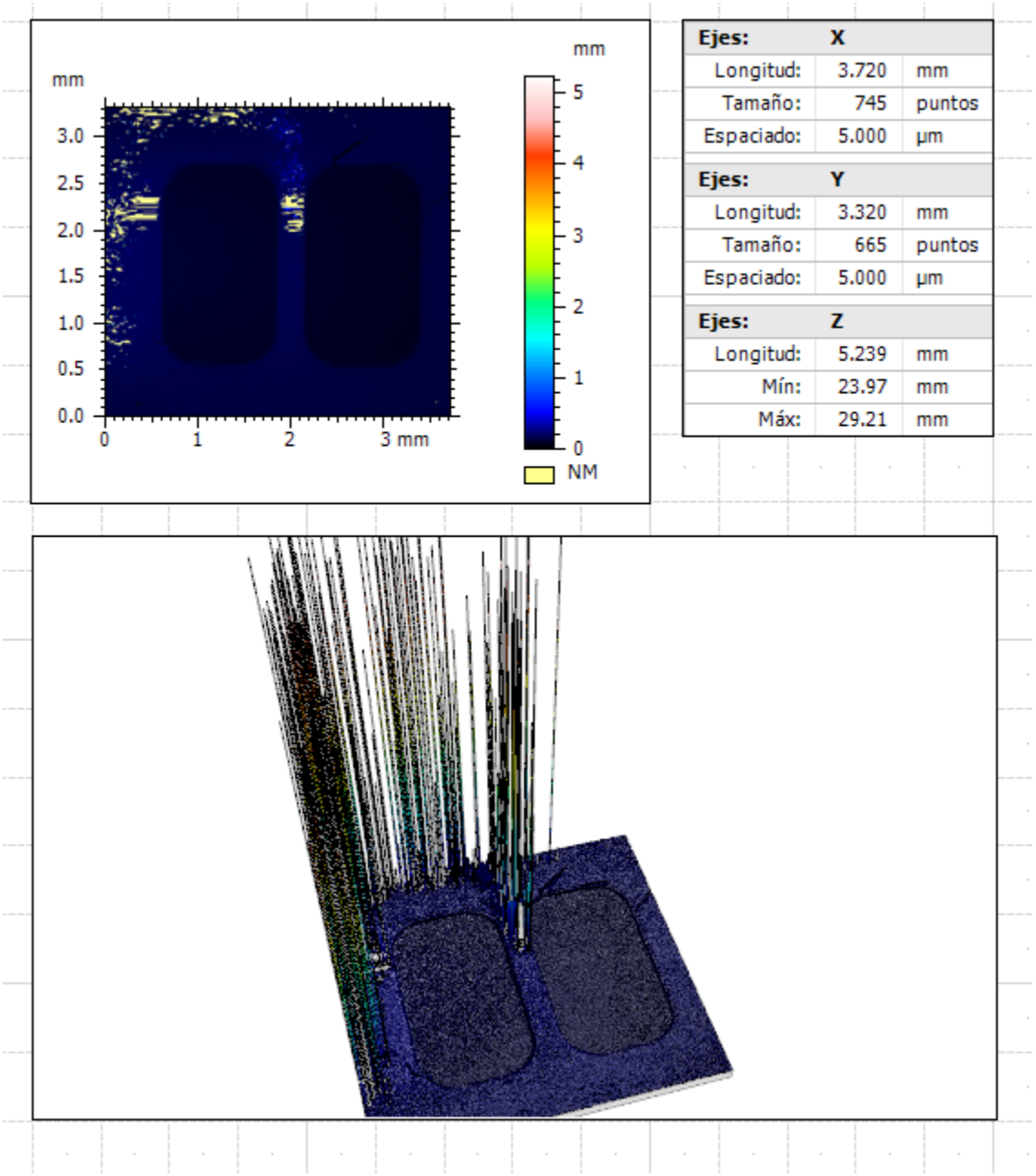



*Ilustración 49: Ejemplo superficie con outliers*

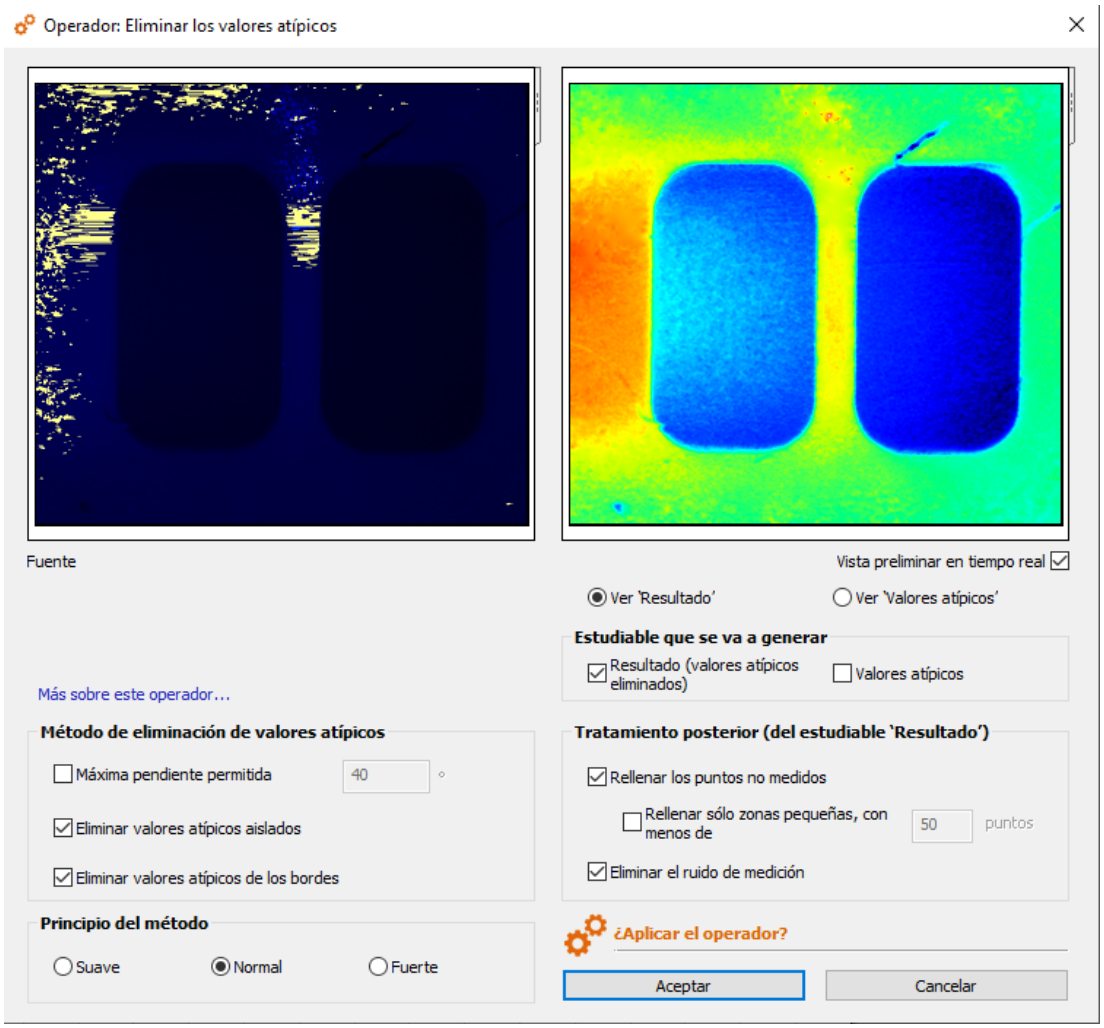



*Ilustración 50: Ejemplo eliminación de valores atípicos*

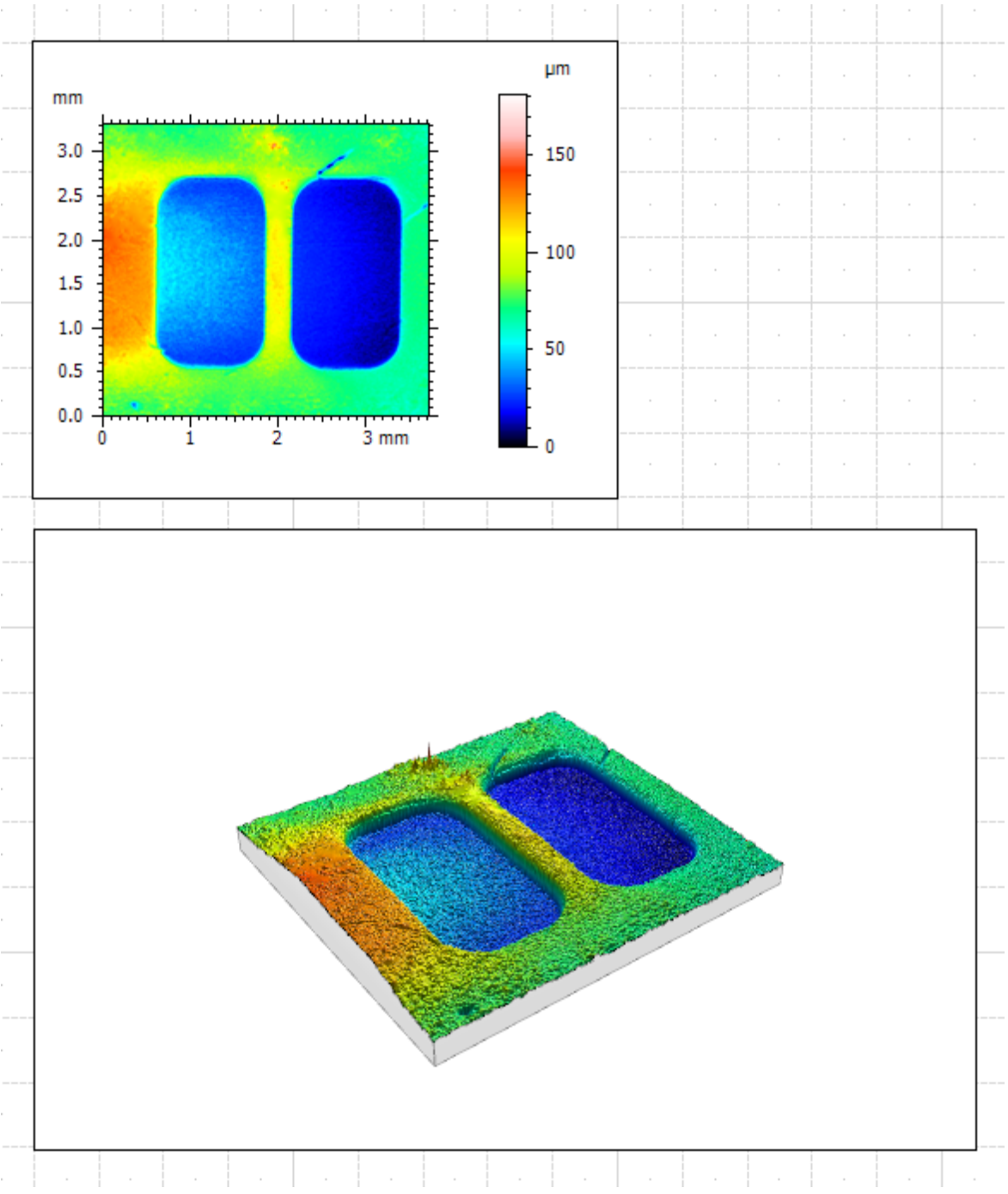



*Ilustración 51: Ejemplo superficie sin valores atípicos*

# 5 Herramienta procesamiento de geometrías 3D

En la sección 4 se evaluó un conjunto de aplicaciones de escritorio para la visualización y transformación de geometrías 3D. Estas herramientas ofrecen una experiencia predefinida en la que el usuario interactúa con la aplicación directamente a través de los elementos de interfaz proporcionados. De esta manera, es posible aplicar transformaciones a una geometría 3D dada desde los menús de la aplicación de escritorio, sin conocimiento alguno de los detalles de implementación de estas.

Utilizando estas herramientas se sacrifica flexibilidad, puesto que no es posible añadir fácilmente rutinas de procesamiento dedicadas. En el caso de las herramientas comerciales, no es posible acceder al código fuente, mientras que en el caso de las alternativas de código abierto, sería posible con un conocimiento avanzado de cada herramienta.

De este aspecto surge la motivación para diseñar e implementar una herramienta software dedicada para el procesamiento y visualización de geometrías 3D, que ofrezca una experiencia de usuario interactiva a partir de una interfaz gráfica de usuario y que facilite la implementación de nuevas rutinas de procesamiento a usuarios programadores.

Los siguientes apartados presentan los requisitos funcionales de la herramienta diseñada, así como un resumen de las tareas realizadas a lo largo del proyecto y una vista de alto nivel de la funcionalidad incluida en la herramienta.

Para un análisis pormenorizado de la funcionalidad incluida en la herramienta desde la perspectiva de un usuario final de la aplicación de escritorio, consultar el documento adjunto **MANUAL DE USUARIO**. Para un análisis detallado de los aspectos de diseño y las principales decisiones de implementación de la herramienta, consultar el documento adjunto **MANUAL DEL PROGRAMADOR**. Este último manual incluye a su vez múltiples consideraciones orientadas a un usuario avanzado para la creación de nuevas rutinas de procesamiento de geometrías 3D en la herramienta.

## 5.1 Consideraciones técnicas

A continuación, se presenta una serie de consideraciones derivadas del análisis de bibliotecas de procesamiento de geometrías 3D realizado anteriormente, que permiten determinar el procedimiento a seguir a la hora de implementar la herramienta desde un punto de vista técnico. En particular, se define la biblioteca de procesamiento de geometrías 3D sobre la que basar la herramienta y la metodología a adoptar para construir una aplicación de escritorio con interfaz gráfica de usuario a partir de la misma.

En la sección 3 se evaluó un conjunto de las bibliotecas de procesamiento de geometrías 3D de código abierto más relevantes en la actualidad, con el objetivo de determinar cual sería la más adecuada para implementar una aplicación de escritorio dedicada.

La mayoría de estas bibliotecas están orientadas al desarrollo de rutinas de procesamiento estáticas y no a flujos interactivos con el usuario. Además, aunque todas ellas ofrecen alguna alternativa para el renderizado y visualización de geometrías 3D, no facilitan la creación de aplicaciones interactivas mediante interfaces gráficas de usuario.

La alternativa, si se quisiera desarrollar una aplicación de escritorio interactiva que utilice estas bibliotecas sería la de su integración con un framework dedicado a la creación de interfaces gráficas de usuario, como PyQt.

No obstante, durante la evaluación de las bibliotecas se identificó la biblioteca Open3D como la más adecuada para el desarrollo de una aplicación de escritorio, siendo la razón principal que esta implementa un módulo para la creación de interfaces gráficas de usuario integrado en la propia biblioteca, cuyas características fueron expuestas en detalle en la sección 3.1.1.

De esta manera, sería posible implementar una aplicación de escritorio interactiva con funcionalidad específica para el procesamiento de geometrías 3D utilizando únicamente la biblioteca Open3D y sus módulos para la creación de interfaces gráficas de usuario. Puesto que Open3D también implementa un módulo dedicado al renderizado de geometrías 3D, sería posible visualizar los cambios en la geometría desde la propia interfaz de la aplicación.

Por lo tanto, se decide implementar la aplicación de escritorio haciendo uso exclusivo de la biblioteca Open3D. Se tratará, en la medida de lo posible, de utilizar la versión más reciente de Open3D, actualizando el código fuente de la aplicación si fuera necesario. Al finalizar el proyecto, la aplicación fue actualizada a la versión más reciente de Open3D (0.13, Junio 2021). Si surge la necesidad específica, se podrá hacer uso de otras bibliotecas del ecosistema de Python para implementar funcionalidades adicionales.

Con respecto a la versión de Python, se utilizará la más reciente soportada por la biblioteca, en este caso Python 3.8.

El código de la aplicación será distribuido junto a la presente memoria, con un manual de instalación dedicado. Se planea a su vez, distribuir la herramienta como una aplicación ejecutable de Windows, lo que eliminaría el proceso de instalación. El proceso de generación del ejecutable a partir de un entorno Python será descrito en detalle en el documento adjunto MANUAL DEL PROGRAMADOR.

## 5.2 Requisitos del sistema

Se plantea, por lo tanto, el desarrollo de una aplicación de escritorio con soporte para la visualización y transformación de geometrías 3D. La aplicación desarrollada, cuyos requisitos funcionales son expuestos en la Tabla 7: Tabla de requisitos, soportará un conjunto diverso de transformaciones de geometrías 3D, lo que posibilitará a su vez la realización de flujos de procesamiento básicos en los que un usuario encadena múltiples transformaciones de manera sucesiva a una misma geometría. Como ejemplo, se plantea un posible escenario que podría ser realizado desde la propia herramienta:

- El usuario importa una geometría de nube de puntos en la herramienta.
- El usuario aplica una función de transformación para eliminar los valores anómalos de la geometría, configurando los parámetros principales de la transformación.
- El usuario aplica una función de transformación para reconstruir la superficie de la nube de puntos, generando una malla triangular.
- El usuario exporta la malla triangular a un fichero independiente.

El conjunto de transformaciones de geometrías incorporado a la herramienta está informado por el análisis funcional de cada herramienta y biblioteca realizados en las secciones 3 y 4, del que se extraen las funciones de mayor utilidad.

La herramienta será diseñada tratando de favorecer posibles extensiones. El objetivo en este aspecto será el de facilitar la implementación de nuevas funciones de transformación de

geometrías 3D, reduciendo al mínimo el conocimiento específico sobre la herramienta, como las particularidades de la interacción con los elementos de interfaz.

Habitualmente, las funciones de transformación requieren de ciertos parámetros de ejecución que modifican el comportamiento de estas. En ciertos casos, resulta interesante que el propio usuario final de la aplicación pueda modificar estos parámetros antes de aplicar la transformación, para adaptarla a las características de la geometría evaluada.

Para posibilitar la modificación de parámetros desde la aplicación, sería necesario crear los elementos de interfaz necesarios para que el usuario introduzca los valores de cada parámetro, además de recogerlos cuando el usuario aplique una transformación. Este es un proceso trabajoso y propenso a errores que se tratará de automatizar en la medida de lo posible, reduciendo al máximo la interacción por parte del usuario programador.

| Id | Requisito | Descripción |
|---|---|---|
| **RF-1** | **Distribución interfaz** | |
| RF-1.1 | Aplicación escritorio | La herramienta se presentará como una aplicación gráfica de escritorio. |
| RF-1.2 | Menú global | La interfaz incluirá un menú superior que permita al usuario acceder a las funciones implementadas. Constará de 2 submenús: uno dará acceso a funcionalidad básica de importado y exportado, mientras que el otro proporcionará acceso a operaciones de transformación de geometrías. |
| RF-1.3 | Panel propiedades visualización | La interfaz incluirá un panel en el lateral izquierdo que permita al usuario interactuar con características de visualización de la aplicación. |
| RF-1.4 | Panel parámetros operaciones | La interfaz incluirá un panel en el lateral derecho que permita modificar los parámetros de las operaciones de transformación de geometrías implementadas. |
| RF-1.5 | Escena principal | La interfaz incluirá un componente que represente una escena 3D en la que visualizar geometrías y navegar libremente. |
| RF-1.6 | Panel información básica geometría | La interfaz mostrará aspectos básicos de la geometría representada en escena (número de vértices, longitud de cada eje…) y será actualizada a medida que el usuario transforme la geometría. |
| **RF-2** | **Funcionalidad básica** | |
| RF-2.1 | Importar nube de puntos | Un usuario podrá importar una nube de puntos a partir de un fichero de geometría. |
| RF-2.2 | Importar malla | Un usuario podrá importar una malla a partir de un fichero de geometría. |
| RF-2.3 | Exportar nube de puntos | Un usuario podrá exportar una nube de puntos en escena a un fichero de geometría. |
| RF-2.4 | Exportar malla | Un usuario podrá exportar una malla en escena a un fichero de geometría. |
| RF-2.5 | Navegación escena 3D | Un usuario podrá alternar entre varios modos de navegación por la escena. |

| | | |
|---|---|---|
| RF-2.6 | Captura de pantalla | Un usuario podrá exportar una captura de pantalla de la escena 3D. |
| **RF-3** | **Características visualización** | |
| RF-3.1 | Tamaño de puntos | El usuario podrá modificar el tamaño de los puntos de una geometría. |
| RF-3.2 | Propiedades de color | El usuario podrá modificar las propiedades de color de la geometría a partir de elementos de interfaz. |
| RF-3.3 | Gradientes de color | El usuario podrá asignar un gradiente de color a una geometría a partir de los valores de sus coordenadas. |
| RF-3.4 | Modos de renderizado | El usuario podrá seleccionar entre múltiples modos de renderizado de una geometría. |
| RF-3.5 | Propiedades material | El usuario podrá modificar propiedades físicas del material de la geometría a partir de elementos de interfaz. |
| RF-3.6 | Perfiles iluminación | El usuario podrá seleccionar entre varios perfiles de iluminación de la escena a partir de elementos de interfaz. |
| RF-3.7 | Foco de iluminación | El usuario podrá modificar la dirección del foco principal de luz de la escena. |
| RF-3.8 | Guardar parámetros visualización | El usuario podrá exportar los parámetros de renderizado a un fichero de texto. |
| RF-3.9 | Cargar parámetros visualización | El usuario podrá importar los parámetros de renderizado a partir de un fichero de texto. |
| RF-3.10 | Refresco de escena | Las modificaciones de los parámetros de renderizado se reflejarán en la escena 3D en tiempo real, a medida que son modificados por el usuario. |
| **RF-4** | **Operaciones geometrías 3D** | |
| RF-4.1 | Filtrado | La aplicación incluirá al menos una operación de filtrado de nubes de puntos. |
| RF-4.2 | Valores anómalos | La aplicación incluirá al menos una operación de eliminación de valores anómalos en nubes de puntos. |
| RF-4.3 | Segmentación de planos | La aplicación incluirá al menos una operación de segmentación de planos en nubes de puntos. |
| RF-4.4 | Clustering | La aplicación incluirá al menos una operación de clustering aplicada a nubes de puntos |
| RF-4.5 | Reconstrucción de superficies | La aplicación incluirá al menos una operación de reconstrucción de superficies en nubes de puntos |
| **RF-5** | **Parametrización operaciones geometrías** | |
| RF-5.1 | Configuración filtrado | El comportamiento de las operaciones de filtrado será configurable por el usuario desde la interfaz de la aplicación. |
| RF-5.2 | Configuración eliminación de valores anómalos | El comportamiento de las operaciones de eliminación de valores anómalos será configurable por el usuario desde la interfaz de la aplicación. |
| RF-5.3 | Configuración segmentación | El comportamiento de las operaciones de segmentación de planos será configurable por el usuario desde la interfaz de la aplicación. |

| | | |
|---|---|---|
| RF-5.4 | Configuración clustering | El comportamiento de las operaciones de clustering será configurable por el usuario desde la interfaz de la aplicación. |
| RF-5.5 | Configuración reconstrucción superficies | El comportamiento de las operaciones de reconstrucción de planos será configurable por el usuario desde la interfaz de la aplicación. |
| **RF-6** | **Cálculo de medidas** | |
| RF-6.1 | Selección de puntos | El usuario podrá seleccionar de puntos de la geometría entre los que realizar medidas. |
| RF-6.2 | Medidas 2 puntos | La aplicación permitirá calcular distancias entre 2 puntos de la geometría. |
| RF-6.3 | Medidas 3 puntos | La aplicación permitirá calcular distancias y ángulos entre 3 puntos de la geometría. |
| RF-6.4 | Visualización medidas | Las medidas calculadas serán representadas en la escena 3D para facilitar su comprensión. |
| RF-6.5 | Resultados medidas | Los resultados de las medidas podrán ser consultados desde la interfaz de la aplicación. |
| **RF-7** | **Aspectos dinámicos** | |
| RF-7.1 | Menú dinámico | El menú de la aplicación será construido de manera dinámica a partir de las operaciones de transformación de geometrías 3D registradas en la aplicación. |
| RF-7.2 | Interfaz parámetros dinámica | Los elementos de interfaz relativos a la parametrización de operaciones serán creados dinámicamente a partir de las operaciones de transformación de geometrías 3D registradas en la aplicación. |
| RF-7.3 | Activación operaciones | El usuario podrá activar desde el menú superior las operaciones de transformación de geometrías registradas. |
| RF-7.4 | Parametrización operaciones | Las operaciones de transformación de geometrías serán invocadas con los valores de cada parámetro configurado por el usuario en la interfaz de la aplicación. |
| RF-7.5 | Control de errores | La aplicación incorporará mecanismos para restaurar el estado previo de una geometría en el evento de una excepción en tiempo de ejecución, notificando al usuario mediante una ventana de diálogo. |
| RF-7.6 | Actualización interfaz | La información de la geometría representada en la aplicación será actualizada a medida que se apliquen operaciones de transformación |

*Tabla 7: Tabla de requisitos*

| Id | Requisito | Descripción |
|---|---|---|
| RNF-1 | Soporte ficheros | Soporte para múltiples ficheros de representación de geometrías 3D (.ply, .pcd) |
| RNF-2 | Rendimiento | La aplicación deberá mostrar un rendimiento aceptable en la visualización y manejo de geometrías 3D complejas. |

| | | |
|---|---|---|
| RNF-3 | Adaptabilidad | La aplicación facilitará la implementación de nuevas funcionalidades. |
| RNF-4 | Distribución ejecutable | Además de mediante el código fuente, la aplicación se distribuirá como una herramienta de escritorio en un archivo ejecutable. |
| RNF-5 | Soporte sistemas operativos | Soporte de múltiples arquitecturas y sistemas operativos, incluyendo MacOS y Windows 10. |
| RNF-6 | Código abierto | La aplicación hará uso exclusivo de tecnologías open-source. |

*Tabla 8: Requisitos no funcionales*

Los requisitos funcionales definidos han sido agrupados según el tipo de objetivo que cubren en el sistema:

- **Distribución interfaz**: Determina los principales elementos de interfaz gráfica de usuario que la herramienta debe presentar, así como la distribución de los mismos.
- **Funcionalidad básica**: Requisitos básicos de la herramienta, como el importado y exportado de geometrías 3D de diferentes tipos.
- **Características visualización**: Requisitos relacionados con elementos de interfaz dedicados a gestionar aspectos relativos a los parámetros de renderizado de la escena, como la iluminación o las propiedades físicas del material que representa la superficie de la geometría 3D. Esta sección también cubre la carga y guardado de los parámetros de renderizado de una escena dada.
- **Operaciones geometrías 3D:** Requisitos asociados a las operaciones de trasformación aplicadas a una geometría 3D (filtrado, eliminación de valores anómalos…).
- **Parametrización operaciones geometrías:** Agrupa los requisitos asociados a la configuración de parámetros de ejecución de las transformaciones definidas desde elementos de interfaz gráfica.
- **Cálculo de medidas:** Agrupa los requisitos asociados con la toma de medidas sobre una geometría 3D representada en la herramienta, así como la visualización de los resultados de las medidas en elementos de interfaz dedicados.
- **Aspectos dinámicos:** Agrupa un conjunto de requisitos dirigidos a mejorar la experiencia de uso desde la perspectiva de un usuario programador. Concentra una serie de automatizaciones que simplifican la extensión de la herramienta con nuevas transformaciones, reduciendo el conocimiento específico de particularidades de la herramienta.

Los requisitos no funcionales recogidos en la tabla agrupan aspectos de usabilidad y rendimiento generales de la herramienta software de escritorio. Los requisitos expuestos en la Tabla 7 pueden ser generalizados en los siguientes casos de uso:

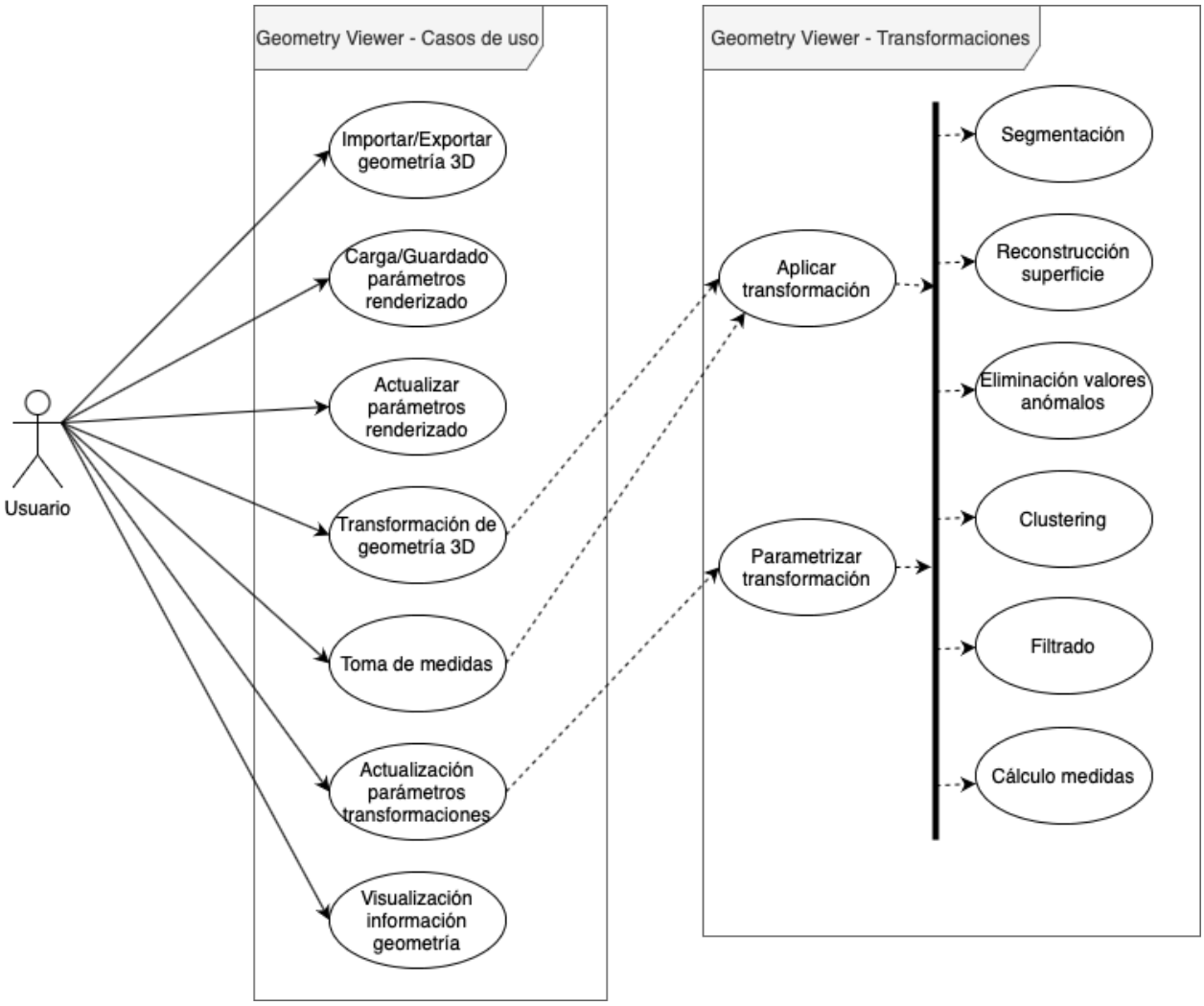



*Ilustración 52: Casos de uso*

Cada caso de uso expuesto en la Ilustración 52 será modelado en la herramienta software a desarrollar como una serie de interacciones entre los principales componentes del sistema. El usuario interactuará directamente con la herramienta desde la interfaz gráfica de usuario, disparando una serie de operaciones que tendrá asociado cada uno de los casos de uso. De esta manera se asegura que la funcionalidad incluida en la herramienta implementada cubre todos los requisitos funcionales definidos.

## 5.3 Solución presentada

A continuación, se presenta una serie de capturas de la aplicación de escritorio desarrollada, resumiendo sus principales características. Para una referencia completa de la funcionalidad incluida, referirse al documento adjunto **MANUAL DE USUARIO**. Alternativamente, el documento adjunto **MANUAL DEL PROGRAMADOR** recoge aspectos de diseño de la herramienta como los principales diagramas de clases, de secuencia y de actividad de las operaciones implementadas.

La Ilustración 53 presenta el aspecto de la aplicación una vez inicializada. En ella se diferencian los principales elementos de interfaz que la componen:

- **Menú superior**: Ofrece acceso directo a todas las funciones implementadas. El submenú *Geometry* incluye funciones básicas como el importado y exportado de geometrías o la carga de parámetros de renderizado, mientras que el submenú *Engine* proporciona acceso a todas las operaciones de transformación de geometrías 3D implementadas.
- **Parámetros de escena**: El panel lateral izquierdo ofrece acceso a múltiples parámetros que gobiernan aspectos de visualización de la geometría representada, como por ejemplo las propiedades del material.
- **Parámetros de transformaciones:** El panel lateral derecho permite al usuario modificar los parámetros de las funciones de transformación de geometrías
- **Escena principal:** La escena principal, representada en gris, será donde se visualicen las geometrías importadas en la herramienta.

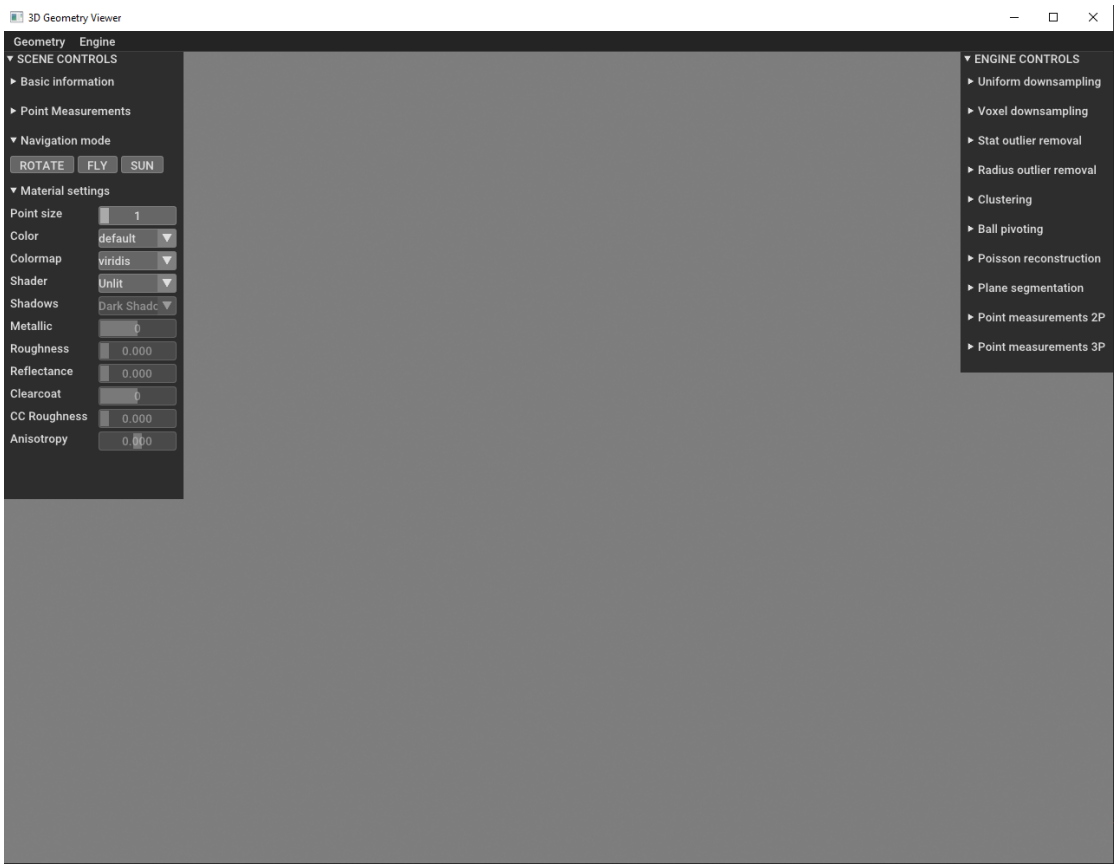



*Ilustración 53: Distribución interfaz*

La Ilustración 54 y la Ilustración 55 demuestran como es posible representar tanto geometrías de nubes de puntos como mallas poligonales. Una vez cargadas en la escena, es posible navegar por la misma, rotando la geometría representada. También es posible modificar el modo de navegación desde la interfaz gráfica de la aplicación.

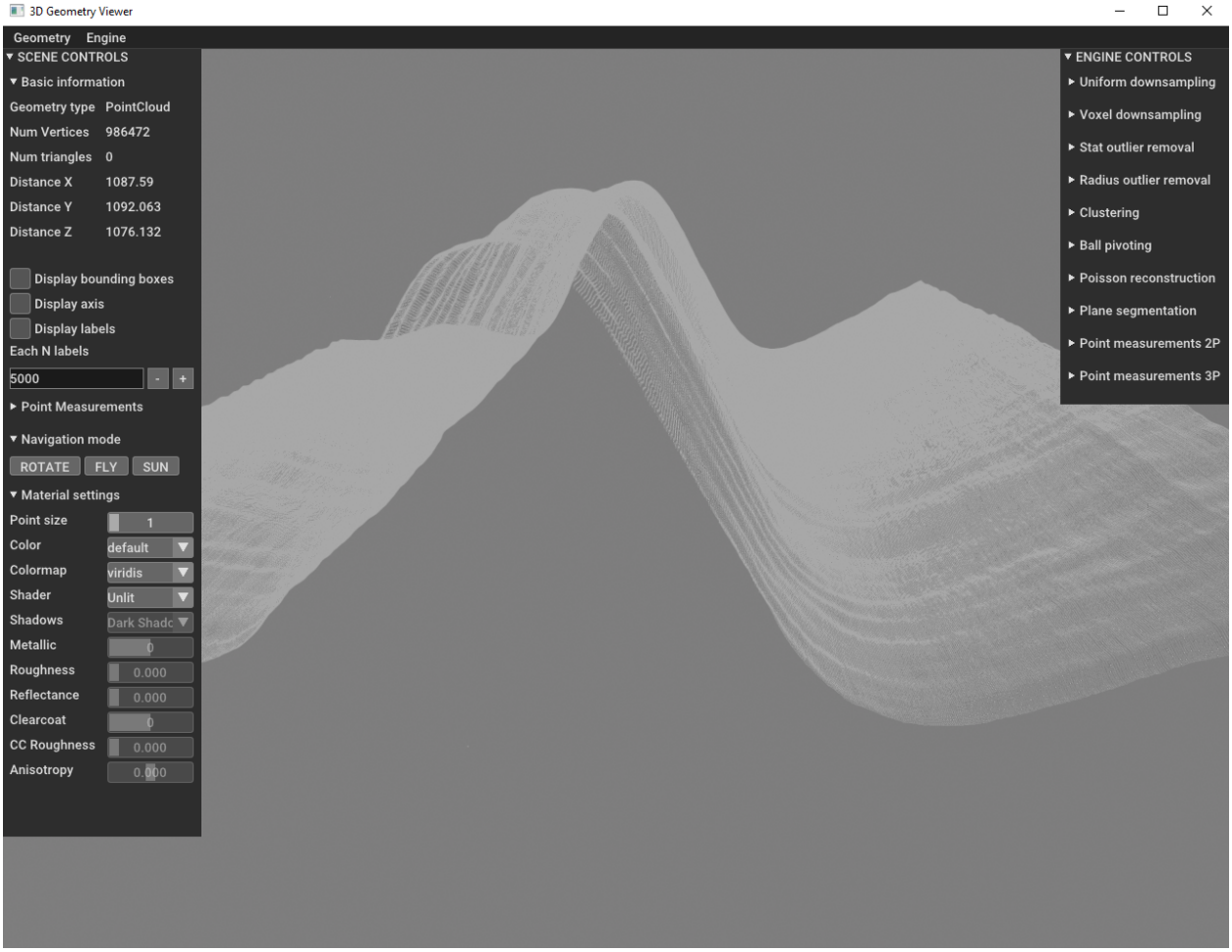



*Ilustración 54: Nube puntos*

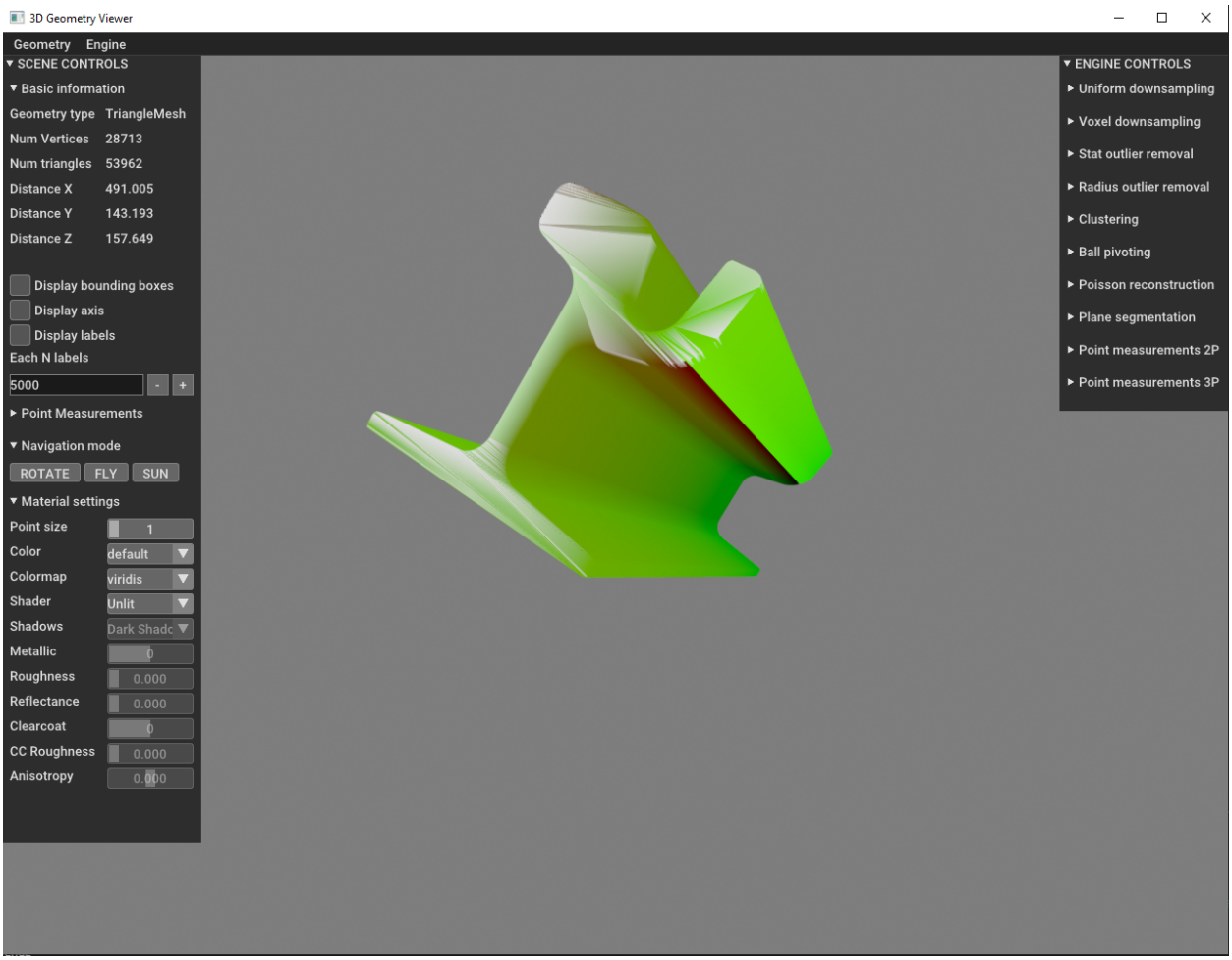



*Ilustración 55: Ejemplo malla*

Las siguientes capturas demuestran como es posible modificar ciertos parámetros de renderizado, modificando la visualización de la geometría representada. En particular la Ilustración 56 representa la geometría asignando colores según las normales de cada vértice de la malla. La Ilustración 57 muestra como es posible configurar las características de iluminación de la escena y las propiedades del material de la geometría para crear configuraciones de renderizado complejas.

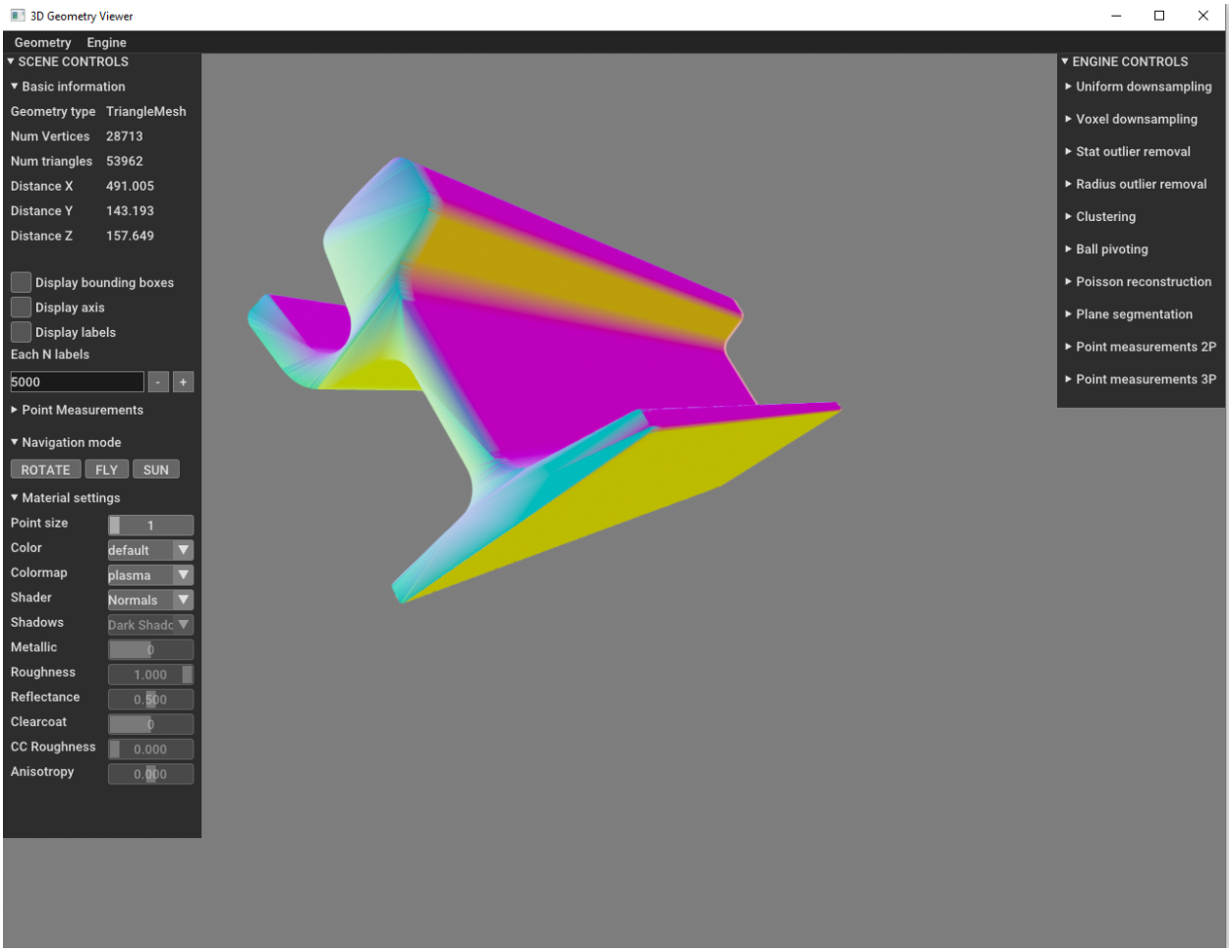



*Ilustración 56: Modo renderizado normales*

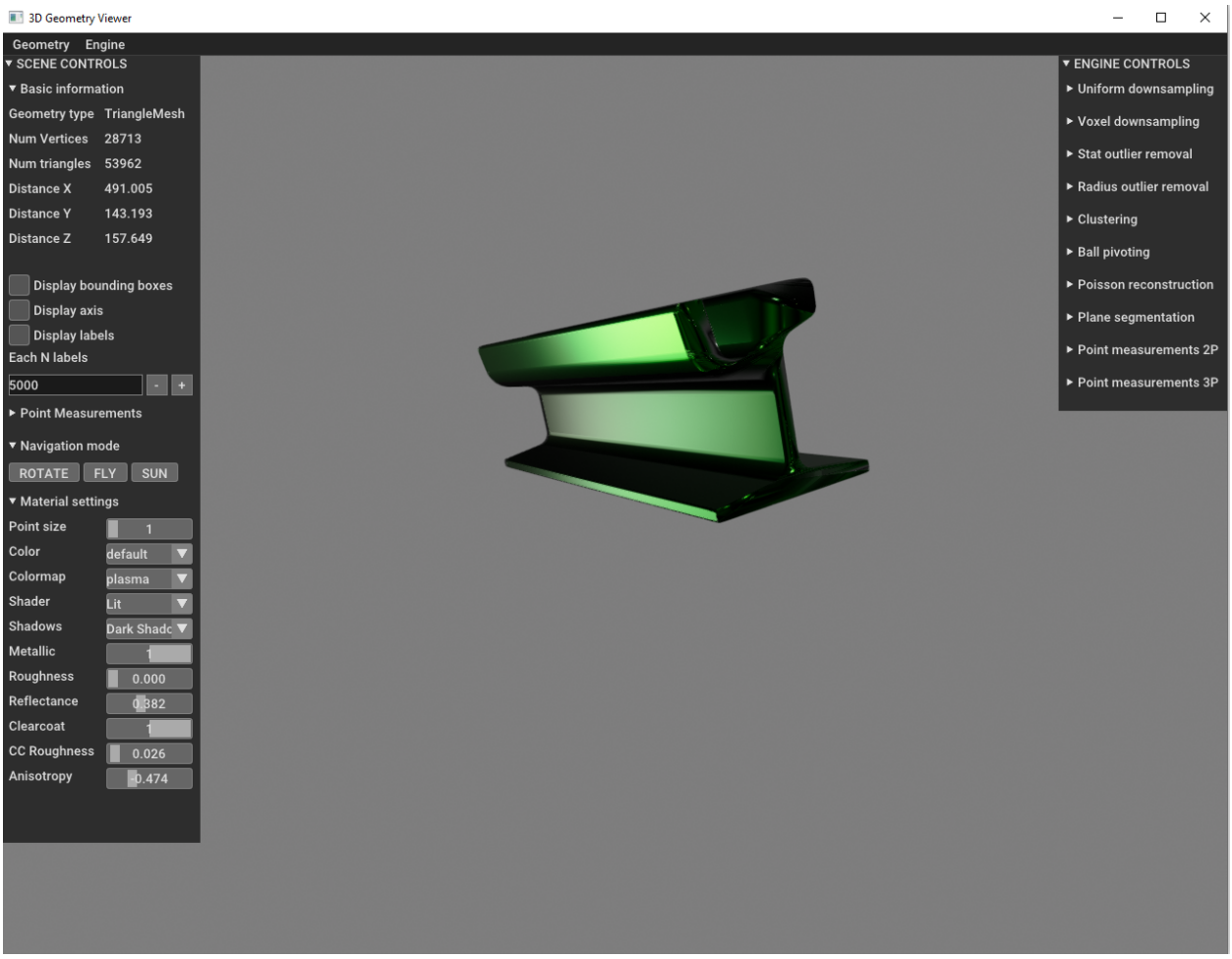



*Ilustración 57: Propiedades material*

La aplicación implementa funcionalidad para guardar los parámetros de renderizado de la aplicación en un fichero de texto plano, que será posible cargar en otra sesión. Esto nos permitirá recuperar la configuración de renderizado de una geometría concreta. La Ilustración 58 muestra un ejemplo del contenido del fichero generado al guardar los parámetros.

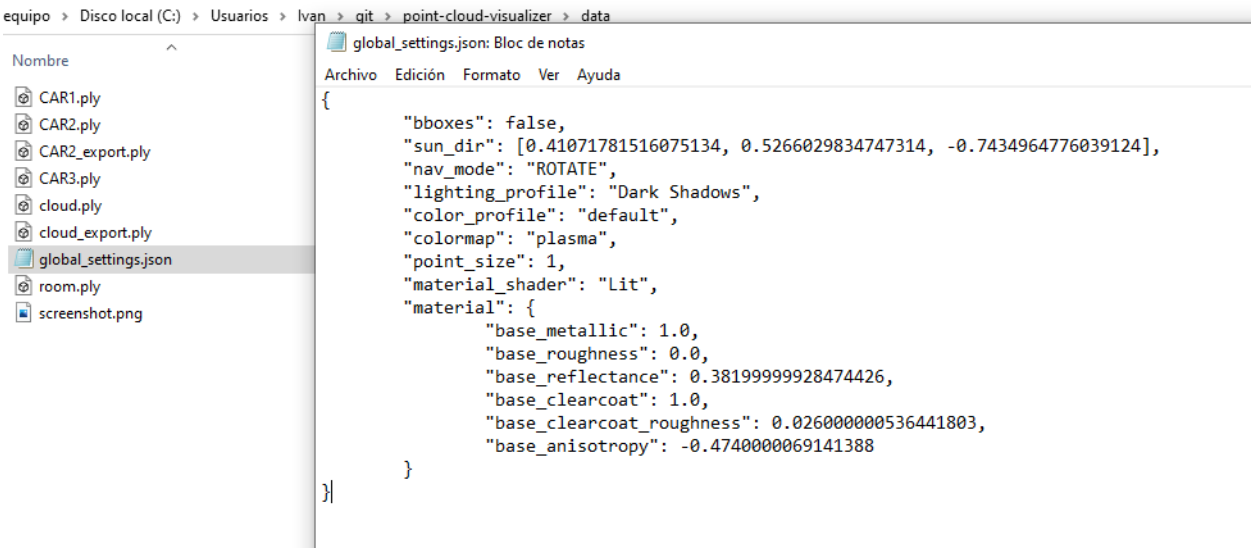



*Ilustración 58: Contenido fichero generado*

La Ilustración 59 muestra como es posible definir puntos de la geometría entre los que realizar medidas básicas de distancias. También demuestra como los puntos y distancias son identificados en la escena 3D. Los resultados de las medidas realizadas son representados en los elementos de interfaz dispuestos en el panel lateral izquierdo de la aplicación.

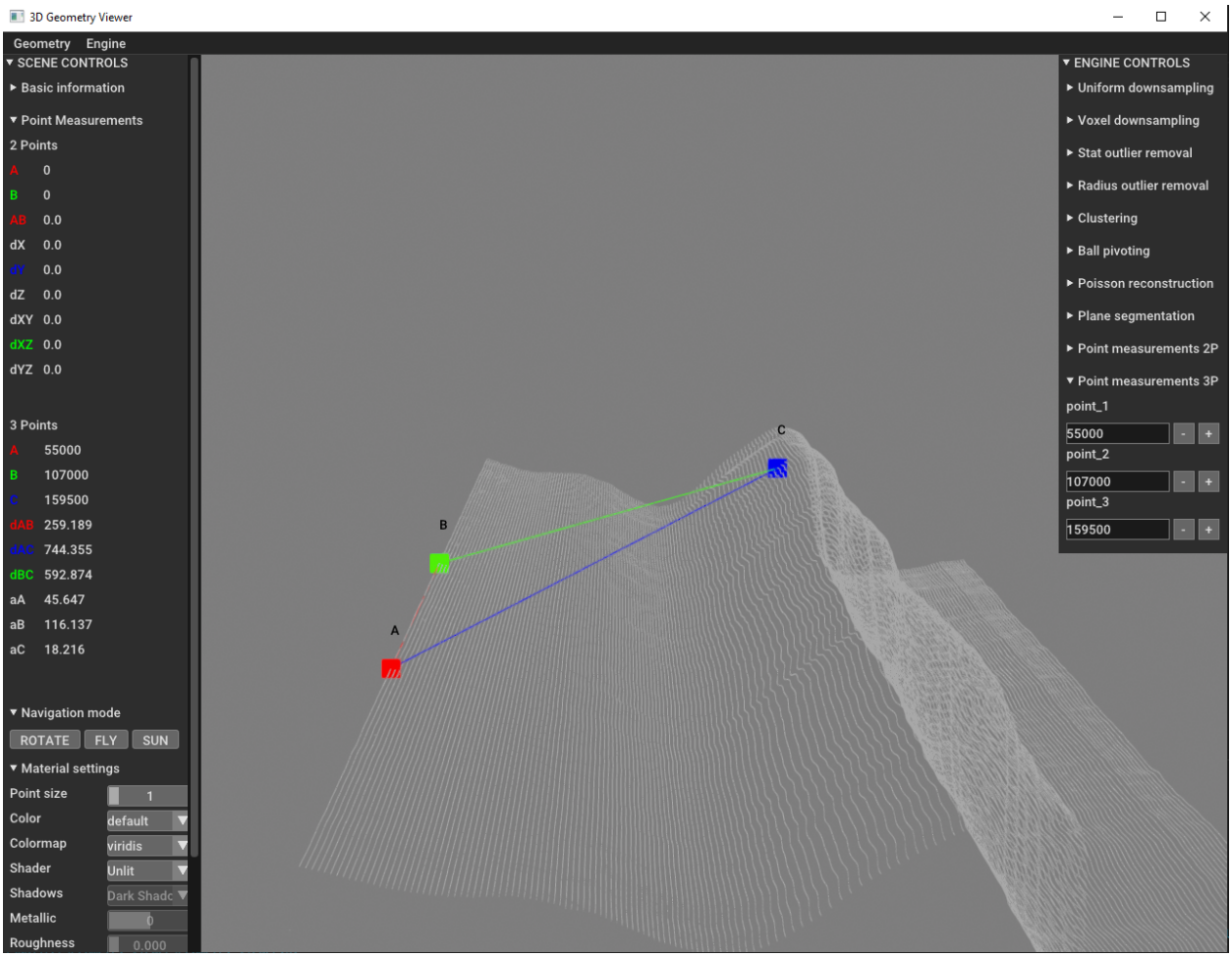



*Ilustración 59: Operación medidas 3 puntos*

La Ilustración 60 muestra como es posible asociar gradientes de color a valores crecientes de un eje de coordenadas seleccionado por el usuario. En el ejemplo planteado se decide colorear la nube de puntos a valores crecientes del eje Z, lo que permite visualizar cambios en su profundidad.

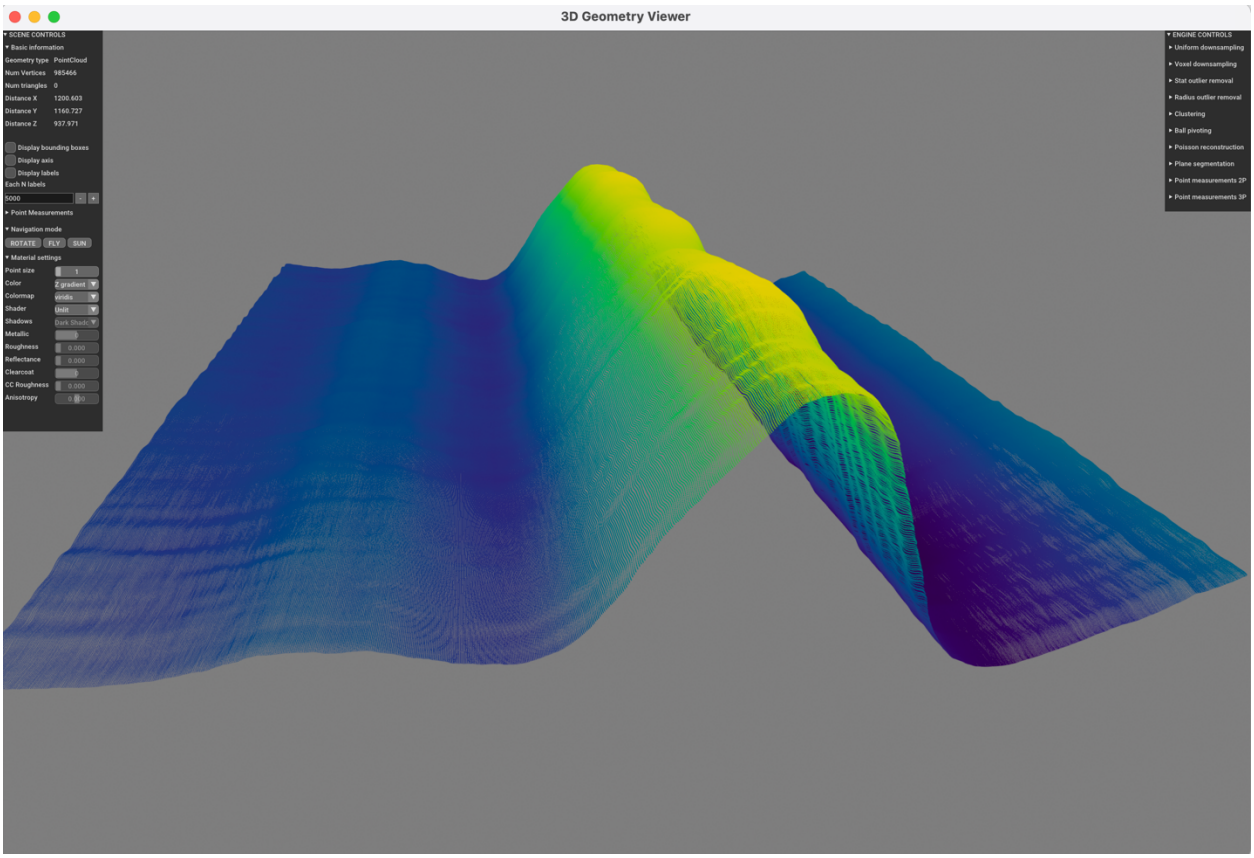



*Ilustración 60: Gradiente eje Z*

A continuación, se presentan una serie de ejemplos de las transformaciones de geometrías 3D incluidas en la herramienta. Para una referencia completa de las mismas y su funcionamiento, referirse al MANUAL DE USUARIO.

La Ilustración 61 muestra como es posible aplicar una rutina de reconstrucción de superficies a una nube de puntos dada. La transformación crea una malla triangular uniforme que une los puntos de la nube de partida. Todas las operaciones de transformación tienen asociado un conjunto de parámetros que pueden ser modificados por el usuario desde el panel lateral derecho de la aplicación. La Ilustración 62 muestra estos parámetros para las dos transformaciones de reconstrucción de superficies incluidas.

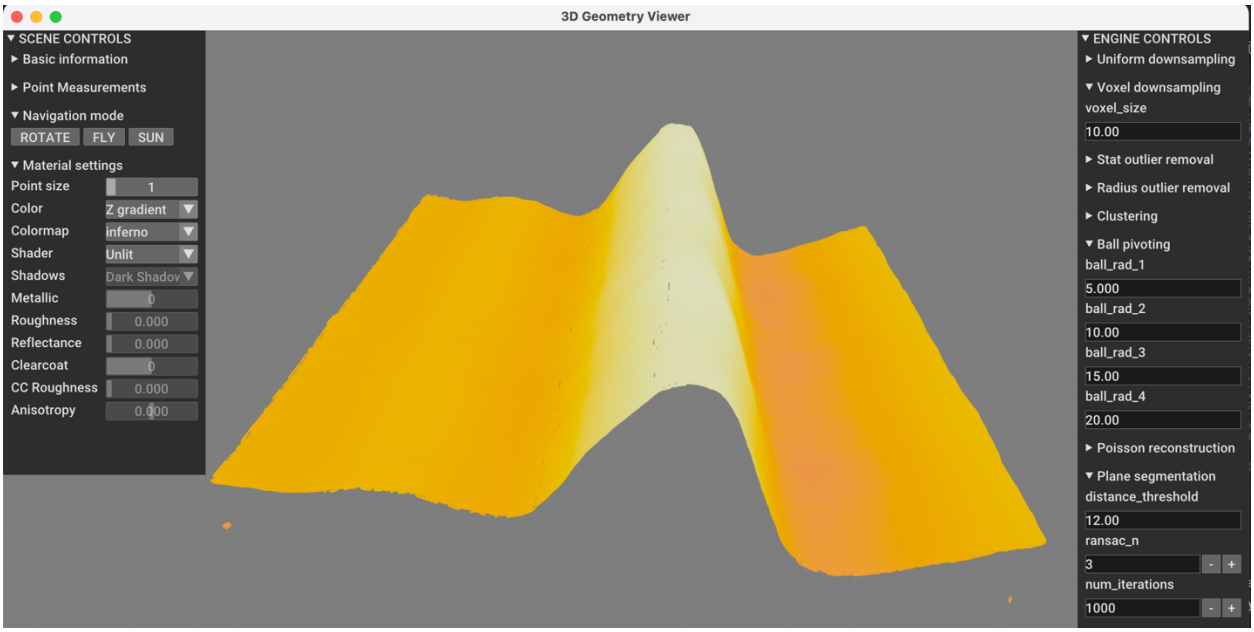



*Ilustración 61: Ejemplo reconstrucción Poisson*

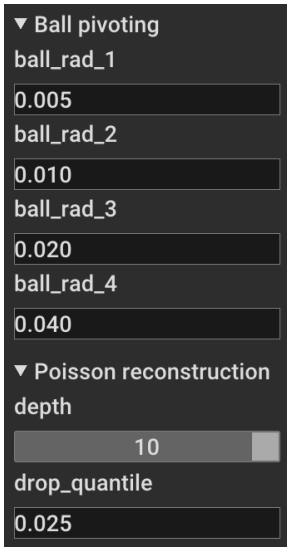



*Ilustración 62: Parámetros reconstrucción de geometrías*

La Ilustración 63 muestra un ejemplo de una operación de filtrado implementada en la herramienta que reduce el número de puntos de la nube de partida de manera uniforme. Al igual que en el resto de transformaciones, estas cuentan con una serie de parámetros de ejecución que pueden ser modificados desde el panel lateral derecho.

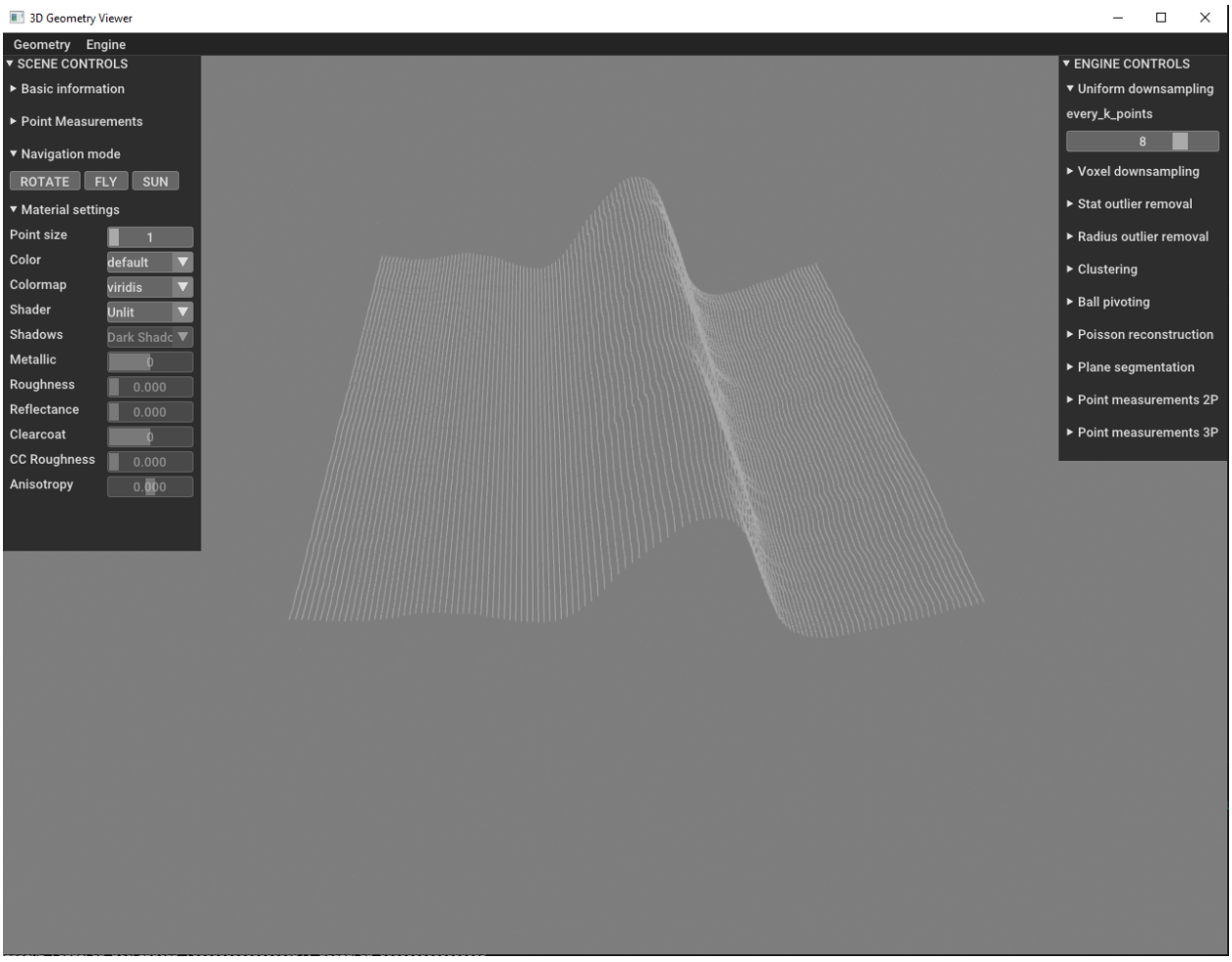



*Ilustración 63: Operación filtrado*

Finalmente, se hace referencia a una serie de aspectos dinámicos que se incluyeron en la herramienta y que simplifican en gran medida la implementación de nuevas rutinas de transformación de geometrías. Estos aspectos no son perceptibles en la interfaz de la aplicación y operan de manera transparente al usuario.

Todos estos elementos están orientados a abstraer al usuario programador de las particularidades de implementación de la herramienta, como aquellos relativos a la interacción con los elementos de interfaz gráfica o el renderizado de la geometría. De esta manera, el usuario que desee implementar una nueva transformación sólo tendrá que preocuparse de la lógica de su función de transformación.

La herramienta implementa un módulo para la creación dinámica de elementos de interfaz, en la que un usuario programador introduce una configuración simplificada que especifica los parámetros requeridos por su función de transformación. La propia herramienta será entonces la encargada de generar los componentes de interfaz necesarios a partir de la configuración.

Cuando un usuario final de la aplicación de escritorio active la transformación desde el menú superior, la aplicación invocará la transformación definida con los valores actuales de cada elemento de interfaz.

El siguiente ejemplo muestra como es posible definir un parámetro requerido por una transformación, definiendo de manera simplificada el elemento de interfaz que lo representa. En particular, se configura un parámetro denominado “every_k_points” que será representado como un slider y que toma un valor por defecto. La Ilustración 64 muestra como este es creado de manera automática y representado en la interfaz.

```
{
      "name": "every_k_points",
      "type": BaseControls.SLIDER,
      "config": {"min": 1, "max": 10, "default": 5}
}
```

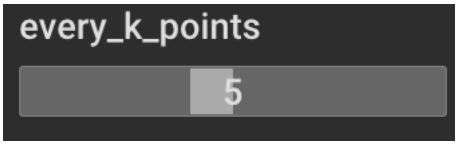



*Ilustración 64: Campo slider*

La Ilustración 65 muestra una captura de la aplicación que recoge todos elementos de interfaz creados de manera dinámica. En ella se puede apreciar como, por ejemplo, una transformación denominada “*Poisson reconstruction*” tiene asociados varios elementos de interfaz, como un slider correspondiente a un parámetro “*depth*” y un campo de texto correspondiente a un parámetro “*drop_quantile*”. Estos elementos son introducidos por el usuario programador mediante una configuración simplificada que es interpretada por la herramienta a la hora de distribuir todos los elementos de interfaz.

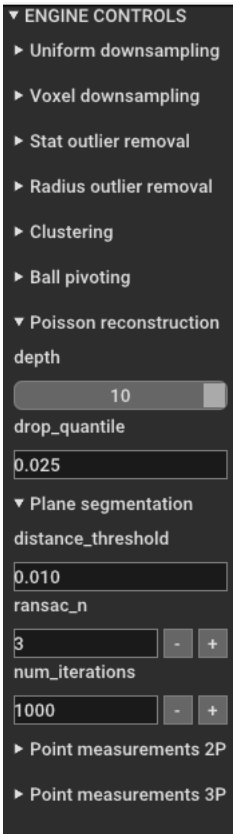



*Ilustración 65: Ejemplo interfaz dinámica*

Para una referencia detallada del funcionamiento de estos mecanismos, referirse al documento adjunto MANUAL DEL PROGRAMADOR.

# 6 Posibles ampliaciones

A continuación, se recoge un conjunto de aspectos identificados durante el desarrollo que podrían ser incorporados en la herramienta para enriquecer su funcionalidad.

**Rutinas en segundo plano**: Todas las operaciones de transformación incorporadas en la herramienta operan en primer plano, lo que en la práctica congela la interfaz de usuario durante la duración de la transformación seleccionada. Esto es particularmente notable en transformaciones complejas que operan en nubes de puntos de gran tamaño.

De manera ideal, las transformaciones deberían ser ejecutadas en un hilo dedicado que notificara al hilo principal cuando finalicen para refrescar la geometría en la escena principal. La documentación de Open3D hace referencia a funciones dedicadas a esta tarea, aunque no se proporcionan ejemplos de uso que tomar como referencia.

Esto también introduciría otros problemas asociados, como la gestión de hilos y el control de las operaciones realizadas por el usuario mientras se ejecutan las transformaciones pendientes en segundo plano. Durante el desarrollo del proyecto se decide no incluir esta funcionalidad debido a la complejidad innecesaria que introduce, teniendo en cuenta el alcance definido para el proyecto.

**Selección de puntos**: La herramienta incluye, en su versión final, una función que permite calcular distancias entre puntos de una geometría. Los puntos entre los que realizar las medidas son introducidos por el usuario de manera manual, a partir de su identificador, que puede ser visualizado habilitando las etiquetas de la geometría.

Este es un proceso innecesariamente complejo que podría ser simplificado si el usuario pudiera seleccionar puntos desde la propia escena, haciendo click en los puntos deseados. Inicialmente, el proceso de selección de puntos se diseñó de esta manera, pero se encontraron una serie de problemas durante la fase de implementación que obligaron a buscar la alternativa incluida finalmente en la herramienta.

Los aspectos implementados en Open3D que facilitan la interacción del usuario con la escena son muy limitados en la versión con la que se desarrolló la herramienta (0.13, Junio 2021), resultando en comportamientos inconsistentes que obligaron a descartar su utilización. Muy posiblemente estos aspectos sean solventados en versiones futuras de la biblioteca. De hecho, en el roadmap definido para la siguiente versión, se incluye soporte para selección de múltiples puntos de una nube definiendo áreas desde la propia escena, lo que invita a pensar que el problema encontrado se solucionará próximamente.

**Visualización web:** La herramienta desarrollada fue diseñada como una aplicación de escritorio desde la fase inicial y así fue presentada. No obstante, sería interesante explorar los nuevos módulos implementados en la versión más reciente de Open3D (Junio 2021) para embeber escenas 3D en navegadores web. Esto permitiría implementar aplicaciones que exploten las rutinas de procesamiento de Open3D y distribuirlas mediante páginas web.

# 7 Conclusiones

El presente proyecto realiza un análisis pormenorizado del ecosistema actual de alternativas software para el procesamiento y visualización de geometrías 3D, introduciendo una nueva referencia en la forma de una aplicación de escritorio portable que facilita la implementación de rutinas de procesamiento de geometrías 3D y que proporciona una interfaz gráfica de usuario para el renderizado y visualización de estas.

En primer lugar, se realiza un estudio de los principales formatos de representación de nubes de puntos y mallas poligonales, exponiendo una referencia de estos y evaluando su eficiencia en la representación de geometrías de distintos tamaños. Este estudio permite determinar el formato de fichero a utilizar a la hora de representar las geometrías 3D que serán utilizadas en el resto del proyecto. Estas geometrías representan datos de superficies y carriles tomadas por cámaras de profundidad y escáneres industriales.

Posteriormente, se evalúan las principales referencias existentes de bibliotecas de bajo nivel para la implementación de rutinas de procesamiento de geometrías 3D, con un objetivo doble. En primer lugar, determinar un conjunto de operaciones de transformación de geometrías comunes que introducir posteriormente en la herramienta propia a desarrollar. En segundo lugar, se trata de identificar la biblioteca de procesamiento 3D más apropiada para basar la herramienta desarrollada. Esta sección está acompañada de un conjunto de pruebas de rendimiento realizadas a las bibliotecas para determinar si cumplen con los requisitos establecidos.

Se dedica un apartado adicional a evaluar un conjunto de herramientas de visualización 3D existentes que ofrecen una experiencia de usuario predefinida e interactiva, en forma de aplicaciones de escritorio. El objetivo fue el de determinar las funciones de transformación de geometrías más habituales que poder incorporar a la herramienta implementada posteriormente, para posibilitar flujos de procesamiento y visualización de geometrías diversos.

A continuación, se introduce una aproximación para el desarrollo de una aplicación de escritorio para la visualización y procesamiento de geometrías 3D. El objetivo principal de la herramienta planteada es el de diseñar una aplicación de escritorio simple dirigida a un público sin conocimientos de programación que facilite la ejecución de flujos de procesamiento y visualización de nubes de puntos de manera interactiva. Esta herramienta toma como base la biblioteca Open3D, de la que se heredan todas las rutinas de procesamiento de geometrías 3D. Alrededor de la misma se construye una interfaz gráfica de usuario que facilita el acceso a aspectos de bajo nivel de Open3D, como las características de iluminación de la escena o las propiedades del material asociado a las geometrías 3D representada.

El aspecto diferenciador de la herramienta con respecto a otras alternativas existentes es que incluye una serie de mecanismos que facilitan la inclusión de nuevas rutinas de procesamiento, abstrayendo al usuario programador de los pormenores de la aplicación, como la gestión y distribución a los elementos de interfaz en la ventana de la aplicación. Las aplicaciones de escritorio existentes dedicadas a la visualización 3D son herramientas complejas que requieren de un conocimiento avanzado si se desea incorporar funcionalidad a la ya incluida de base y normalmente requieren de procesos de compilación complejos si se requiere su reconstrucción.

Habiendo sido implementada en Python, toda la lógica de la herramienta es evaluada en tiempo de ejecución, lo que simplifica su extensibilidad, además de soportar aspectos dinámicos que permiten automatizar múltiples aspectos de la creación de elementos de interfaz gráfica. Estos

aspectos no sacrifican el rendimiento de la herramienta en el procesamiento de geometrías 3D, puesto que Open3D opera en C++ a bajo nivel.

La herramienta desarrollada responde a los objetivos iniciales planteados, resumidos a continuación:

- Dar soporte a escenarios de uso genéricos y flujos de transformación y visualización de geometrías 3D en los que un usuario final aplica un conjunto de transformaciones a una geometría sucesivamente, hasta estar satisfecho con el resultado.
- Facilitar la configuración de aspectos de renderizado de la geometría representada mediante elementos de interfaz gráfica dedicados.
- Implementar una serie de transformaciones de geometrías 3D diverso y configurable desde la propia interfaz de la aplicación.
- Incorporar mecanismos que faciliten la extensión de la herramienta mediante la implementación de nuevas rutinas de transformación de geometrías.

# 8 Referencias

**Especificaciones formato**

Se incluyen a continuación las referencias a la especificación completa de cada formato de representación de nubes de puntos evaluado. En los casos en los que no existe una especificación oficial, se refiere a la web de sus creadores:

Especificación LAS: http://www.asprs.org/wp-content/uploads/2019/07/LAS_1_4_r15.pdf

Especificación PLY: https://web.archive.org/web/20161204152348/http://www.dcs.ed.ac.uk/teaching/cs4/www/graphics/Web/ply.html

Especificación PCD: https://pcl.readthedocs.io/projects/tutorials/en/latest/pcd_file_format.html

**Bibliotecas procesamiento**

Se menciona a continuación, por completitud, cada una de las webs de las bibliotecas evaluadas. En los casos en los que existe, se presenta la referencia al artículo científico donde fueron presentadas.

*Visualization Toolkit, VTK*: https://vtk.org/about/#overview

*Schroeder, Will; Martin, Ken; Lorensen, Bill (2006)*, **The Visualization Toolkit (4th ed.)**, Kitware, ISBN 978-1-930934-19-1

*Potree*: https://github.com/potree/potree

*Markus Schuetz, (2016)*, **Potree: Rendering Large Point Clouds in Web Browsers.** (Thesis: https://www.cg.tuwien.ac.at/research/publications/2016/SCHUETZ-2016-POT/SCHUETZ-2016-POT-thesis.pdf)

*Open3D*: https://github.com/IntelVCL/Open3D

*Qian-Yi Zhou, Jaesik Park, Vladlen Koltun, (2018)*. **Open3D: A Modern Library for 3D Data Processing**. https://arxiv.org/abs/1801.09847

*PyVista*: https://github.com/pyvista/pyvista

*Sullivan and Kaszynski, (2019),* **PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)**. Journal of Open Source Software, 4(37), 1450, https://doi.org/10.21105/joss.01450

*Documentación PyVista*: https://docs.pyvista.org/index.html

*Cilantro*: https://github.com/kzampog/cilantro

*Documentación Cilantro*: https://cilantro.readthedocs.io/en/latest/?badge=latest

*Konstantinos Zampogiannis et al., (2018)*, **cilantro: A Lean, Versatile, and Efficient Library for Point Cloud Data Processing**. https://arxiv.org/abs/1807.00399

*Point Cloud Library (PCL)*: https://pointclouds.org

*R. B. Rusu and S. Cousins, (2011),* **3D is here: Point Cloud Library (PCL)**, IEEE International Conference on Robotics and Automation, 2011, pp. 1-4, doi: 10.1109/ICRA.2011.5980567

*Paraview*: https://www.paraview.org

*Ahrens, James, Geveci, Berk, Law, Charles*, (2005), **ParaView: An End-User Tool for Large Data Visualization**, Visualization Handbook, Elsevier, 2005, ISBN-13: 978-0123875822

*Ayachit, Utkarsh*, (2015), **The ParaView Guide: A Parallel Visualization Application**, Kitware, 2015, ISBN 978-1930934306

*Paraview Point Cloud*: https://www.paraview.org/lidar/

*ParaView PCL*: https://www.paraview.org/Wiki/ParaView/PCL_Plugin

*MeshLab*: https://www.meshlab.net

*P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia*, (2008), **MeshLab: an Open-Source Mesh Processing Tool**. Sixth Eurographics Italian Chapter Conference, page 129-136, 2008

*Mountains*: https://www.digitalsurf.com

*Mountains video tutorials*: https://www.digitalsurf.com/learning/video-tutorials/

**Otros**

A continuación, se incluyen referencias a otras herramientas software utilizadas en la elaboración del proyecto, además de artículos científicos consultados.

*Physically Based Rendering in Filament*: https://google.github.io/filament/Filament.html

*Filament Materials Guide*: https://google.github.io/filament/Materials.html

*Filament Material Properties*: https://google.github.io/filament/Material%20Properties.pdf

*Surface metrology guide*: https://guide.digitalsurf.com/en/guide.html

*Hänsch et al.*, (2014). **Comparison of 3D Interest Point Detectors and Descriptors for Point Cloud Fusion**

*ImGui*: https://github.com/ocornut/imgui

*Three.js*: https://threejs.org

*Documentación Three.js*: https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene

*OpenGL Profiler*: https://developer.apple.com/library/archive/documentation/GraphicsImaging/Conceptual/OpenGLProfilerUserGuide/Introduction/Introduction.html

*PyQt*: https://riverbankcomputing.com/software/pyqt/intro