Universidad Carlos III de Madrid

Reconstrucción de interiores basada en proyecciones de planos

Proyecto fin de carrera ingeniería industrial superior

Autor: Javier Moreno García

Tutor: Javier V. Gómez González

Tutor: David Álvarez Sánchez

Título: Reconstrucción de interiores basada en proyecciones de planos.

Autor: Javier Moreno García

Tutor: Javier V. Gómez González

EL TRIBUNAL

Presidente: _

Vocal: _

Secretario: _

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día _ _ _ de _ _ _ _ _ _ _ _ _ _20 _ _ _ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

En la actualidad, las aplicaciones de la robótica están evolucionando de tal forma que requieren de sistemas capaces de percibir el entorno cada vez más sofisticados. Estos sistemas han de poder de recoger datos del entorno y procesarlos de forma eficaz para que se puedan extraer de ellos la información requerida para la aplicación.

Uno de los sistemas de percepción más utilizados en la robótica son los sistemas de captura por láser 3D. Estos dispositivos capturan medidas de distancia del entorno mediante haces de luz láser y las almacena en listados de miles de puntos cartesianos.

El objetivo de este proyecto es la creación de una metodología capaz de extraer, completar e interpretar la información necesaria para formar un modelo geométrico de un recinto cerrado a partir de los datos generados por un dispositivo láser 3D.

Abstract

Nowadays, robotic applications are evolving so that they require increasingly sophisticated systems capable of perceiving the environment. These systems must be able to collect and process environmental data efficiently so that you can draw from them the information required for the application.

One of the mostly used sensing systems used in robotic systems are 3D laser capture. These devices capture measure distances using laser beams and store them in thousands of Cartesian points.

The objective of this project is to create a methodology able to draw, complete and interpret information to form a closed geometric model from data generated by a 3D laser device.

Índice

Pag.

1. INTRODUCCIÓN

Cada vez nos dirigimos más hacia un mundo donde las aplicaciones de robótica están más presentes en el día a día. Uno de los problemas con los que tropieza la robótica es la forma que se tiene de percibir el entorno no estructurado. Durante los últimos 20 años, se ha ido recorriendo un largo camino, desde simples sensores de rango basados en el sonar o infra rojos (IR) que proporciona unos pocos bytes de información sobre el mundo, a las cámaras omnidireccionales de los escáneres láser.

En los últimos años, los sensores como el *LIDAR Velodyne* [1], mostrado en la *figura 1.1*, nos han proporcionado nubes de puntos tridimensionales que suponen una representación del mundo exterior de alta calidad. Desafortunadamente, estos sistemas son caros, con lo que han quedo fuera del alcance de muchos proyectos de robótica.

Figura 1.1. Sensor de hilado LIDAR Velodyne.
Imagen cortesía de: http://velodynelidar.com/lidar/lidar.aspx

Sin embargo, muy recientemente, los sensores 3D han empezado a estar disponibles por un coste más moderado. Por ejemplo, el sensor *Kinect* [2] para el sistema de juego de Microsoft Xbox 360, basado en la tecnología subyacente de *PrimeSense* [3], se puede comprar por menos de $ 150, y proporciona nubes de puntos en tiempo real, así como imágenes 2D.

Figura 1.2. Sensor Kinect.

Actualmente, existen proyectos tales como Barcelona Robot Lab Dataset [4] o el KITTI Vision Benchmark [5] que a través de este tipo de tecnología tratan de obtener representaciones tridimensionales del entorno. Este tipo de proyectos han conseguido grandes progresos en este campo debido a la creciente mejora de las capacidades de captura de este tipo de sensores.

Como resultado, podemos esperar que la mayoría de los robots en el futuro sean capaces de "ver" el mundo en 3D. Todo lo que se necesita entonces es un mecanismo para el manejo y la interpretación de las nubes de puntos de manera eficiente, y ahí es donde entra en juego la idea de este proyecto.

1.1. **Algoritmos de reconstrucción tridimensional existentes**

La reconstrucción tridimensional es el proceso mediante el cual objetos o espacios tridimensionales capturados por sensores 3D, son procesados en la memoria de una computadora para obtener una serie de características físicas (dimensiones, volumen y forma). Existen multitud de técnicas de reconstrucción cuyo objetivo principal es obtener información concreta de un conjunto de puntos representativos del objeto o espacio.

Los algoritmos desarrollados hasta el momento, se debaten entre el coste computacional y la calidad del mallado obtenido. Un ejemplo de algoritmo de este tipo es la "reconstrucción de un entorno mediante triangulaciones de puntos". A priori, estos algoritmos tratan de obtener la denominada matriz de conexiones. Esta matriz, almacena los puntos del conjunto inicial que deben estar conectados entre sí. Si empleamos triángulos (método bastante común), esta matriz tiene la forma de 3x3xn (siendo "n" el número total de triángulos que contiene la pieza), es decir que cada fila de la matriz representa un triángulo en el plano o en el espacio. En la *figura 1.3* se observa una reconstrucción mediante triangulación de puntos. Independientemente del método que se utilice para generar las aristas de los triángulos de reconstrucción, una vez trazadas todas, podrá observarse que la figura generada puede descomponerse siempre en triángulos.

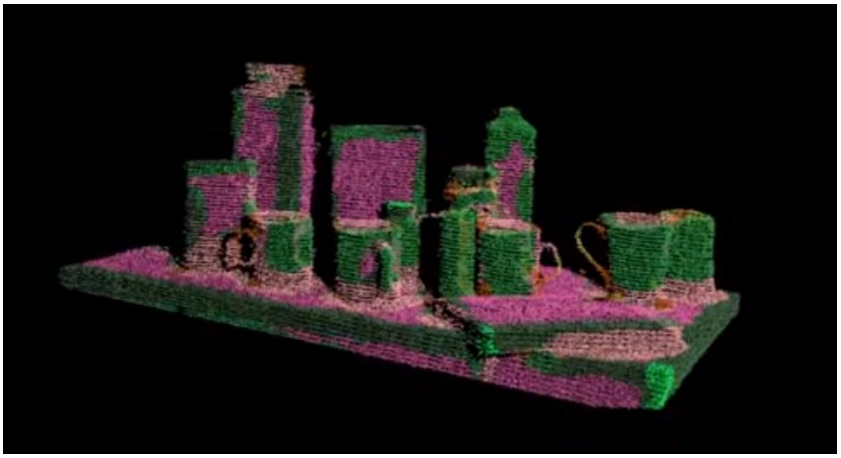


Figura 1.3. Reconstrucción mediante triangulación de puntos.
Imagen cortesía de: http://pointclouds.org/

La eficiencia de este tipo de algoritmos es la que define la calidad final del mallado. Si suponemos un conjunto de puntos mal representado, existirán puntos definidos que no cumplan las condiciones óptimas para el mallado. Los puntos que se encuentran muy cercanos entre sí, los puntos ruidosos y los puntos redundantes, no ofrecen ninguna información para la reconstrucción. Imaginemos por ejemplo, que si queremos representar un cubo en el espacio, simplemente con ocho puntos y doce triángulos serían suficientes, el resto de la información sería redundante.

Otro ejemplo de algoritmo de reconstrucción tridimensional es la "Reconstrucción mediante la obtención de bordes basada en DoN (Difference of normal)". Este tipo de reconstrucción proporciona un enfoque computacionalmente eficiente para el procesamiento de grandes nubes de puntos 3D. La idea es muy simple en concepto, y sin embargo, sorprendentemente eficaz en la segmentación de escenas con diferentes escalas de detalle. La idea básica de este algoritmo es que para cada punto de una nube se calculen dos normales, una para un entorno de puntos pequeño y otra en un entorno más grande. Si la diferencia entre estas dos normales es significativa, se considerara este punto como un cambio de tendencia de la superficie, un borde. Con este tipo de procesamiento de datos se conseguirá filtrar los puntos obtenidos quedándose únicamente con una información referente a los bordes. En la *figura 1.4* se muestra cómo mediante el método DoN se consigue reducir los datos de la nube únicamente a la información de los bordes.

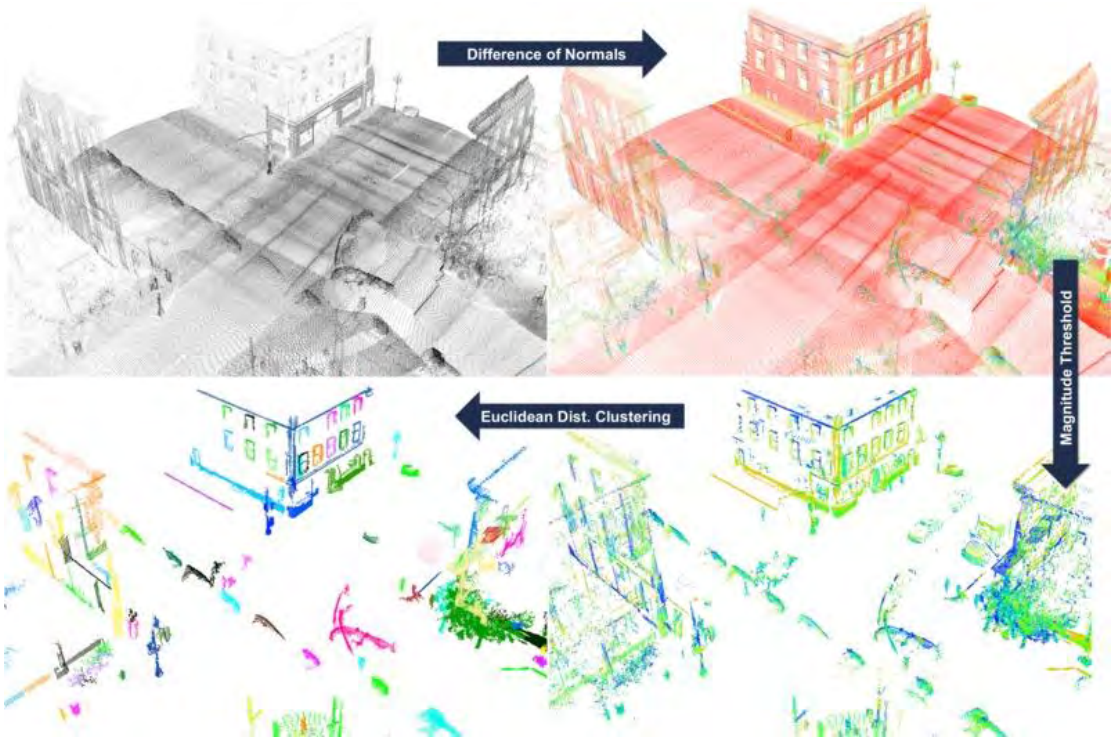


Figura1. 4. Obtención de bordes mediante método DoN.
Imagen cortesía de: http://pointclouds.org/

1.2. **Objetivo general del proyecto**

El objetivo de este proyecto es la creación de un algoritmo de reconstrucción tridimensional, para establecer un modelo geométrico a partir de una nube de puntos de un recinto cerrado, que típicamente puede ser una habitación o sala cerrada de dimensiones medias. Este modelo será capaz de recopilar los datos necesarios para reconstruir las paredes, el techo y el suelo del recinto.

El modelo geométrico se establecerá a partir de los datos recopilados por un equipo láser 3D, que a través de medidas radiales de distancia, permitirán disponer de un conjunto de puntos que definen con gran exactitud el interior del recinto.

El núcleo de este proyecto es el algoritmo que permite el procesado de la nube de puntos, cuya información está sesgada y distorsionada por las obstrucciones causadas por los obstáculos que quedan fuera del modelo y a otros errores de la medida. Este algoritmo se empleará para extraer de la nube información que permita la reconstrucción del modelo del recinto cerrado, ignorando el resto de la información contenida en la nube de puntos y teniendo en cuenta errores asociados al procedimiento de obtención de los datos.

Con los algoritmos existentes no se consiguen reconstruir de forma eficiente las dimensiones del interior de un recinto cerrado, ya que no están construidos para tal fin. Utilizando "la reconstrucción tridimensional por triangulación" se tendrían en cuenta todos los objetos capturados por el láser por lo que no se podría diferenciar entre los diferentes objetos del recinto cerrado. Por otro lado, en la reconstrucción mediante el método "DoN" no se obtendría una representación exacta de las intersecciones del recinto cerrado debido a que el número de puntos que dan información sobre la pared es reducido.

Los objetivos disgregados para este proyecto son:

- Eliminar toda la información innecesaria de la nube.
- Posibilidad de reconstrucción de los planos que forman el recinto cerrado
- Marcar las fronteras del recinto cerrado.
- Utilizar herramientas "open source" para el procesado de datos.
- Exportación de la información obtenida a programas de uso común tales como "AutoCAD", "3ds Max", "Draftsight", o "Blender".

1.3. Utilidad practica

Este proyecto puede servir como punto de partida para elaboración de herramientas de medida de espacios tridimensionales, los cuales pueden aplicarse en campos tales como la arquitectura, la climatización de una sala, el análisis de acústico de un recinto cerrado, etc.

1.4. Estructuración de la memoria

La memoria está dividida en tres partes. Estas partes son; obtención de datos, descripción del algoritmo teórico desarrollado y resultados obtenidos tras la ejecución del algoritmo diseñado en un software construido en C++.

Inicialmente se comienza por la obtención de datos. Aquí se explica el tipo de datos con el que se va a tratar, el procedimiento de captura y el procedimiento de almacenamiento. También se procura una descripción del dispositivo de captura utilizado en el proyecto así como el funcionamiento de éste.

Después se procede a explicar el algoritmo diseñado para el procesamiento de los datos. En él se procuran explicaciones acompañadas de imágenes para facilitar la comprensión. Se explican también los procesos realizados dentro del algoritmo principal acompañados de sus correspondientes referencias.

Por último se muestran los resultados obtenidos tras la ejecución del programa diseñado en C++ aplicado sobre varias muestras tomadas. En este apartado se muestran imágenes de las reconstrucciones obtenidas mostrando los resultados de reconstrucción. También se muestran los resultados exportados a diferentes formatos soportados por softwares de diseño gráfico. El apartado finaliza explicando con detalle cómo, con los resultados obtenidos, se cumplen los objetivos del proyecto.

2. DATOS DE PARTIDA

Los datos de partida se generan gracias a un equipo láser 3D diseñado en el proyecto "Mapeado tridimensional con escáner láser integrado en ROS [6]". El equipo captura medidas de distancias puntuales del entorno que lo rodea, y almacena estas medidas como coordenadas cartesianas mediante un proceso de conversión. Este conjunto de puntos cartesianos se denomina como nube de puntos. Se puede encontrar una descripción más detallada del equipo láser utilizado en el *ANEXO I.*

La nube de puntos capturada por el dispositivo láser 3D, el cual proporciona una descripción el entorno, se almacena en un formato denominado "pcd". Este formato, específico de las librerías "PCL", fue diseñado para complementar deficiencias de formatos de archivos existentes, que por una razón u otra, no son compatibles con algunas de las extensiones que "PCL" aporta para el procesamiento de nubes de puntos. En la *figura 2.1* se observa un entorno descrito mediante una nube de puntos.

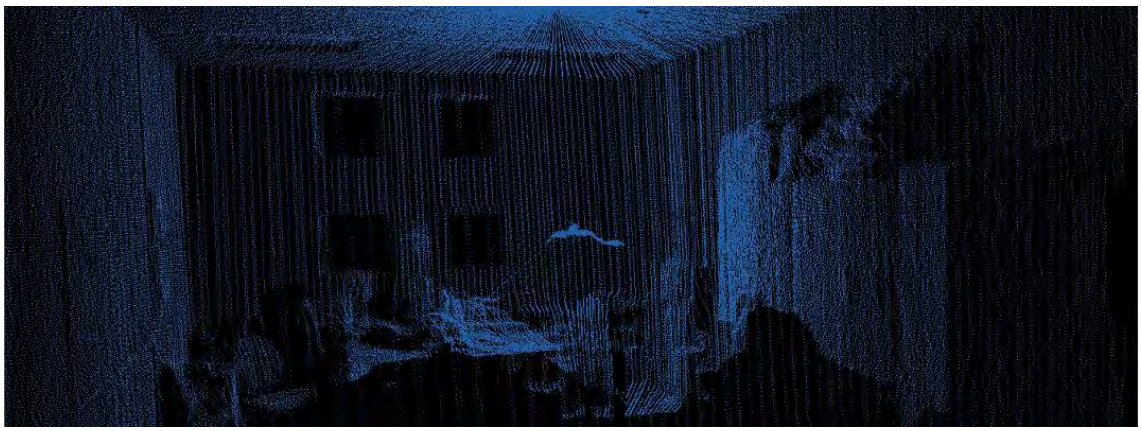


Figura 2.1. Descripción de un entorno mediante una nube de puntos.

Cada archivo de este formato contiene un encabezado que identifica y declara ciertas propiedades de los datos de nubes de puntos almacenados en él. En la *figura 2.2* se muestra un ejemplo de cabecera de un archivo ".pcd".

```
1  # .PCD v0.7 - Point Cloud Data file form
2  VERSION 0.7
3  FIELDS x y z
4  SIZE 4 4 4
5  TYPE F F F
6  COUNT 1 1 1
7  WIDTH 4857
8  HEIGHT 1
9  VIEWPOINT 0 0 0 1 0 0 0
10 POINTS 4857
11 DATA ascii
12 -0.91847998 -0.43101001 -1.5913
13 -0.93092 -0.42414001 -1.5908
14 -0.92878002 -0.42473999 -1.5931
15 -0.94810998 -0.43985 -1.5801001
16 -0.94328666 -0.43232664 -1.5755
17 -0.90842998 -0.43718001 -1.5705
18 -0.93958002 -0.42805001 -1.5741
19 -0.89754498 -0.44069499 -1.56795
20 -0.91758502 -0.434275 -1.5631001
21 -0.898 -0.43533999 -1.5639
22 -0.93269002 -0.42789999 -1.5615
23 -0.91104001 -0.435 -1.5506999
24 -0.89276505 -0.432735 -1.5546
25 ...
```

Figura 2.2. Ejemplo de formato de archivo ".pcd".

Con lo que respecta a la calidad de información obtenida por el dispositivo láser 3D utilizado, debido al diseñado de éste, las nubes no son homogéneas en cuanto a la densidad de puntos se refiere. En la *figura 2.3 y figura 2.4* se observan estas diferencias de densidad, diferencias que hay que tener en cuenta a la hora del procesamiento de los datos. En la *figura 2.3* se puede observar que el número de puntos capturados por unidad de superficie en la zona superior e inferior al dispositivo láser 3D es mayor que en zonas laterales. En la *figura 2.4* se puede observar que el número de puntos capturados por unidad de superficie en zonas más cercanas a dispositivo láser 3D es mayor que en zonas más alejadas.

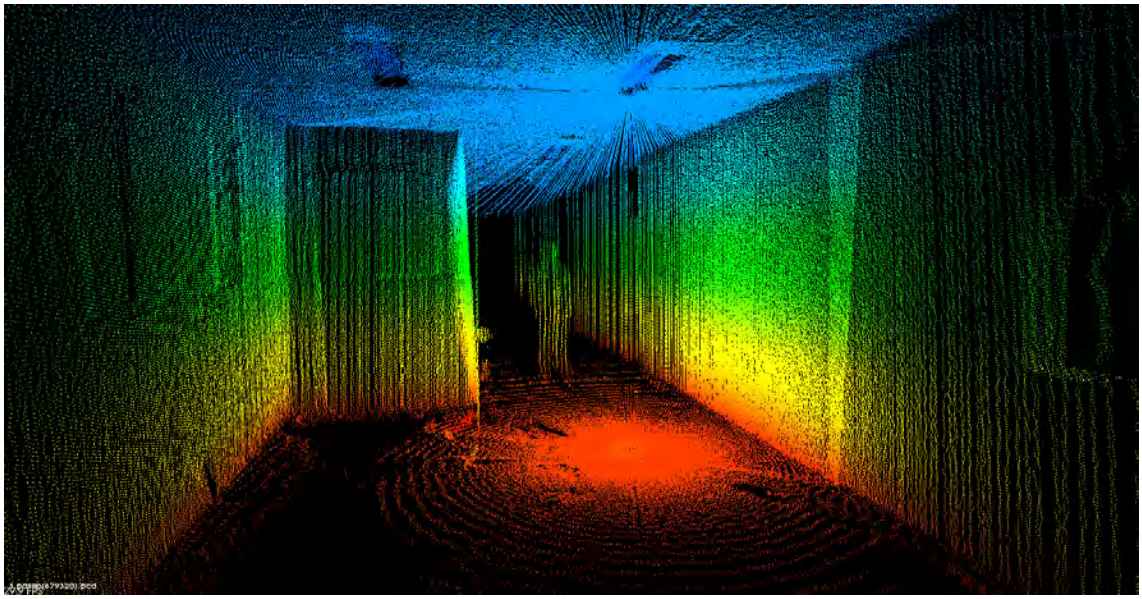


Figura 2.3. Diferencias de densidad en las nubes capturadas.

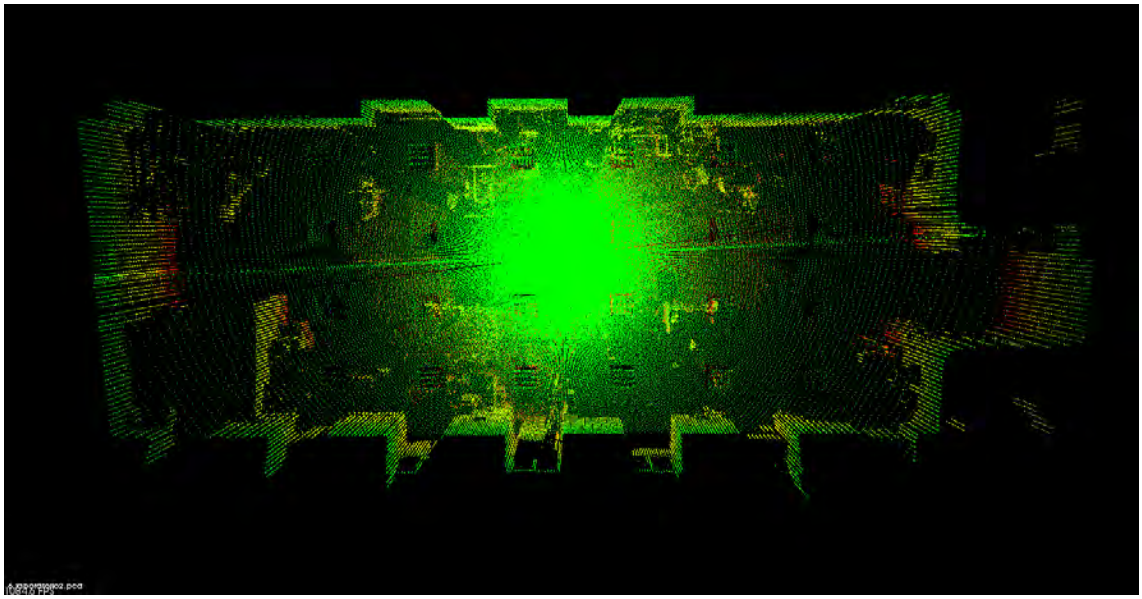


Figura 2.4. Diferencias de densidad en las nubes capturadas.

En cuanto a la cantidad de información útil para la reconstrucción del recinto cerrado, hay que tener en cuenta que, debido a los diferentes objetos situados dentro del recinto cerrado, la información que describe la forma geométrica de éste se encuentra sesgada, reducida y distorsionada. Como se puede observar en la *figura 2.5*, debido a los numerosos obstáculos existen zonas de las superficies de la pared y del suelo en donde apenas existen puntos capturados que las describan. En la figura 2.6 se muestra un ejemplo grafico de como, por diferentes causas, de pierde información acerca de recinto.

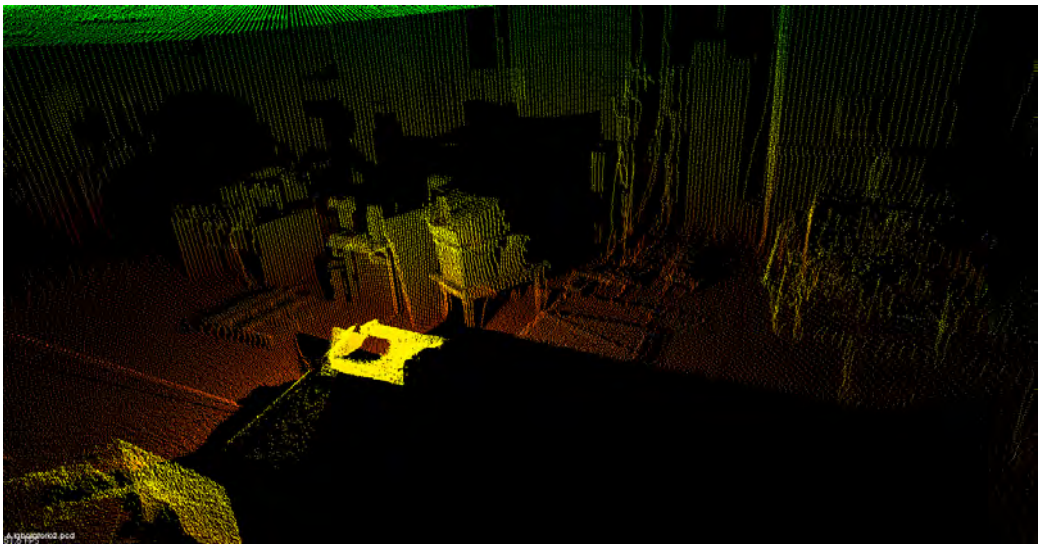


Figura 2.5.Falta de información sobre la pared y suelo del recinto cerrado.

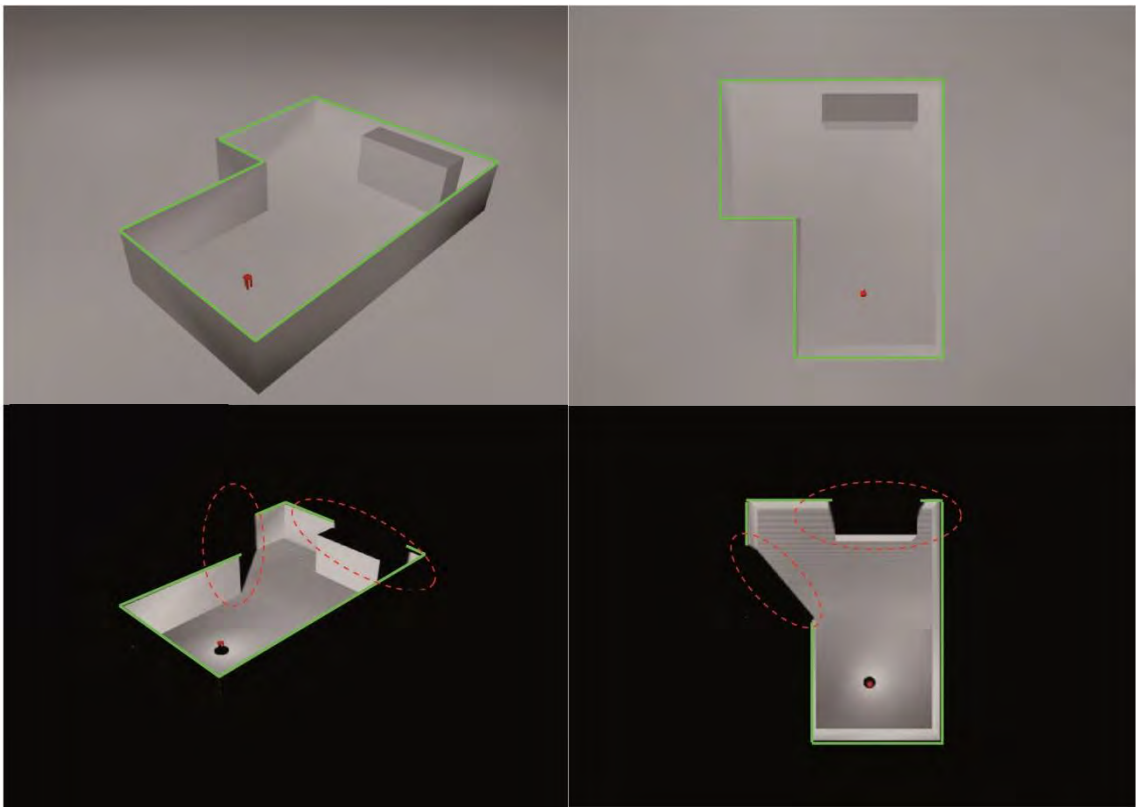


Figura 2.6 Ejemplo grafico de la perdida de información debida a las interferencias de objetos situados dentro del recinto.

3. DESCRIPCIÓN DEL ALGORITMO

Con el objeto de facilitar la comprensión del algoritmo, a continuación se detalla una descripción general teórica de éste. Tras la descripción general, se detalla cada uno de los procesos que intervienen en dicho algoritmo.

3.1. Descripción general del algoritmo

Como dato de entrada se tiene una nube de puntos capturada por el dispositivo láser 3D que describe el interior del recinto cerrado. Tras la aplicación de todos los procesos que se detallan más adelante, se obtiene como dato de salida un listado de puntos con los que se podrá reconstruir la forma y dimensiones del recinto cerrado. De esta marera, el conjunto de miles de puntos cartesianos tomados por el dispositivo láser 3D, se quedan reducidos a unos cientos de puntos cartesianos. Estos nuevos puntos, aportan de manera más sencilla y exacta la forma y dimensión del recinto. Cada punto del listado devuelto adjunta una serie de conexiones con los que se genera un recinto cerrado mediante la formación de polígonos cerrados. Estos polígonos cerrados se forman creando segmentos al unir entre si los puntos del listado devuelto. Gracias a dichos polígonos se describen las numerosas superficies con las que se delimita el recinto cerrado.

Todo el procedimiento para obtener el listado de puntos de reconstrucción se llevan a cabo mediante la ejecución de una serie de procesos secuenciales. Los procesos son los siguientes:

- Filtrado.
- Búsqueda y clasificación de coeficientes de los planos del recinto.
- Obtención de los coeficientes de las rectas de intersecciones de los planos.

- Obtención de intersecciones entre superficies del recinto.
- Detección de intersecciones erróneas.
- Obtención de puntos de reconstrucción.
- Formación de los polígonos cerrados que describen las superficies de las paredes del recinto.

En los puntos siguientes se profundiza en la metodología seguida para la realización de cada uno de los procesos antes mencionados.

3.2. **Filtrado**

En primer lugar se realiza un proceso de filtrado de la densidad de puntos de la nube. Se reduce la densidad de forma que queden eliminados aquellos puntos que aporten una misma información para la búsqueda del objetivo final. Esto proporciona agilidad a la hora de procesamiento de los puntos. Además con el filtrado se consigue que la nube con la que se trabaja tenga una cota superior del número de puntos por unidad de volumen.

Para filtrar se divide el espacio cartesiano de la nube de puntos en un conjunto de pequeñas celdas tridimensionales de tamaño configurable. Cada punto de la nube es clasificado en función de la cuadrícula en la que esté contenido. Identificados qué puntos están contenidos en qué cuadrículas, se calcula para cada cuadrícula tridimensional su centro de gravedad. Para ello se utiliza la *ecuación 3.1.* Posteriormente, los puntos recogidos en el interior de cada cuadrícula tridimensional son sustituidos por sus correspondientes centros de gravedad.

$$X_{cdg} = \frac{\sum_{i=0}^{n} X_i}{n} \quad Y_{cdg} = \frac{\sum_{i=0}^{n} Y_i}{n} \quad Z_{cdg} = \frac{\sum_{i=0}^{n} Z_i}{n}$$

$X_{cdg} \rightarrow$ Coordenada X del centro de gravedad de la celda.
$Y_{cdg} \rightarrow$ Coordenada Y del centro de gravedad de la celda.
$Z_{cdg} \rightarrow$ Coordenada Z del centro de gravedad de la celda.
$X_i \rightarrow$ Coordenada X del punto i de la celda.
$Y_i \rightarrow$ Coordenada X del punto i de la celda.
$Z_i \rightarrow$ Coordenada X del punto i de la celda.
$n \rightarrow$ Numero de puntos en la celda.

Ecuación 3.1.

En las *figura 3.7* y *3.8* se muestra gráficamente la secuencia del proceso descrito. Inicialmente (*figura 3.7)* los puntos se clasifican en función de la cuadricula donde estén contenidos. Tras clasificarlos, se calculan los centros de gravedad de las cuadriculas que sustituyen a los puntos originales (*figura 3.8).*

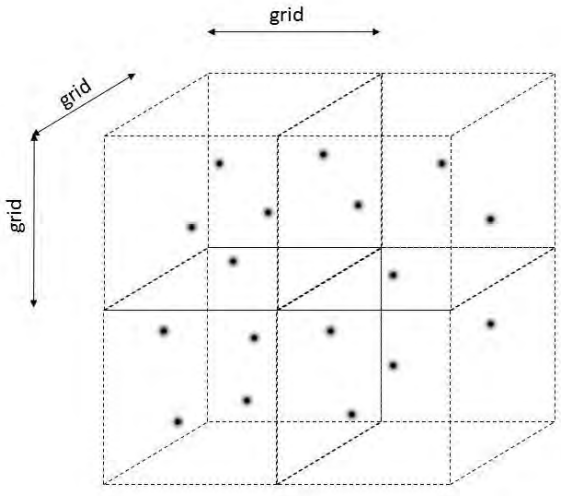


Figura 3.7. Clasificación de los puntos en celdas.

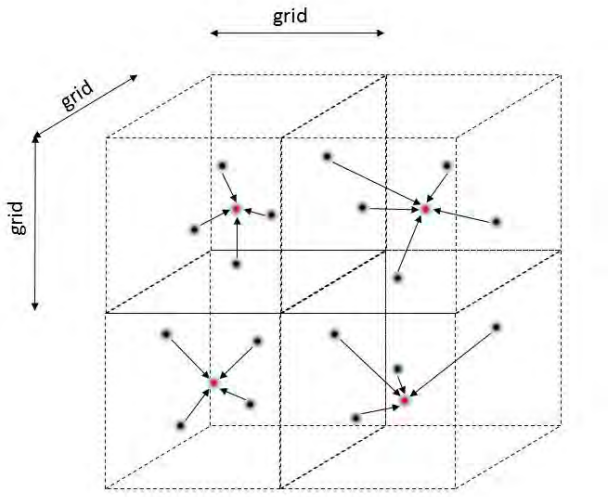


Figura 3.8. Cálculo del centro de gravedad de cada celda.

A continuación se muestran un ejemplo de cómo queda la nube de puntos antes (328.320 puntos) y después (35.899 puntos) del proceso de filtrado. Cabe destacar que aunque el nivel de detalle de las formas contenidas en el recinto haya disminuido debido a la pérdida de información tras el filtrado, la información referente a las paredes del recinto apenas varía.

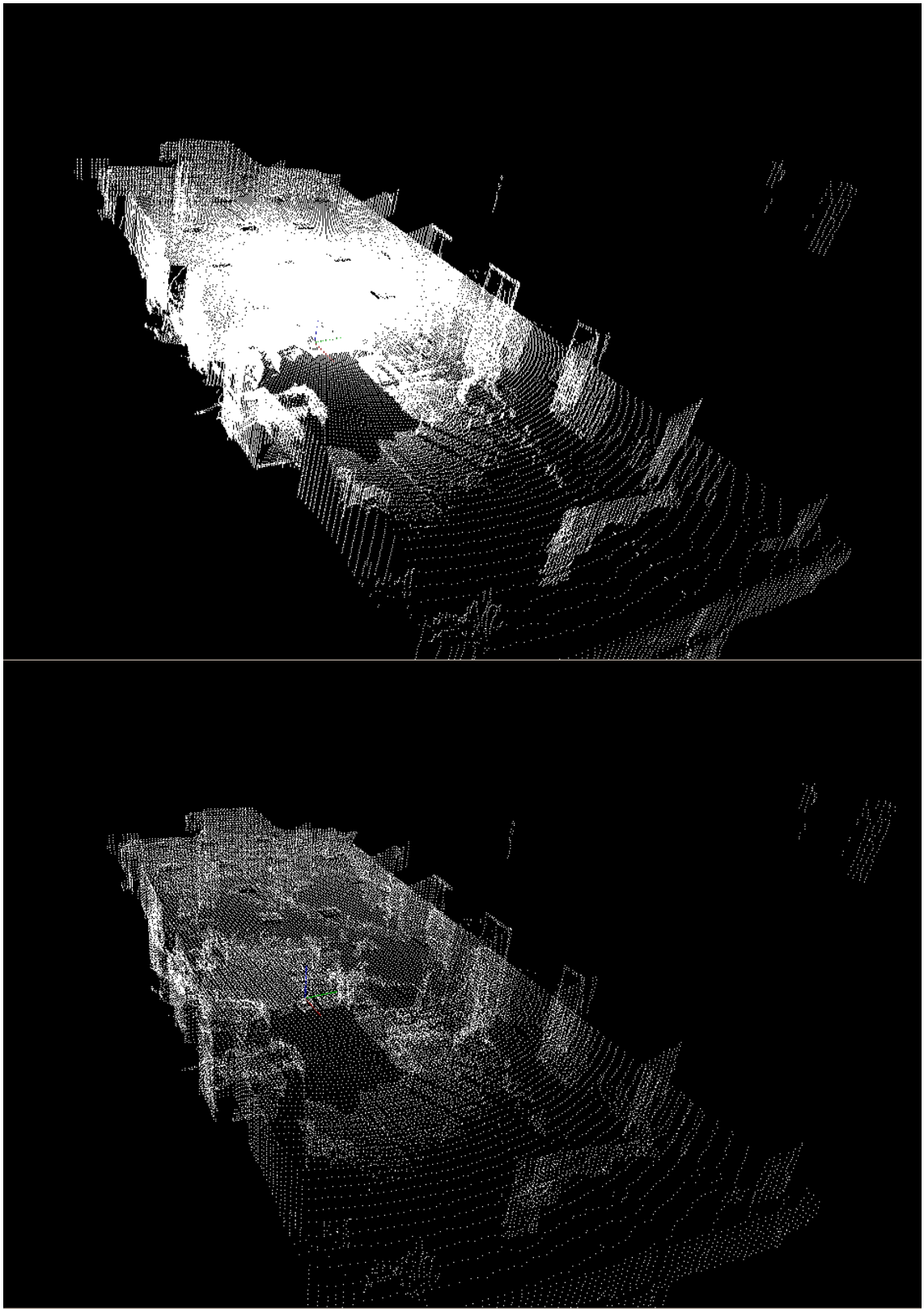


Figura 3.9. Perspectiva 1 del antes (arriba) y el después (abajo) del proceso de filtrado.

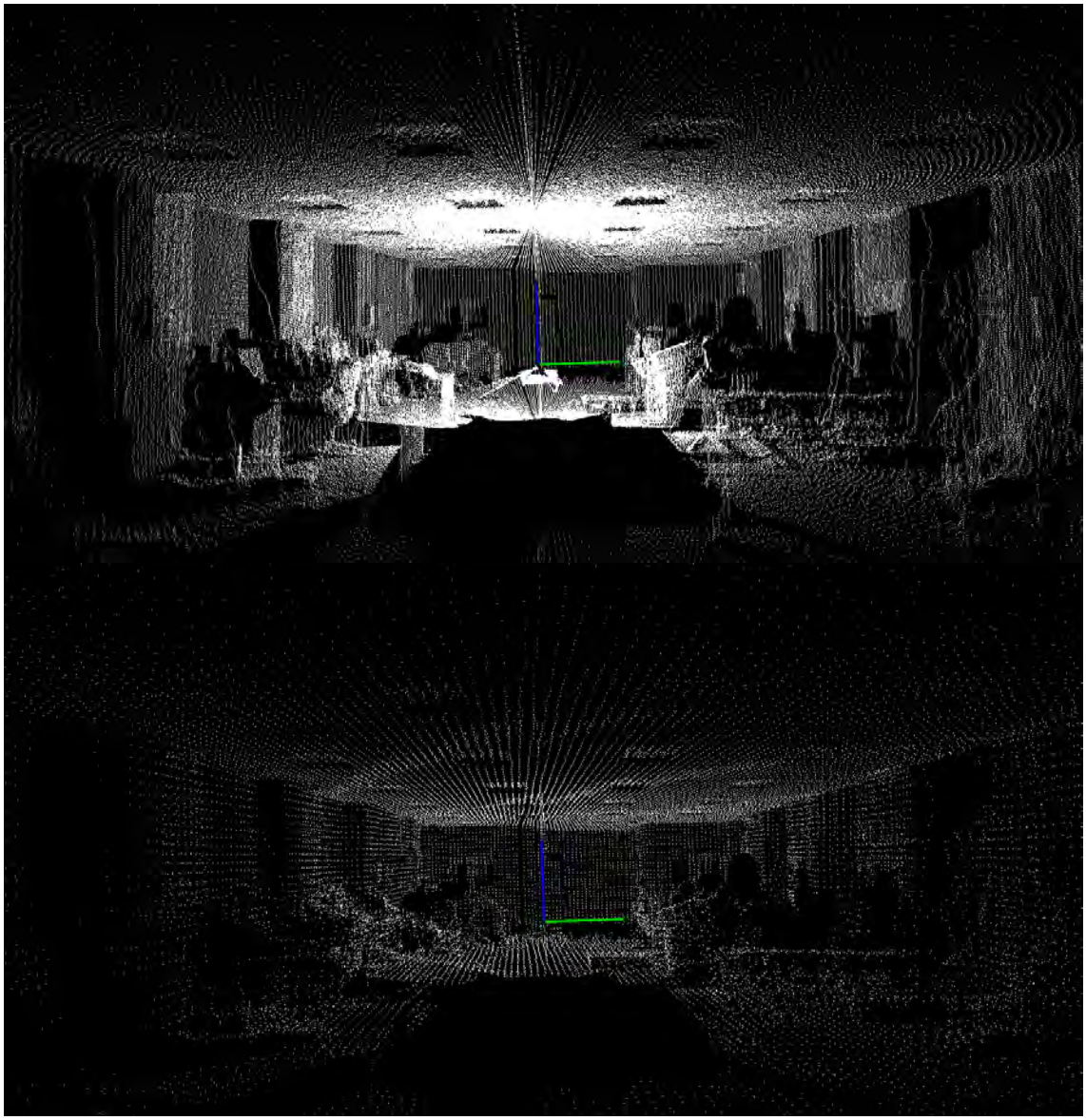


Figura 3.10. Perspectiva 2 del antes (arriba) y el después (abajo) del proceso de filtrado.

En conclusión, tras realizar el proceso de filtrado, la densidad queda reducida un nivel inferior al original, pero se mantiene casi toda la información necesaria para poder reconstruir la forma geométrica del recinto cerrado.

3.3. **Búsqueda y clasificación de los coeficientes de los planos del recinto**

Reducida la densidad se procede a buscar y clasificar las ecuaciones de los planos en donde, posiblemente, estén contenidos los puntos que describen las superficies de las paredes. El fin de este proceso es el de obtener un listado con estas ecuaciones generales de planos. De esta forma, en pasos posteriores se puede extraer información contenida en la nube que haga referencia únicamente a una superficie de pared del recinto en concreto.

Figura 3.11. Esquema general del proceso.

La búsqueda de los coeficientes de las ecuaciones generales de los planos se realiza mediante el método conocido como RANSAC [7]. Este método genera los coeficientes de las ecuaciones generales de planos basándose en números generados de manera aleatoria. Para cada ecuación que se genere se calcula el número de puntos de la nube que cumplan con ella. Después de calcular el número de puntos que cumplen con cada ecuación, se selecciona la que más puntos logre verificar y se descartan las demás. En la secuencia de la *figura 3.12* se muestra un ejemplo de como este proceso localiza la ecuación general del plano. El proceso RANSAC se repetirá mientras se siguán generando ecuaciones de planos con las que se verifique un número de puntos de la nube superior a un cierto valor establecido. Cada ecuación encontrada se almacena temporalmente para luego decidir si guardarla o descartarla. Hay que tener en cuenta que, ya que el proceso RANSAC obtiene la ecuación que más los puntos de la nube verifique, una vez almacenada la ecuación, habrá que extraer todos los puntos de la nube que cumplan con ella. Esto se hace por que si no se extrajesen estos puntos el método localizaría una y otra vez la misma ecuación del plano. En la *figura 3.13* se muestra un esquema grafico de los pasos seguidos en este proceso.

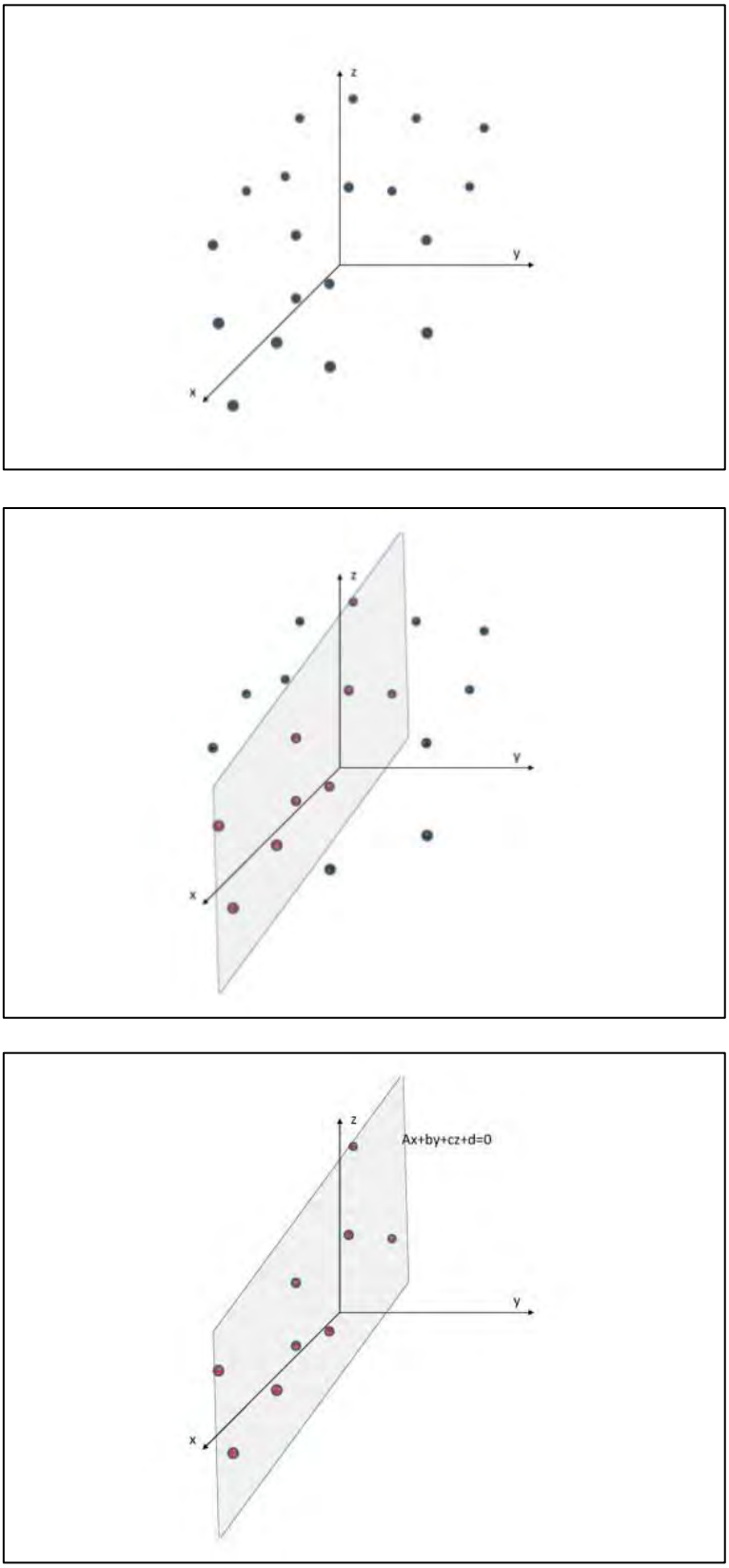


Figura 3.12. Método RANSAC.

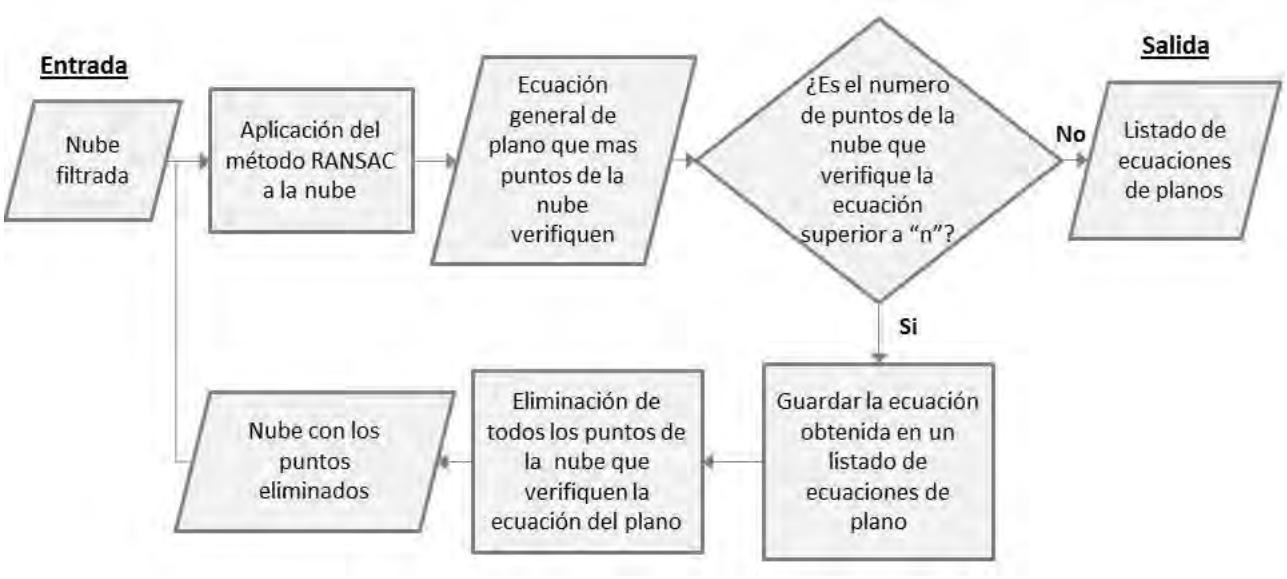


Figura 3.13. Esquema gráfico del proceso seguido para la obtención del primer listado de ecuaciones de plano.

Las ecuaciones de planos que se hayan guardado pueden verificar tanto a un conjunto de puntos que formen una superficie amplia (por ejemplo los puntos correspondientes a las superficies de las paredes), como a un conjunto puntos que formen superficies pequeñas (por ejemplo los puntos correspondientes a diferentes superficies de objetos). Por ello, es necesario identificar cuáles de las ecuaciones de planos guardadas no verifican ningún conjunto de puntos que formen una superficie amplia para así poder eliminarlas de la lista. Para identificar el tamaño de las superficies formadas se calcula la relación superficie-borde mediante la *ecuación 3.2*. Si una ecuación de plano tiene una relación superficie-borde inferior a un valor umbral establecido es eliminada de la lista ecuaciones. En los ejemplos de las *figuras 3.14-A* y *3.14-B* se muestran las relaciones superficie-borde que se obtiene para superficies de distinto tamaño.

$$RSB = \frac{n_s}{n_b}$$

$RSB \rightarrow Relacion\ superficie - borde$

$n_s \rightarrow numero\ de\ puntos\ que\ forman\ la\ superficie$

$n_b \rightarrow numero\ de\ puntos\ que\ forman\ los\ bordes\ de\ la\ superficie$

Ecuación 3.2. Relación superficie-borde.

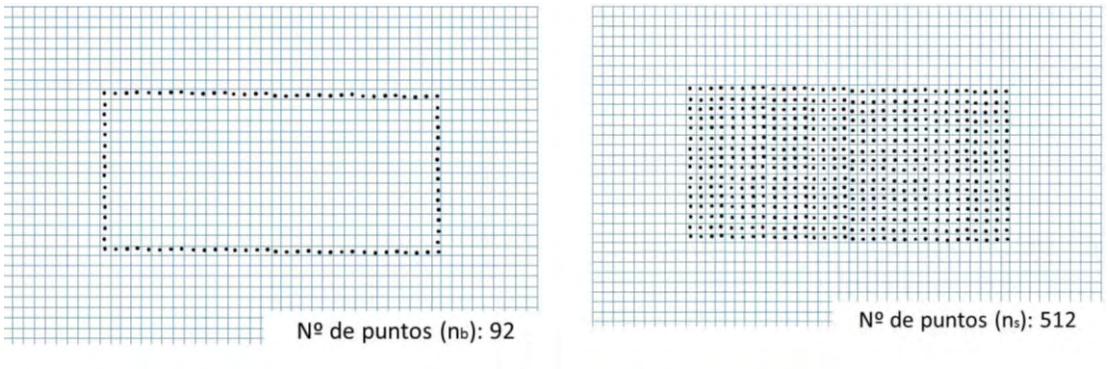


Figura 3.14-A. Comparación de superficie-borde de una superficie amplia (RSB = 5,56).

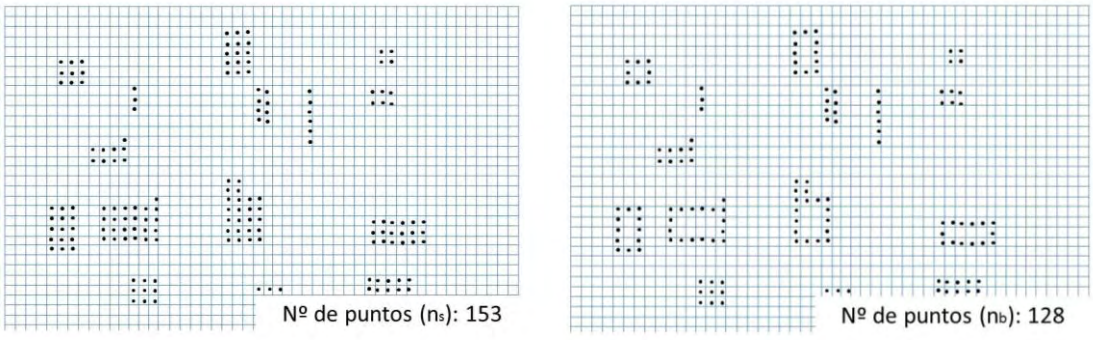


Figura 3.14-B. Comparación de superficie-borde de una superficie pequeña (RSB = 1,19).

Cabe indicar que aunque todas las ecuaciones de planos guardadas verifiquen conjuntos de puntos de la nube que formen una superficie amplia, pueden existir superficies amplias que no correspondan a las superficies de paredes. Estas superficies pueden ser por ejemplo superficies de armarios o mesas situadas dentro del recinto. Las ecuaciones de planos que verifique a estas superficies habría que eliminarlas de la lista. Pero por el momento no se tienen datos suficientes para diferenciarlas. En la *figura 4.15* se muestra un ejemplo grafico como una superficie amplia que no corresponde a la de una pared puede ser confundida por sus similitudes. Estas ecuaciones de planos son eliminadas en un proceso que se realiza más adelante.

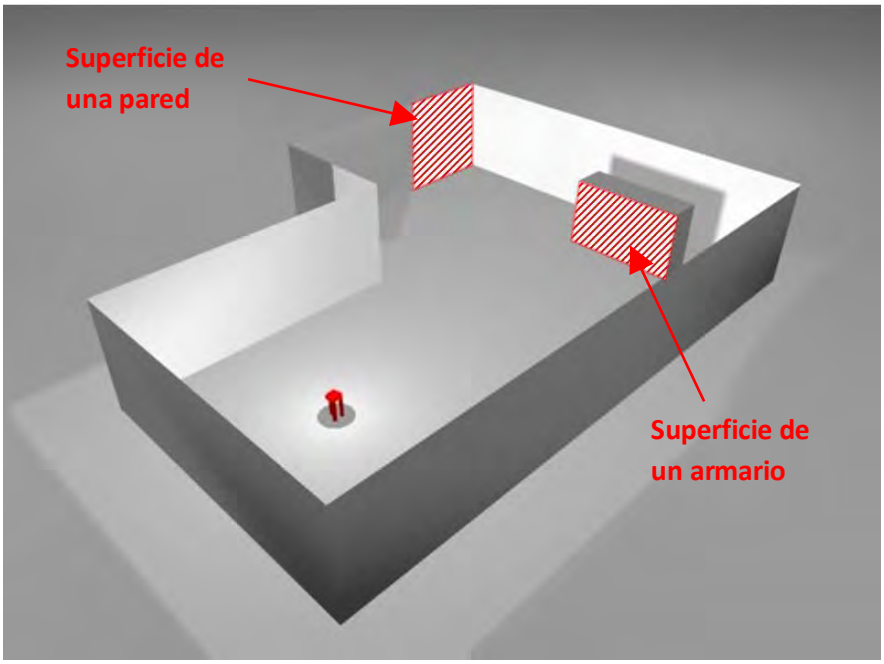


Figura 4.15. Similitudes de superficies del recinto.

Por otro lado también cabe indicar que en el listado de ecuaciones de planos guardadas no están representadas todas las superficies de pared del recinto. Esto es debido a que por diferentes causas, pueden existir superficies de pared con escasa o nula representación en la nube de puntos. Aunque estos planos no se localicen inicialmente, sí que estarán representados en el modelo del recinto generado tras finalizar el algoritmo.

Tras obtener el listado de ecuaciones planos se procede a clasificar las ecuaciones en tres grupos de reconstrucción. Las ecuaciones se clasifican en tres grupos:

- Ecuación de plano tipo H_s (tipo horizontal superior).
- Ecuación de plano tipo H_i (tipo horizontal inferior).
- Ecuación de plano tipo V (tipo vertical).

Esta clasificación es necesaria para poder proceder con la reconstrucción en procesos posteriores. Para clasificarlas se tiene en cuenta el vector normal a la cara exterior de los planos de cada ecuación. La clasificación se hace observando la posición del vector con respecto a dos conos enfrentados. Para el caso de que el vector normal quede fuera de los dos conos, el plano al que pertenece este vector es clasificado como un plano "tipo V". Por el contrario, para el caso de que el vector normal caiga dentro de alguno de los dos conos, sé que averigua cuál de los dos conos es. Si cae dentro del cono superior es considerado "tipo H_s", en caso contrario será considerado "tipo H_i". En la *figura 3.16* se muestra gráficamente como se identifican los tipos de planos mediante conos enfrentados.

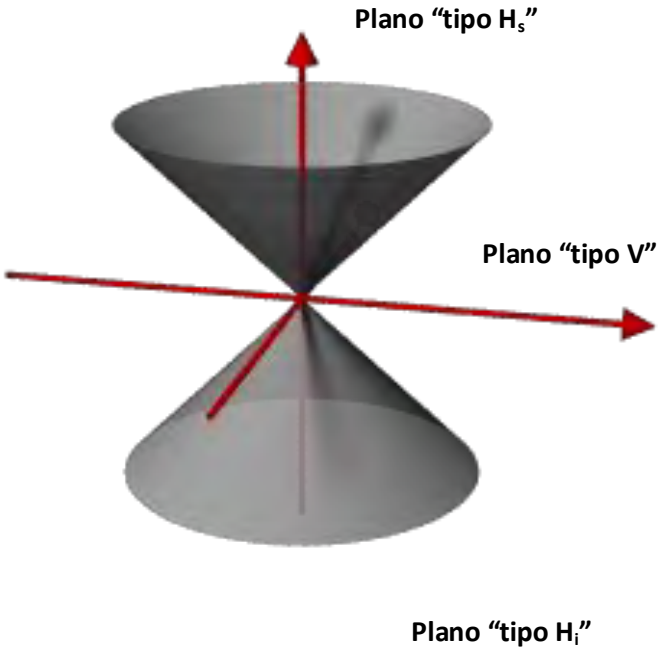


Figura 3.16. Clasificación del plano mediante conos enfrentados.

En las *figuras 3.17, 3.18 y 3.19* se muestran, marcados por diferentes colores, un ejemplo de las diferentes superficies que se han localizado mediante este proceso en una nube de puntos real.

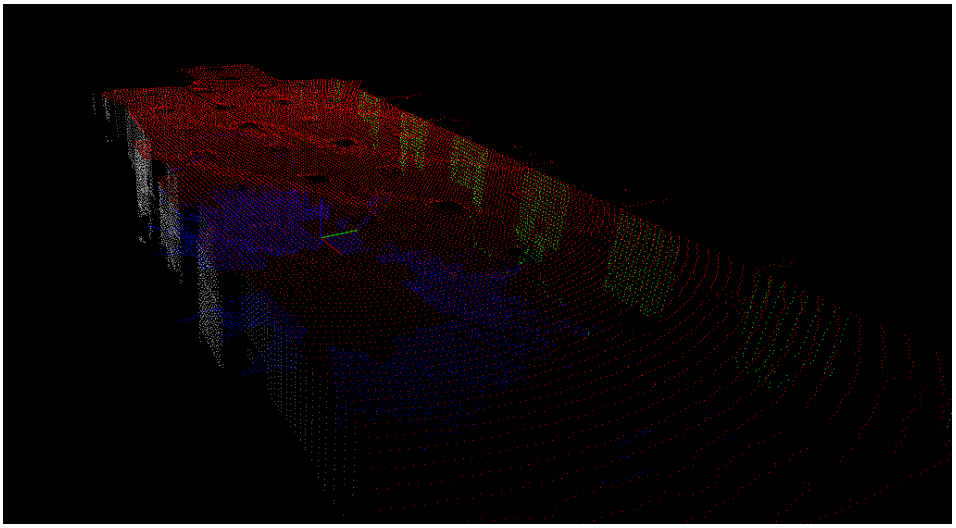


Figura 3.17. Diferentes superficies localizadas en una nube de puntos.

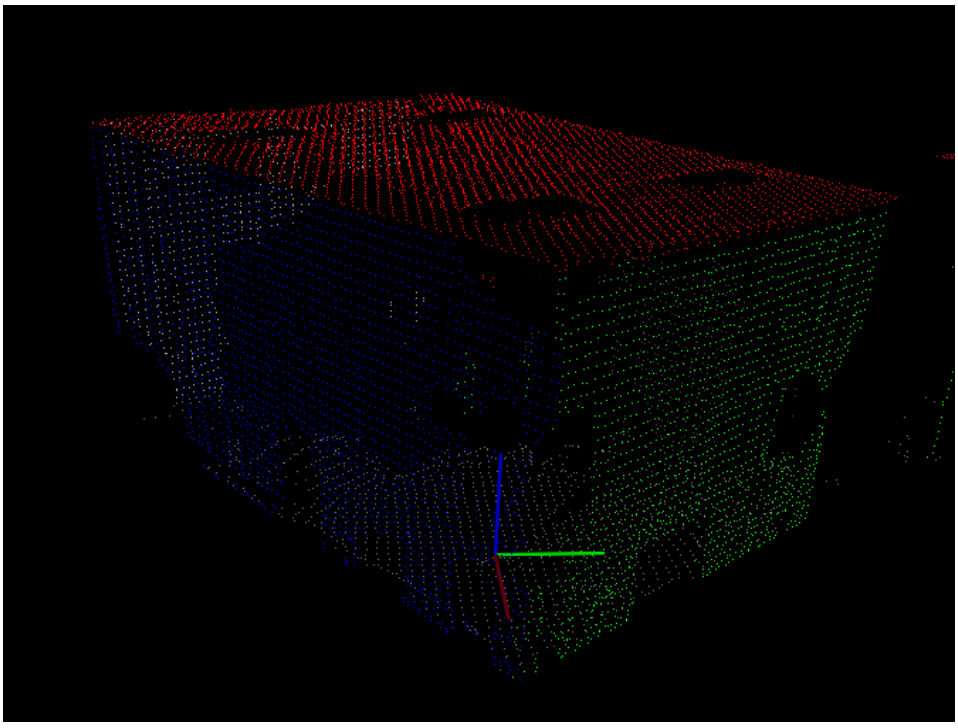


Figura 3.18. Diferentes superficies localizadas en una nube de puntos.

3.4. Obtención de los coeficientes de las rectas de intersecciones entre planos

Obtenido el listado con las ecuaciones de los diferentes planos que forman el recinto, se procede a calcular las ecuaciones de las rectas que describen las intersecciones que existentes entre estos. El fin de este proceso busca obtener un listado con las ecuaciones generales de las rectas de intersección. En cada ecuación de recta devuelta se adjuntan las ecuaciones de los dos planos que la forman.

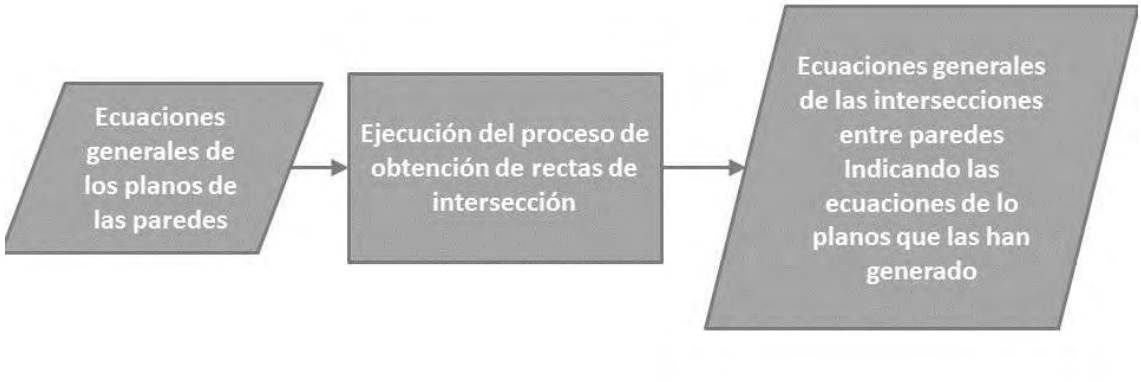


Figura 3.19. Esquema general del proceso.

Cada recta de intersección devuelta se representa mediante un vector director y un punto perteneciente a ésta. El vector director se calcula utilizando la *ecuación 3.3* y el punto perteneciente a la recta se calcula resolviendo el sistema de ecuaciones de la *ecuación 3.4.* Para saber con qué combinaciones de planos se calculan las rectas de intersección se utiliza la clasificación realizada en el apartado anterior. De esta manera para calcular las intersecciones entre planos únicamente se buscan combinaciones de planos de "tipo V" con planos de "tipo H_i" y combinaciones de planos de "tipo V" con planos de "tipo H_i".

$$\bar{V} = \begin{vmatrix} \bar{\imath} & \bar{\jmath} & \bar{k} \\ A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \end{vmatrix}$$

$A_1 \rightarrow Coeficiente\ A\ de\ la\ ecuacion\ del\ plano\ 1$
$B_1 \rightarrow Coeficiente\ B\ de\ la\ ecuacion\ del\ plano\ 1$
$C_1 \rightarrow Coeficiente\ C\ de\ la\ ecuacion\ del\ plano\ 1$
$A_2 \rightarrow Coeficiente\ A\ de\ la\ ecuacion\ del\ plano\ 2$
$B_2 \rightarrow Coeficiente\ B\ de\ la\ ecuacion\ del\ plano\ 2$
$C_2 \rightarrow Coeficiente\ C\ de\ la\ ecuacion\ del\ plano\ 2$
$\bar{V} \rightarrow vector\ director\ de\ la\ recta$

Ecuación 3.3. Calculo del vector director de la recta.

$$\begin{Bmatrix} A_1 \cdot X + B_1 \cdot Y + C_1 \cdot Z + D_1 = 0 \\ A_2 \cdot X + B_2 \cdot Y + C_2 \cdot Z + D_2 = 0 \end{Bmatrix}$$

$A_1 \rightarrow Coeficiente\ A\ de\ la\ ecuacion\ del\ plano\ 1$
$B_1 \rightarrow Coeficiente\ B\ de\ la\ ecuacion\ del\ plano\ 1$
$C_1 \rightarrow Coeficiente\ C\ de\ la\ ecuacion\ del\ plano\ 1$
$D_1 \rightarrow Coeficiente\ D\ de\ la\ ecuacion\ del\ plano\ 1$

$A_2 \rightarrow Coeficiente\ A\ de\ la\ ecuacion\ del\ plano\ 2$
$B_2 \rightarrow Coeficiente\ B\ de\ la\ ecuacion\ del\ plano\ 2$
$C_2 \rightarrow Coeficiente\ C\ de\ la\ ecuacion\ del\ plano\ 2$
$D_2 \rightarrow Coeficiente\ D\ de\ la\ ecuacion\ del\ plano\ 2$

$X \rightarrow Coordenada\ X\ del\ punto\ perteneciente\ a\ la\ recta$
$Y \rightarrow Coordenada\ Y\ del\ punto\ perteneciente\ a\ la\ recta$
$Z \rightarrow Coordenada\ Z\ del\ punto\ perteneciente\ a\ la\ recta$
$(generada\ de\ forma\ aleatoria\ para\ solventar\ el\ problema\ del\ grado\ de\ libertad)$

Ecuación 3.4. Sistema de ecuaciones para obtener un punto de la recta de intersección.

3.5. Obtención de intersecciones entre superficies del recinto

El fin de este proceso es el de obtener una primera aproximación de las intersecciones entre las diferentes superficies de las paredes del recinto. Para ello, como se muestra en la *figura 3.20*, se proyectan los puntos que verifiquen las ecuaciones de los planos clasificados como "tipo V". Se proyectan sobre las diferentes rectas de intersección que se generaron en el proceso anterior. De este modo todos los puntos pertenecientes a un plano se proyectan sobre la rectas de intersección que ese plano haya generado con otros planos. El fin de este proceso busca obtener un conjunto de puntos que unidos entre si formen los segmentos que describan las intersecciones. En la *figura 3.21* se muestra un ejemplo grafico de como proyectando los puntos se obtiene una primera aproximación de las intersecciones del recinto.

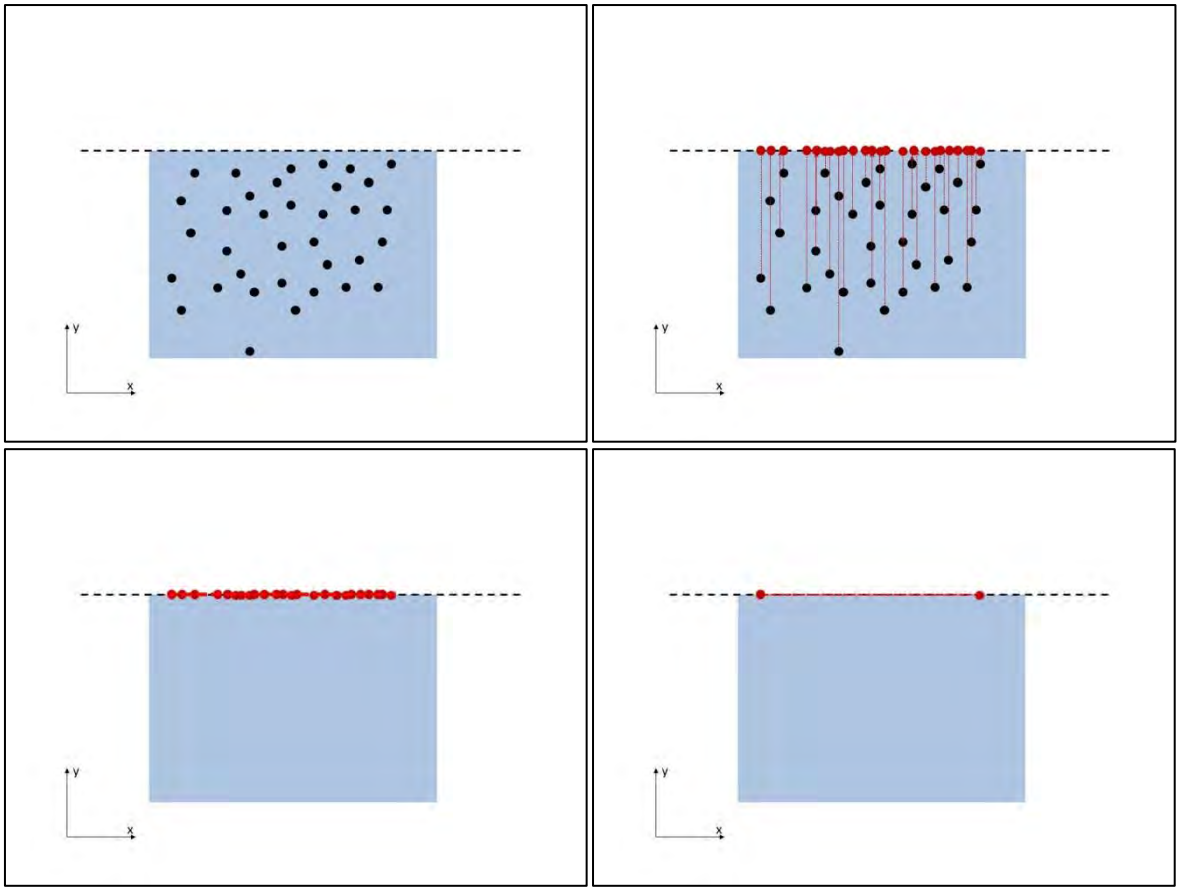


Figura 3.20. Calculo de primera aproximación de la intercesión mediante la proyección de los puntos.

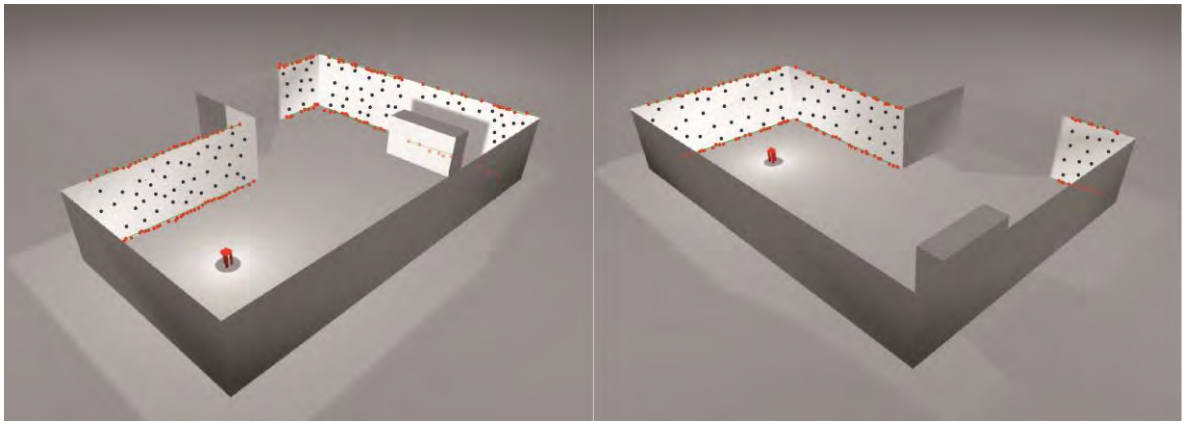


Figura 3.21. Primera aproximación de las intersecciones (puntos en rojo).

Existe un inconveniente al obtener las intersecciones de esta manera mencionada. Este inconveniente es que al extraer todos los puntos que verifiquen la ecuación de un plano para proyectarlos, se pueden estar extrayendo puntos que no pertenezcan a la superficie de la pared. Estos puntos pueden estar ahí por múltiples causas (ruido, interferencia con otros objetos, etc…). Esto implica obtener segmentos de intersección que describen intersecciones inexistentes. En la *figura 3.22* se muestra un ejemplo grafico de este inconveniente.

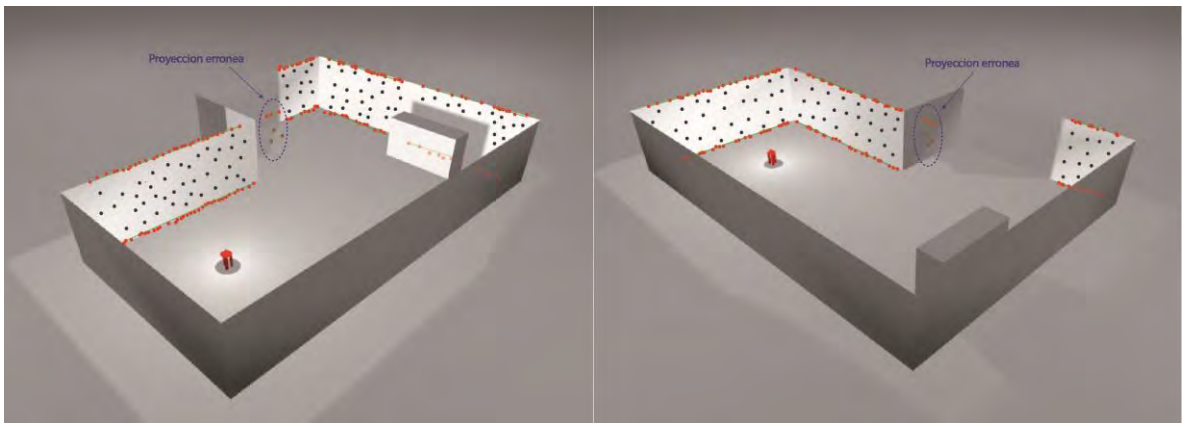


Figura 3.22. Proyección errónea de un plano.

Para poder eliminar los puntos erróneos de la recta de intersección formada se realiza un análisis estadístico de los puntos proyectados. Este análisis estudia la densidad de puntos por unidad de longitud a lo largo de la recta. El análisis tiene en cuenta el tamaño de las celdas del proceso de filtrado que se realizó inicialmente. Gracias este filtrado se sabe que en cada celda de filtrado existe como máximo un punto cartesiano. En la *figura 3.23* se muestra un ejemplo de la disposición de los puntos. Al realizar la proyección de los puntos sobre la recta intersección, el número de puntos dentro de las celdas a lo largo de la recta se ve incrementado. En zonas de la recta donde los puntos proyectados procedan de las superficies de la pared, el incremento de puntos por celda es elevado. Por el contrario, en zonas de la recta donde los puntos proyectados no procedan de las superficies de la pared, el incremento de puntos por celda es reducido. Para poder eliminar las proyecciones que no procedan de la superficie de la pared se calcula el valor percentil [8] de todos los incrementos de densidad de puntos en las celdas de filtrado a lo largo de la recta intersección. Calculado el valor percentil se eliminan todos los puntos que pertenezcan a una celda cuyo incremento de densidad de puntos no esté por encima de ese valor percentil. En la *figura 2.24* se muestra como queda la intersección con y sin análisis de proyecciones.

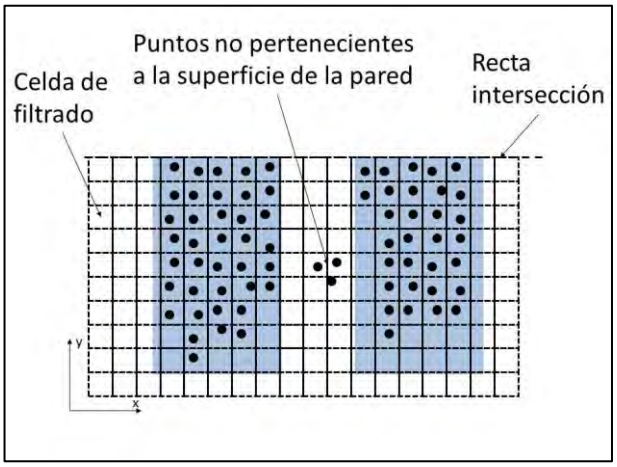


Figura 3.23. Ejemplo de la disposición de los puntos de una superficie.

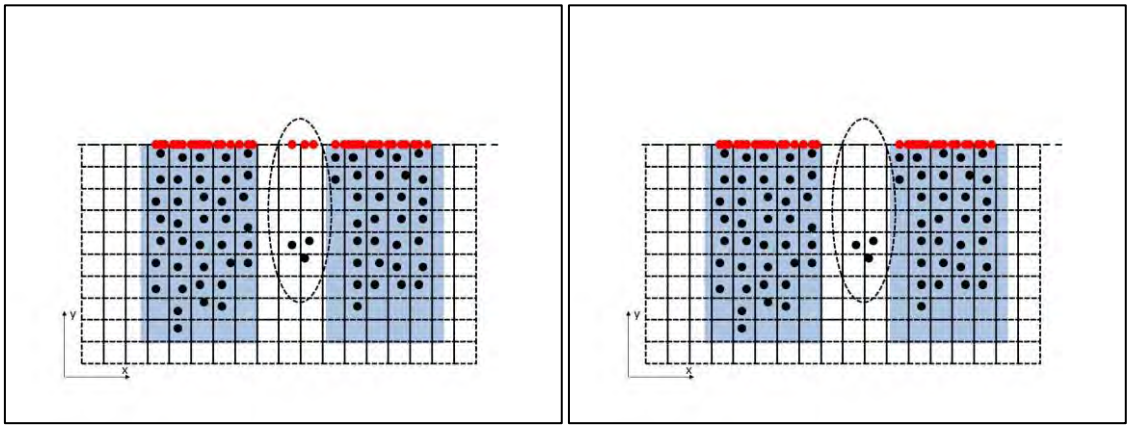


Figura 3.24. Ejemplo de proyección con (derecha) y sin (izquierda) la realización del análisis estadístico de las proyecciones.

Con los puntos erróneos de las proyecciones eliminados se procede a formar los segmentos que describen las intersecciones. Estos segmentos están descritos por los dos puntos de sus extremos. Debido a que una recta puede contener varios segmentos, primero se identifican agrupaciones de puntos dentro de los puntos proyectados sobre la recta. De esta forma, con cada agrupación que se identifique identificada se genera un segmento. El procedimiento para identificar agrupaciones es el siguiente:

1. Para cada punto proyectado se obtiene la distancia a su punto más cercano.

2. Cuando se tengan todas las distancias se calcula el percentil todas estas. Este percentil establece una distancia mínima que han de tener dos puntos para pertenecer a una misma agrupación.

3. Se examinan de nuevo los puntos proyectados para formar las agrupaciones. Se forman las agrupaciones teniendo en cuenta esta dos reglas:

   - Si el punto A y el punto B se encuentran a una distancia inferior a la distancia del percentil, ambos pertenecen a la misma agrupación.

   - Un punto solo puede pertenecer a una única agrupación, luego:

     - Si el punto A y el punto B pertenecen por distancia a una misma agrupación.

- Si el punto B y el punto C pertenecen por distancia a una misma agrupación.
- El punto A y el punto C tienen que pertenecer a una misma agrupación.

➢ Para evitar posibles errores se establece un número mínimo de puntos por agrupación.

Después, cada agrupación identificada se reduce a sus dos puntos más alejados. Estos dos puntos corresponden a los extremos del segmento. En la figura 3.25 se muestra un ejemplo de cómo se generan los segmentos mediante las proyecciones. Para cada punto que forme un extremo de un segmento se guarda:

➢ Su coordenada cartesiana.

➢ Las ecuaciones de los planos a los que pertenece el punto.

➢ Un identificador del punto con el que habrá que conectar para formar un segmento.

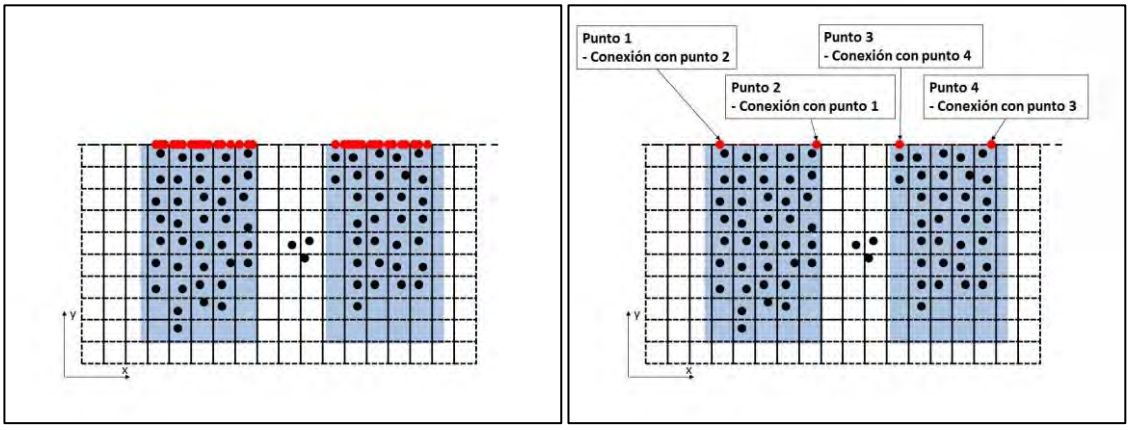


Figura 3.25. Ejemplo de cómo quedan las agrupaciones reducidas a los puntos de sus extremos.

A continuación, en las *figuras 3.26 y 3.27* se muestran los resultados que se obtienen tras la aplicación de este proceso en nubes obtenidas en diferentes recintos. En ambas figuras se muestran:

- Los planos “tipo V” en azul.
- Los plano “tipo H” en rojo.
- Las intersecciones (libres de errores) en blanco.

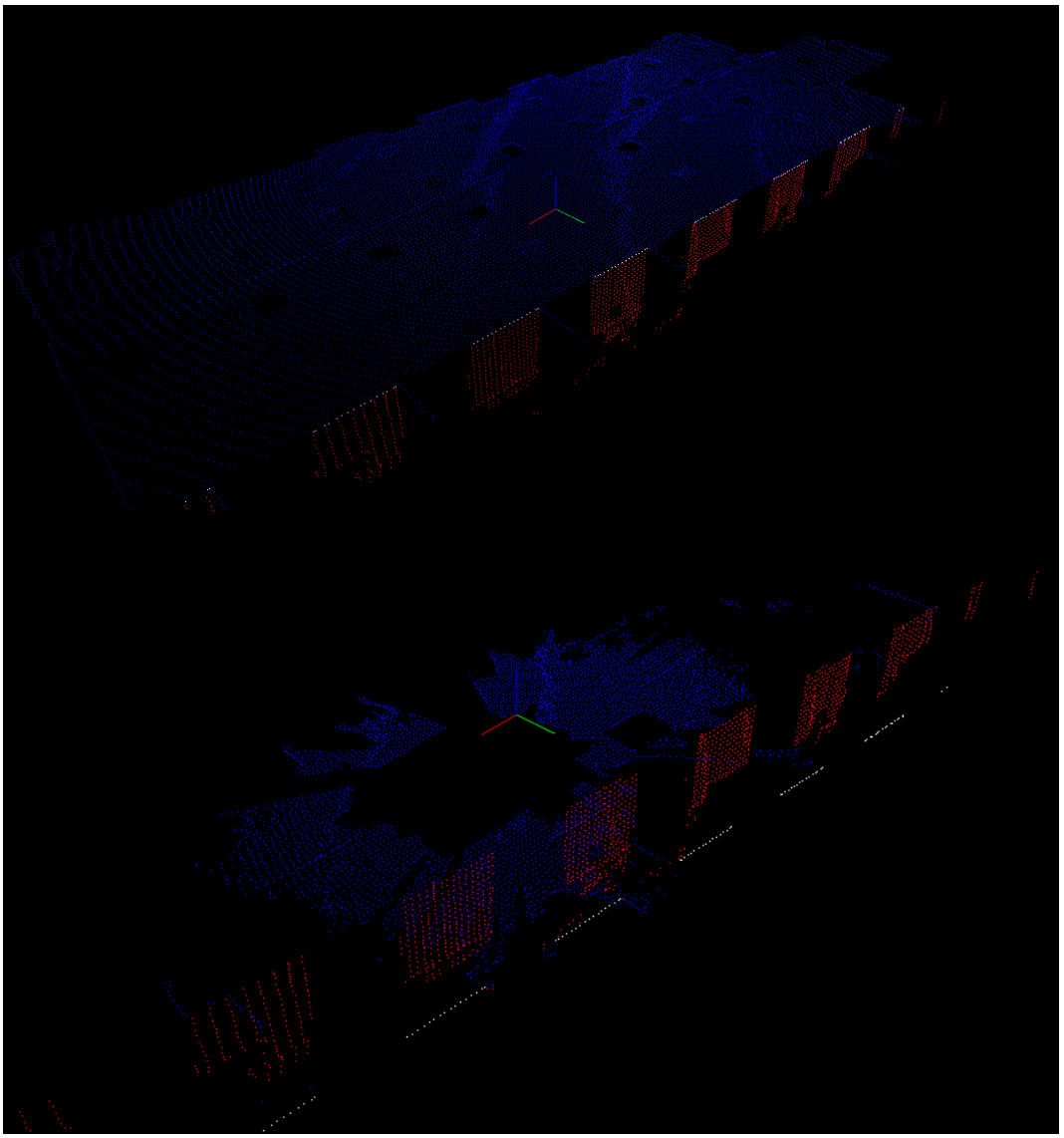


Figura 3.26. Ejemplo aplicado de la proyección de un plano “tipo V” sobre un plano “tipo H_s” (arriba) y “tipo H_i” (abajo).

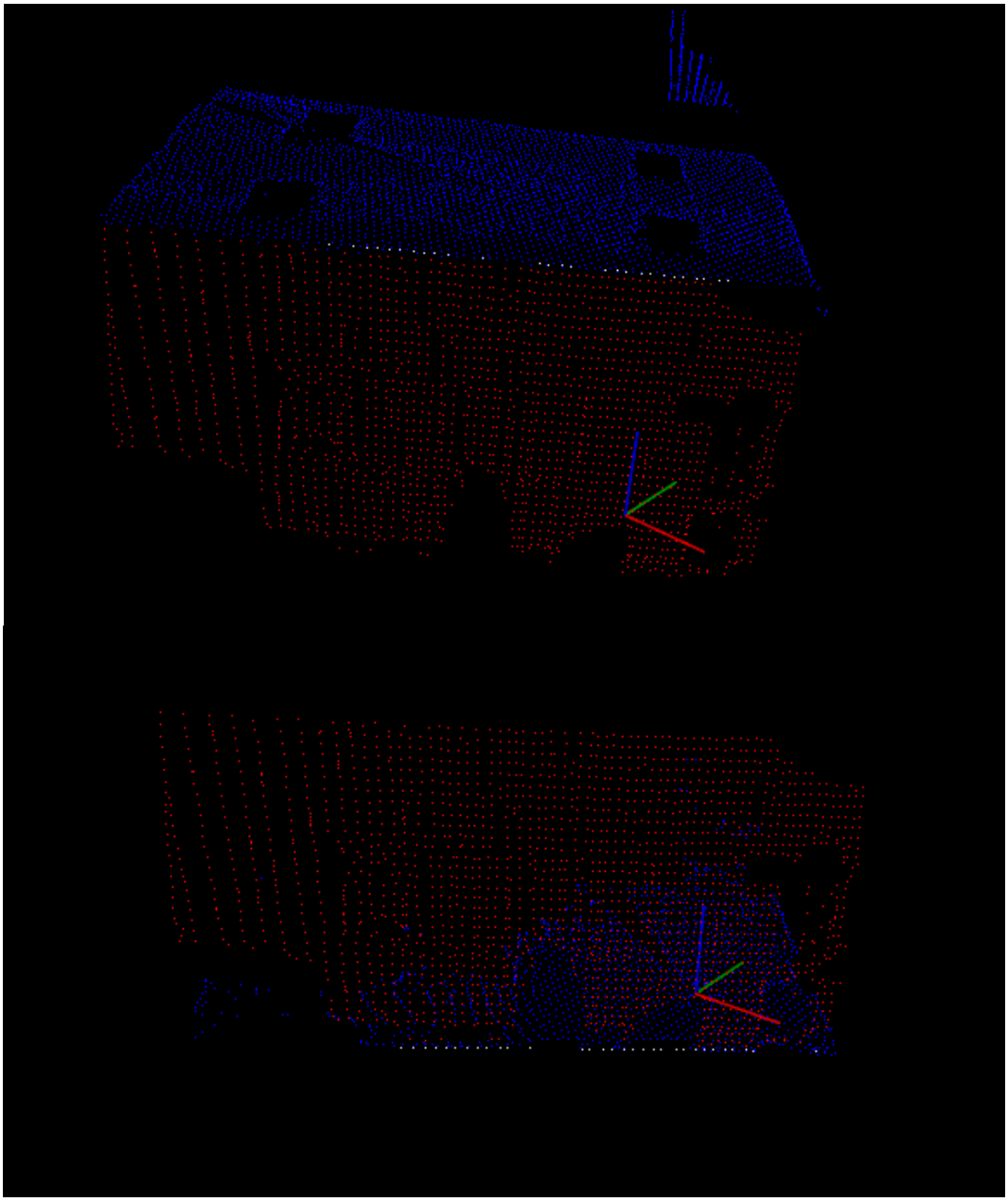


Figura 3.27. Ejemplo aplicado de la proyección de un plano "tipo V" sobre un plano "tipo H_s" (arriba) y "tipo H_i" (abajo).

Este proceso se realiza para todos los planos clasificados como "tipo V" de la lista de planos. Tras realizarse se obtiene una primera aproximación de todas las intersecciones entre las diferentes superficies del recinto. En las f*iguras 3.28* y *3.29* se muestran, junto con su nube original, los resultados que se obtienen tras terminar con este proceso.

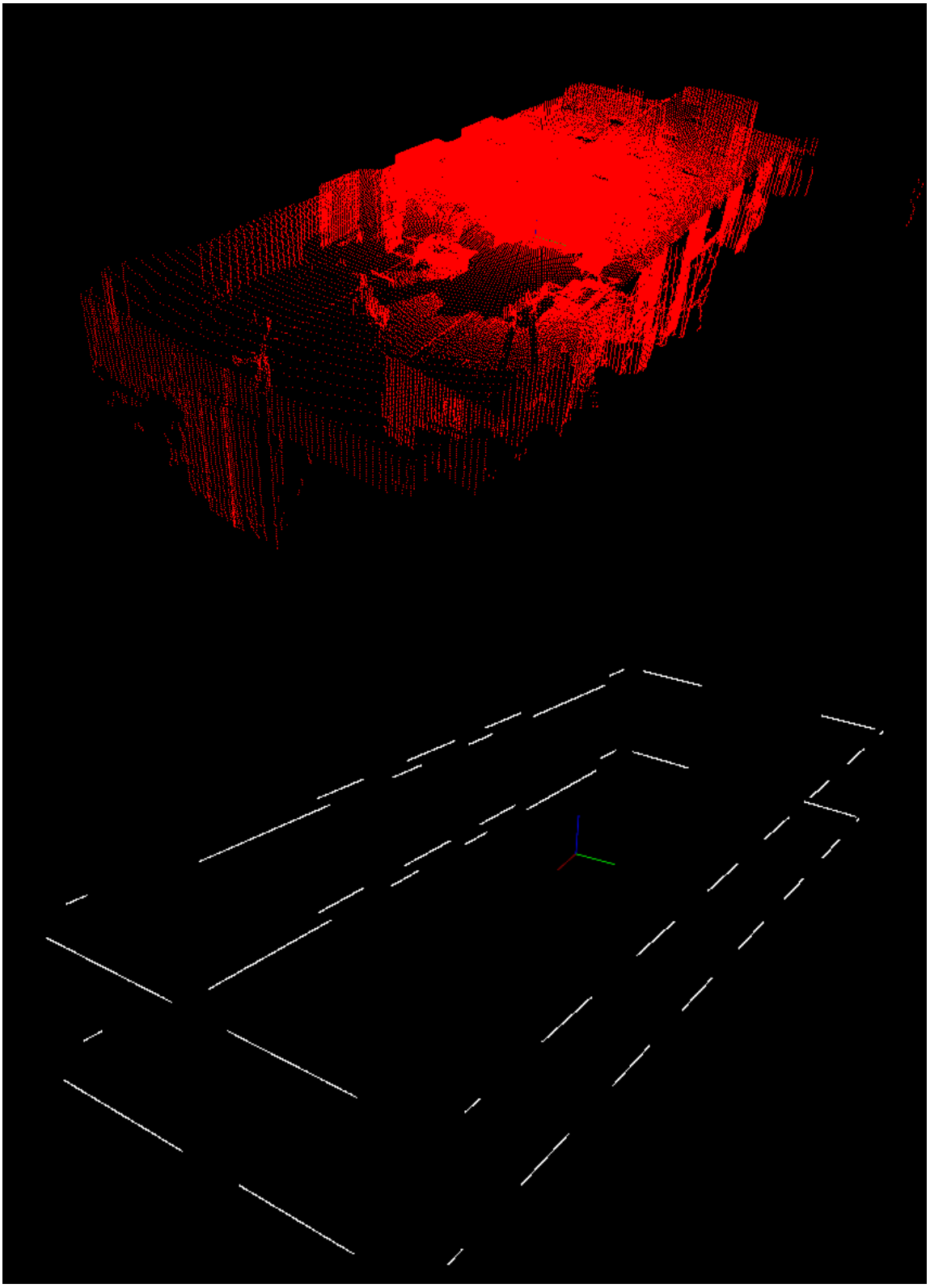


Figura 3.28. Ejemplo de obtención de la primera aproximación de las intersecciones.

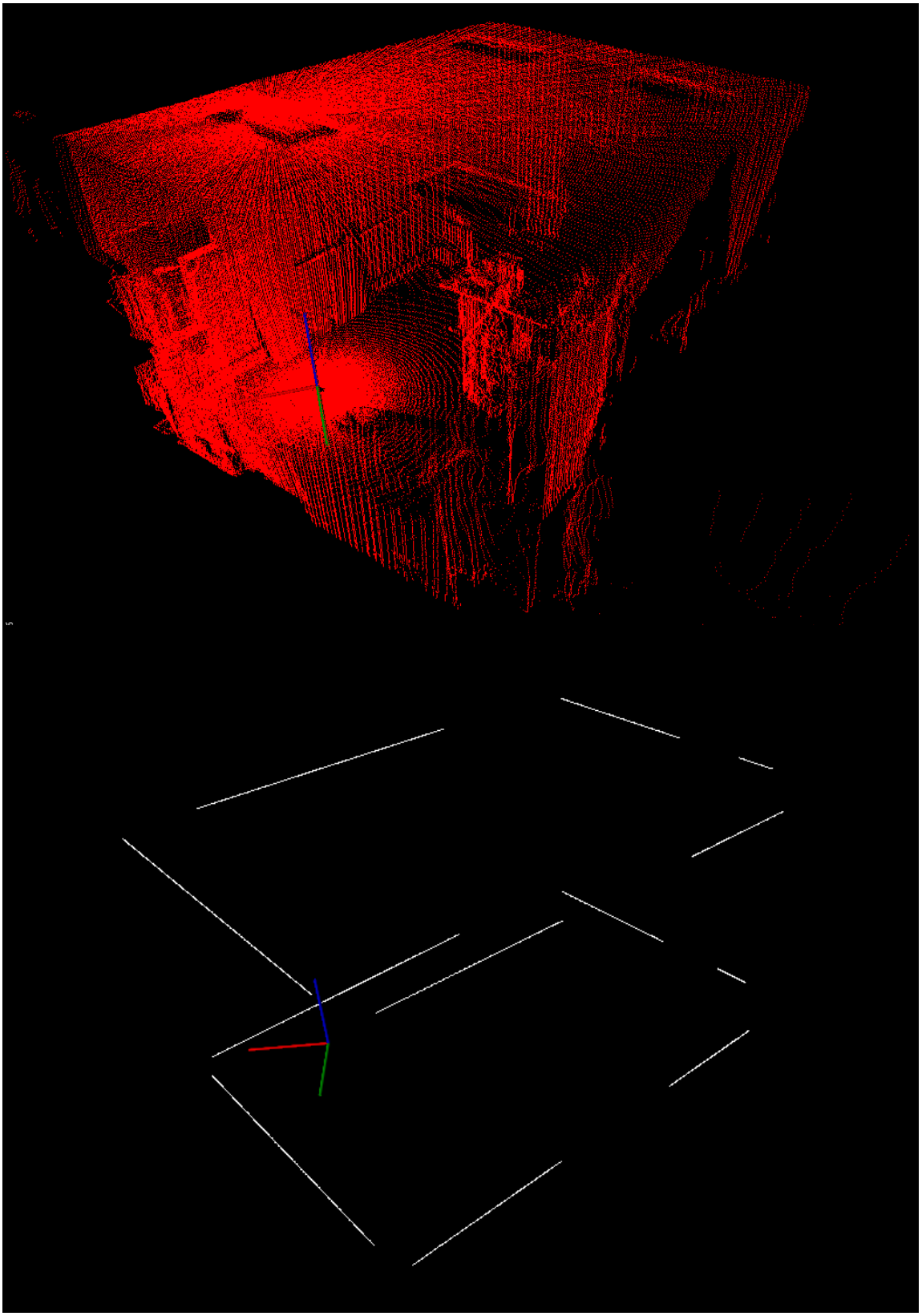


Figura 3.28. Ejemplo de obtención de la primera aproximación de las intersecciones.

Con los resultados mostrados se observa que las intersecciones calculadas se asemejan a las del espacio de captura, pero no son suficientes para reconstruir el recinto. Para mejorar la exactitud de la forma del recinto es necesario incorporar nuevos puntos. Estos nuevos puntos se añaden en un proceso posterior.

3.6. Detección de intersecciones erróneas

Como ya se comentó en el apartado de detección de planos, las superficies amplias tales como armarios, estanterías, cuadros, etc… pueden ser consideradas como superficies de paredes. Esto ocasiona que, en el momento de realizar las proyecciones para el cálculo de intersecciones, se tengan en cuenta planos que contienen superficies distintas a la de una pared. Al considerar estos planos se forman intersecciones que en la realidad no existen. En la *figura 3.29* de muestra un ejemplo de cómo considerar una superficie distinta a la de una pared puede ocasionar un error.

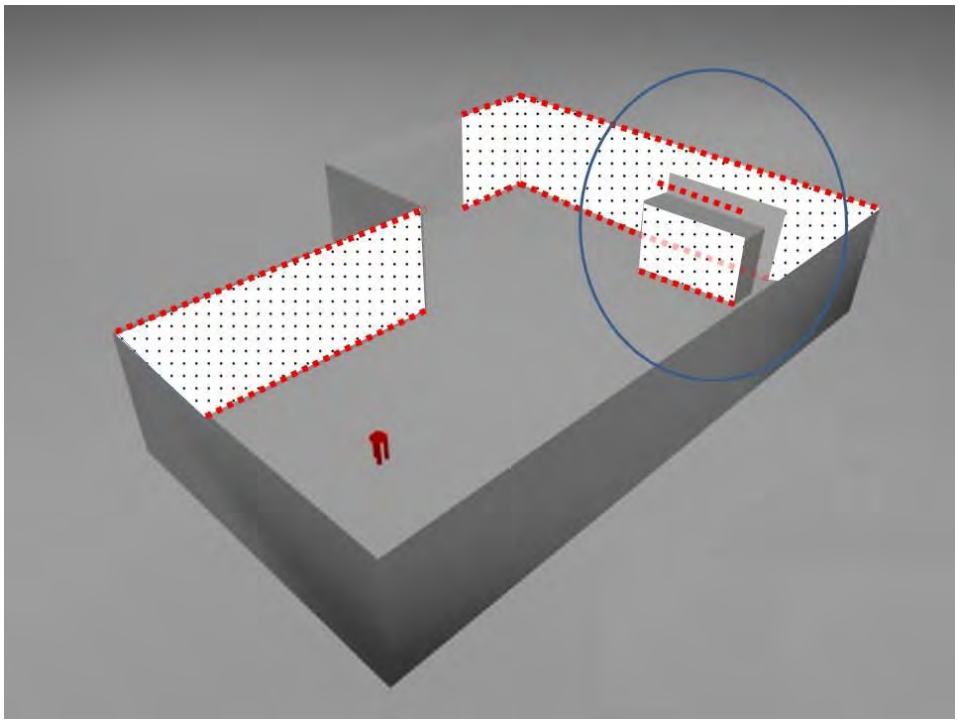


Figura 3.29. Ejemplo de obtención de intersección no real.

Este tipo de planos tienen que ser eliminados. Pueden existir tres tipos de planos erróneos:

- Los clasificados como “tipo V”.
- Los clasificado como “tipo H_s”.
- Los clasificados como “tipo H_i”.

El procedimiento para calcular los planos “tipo V” es distinto que el procedimiento para calcular los planos erróneos “tipo H_s” y “tipo H_i”. Para detectar los planos erróneos “tipo V”, el algoritmo se ayuda de la disposición del láser en el momento de la captura y de las primeras aproximaciones de intersección antes calculadas. Para ello se tienen en cuenta las siguientes afirmaciones:

- El origen de coordenadas cartesianas de la nube es el punto de captura del dispositivo láser 3D.
- El dispositivo láser realiza un barrido de 360º sobre los planos clasificados como “tipo H_s” y “tipo H_i”.
- No es posible que el dispositivo láser 3D capture, dentro de un mismo plano (clasificado como “tipo H_s” o “tipo H_i”), dos intersecciones entre superficies de pared que compartan un mismo intervalo de ángulo de barrido.

En la *figura 3.30* se muestra un ejemplo grafico de las dos primeras afirmaciones. Seguidamente, en la *figura 3.31* se muestra un ejemplo de la tercera afirmación. En ésta, se puede observar como, por el hecho de compartir un mismo ángulo de barrido, se puede detectar que una de las dos intersecciones procede de una superficie que no es de una pared. Con estas afirmaciones se localizan aquellas intersecciones que no corresponden a las superficies de alguna de las paredes. Una vez localizada se elimina el plano clasificado como “tipo V” que la generó.

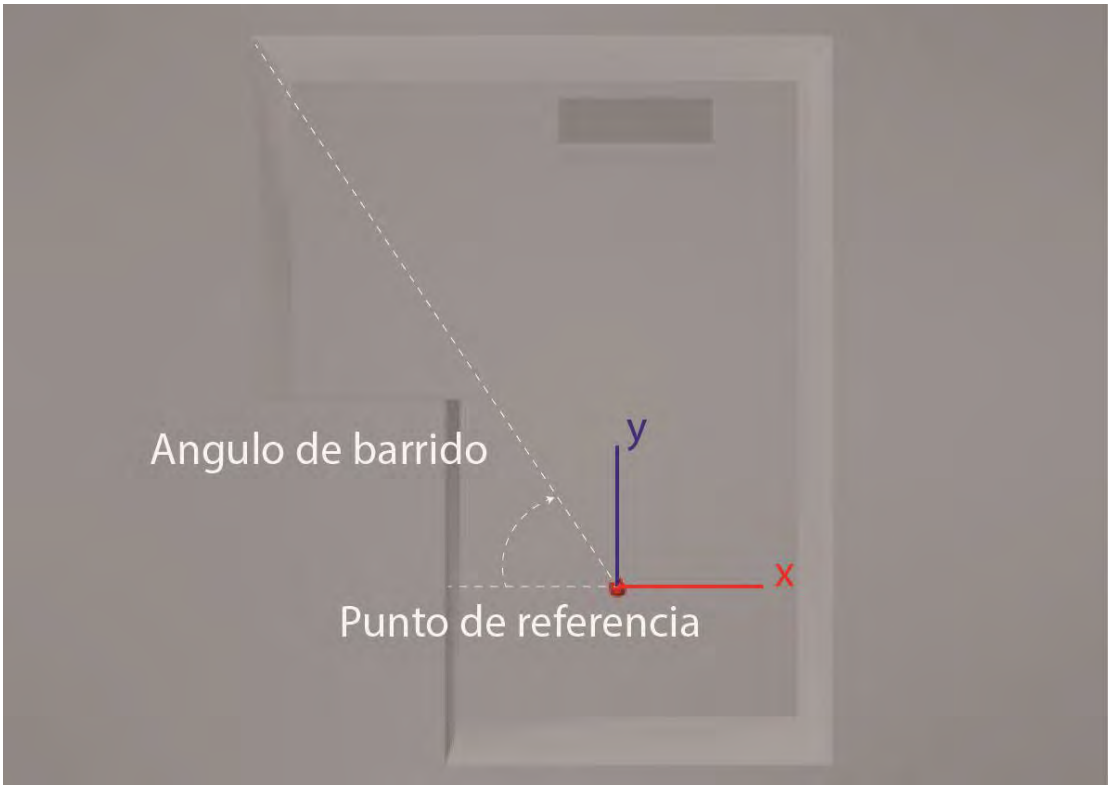


Figura 3.30. Definición de ángulo de barrido y de punto de referencia.

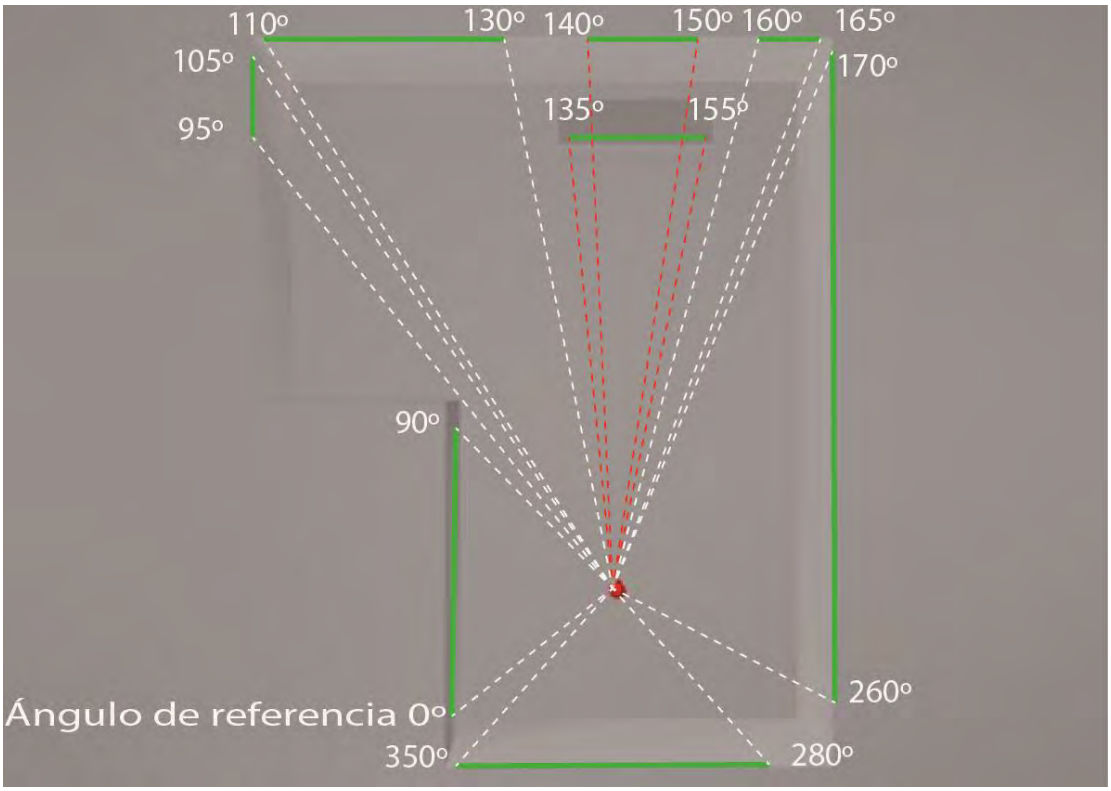


Figura 3.31. Detección de una intersección imposible.

Para detectar los planos erróneos clasificados como "tipo H_s" y "tipo H_i", el algoritmo se ayuda de la distancia del láser al plano. De esta manera, el algoritmo calcula las distancias del dispositivo láser de los todos los planos con esta clasificación. Después descarta todos los planos salvo el más alejado clasificado como "tipo H_s" y el más alejado clasificado como "tipo H_i". Junto a los planos se eliminan todas las intersecciones que hayan generado los planos eliminados. En la *figura 3.32* se muestra un ejemplo de un plano erróneo "tipo H_i" y en la *figura 3.33* como se detecta.

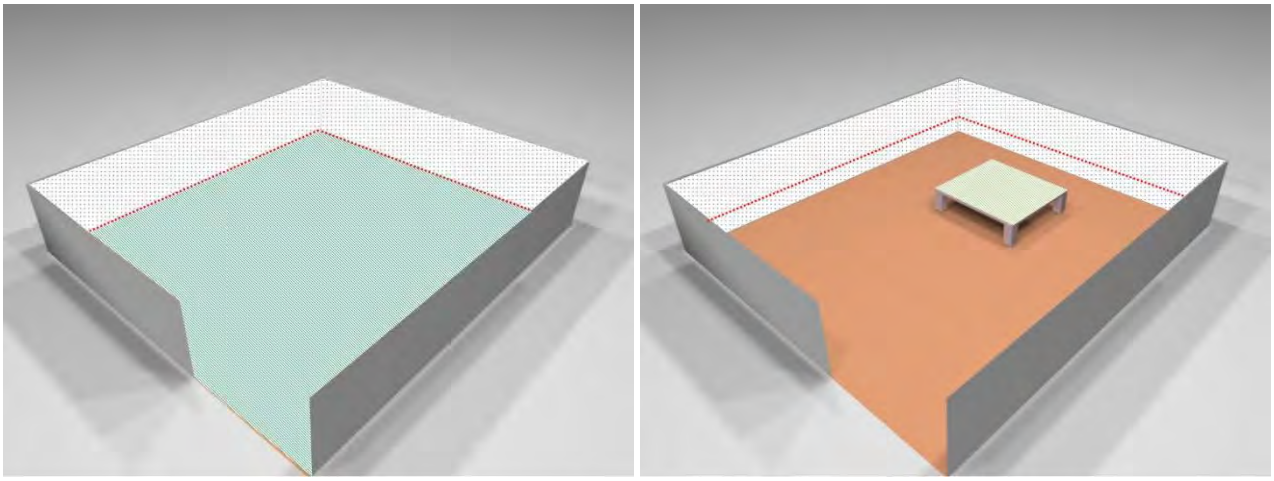


Figura 3.32. Proyección correcta debida al plano del suelo (izquierda). Proyección incorrecta debida al plano de la mesa (derecha).

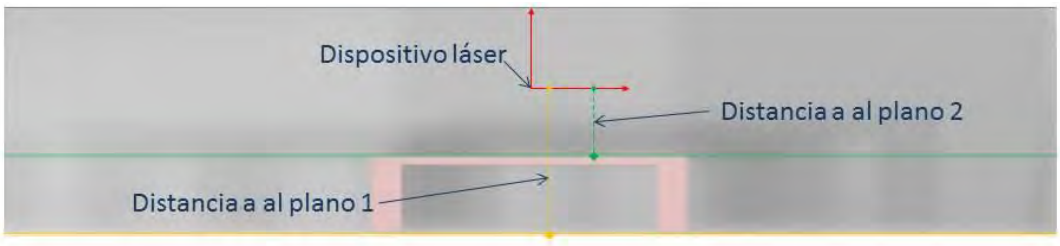


Figura 3.33. Como detectar el plano clasificado como "tipo H_i" erróneo mediante el cálculo de las distancias de los planos al origen de coordenadas.

3.7. Obtención de puntos de reconstrucción

El objetivo de este proceso es el de generar un listado de puntos que formen los vértices de los diferentes polígonos cerrados que describen las superficies del recinto cerrado. A estos puntos se les denomina puntos de reconstrucción del recinto. En la *figura 3.34* se muestra los puntos de reconstrucción de un recinto cerrado de ejemplo. Para generar estos puntos se calculan las soluciones de sistemas de tres ecuaciones. Dos de las tres ecuaciones de estos sistemas son ecuaciones de planos "tipo V". La tercera ecuación de los sistemas es la ecuación del plano "tipo H_s" o la del plano "tipo H_i". Cabe recordar que solo debe existir un plano "tipo H_s" y otro plano "tipo H_i". En la *figura 3.35* se muestra un ejemplo grafico de cómo se generan los puntos de reconstrucción de un recinto cerrado de ejemplo. El problema residen en saber con qué combinación de planos "tipo V" generar el sistema de ecuaciones ya que no todas las combinaciones con correctas. En la *figura 3.36* se demuestra gráficamente como no todas las combinaciones de planos "tipo V" generan una solución correcta. Para generar las combinaciones de planos "tipo V" se utilizan las los segmentos generados en los procesos anteriores. Estos segmentos, tal como se muestra en la *figura 3.37,* están formados por los puntos de sus extremos y describen una aproximación de las intersecciones entre superficies contenidas en planos "tipo V" con superficies contenidas en planos "tipo H_s" o en planos "tipo H_i".

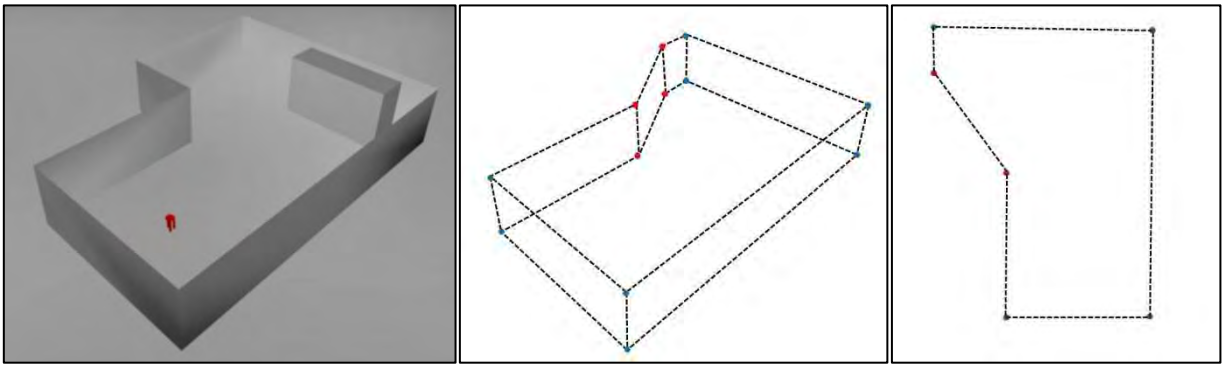


Figura 3.34. Recinto representado mediante la conexión de sus puntos de reconstrucción.

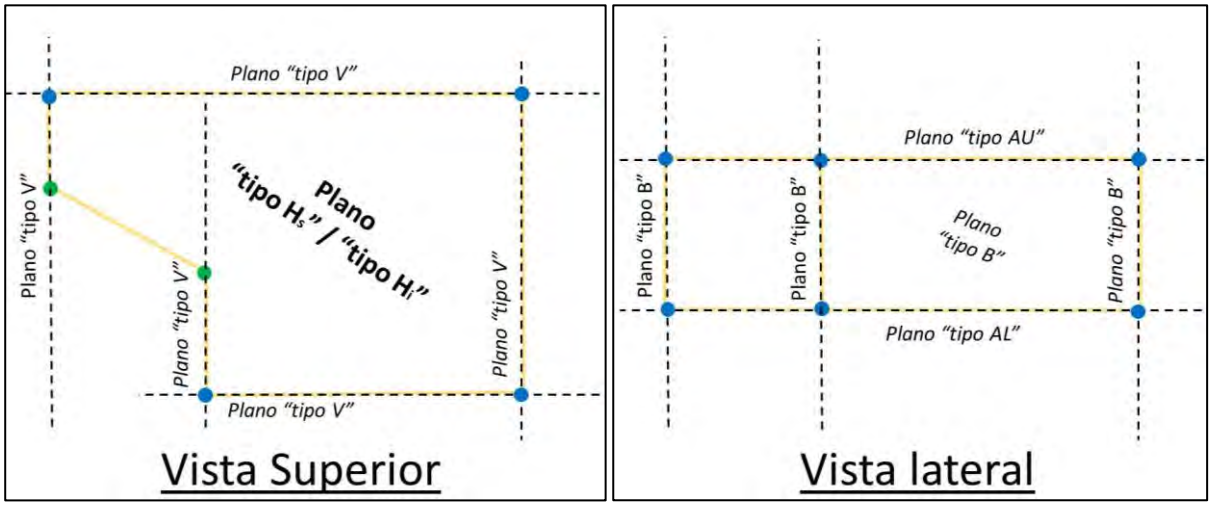


Figura 3.35. Obtención de los puntos de reconstrucción mediante intersecciones de planos.

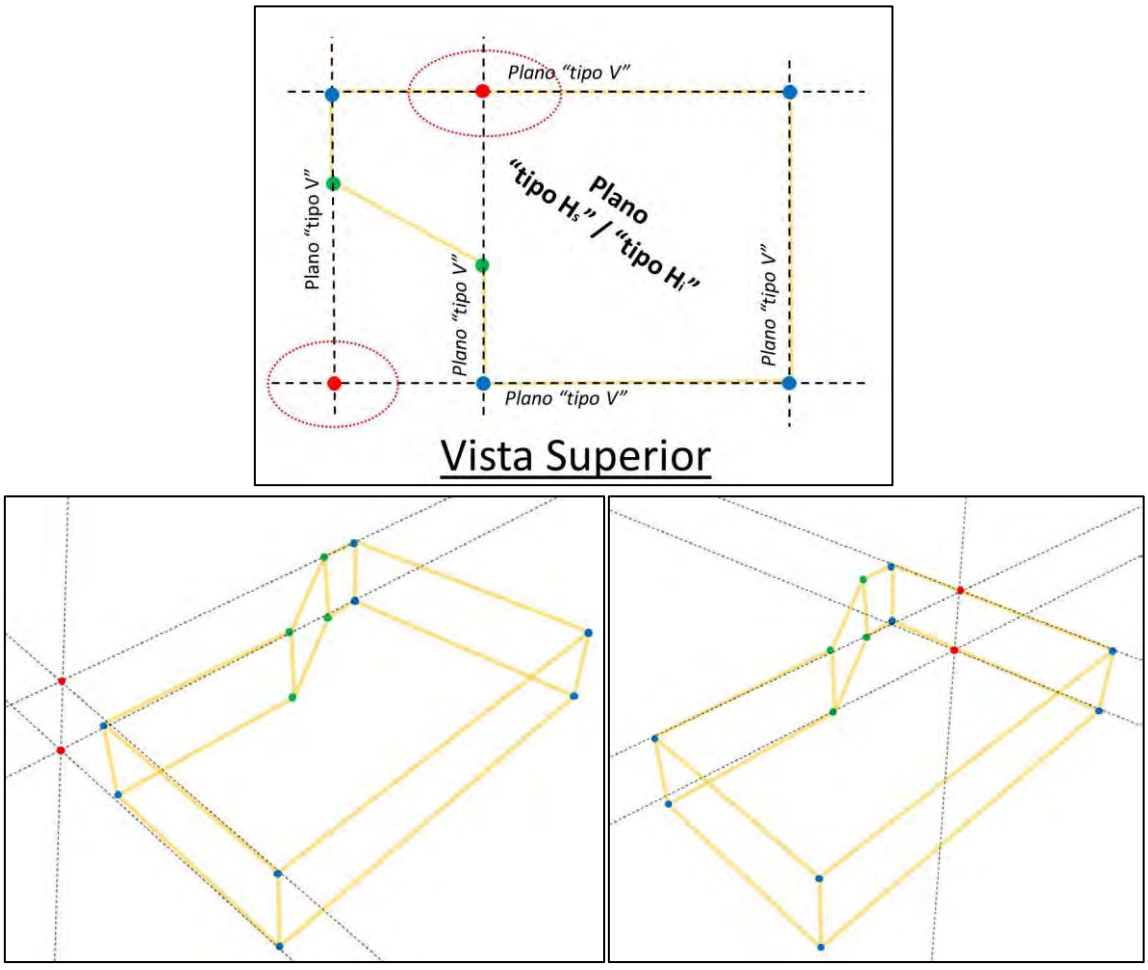


Figura 3.36. Obtención de puntos de intersección erróneos.

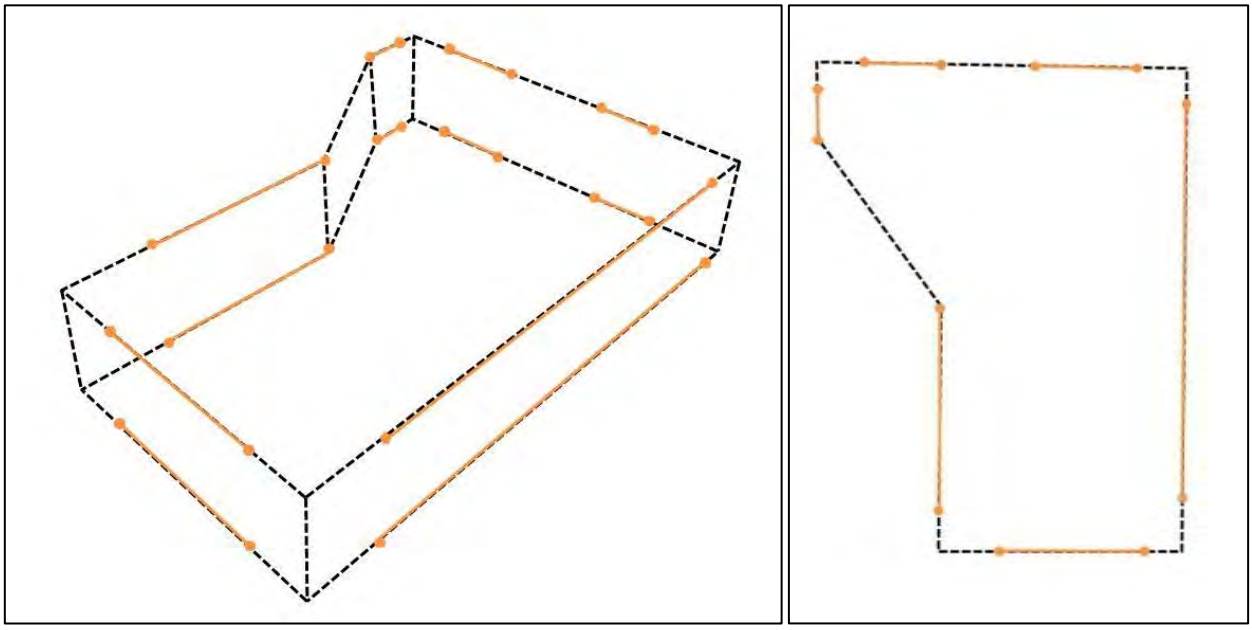
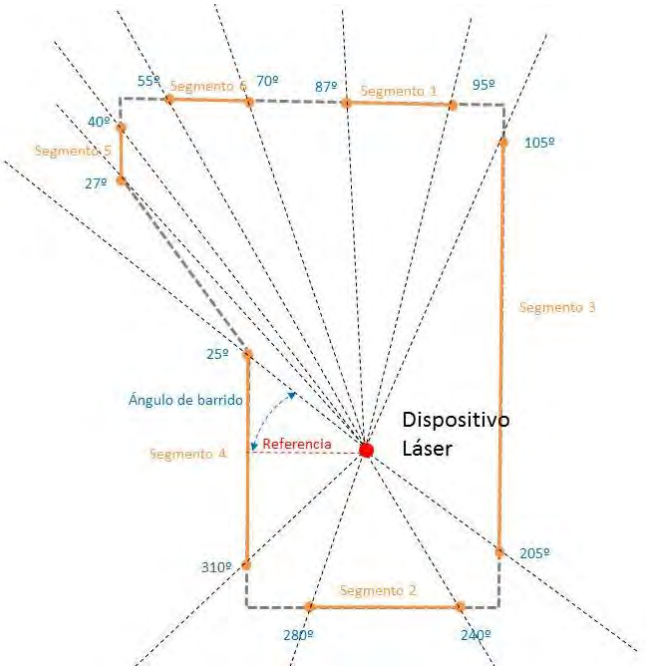


Figura 3.37. Segmentos que representan las diferentes intersecciones obtenidas.

A continuación se detallan los pasos del procedimiento para calcular los puntos de reconstrucción mediante la generación de sistemas de ecuaciones. Solo se van a detallar los pasos para generar los puntos de reconstrucción mediante sistemas de ecuaciones que contienen la ecuación del plano "tipo H_s". Los pasos para generar los puntos de reconstrucción mediante sistemas de ecuaciones que contienen la ecuación del plano "tipo H_i" son similares, salvo por el hecho de utilizar el plano "tipo H_i" en lugar del "tipo H_s".

1. Se identifican todos los segmentos que estén contenidos en el plano "tipo H_s". Una vez identificados se calculan los intervalos de ángulos de barrido de cada segmento. En la *figura 3.38* se muestra en ejemplo grafico de cómo se calculan los intervalos de ángulos de barrido de cada segmento.



Segmento	Ángulo inicial	Ángulo final
Segmento 1	87º	95º
Segmento 2	240º	280º
Segmento 3	105º	205º
Segmento 4	25º	310º
Segmento 5	27º	40º
Segmento 6	55º	70º

Figura 3.38. Obtención de los ángulos de barrido de los segmentos.

2. Se ordenan los segmentos de mayor a menor los segmentos según su intervalo de ángulos de barrido. En la *figura 3.39* se muestra un ejemplo.

Segmento	Ángulo inicial	Ángulo final
Segmento 1	87º	95º
Segmento 2	240º	280º
Segmento 3	105º	205º
Segmento 4	25º	310º
Segmento 5	27º	40º
Segmento 6	55º	70º

Segmento	Ángulo inicial	Ángulo final
Segmento 5	27º	40º
Segmento 6	55º	70º
Segmento 1	87º	95º
Segmento 3	105º	205º
Segmento 2	240º	280º
*Segmento 4	310º	25º

Figura 3.39. Lista de segmentos sin ordenar (izquierda). Lista de segmentos ordenados (derecha).

3. Se compara el ángulo de cada segmento de la lista con el ángulo del segmento inmediatamente superior. Es decir se compara el ángulo que hay entre segmento situado en la posición "i" de la lista y el segmento situado en la posición "i+1" de la lista. En el caso de ser "i" la última posición de la lista, se compara con el segmento situado en la primera posición. En *la figura 3.40* se muestra un ejemplo gráfico de cómo se realiza la comparación. Se toma una decisión en función del ángulo.

   - Si el ángulo es mayor que un cierto valor establecido se considera que el sistema de ecuaciones sí que tiene solución. Se muestra un ejemplo grafico de este caso en la *figura 3.41*. El punto de reconstrucción se calcula resolviendo el sistema de ecuaciones formado por:

     - La ecuación del plano "tipo V" al que pertenece el segmento situado en la posición "i" de la lista.

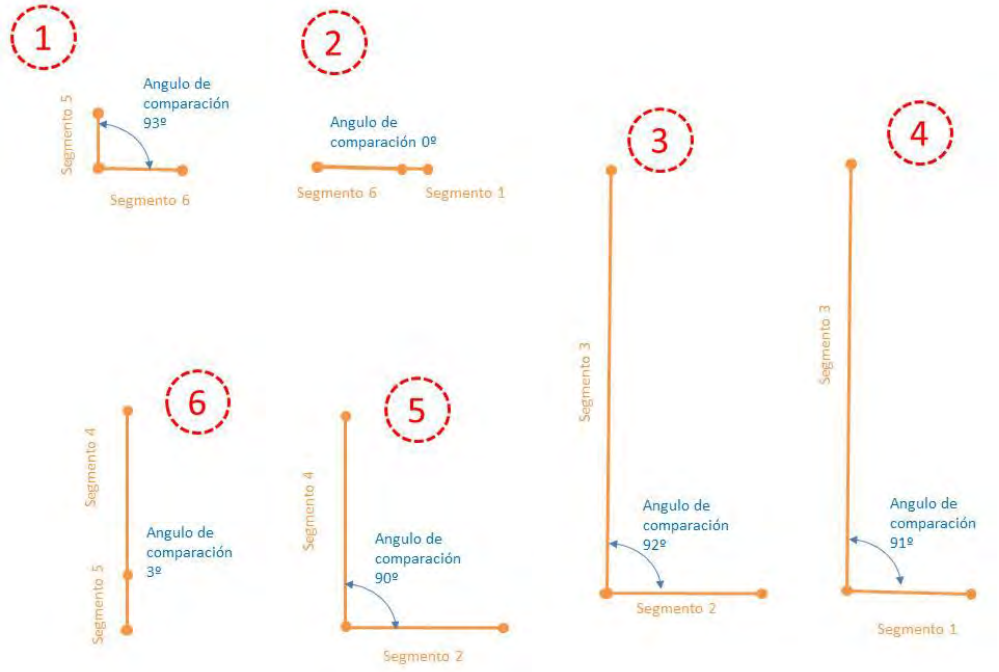
     - La ecuación del plano "tipo V" al que pertenece el segmento situado en la posición "i+1" de la lista. En el caso de ser "i" la última posición de la lista, se usa el plano "tipo V" del segmento situado en la primera posición de la lista.

     - La ecuación del plano "tipo H_s".

   - Si el ángulo es menor que un cierto valor establecido se considera que el sistema de ecuaciones no tiene solución. En este caso se comprueban si el plano "tipo V" de ambos segmentos es el mismo. En el caso de ser el mismo plano no se genera ningún punto de reconstrucción. En la *figura 3.42* se muestra un ejemplo de este caso. En el caso de no ser el mismo generan dos puntos de reconstrucción. En la *figura 3.43* se muestra un ejemplo de este otro caso. Los puntos se reconstrucción son:

     - El punto de final del segmento "i".

     - El punto de inicio del segmento "i+1". En el caso de ser "i" la última posición de la lista, se usa el segmento en la primera posición en lugar del segmento en la posición "i+1".



Segmento		Ángulo inicial	Ángulo final
1	Segmento 5	27º	40º
2	Segmento 6	55º	70º
3	Segmento 1	87º	95º
4	Segmento 3	105º	205º
5	Segmento 2	240º	280º
6	*Segmento 4	310º	25º

Figura 3.40. Comparación de ángulos de segmento (arriba). Tabla de comparación (abajo).

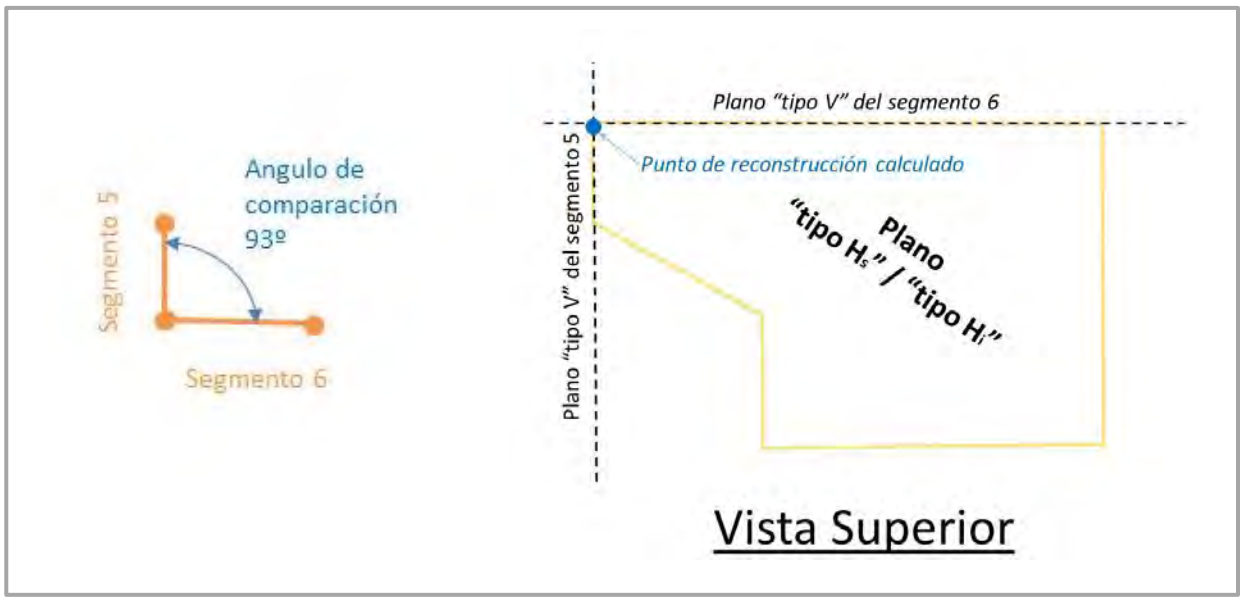


Figura 3.41.Caso de ángulo de comparación mayor (izquierda). Cálculo del punto de reconstrucción mediante resolución del sistema de ecuaciones (derecha).

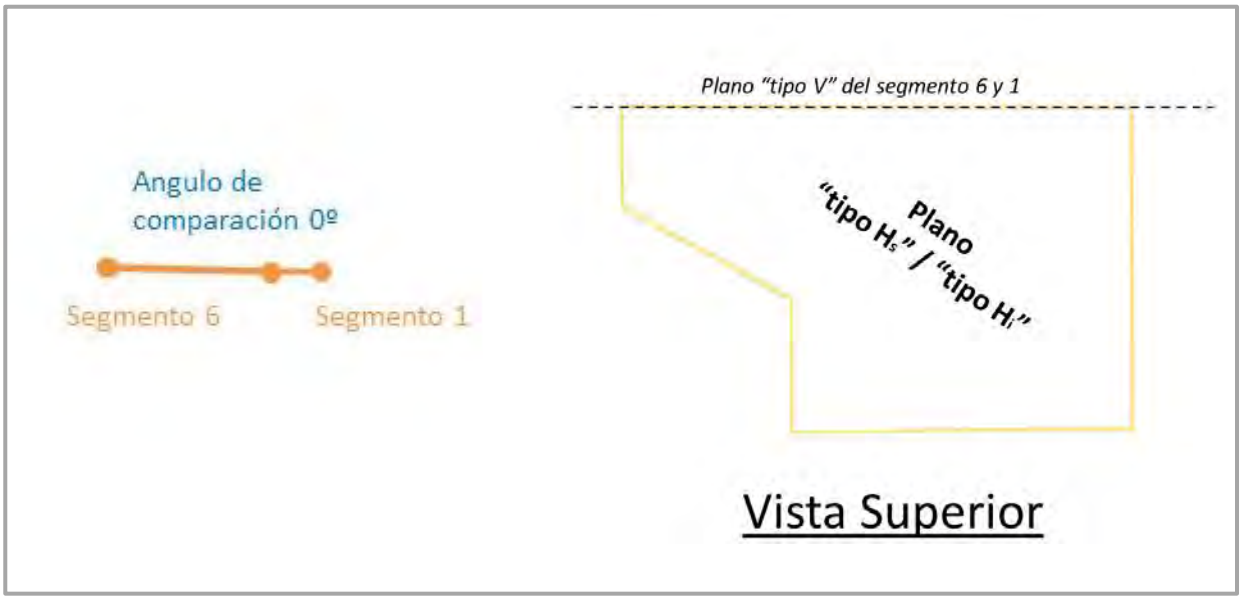


Figura 3.42.Caso de ángulo de comparación menor (izquierda). No se calcula punto de reconstrucción al pertenecer los segmentos al mismo plano "tipo V" (derecha).

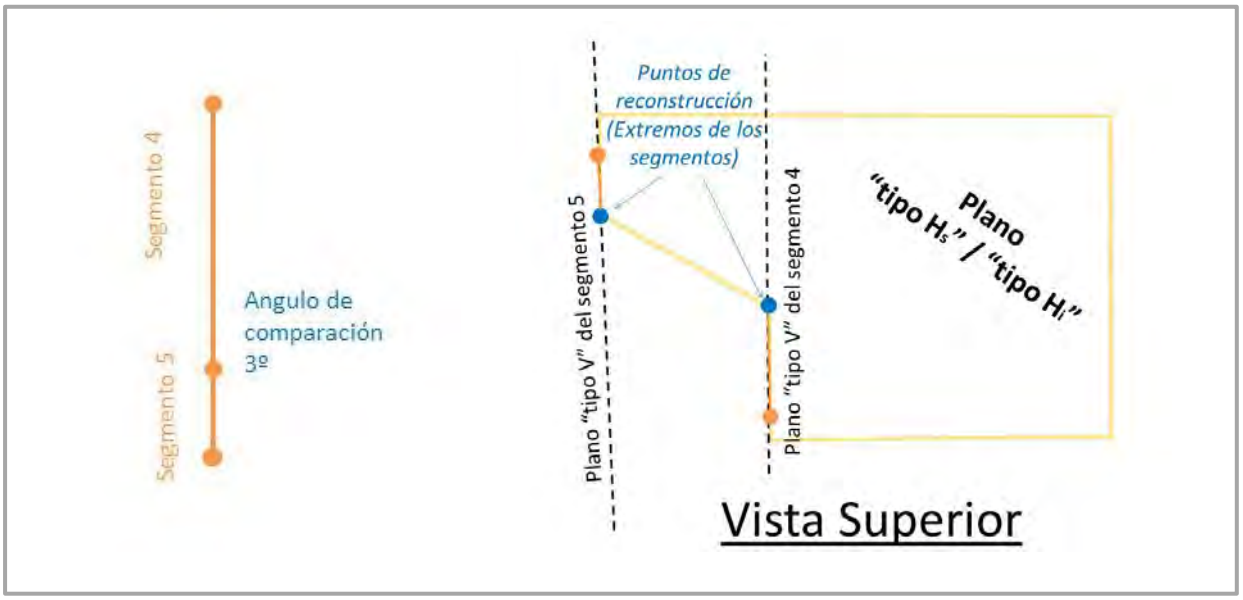


Figura 3.43.Caso de ángulo menor (izquierda). Cálculo de los puntos de reconstrucción mediante los extremos de los segmentos (derecha).

Cada punto de reconstrucción generado se almacena junto con las ecuaciones generales de los planos donde está contenido. Tras repetir estos pasos para el plano "tipo H_s" se tendrá completa la lista de puntos de reconstrucción tal como se mostró en la *figura 3.34*.

3.8. Formación de polígonos cerrados que describan las superficies de las paredes del recinto

Llegados a este punto, solo queda crear los polígonos cerrados que describan las superficies de las paredes. Estos polígonos se forman al crear conexiones entre los puntos de reconstrucción guardados. Estas conexiones entre puntos forman las aristas de los polígonos cerrados. El problema reside en cómo conectar los puntos de reconstrucción ya que existen multitud de combinaciones posibles. La metodología establecida para generar las conexiones es la siguiente. Se proyectan los puntos de reconstrucción sobre la superficie lateral de un cilindro de radio arbitrario. El eje de giro de este cilindro pasa por el origen de coordenadas de la nube y su orientación es la del eje Z. Cabe recordar que el origen de coordenadas de la nube es el punto donde se encuentra situado el dispositivo láser y que el eje Z de la nube es el eje Z del dispositivo láser. Manteniendo la relación entre el punto proyectado y el punto original, se generan conexiones entre los puntos proyectados de manera que se formen las superficies del cilindro. Cada conexión que se realice sobre los puntos proyectados se realiza sobre los puntos originales.

En la secuencia de *figuras 3.44, 3.45. 3.46 y 3.47* se muestra un gráficamente de cómo generar, con ayuda de un cilindro, las conexiones en un recinto de ejemplo. Inicialmente en la *figura 3.44* se muestran las intersecciones que existen en el recinto de ejemplo y las intersecciones que el dispositivo es capaz de capturar. Después, en la *figura 3.45* se muestran los diferentes puntos de reconstrucción que se obtienen realizando los procesos mencionados hasta el momento. En *la figura 3.46* se muestra como quedan proyectados los puntos sobre la superficie lateral del cilindro. Se proyectan todos los puntos de reconstrucción existentes. Por último, en *la figura 3.47* se muestra como, mediante la construcción de una figura cilíndrica con los puntos proyectados, se generan las conexiones necesarias en los puntos originales para reconstruir el recinto.

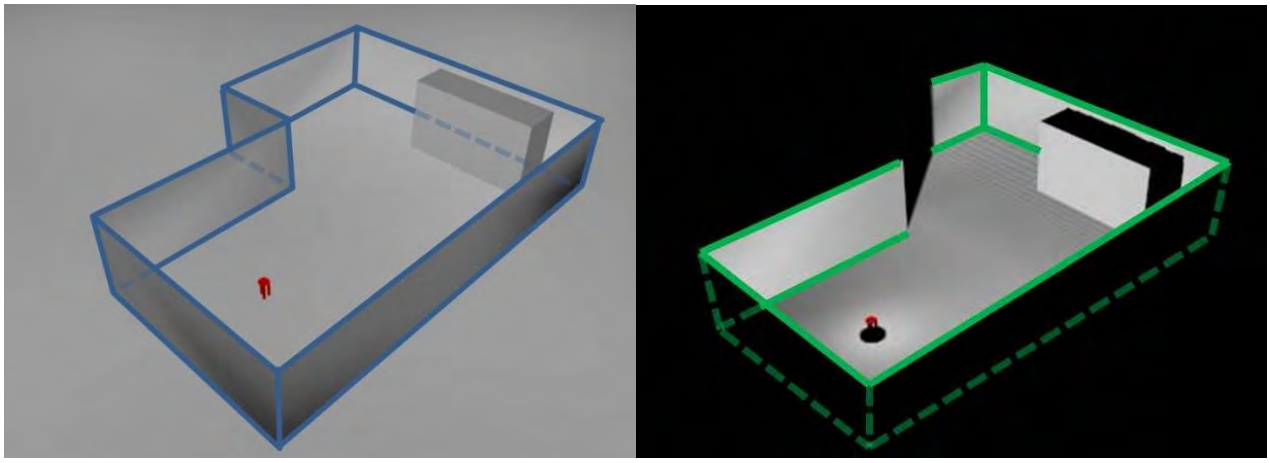


Figura 3.44. (Izquierda) Intersecciones entre superficies existentes en un recinto. (Derecha) Intersecciones entre superficies que el dispositivo láser puede captura.

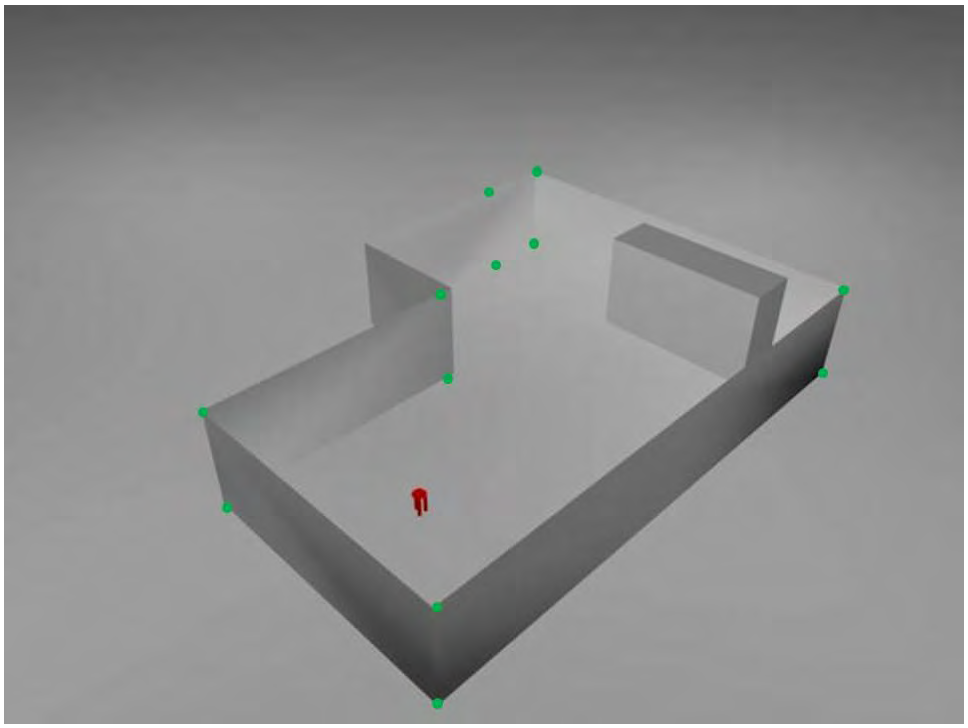


Figura 3.45. Puntos de reconstrucción.

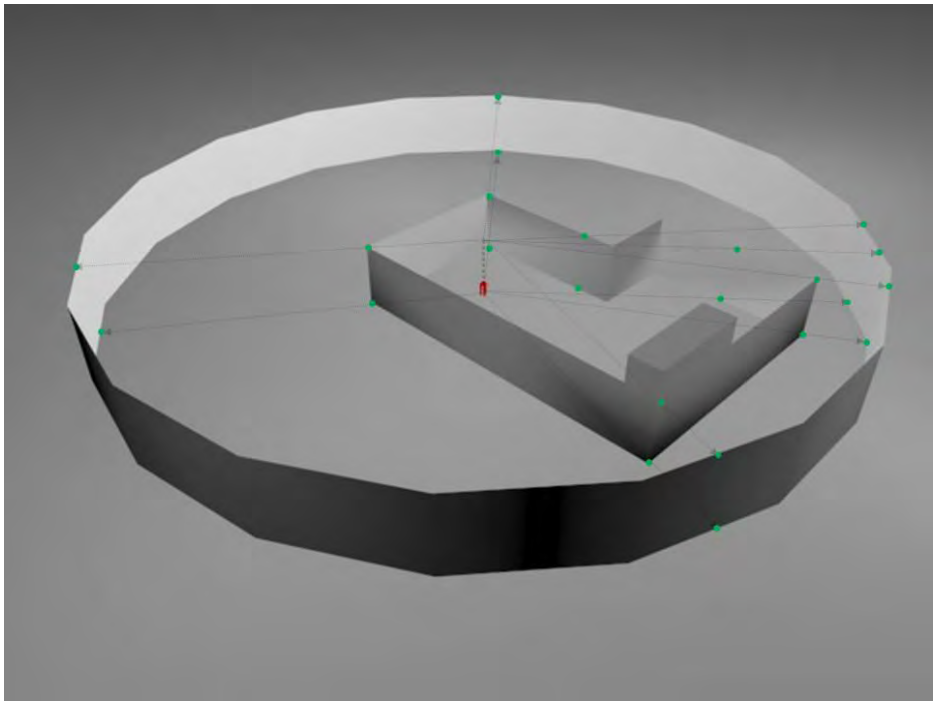


Figura 3.46. Proyección de los puntos de reconstrucción sobre la superficie de un cilindro.

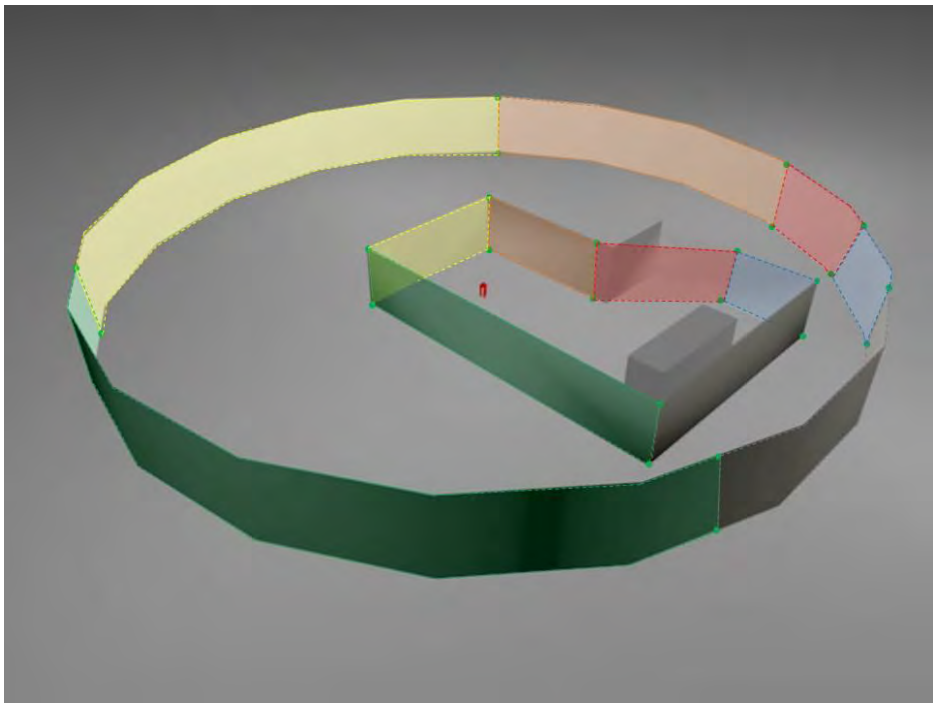


Figura 3.47. Generación de conexiones entre puntos de reconstrucción gracias a la formación del cilindro con los puntos proyectados.

El hecho de proyectar los puntos sobre la superficie de un cilindro para generar las conexiones tiene como ventaja que sea cual sea la forma del recinto, el mecanismo para generar los las conexiones entre puntos es siempre el mismo, el de formar el cilindro. Este mecanismo para unir los puntos proyectos sobre la superficie cilindra se divide en dos partes:

- Formación de superficies de las tapas del cilindro.
- Formación superficies laterales del cilindro.

Formación las superficies de las tapas del cilindro

Para formar estas superficies se realizan conexiones entre puntos proyectados cuyos puntos originales estén contenidos en planos clasificados con el mismo tipo. Esto quiere decir que para generar el polígono que describa la superficie de la tapa superior, se crean conexiones entre puntos proyectados cuyos puntos originales estén contenidos en planos clasificados como "tipo H_s". De la misma manera, para generar la circunferencia que describa la superficie inferior, se crean conexiones entre si puntos proyectados cuyos puntos originales estén contenidos en planos clasificados como "tipo H_i".

Para realizar las conexiones se realizan los siguientes pasos:

1. Se generan dos listas.
    - Una lista con todos los puntos proyectados cuyos puntos originales están contenidos en un plano clasificado como "tipo H_s".
    - Una lista con todos los puntos proyectados cuyos puntos originales están contenidos en planos clasificados como "tipo H_i".
2. Se ordenan los puntos de las listas de menor a mayor ángulo central. El ángulo central se calcula con respecto a un punto de referencia tal como se muestra en *la figura 3.48*.
3. Con las listas de puntos ordenadas se conecta, dentro de cada lista, cada punto con su inmediato superior y su inmediato inferior. De esta manera, dentro de

una misma lista, el punto en la posición “i” de queda conectado con el punto en la posición “i+1” y con el punto en la posición “i-1”.

4. Para generar una conexión cerrada, se conecta dentro de cada lista, el punto en primera posición con el punto en última posición.

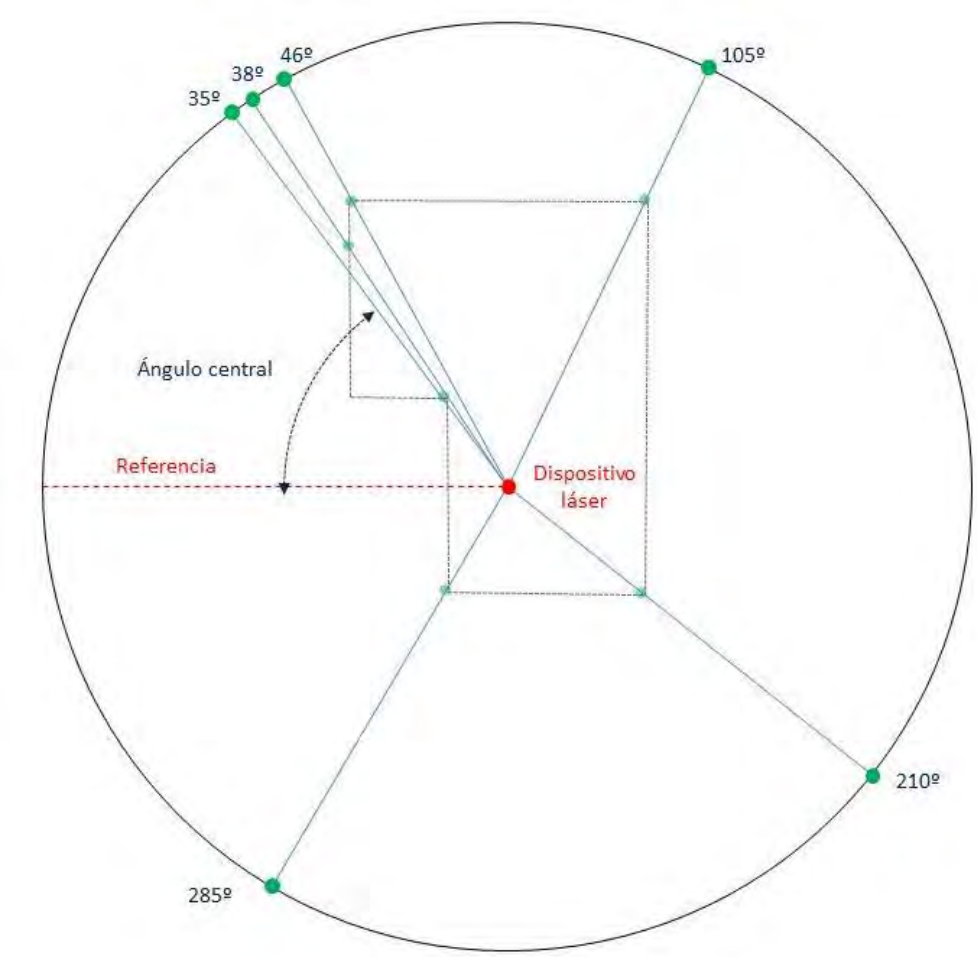


Figura 3.48. Calculo de ángulos centrales.

En la *figura 3.49* se muestra un ejemplo de la generación estas superficies. En la figura se observa como al general las conexiones necesarias para generar las

superficies de las tapas del cilindro, y realizando las mismas conexiones en los puntos originales, se generan dos polígonos que describen dos superficies del recinto.

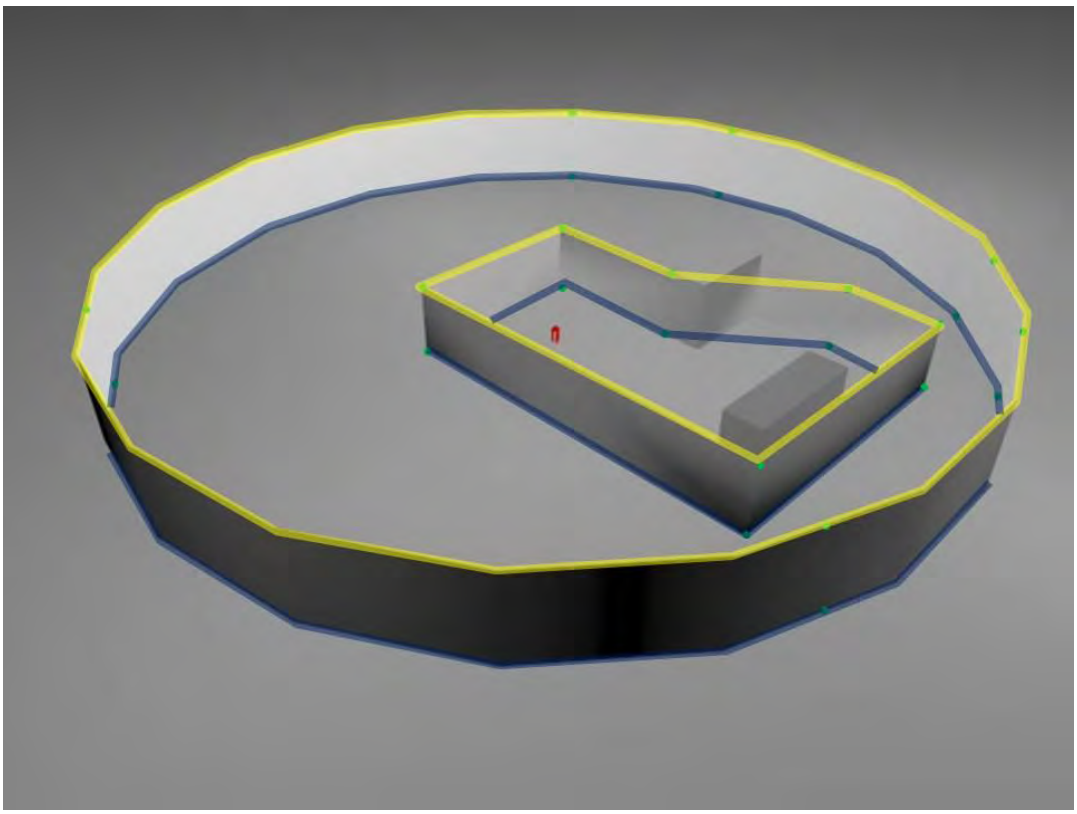


Figura 3.49. Ejemplo de formación de las circunferencias de las superficies de las caras superior e inferior del cilindro.

Formación de las superficies laterales del cilindro.

Para formar estas superficies se conectan: puntos proyectados cuyos puntos originales están contenidos en planos clasificados como "tipo H_s", con puntos proyectados cuyos puntos originales están contenidos en planos clasificados como "tipo H_i". Estas conexiones, más las conexiones formadas en el paso anterior, generan las diferentes superficies que componen la superficie lateral del cilindro. Realizando estas mismas conexiones en los puntos originales se generan el resto de los polígonos que describen las superficies del recinto.

Para realizar las conexiones se realizan los siguientes pasos:

1. Se generan dos listas.
    - Una con todos los puntos proyectados cuyos puntos originales están contenidos en un plano clasificado como "tipo H_s",
    - Otra con todos los puntos proyectados cuyos puntos originales están contenidos en planos clasificados como "tipo H_i".
2. Se ordenan los puntos de las listas de menor a mayor ángulo central. El ángulo central se calcula con respecto a un punto de referencia tal como se muestra en la *figura 3.48.*
3. Con las listas de puntos ordenadas, se conecta un punto de la primera lista con un punto de la segunda lista según su posición en esta. De esta manera se conecta el punto situado en la posición "i" de la primera lista con el punto situado en la posición "i" de la segunda lista.

En la *figura 3.50* se muestra un ejemplo de la generación estas superficies. Se observa como al general las conexiones necesarias para formar las superficies laterales del cilindro, y realizando las mismas conexiones en los puntos originales, se generan los polígonos que describen en resto de las superficies del recinto cerrado.

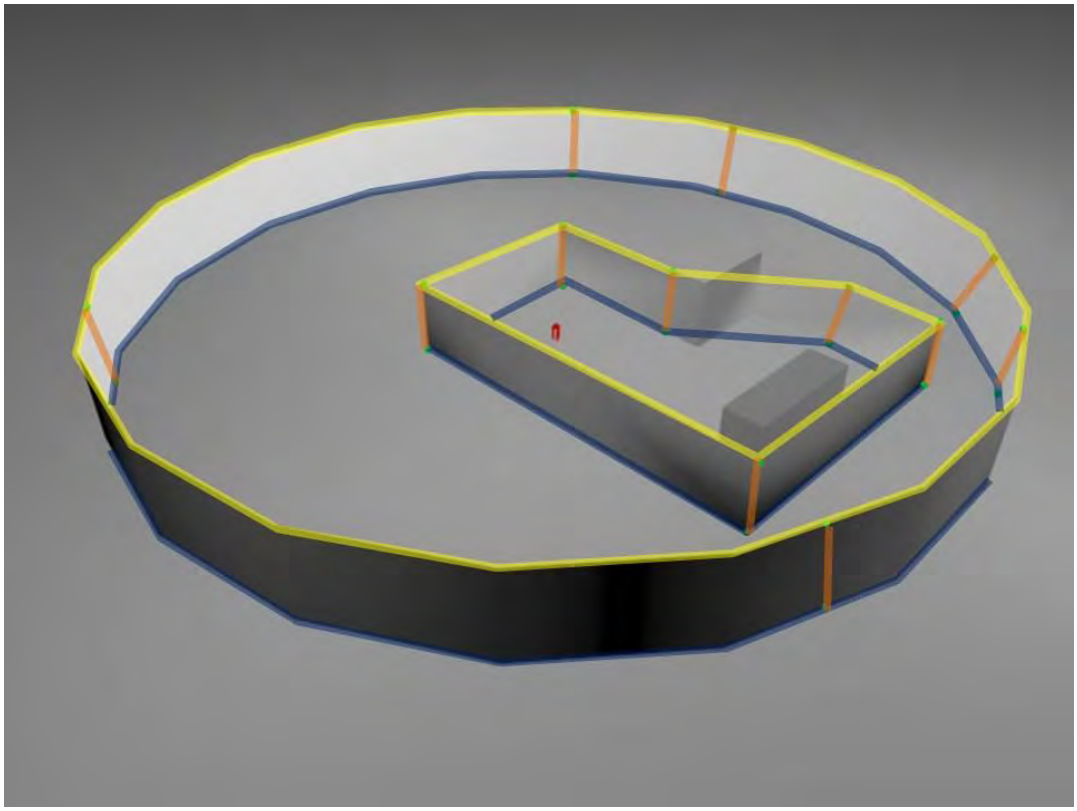


Figura 3.50. Ejemplo de formación de polígonos de las superficies de la cara lateral del cilindro.

Tras la realizar este proceso en nubes reales se muestran los resultados mostrados en las *figuras 3.51* y *3.52*. Se puede observar cómo tras generar las conexiones pertinentes entre los puntos proyectados para formar el cilindro, realizando las mismas conexiones en los puntos originales equivalentes, se genera el modelo geométrico del recinto.

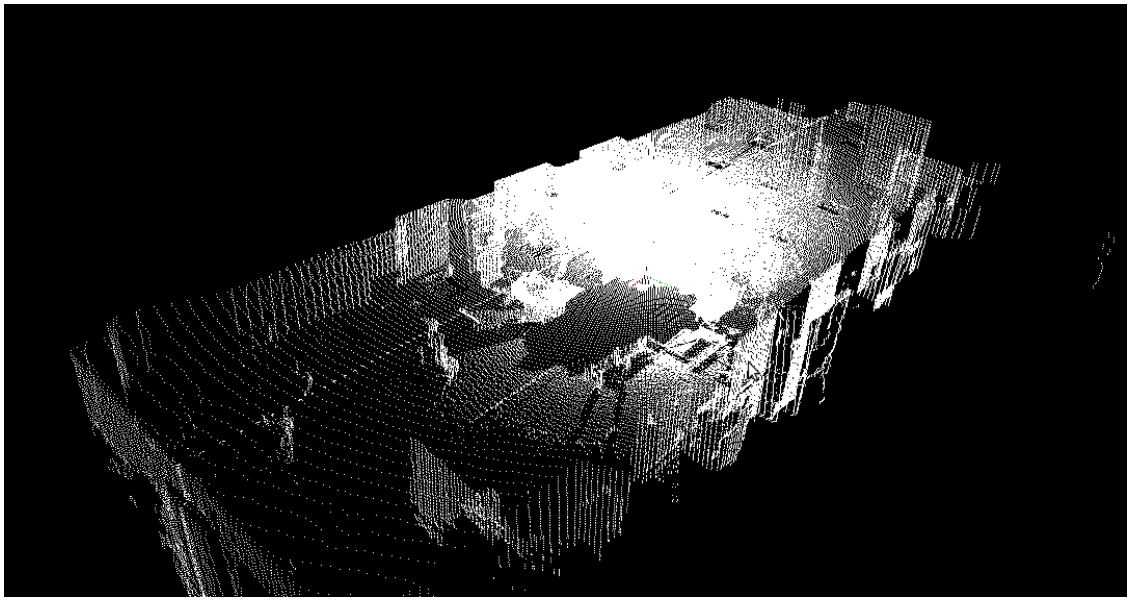


Figura 3.42-A. Nube original.

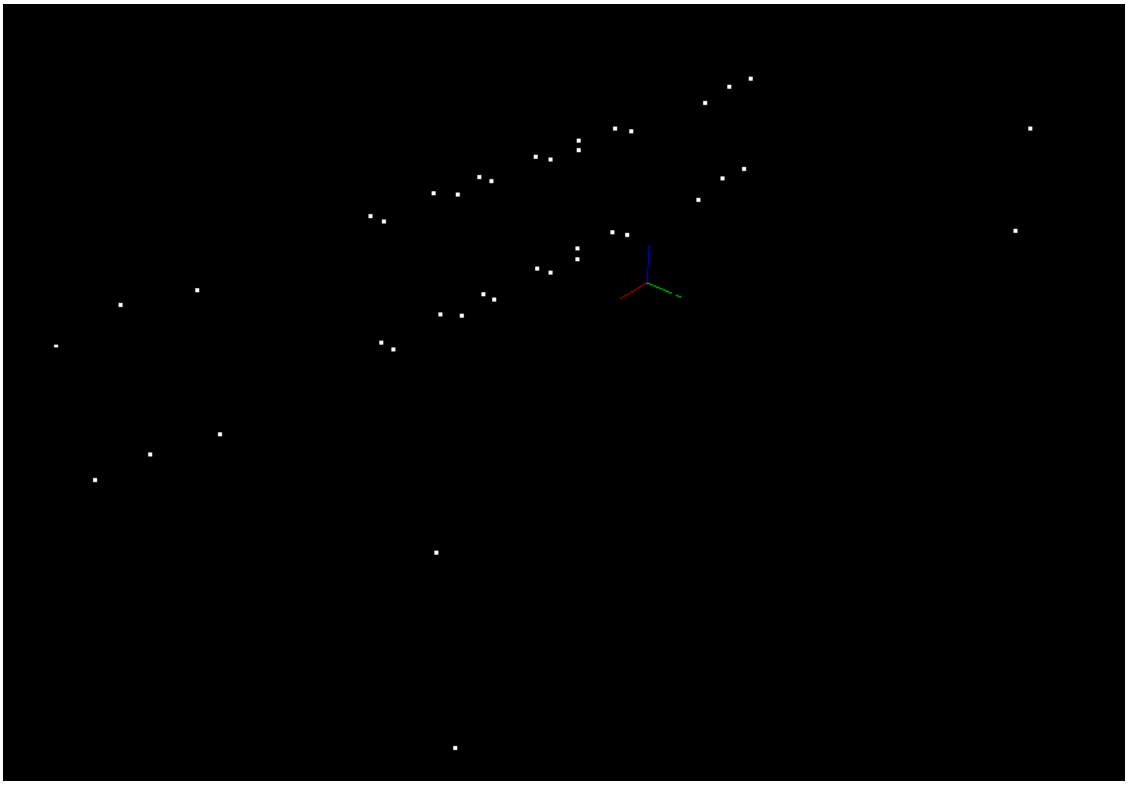


Figura 3.42-B. Primera aproximación de las intersecciones.

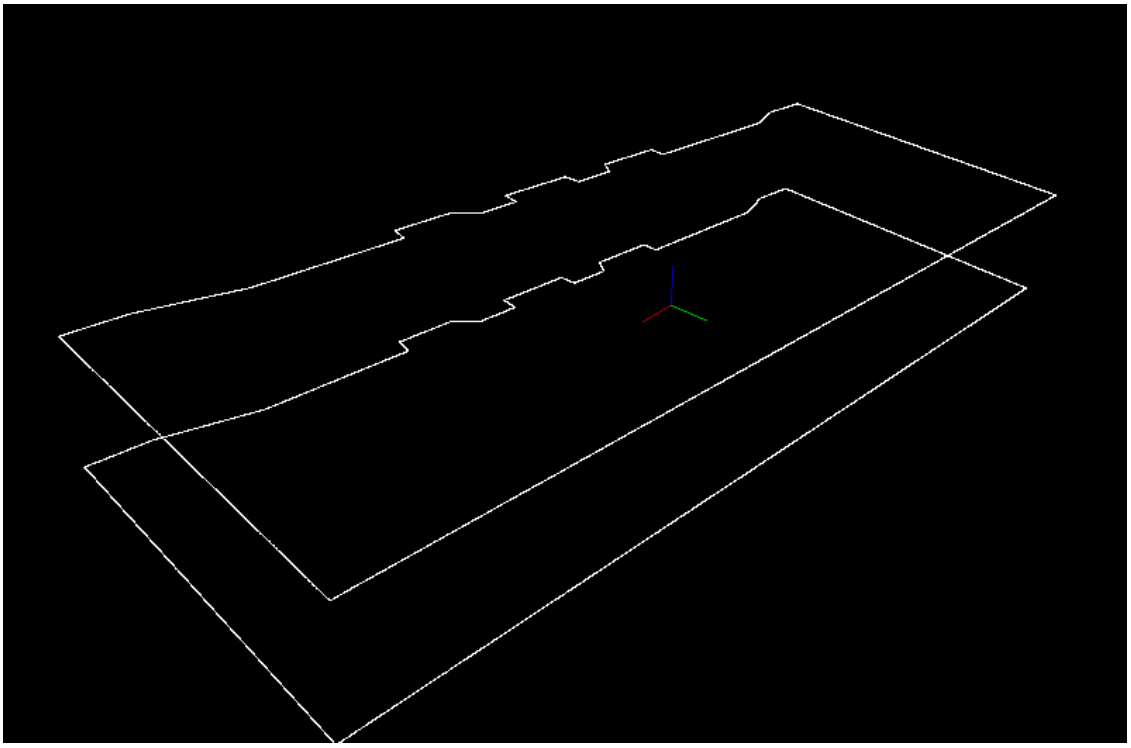


Figura 3.42-C. Formación de los polígonos cerrados que describan las tapas del cilindro.

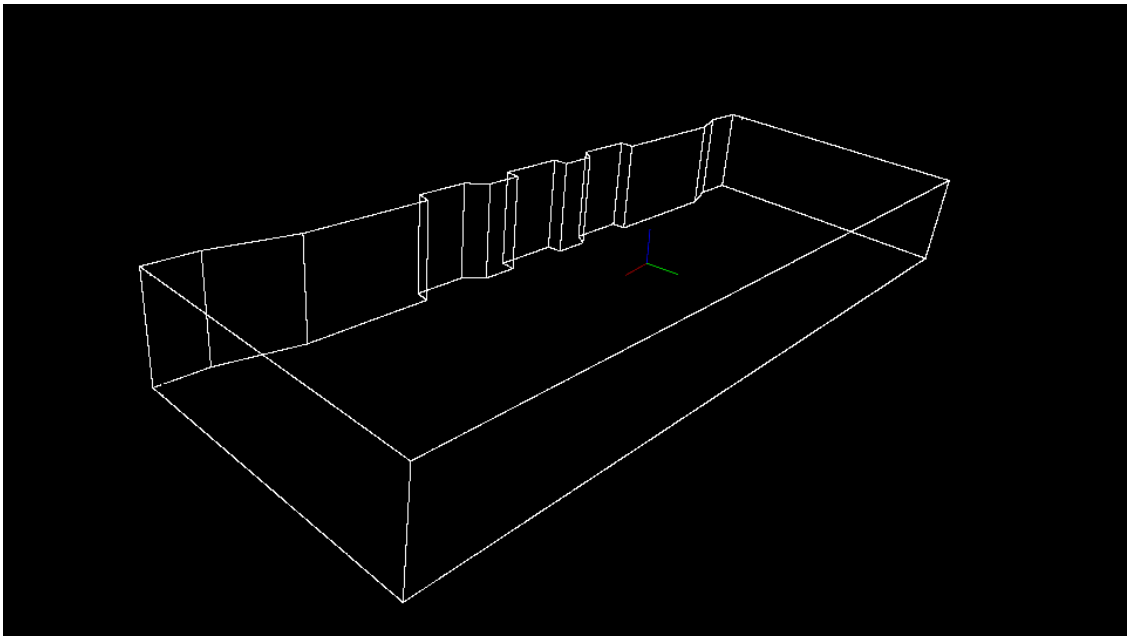


Figura 3.42-D. Formación de los polígonos que describan las superficies laterales del cilindro.

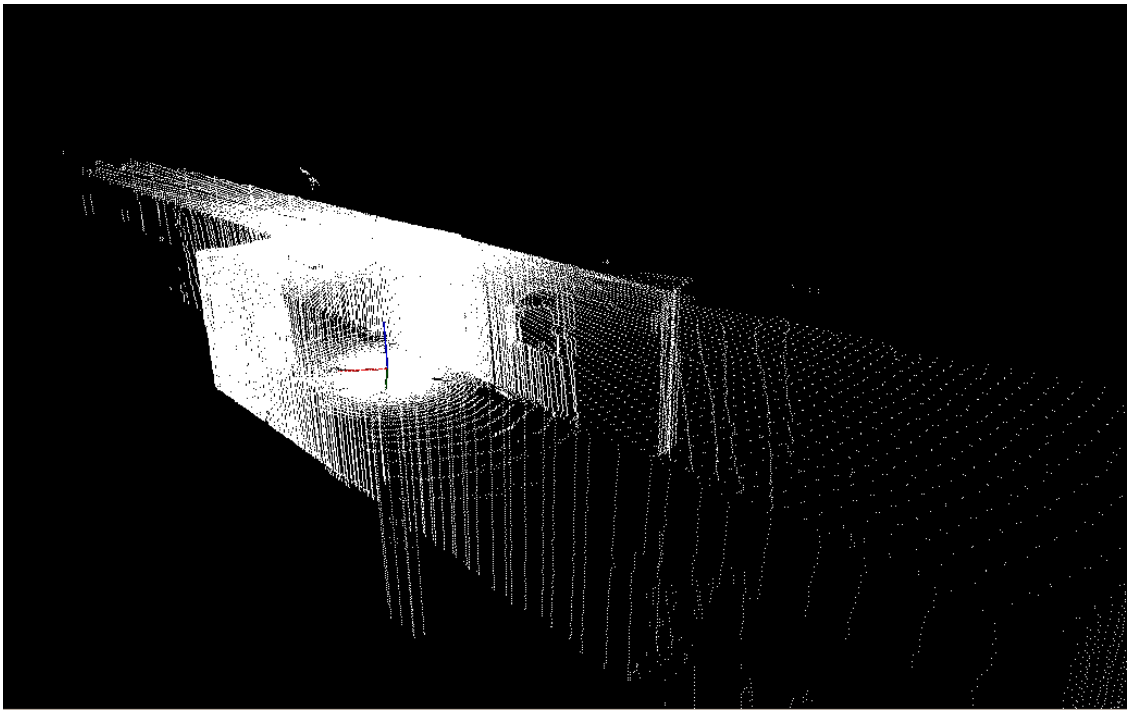


Figura 3.43-A. Nube original.

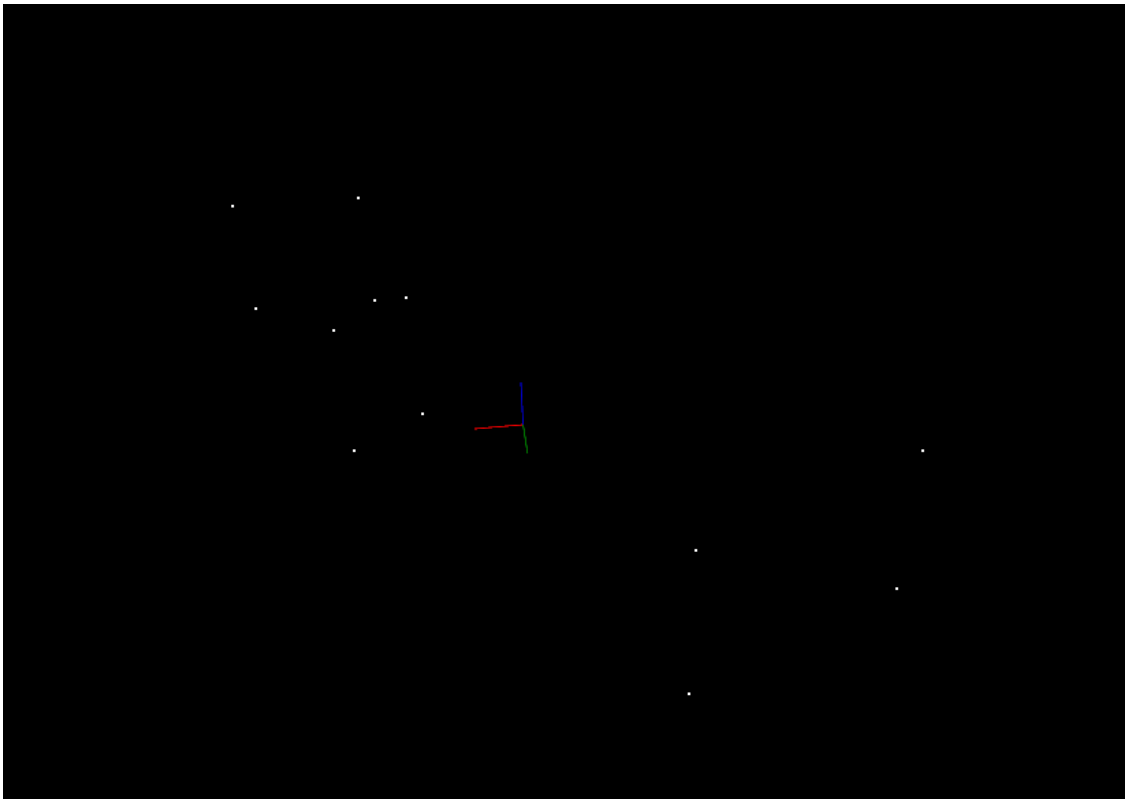


Figura 3.43-B. Puntos de reconstrucción.

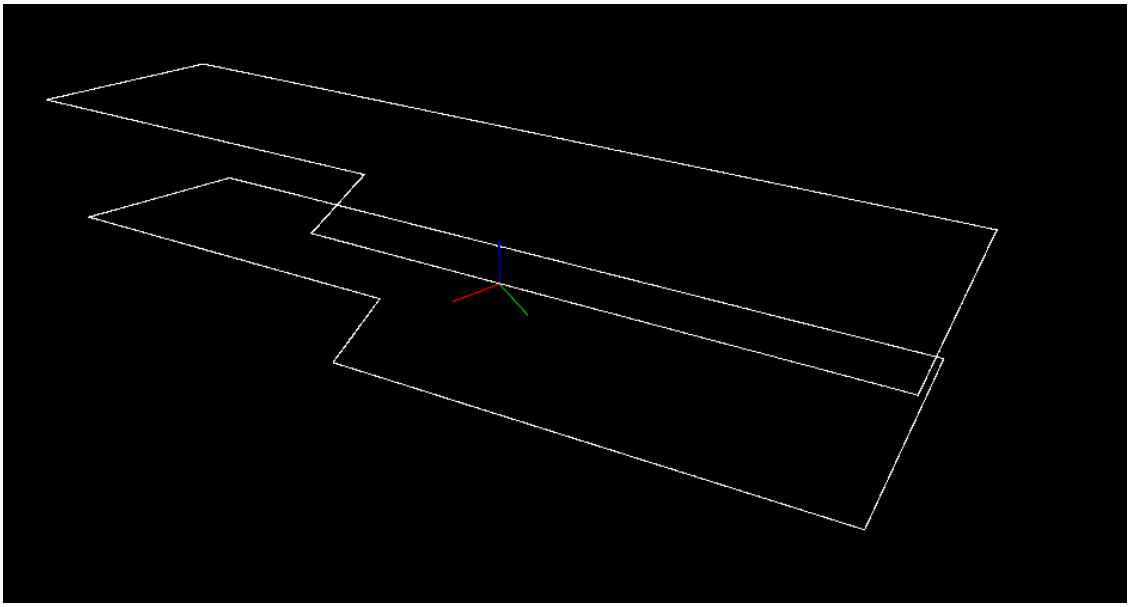


Figura 3.43-C. Formación de los polígonos cerrados que describan las tapas del cilindro.

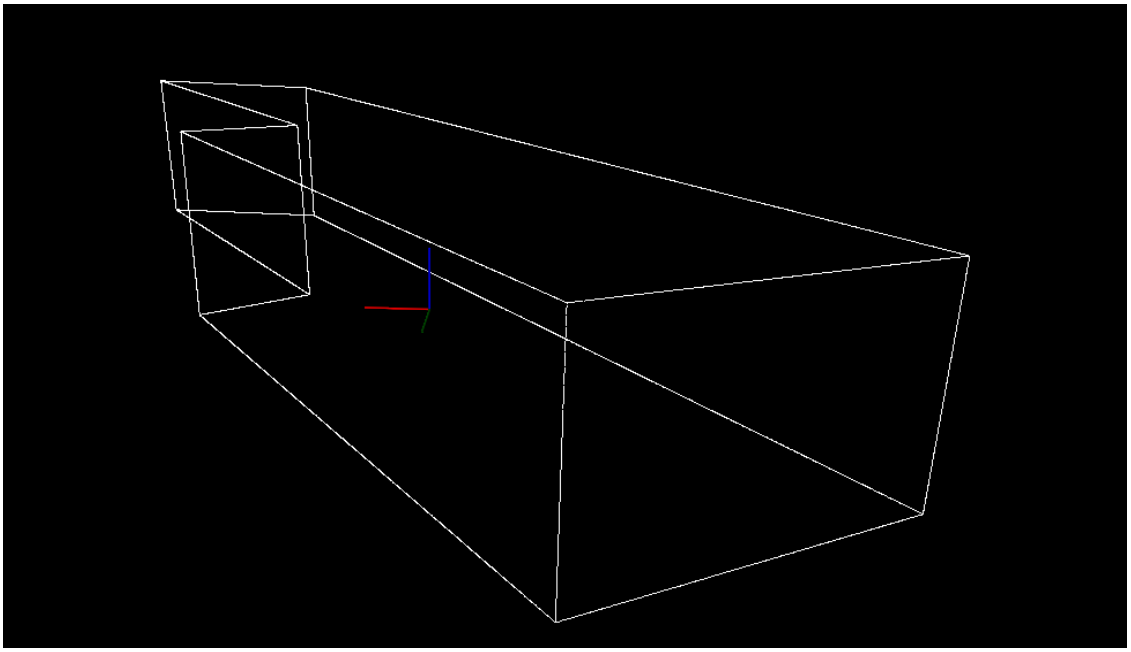


Figura 3.43-D. Formación de los polígonos que describan las superficies laterales del cilindro.

4. IMPLEMENTACIÓN DEL ALGORITMO

Para llevar a cabo una demostración del funcionamiento de del algoritmo diseñado se ha implementado un software programado en C++. Este es capaz de procesar los datos obtenidos por el dispositivo láser 3D para obtener el modelo geométrico del recinto. Toda la programación de este software se ha llevado a cabo bajo el sistema operativo de Linux Ubuntu.

4.1. Librerías utilizadas

La implementación del software se ha llevado a cabo mediante el uso de diversas librerías de "open source". Entre las diferentes librerías utilizadas destacan:

- **Librerías estándar "STL"**

La Standard Template Library (STL) [9] es una colección de estructuras de datos genéricas y algoritmos escritos en C++. Esta librería implementa un gran número de patrones de clases que describen contenedores genéricos para el lenguaje C++. La librería STL además proporciona algoritmos que permiten manipular fácilmente contenedores de información (para inicializarlos, buscar valores, etc.). También introduce el concepto de "iterador" que permite recorrer fácilmente un contenedor de información sin tener en cuenta la manera en que ha sido implementado.

- **Librerías de procesamiento de nubes de puntos "PCL"**

La point Cloud Library (PCL) [10] es proyecto abierto para el procesamiento de imágenes 2D/3D y para el procesamiento de nubes de puntos. Esta librería contiene numerosos algoritmos de procesamiento de última generación que incluyen entre otros:

- Algoritmos de filtrado.
- Algoritmos de estimación de función.
- Algoritmos de reconstrucción de la superficie.
- Algoritmos de registro y de segmentación.

Estos algoritmos se pueden utilizar, por ejemplo:

- Para filtrar los valores extremos de los datos ruidosos.
- Para obtener los segmentos correspondientes partes de una escena.
- Para extraer puntos clave o descriptores de cómputo para reconocer objetos en el mundo en base a su apariencia geométrica.
- Para crear superficies de nubes de puntos y visualizarlos.

➢ **Librerías de procesamiento matemático "Eigen"**

La Eigen [11] es una librería de alto nivel de C ++. Es un proyecto abierto que contiene funciones de algebra lineal. Sus principales características son:

- Proporciona todos los métodos y operadores habituales de vectores y matrices cuadradas (Eigen no permite matrices no cuadradas).
- Proporciona clases fáciles de utilizar para resolver sistemas lineales de ecuaciones.
- Permite el uso de matrices y vectores tanto de tamaño fijo como de tamaño dinámico.
- Permite realizar una descomposición LU, cálculo de la inversa de una matriz, cálculo del rango, cálculo del kernel, etc.
- Permite resolver sistemas de ecuaciones sin asignación de memoria dinámica.
- Totalmente compatible con la librería STL.

4.2. Estructuración del software

El software está estructurado en tres clases. Cada una de estas clases tiene un objetivo diferenciado. La relación entre estas clase se muestra en el diagrama UML de la *figura 4.1.* Las clases son las siguientes:

- Clase “laucher”.
- Clase “viewer_rt”.
- Clase “room_inside_detect”.

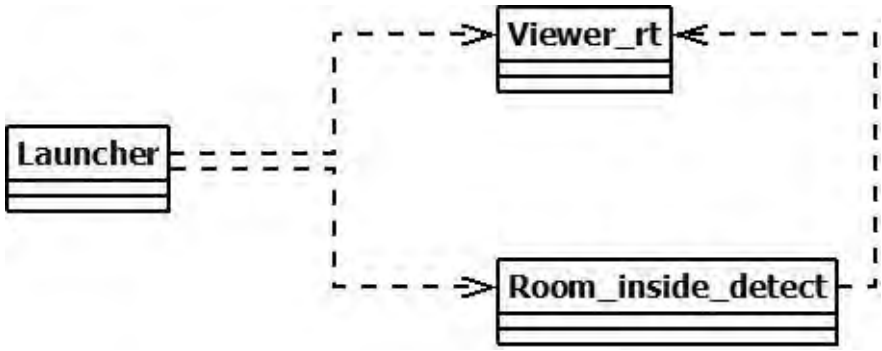


Figura 4.1. Diagrama UML de clases del software.

- **Clase “laucher”**

El objetivo de esta clase es administrar la interacción entre el usuario, el proceso de reconstrucción y el visualizador. Esta clase lanza un menú con el que se pueden administrar los cambios de variables, los tipos de reconstrucción, la visualización de la nube, el almacenamiento de los resultados, etc…

- **Clase "viewer_rt"**

El objetivo de esta clase es visualizar en tiempo real el procesado realizando sobre la nube de puntos. Esta clase se encarga de dibujar los puntos de una determinada nube de puntos. La ejecución de esta clase se realiza de forma paralela a la ejecución del proceso. De esta forma se permite visualizar las nubes que se van generando durante el proceso de reconstrucción sin invertir mucho tiempo de ejecución. Las nubes de puntos son enviadas desde la línea de ejecución principal a la line de ejecución paralela a través de hilos de comunicación. Para gestionar estas comunicaciones la clase incorpora un conjunto de funciones.

- **Clase "room_inside_detect"**

El objetivo de esta clase obtener el modelo geométrico del recinto. Este modelo se obtiene aplicando a la nube de puntos de todos los procesos del algoritmo de reconstrucción implementado. Estos procesos se dividen en una secuencia de funciones ejecutadas de manera secuencial. Toda la información que se genera se almacena dentro de la misma clase. Para poder visualizar toda la información que se genera, la clase se comunica con la clase del visualizador y la terminal de Linux. De esta forma toda la información se puede ir visualizando a medida que se va generando. La clase también incorpora un conjunto de funciones con las que poder establecer las variables de configuración de los diferentes procesos de reconstrucción. Los procesos del algoritmo están distribuidos en funciones tal como se muestra en la tabla de la *tabla 4.2.* En la *tabla 4.3* se muestran también los datos de entrada y de salida de cada función.

Proceso	Función
Filtrado	density_filter ()
Búsqueda y clasificación de coeficientes de los planos del recinto	find_wall_coeficients ()
	identify_type_of_plane()

Obtención de los coeficientes de las rectas de intersecciones de los planos	get_intersections_for_planes_type_A()
Obtención de intersecciones entre superficies del recinto	proyect_planes_type_B()
Detección de intersecciones erróneas	delete_incorrect_planes()
Obtención de puntos de reconstrucción	insert_corners_points()
Formación de los polígonos cerrados que describen las superficies de las paredes del recinto	make_polygons_in_planes_type_A()
	make_polygons_in_planes_type_B()
	find_polygons()

Tabla 4.2. Distribución los procesos en funciones.

Función	**Datos de entrada**	**Datos de salida**
density_filter ()	Nube de puntos original	Nube de puntos filtrada
	Tamaño de la celda de filtrado	
find_wall_coeficients ()	nube de puntos filtrada	Vector con los coeficientes de los planos (sin identificar)
	porcentaje de búsqueda de putos	
	Porcentaje de plano admisible	
identify_type_of_plane()	Vector con los coeficientes de los planos (sin identificar)	Vector con los coeficientes de los planos (identificados)

get_intersections_for_planes_type_A()	Vector con los coeficientes de los planos (identificados)	Vector de rectas de intersección
proyect_planes_type_B()	Vector con los coeficientes de los planos (identificados)	Vector de puntos de reconstrucción
	Vector de rectas de intersección	
	Nube de puntos filtrada	
	Tamaño de la celda de filtrado	
	Valor del percentil a buscar	
delete_incorrect_planes()	Vector con los coeficientes de los planos (identificados)	Vector con los coeficientes de los planos (identificados)
	Vector con los puntos de reconstrucción	Vector de puntos de reconstrucción
insert_corners_points()	Vector con los coeficientes de los planos (identificados)	Vector de puntos de reconstrucción
	Vector con los puntos de reconstrucción	
make_polygons_in_planes_type_A()	Vector con los coeficientes de los planos (identificados)	Vector con los puntos de reconstrucción
	Vector con los puntos de reconstrucción	
make_polygons_in_planes_type_B()	Vector con los coeficientes de los planos (identificados)	Vector con los puntos de reconstrucción
	Vector con los puntos de reconstrucción	

find_polygons()	Vector con los coeficientes de los planos (identificados)	Vector con los índices de los puntos de cada polígono cerrado
	Vector con los puntos de reconstrucción	

Tabla 4.3. Datos de entrada y de salida de cada función.

4.3. Estructuración de los datos

Para gestionar los diferentes datos que se van generando a los largo de la reconstrucción del recinto se han generado un conjunto de estructuras de datos. La relación de estas estructuras se muestra en la *figura 4.4.*

- **Estructura "plane"**

Esta estructura se ha creado para almacenar todos los datos referentes a un plano del recinto. Dentro de esta estructura se almacena:

- Coeficientes que definan la ecuación general del plano.
- Clasificación del plano (Tipo H_s, Tipo H_i o Tipo V).

- **Estructura "point"**

Esta estructura se ha creado para almacenar todos los datos referentes a un punto de reconstrucción del recinto. Esta estructura también se usa para almacenar los extremos de los segmentos. Dentro de esta estructura se almacena:

- Coordenadas cartesianas del punto de reconstrucción.
- Índices de los otros puntos del vector de puntos de reconstrucción con los que conectar para generar un segmento.
- Índices de los planos (del vector de planos) a los que el punto pertenece.

- **Estructura "polygon_vertices"**

Esta estructura se ha creado para almacenar todos los datos referentes a un polígono cerrado. Esta estructura almacena:

- Índices de los puntos (del vector de puntos de reconstrucción) que forman los vértices del polígono cerrado.
- Índice del plano (del vector de planos) que contiene el polígono.
- Coordenadas del centro de gravedad del polígono.

- **Estructura "segment"**

Esta estructura se ha creado para almacenar todos los datos referentes a un segmento. Esta estructura almacena:

- Índices de los dos puntos (del vector de puntos de reconstrucción) que forman el segmento.
- Índices de los planos (del vector de planos) a los que pertenece el segmento.
- Ángulos de barrido de los dos extremos del segmento. Estos dos ángulos constituirán el intervalo del ángulo de barrido del segmento.
- Coordenadas del centro del segmento.

- **Estructura "line"**

Esta estructura se ha creado para almacenar todos los datos referentes a una recta de intersección. En esta estructura se almacena:

- Las tres coordenadas del vector director (V_X, V_Y, V_Z).
- Las tres coordenadas de un punto perteneciente a la recta (X, Y, Z).
- Índices de los dos planos (del vector de planos) que generan la recta de intersección.

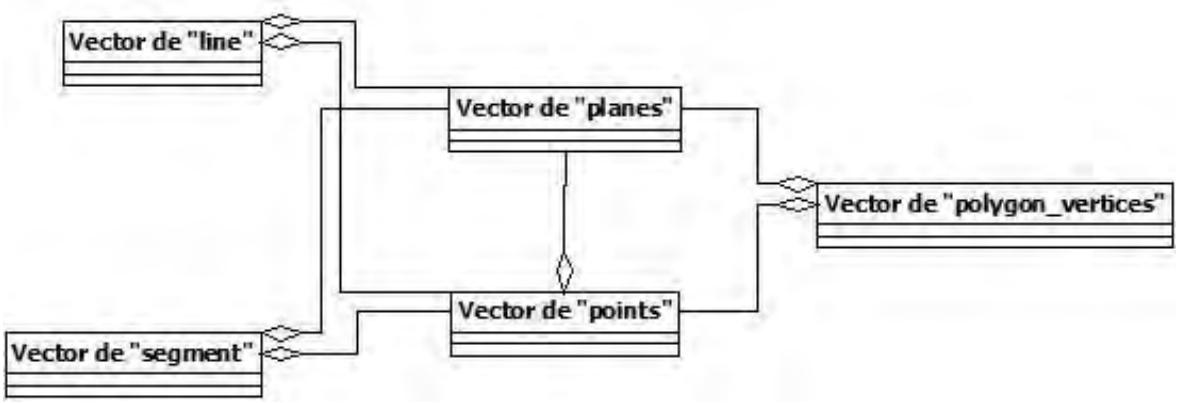


Figura 4.4. Diagrama UML de clases con la relación entre las diferentes estructuras de datos.

5. RESULTADOS

Después de implementar todos los procesos del algoritmo en el software, se ha probado sobre varias muestras. Las muestras con las que se ha probado el software son las siguientes:

- Laboratorio 1.
- Laboratorio 2.
- Pasillo.

Como se puede observar en la tabla de la *figura 5.1*, el número de puntos de la nube se ve reducido considerablemente sin perder información acerca de la forma o dimensiones de este.

Muestra	Número de puntos originales	Número de puntos finales
Laboratorio 1	328320	40
Laboratorio 2	328320	8
Pasillo	328320	12

Figura 5.1 Tabla de resultados.

En los siguientes sub-apartados se muestran los resultados obtenidos para cada muestra. A estos resultados se les han aplicado diferentes métodos de reconstrucción implementados dentro del software. Estos métodos son los siguientes:

1. **Reconstrucción mediante mallado de puntos.** Este método reconstruye las superficies mediante un mallado de puntos cartesianos de manera uniforme.
2. **Reconstrucción mediante generación de las rectas intersección.** Este método genera las rectas de intersección entre las diferentes superficies del recinto.
3. **Reconstrucción mediante un mallado de polígonos triangulares**. Este método, también conocido como polygon mesh [13], reconstruye las superficies del recinto generando polígonos triangulares. Esta metodología de reconstrucción es necesaria para poder exportar los resultados a programas de diseño gráfico.

5.1. **Resultados del laboratorio 1**

- Estado original:

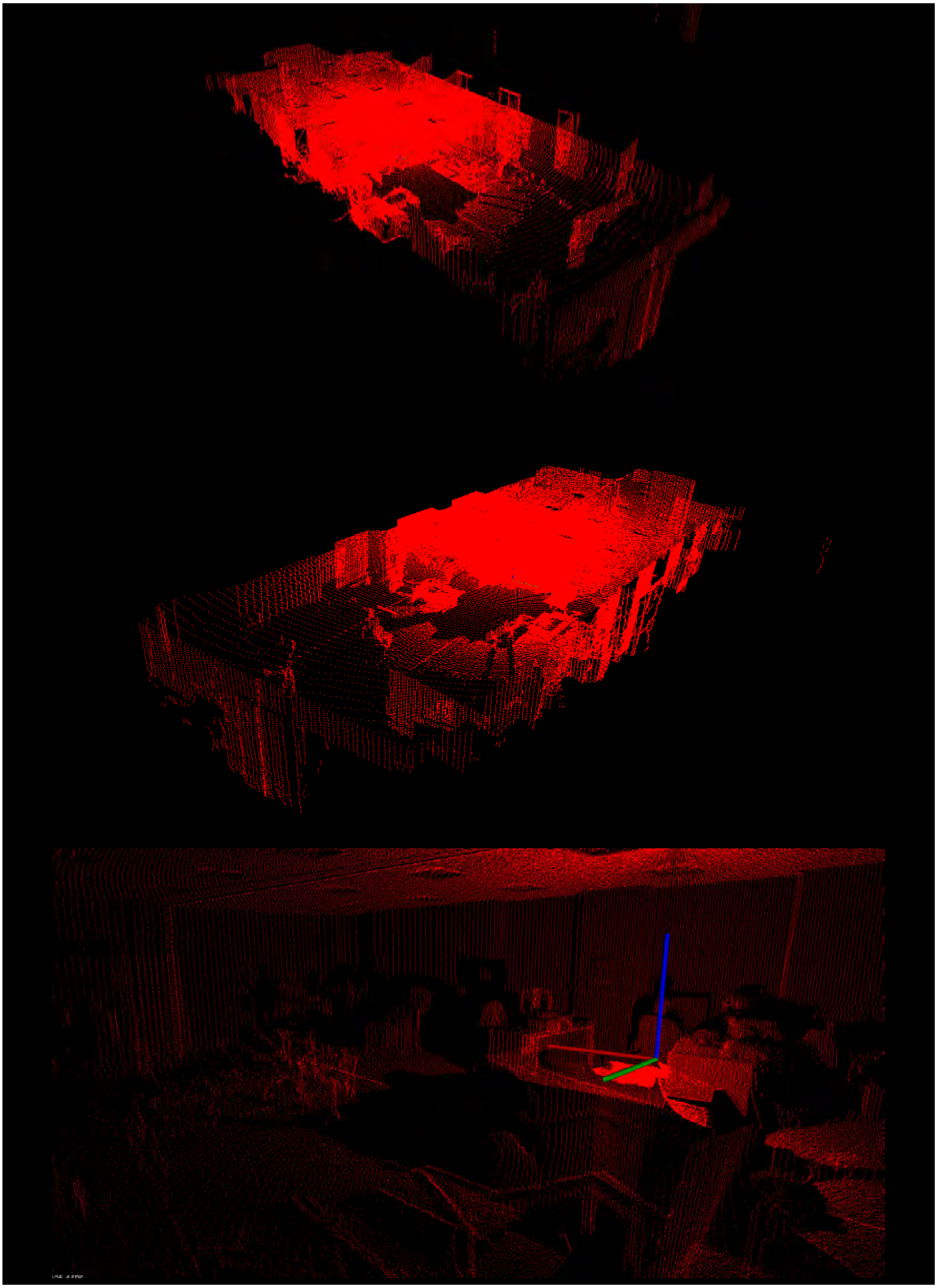


Figura 5.2. Nube original del laboratorio 1.

- Reconstrucción mediante las metodologías 1 y 2:

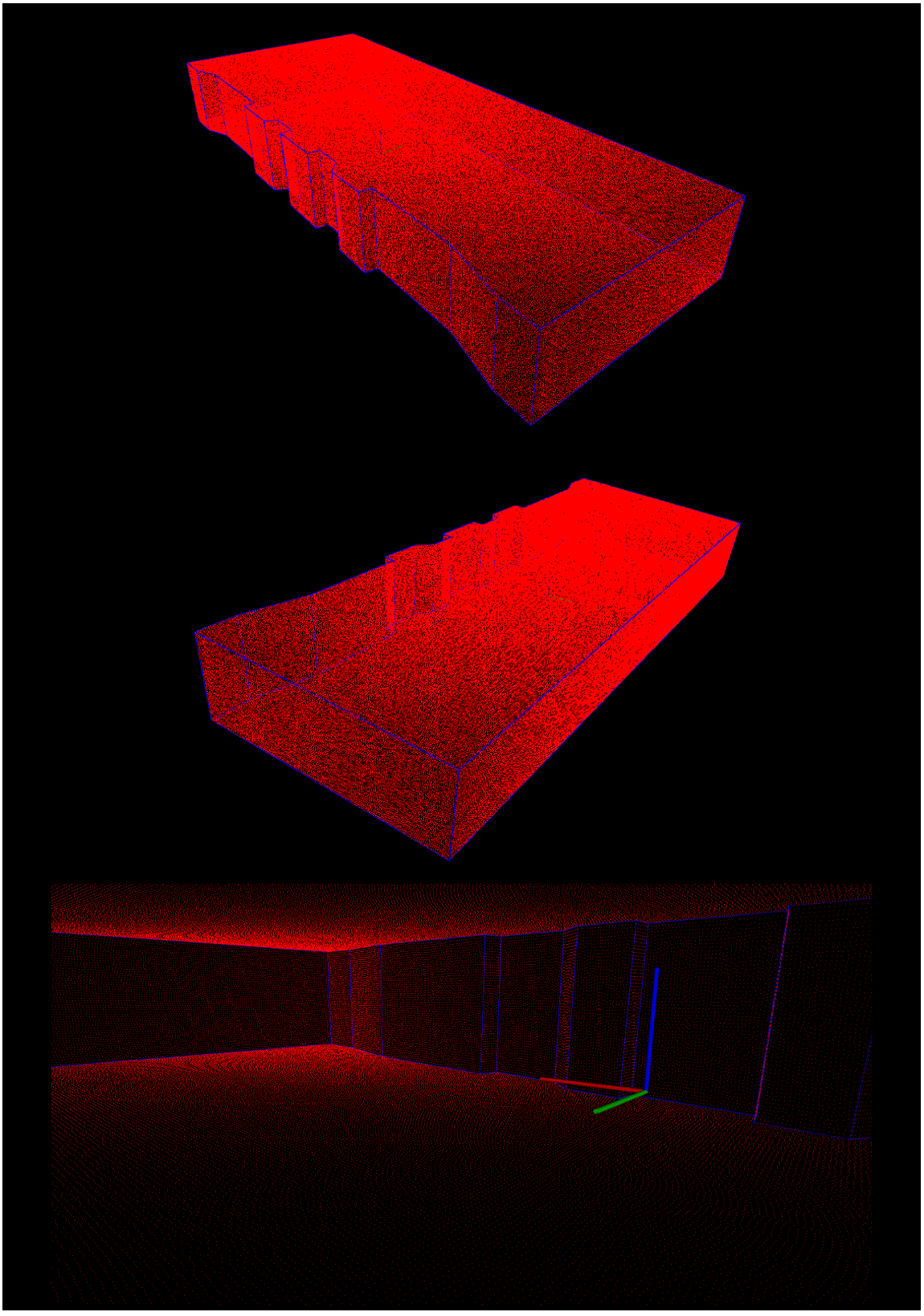


Figura 5.3. Reconstrucción del laboratorio 1 mediante metodologías 1 (rojo) y 2 (azul).

- Reconstrucción mediante las metodologías 2 y 3:

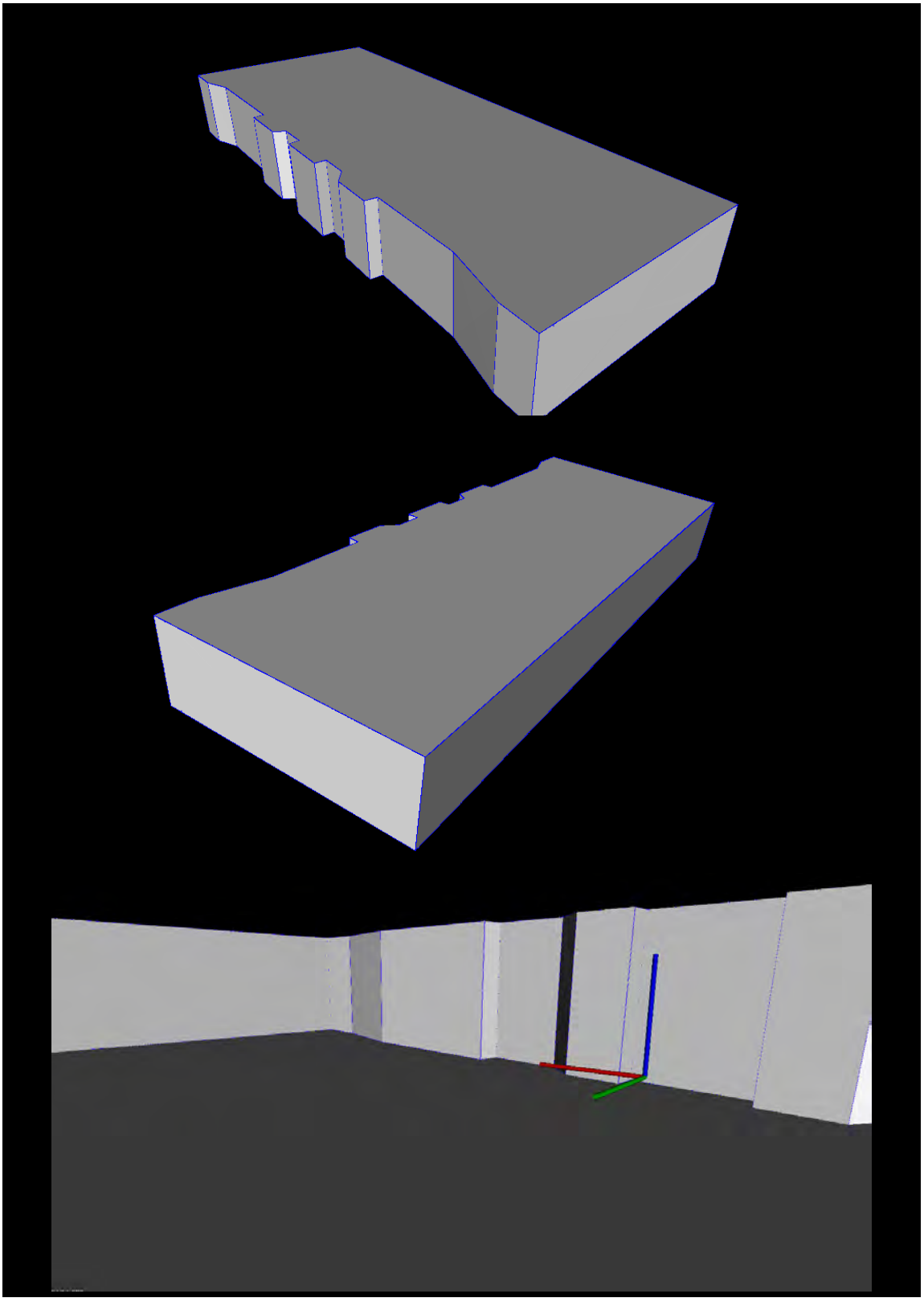


Figura 5.4. Reconstrucción del laboratorio 1 mediante metodologías 2 (azul) y 3 (gris).

5.2. Resultados del laboratorio 2

- Estado original:

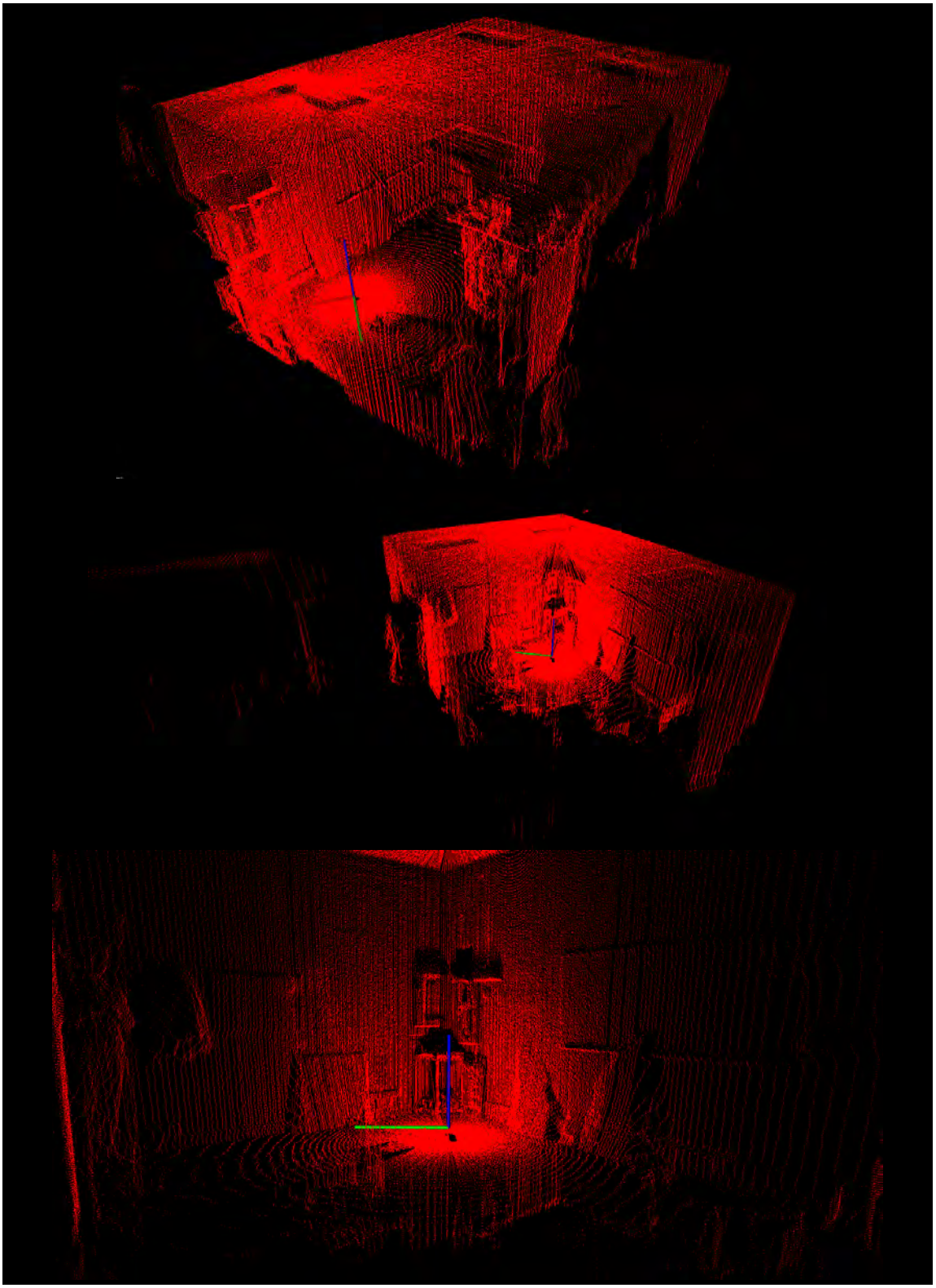


Figura 5.5. Nube original del laboratorio 2.

- Reconstrucción mediante las metodologías 1 y 2:

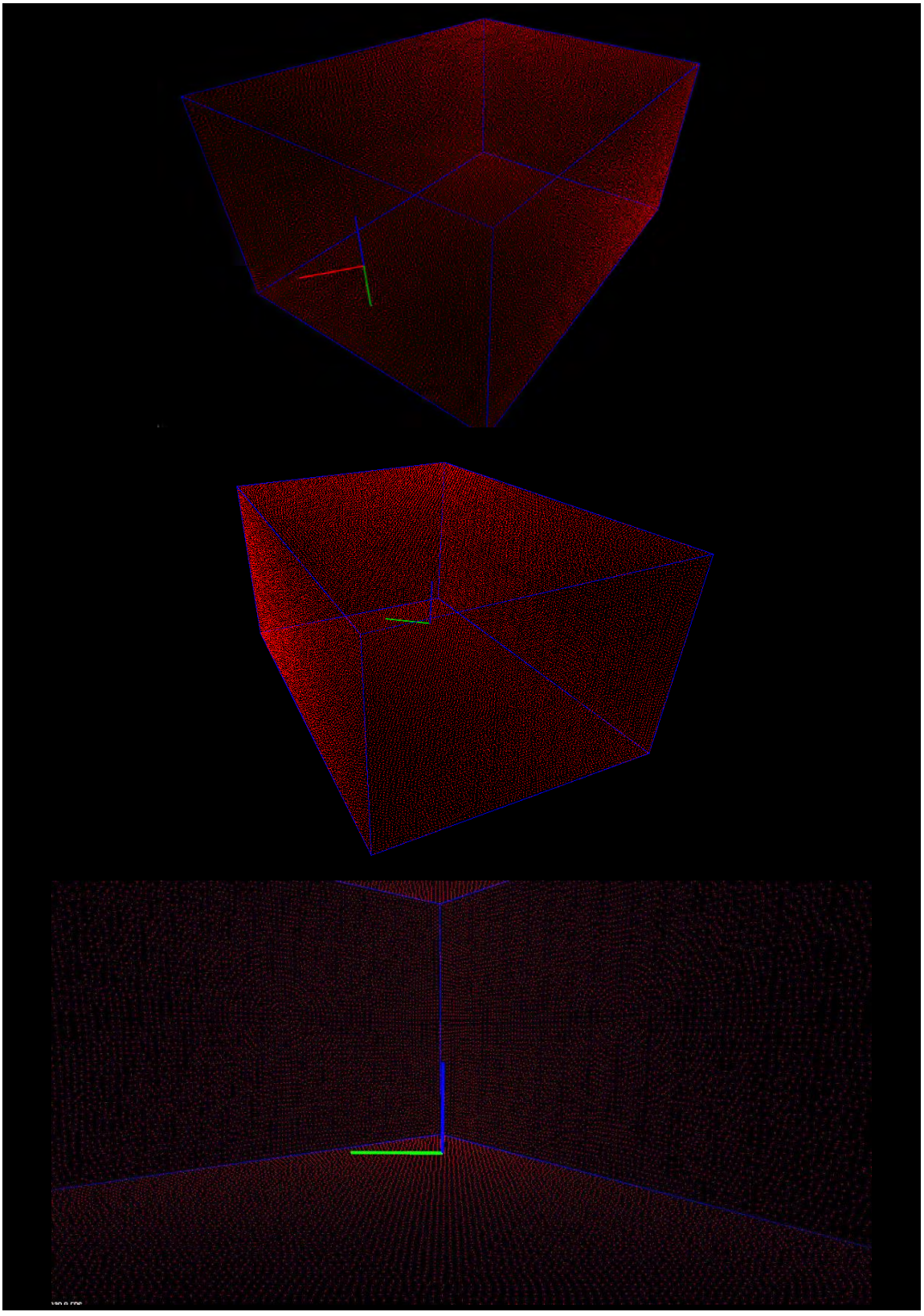


Figura 5.6. Reconstrucción del laboratorio 2 mediante metodologías 1 (rojo) y 2 (azul).

- Reconstrucción mediante las metodologías 2 y 3:

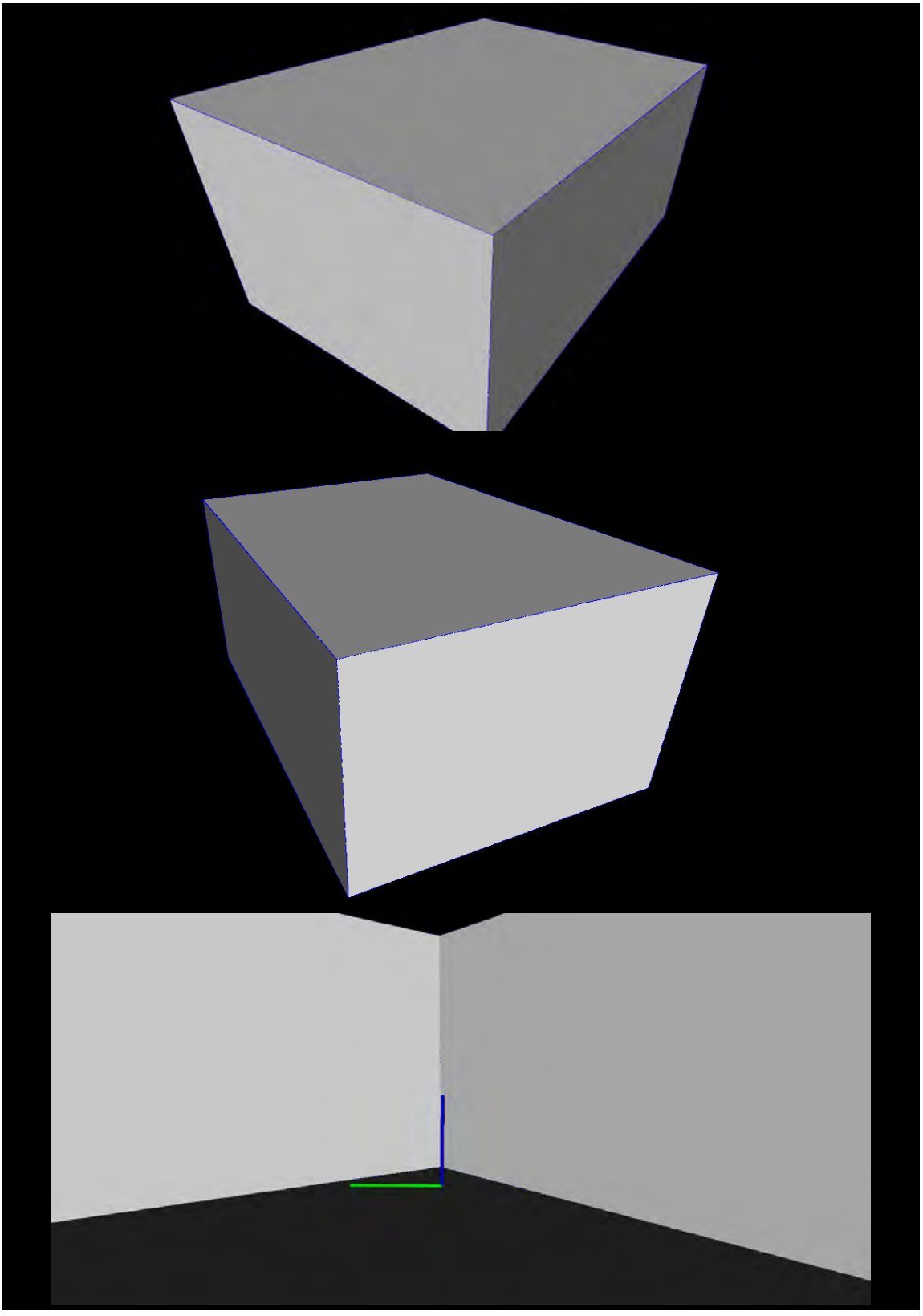


Figura 5.7. Reconstrucción del laboratorio 2 mediante metodologías 2 (azul) y 3 (gris).

5.3. Resultados del pasillo

- Estado original:

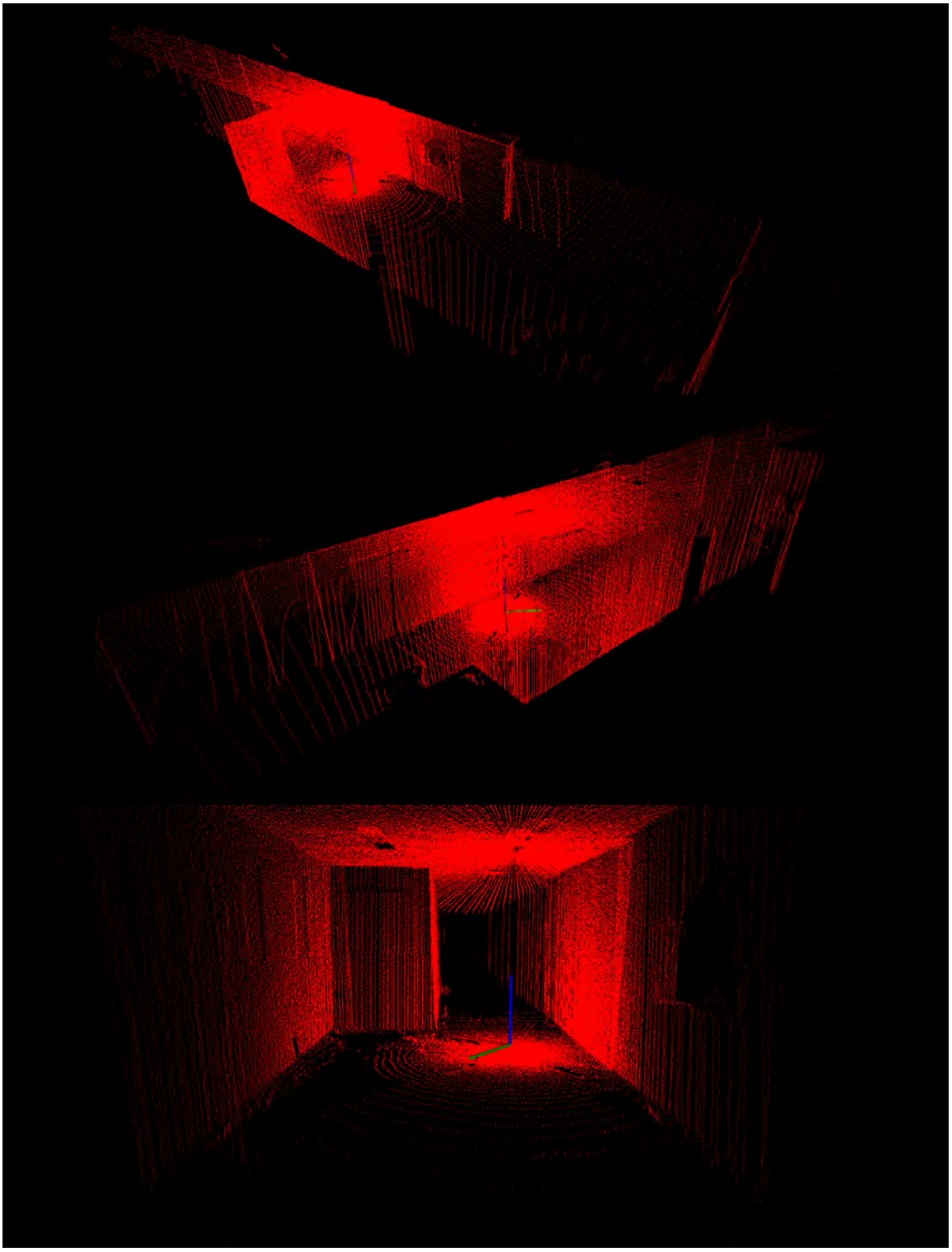


Figura 5.8. Nube original del pasillo.

- Reconstrucción mediante las metodologías 1 y 2:

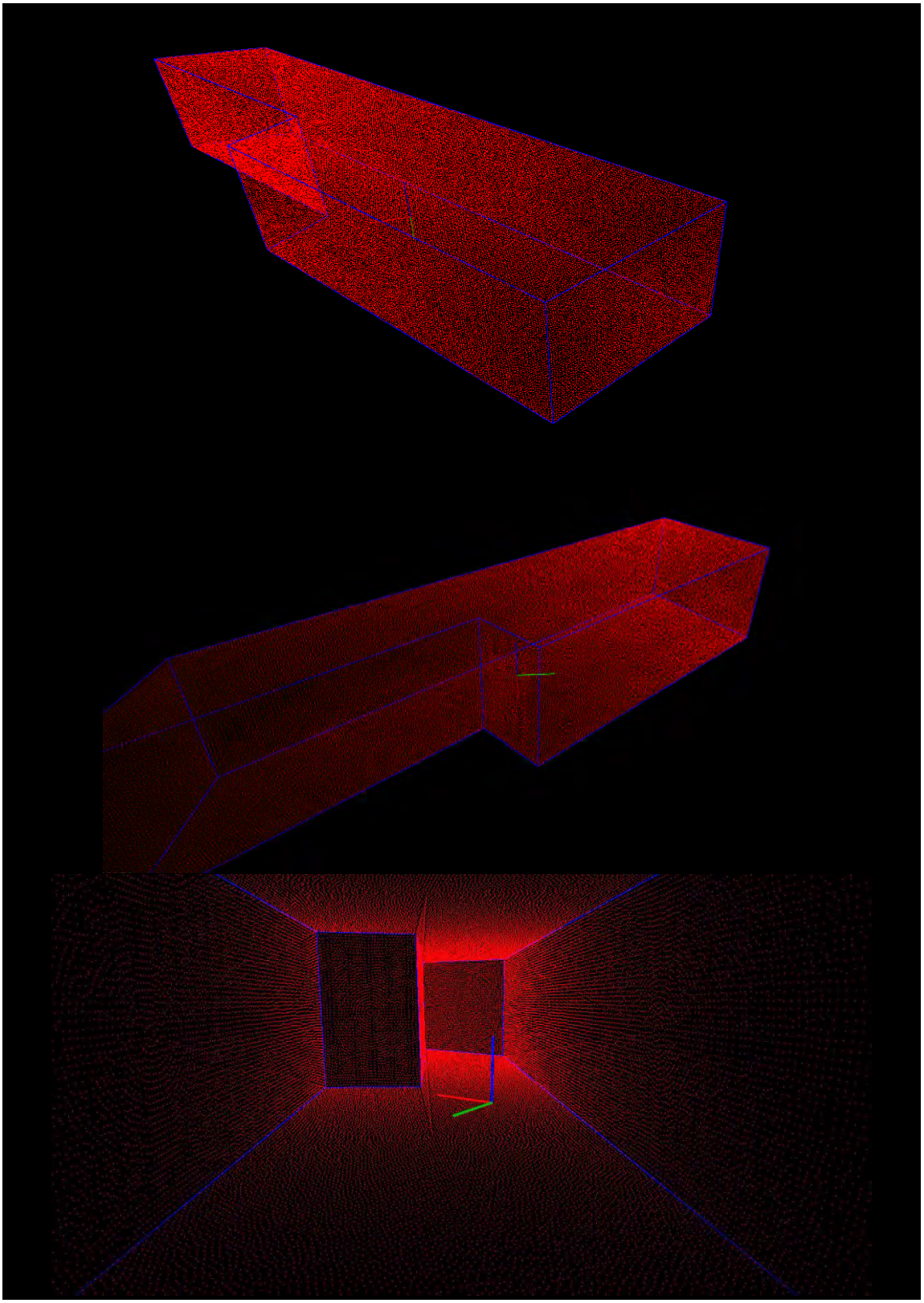


Figura 5.9. Reconstrucción del pasillo mediante metodologías 1 (rojo) y 2 (azul).

- Reconstrucción mediante las metodologías 2 y 3:

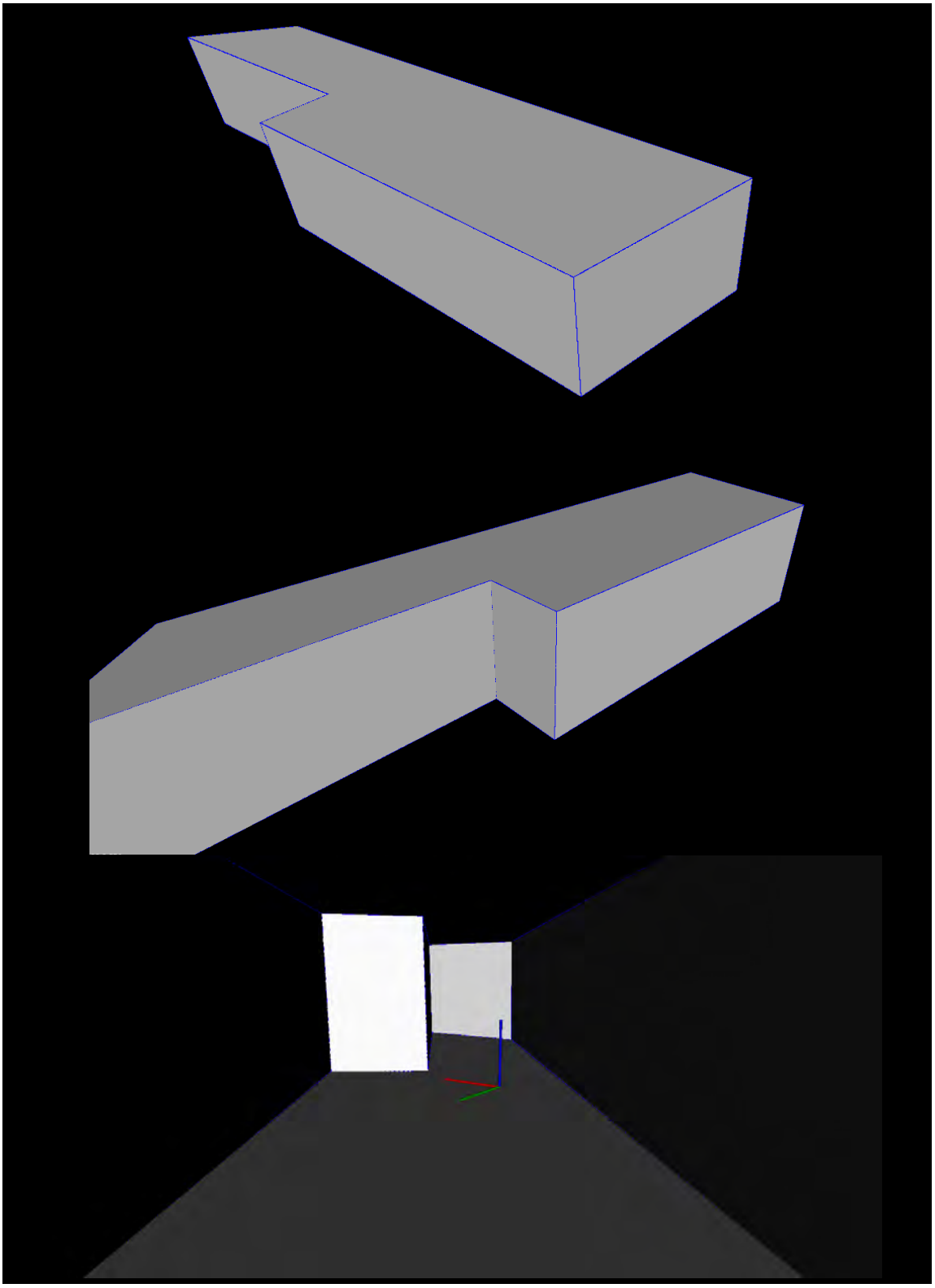


Figura 5.10. Reconstrucción del pasillo mediante metodologías 2 (azul) y 3 (gris).

5.4. Exportación de resultados a programa "3ds max"

- Exportación de los resultados del laboratorio 1 (se le ha quitado el polígono del techo para poder ver el interior):

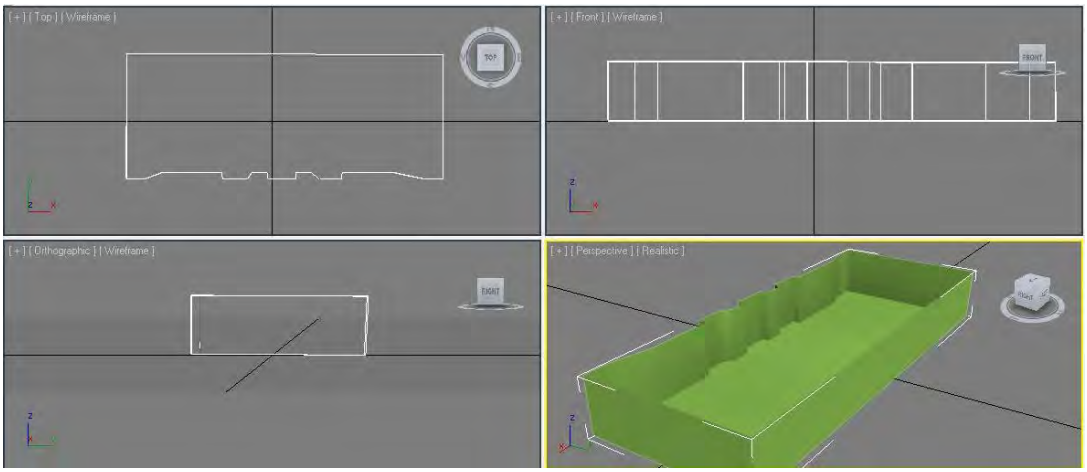


Figura 5.11. Diferentes perspectivas del laboratorio 1 exportado al programa "3ds max".

- Exportación de los resultados del laboratorio 2(se le ha quitado el polígono del techo para poder ver el interior):

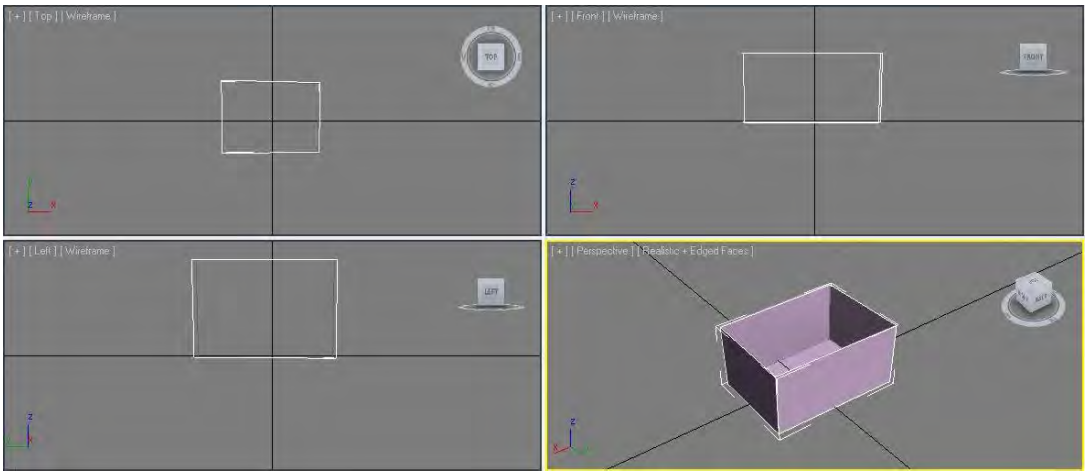


Figura 5.12. Diferentes perspectivas del laboratorio 2 exportado al programa "3ds max".

- Exportación de los resultados del pasillo (se le ha quitado el polígono del techo para poder ver el interior)

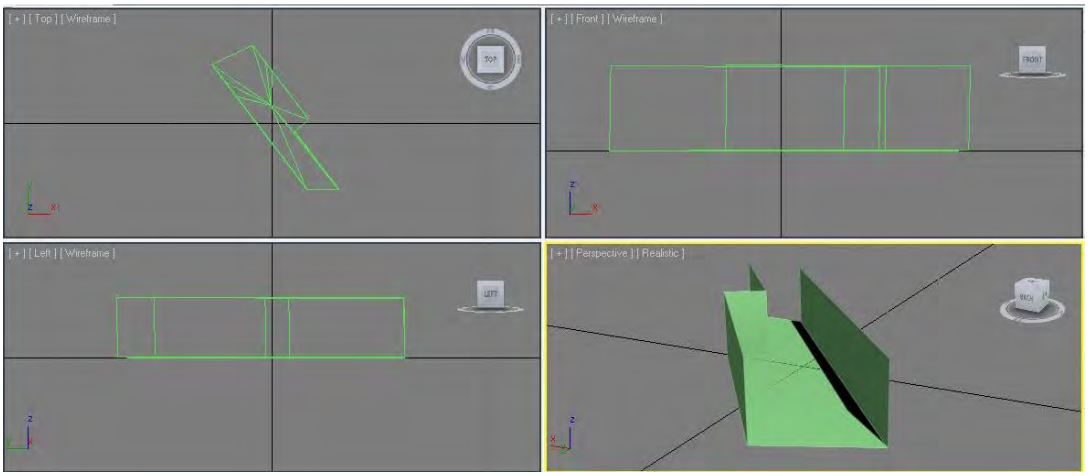


Figura 5.13. Diferentes perspectivas del pasillo exportado al programa "3ds max".

5.5. Exportación de resultados a programa "AutoCAD"

- Exportación de los resultados del laboratorio 1:

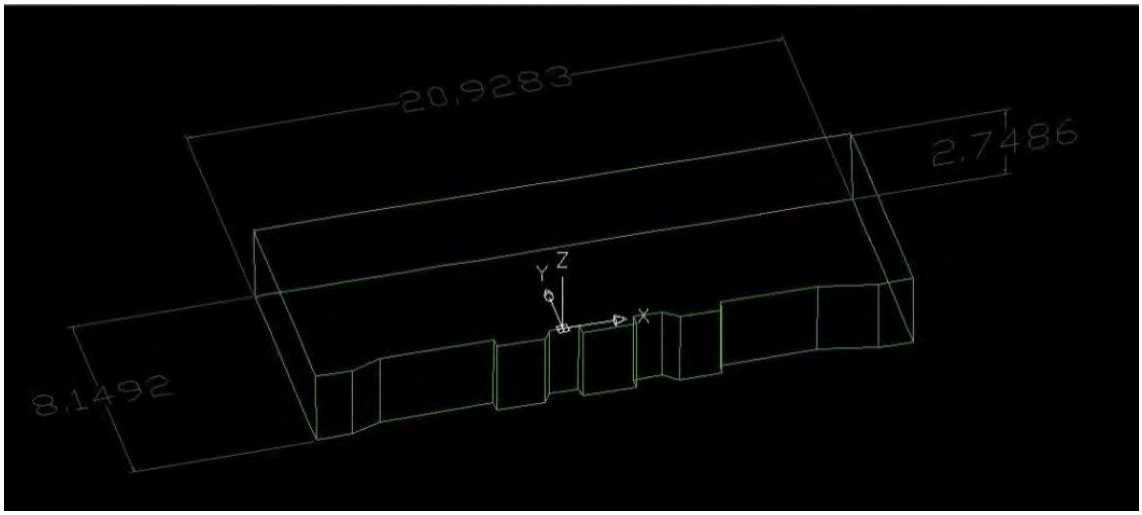


Figura 5.14. Acotaciones del laboratorio 1 en AutoCAD.

- Exportación de los resultados del laboratorio 2:

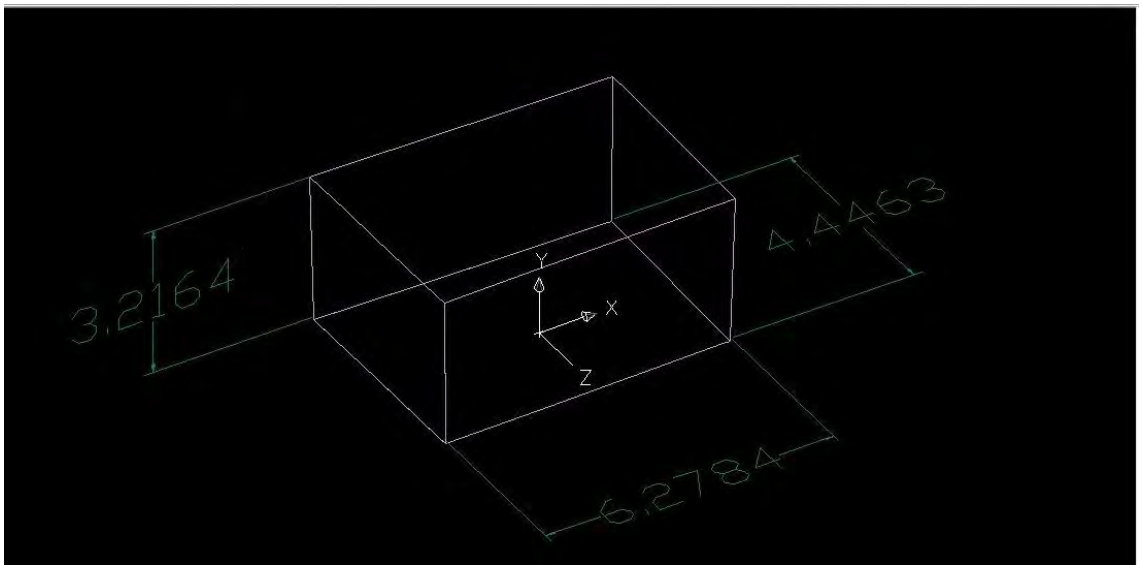


Figura 5.15. Acotaciones del laboratorio 2 en AutoCAD.

- Exportación de los resultados del pasillo:

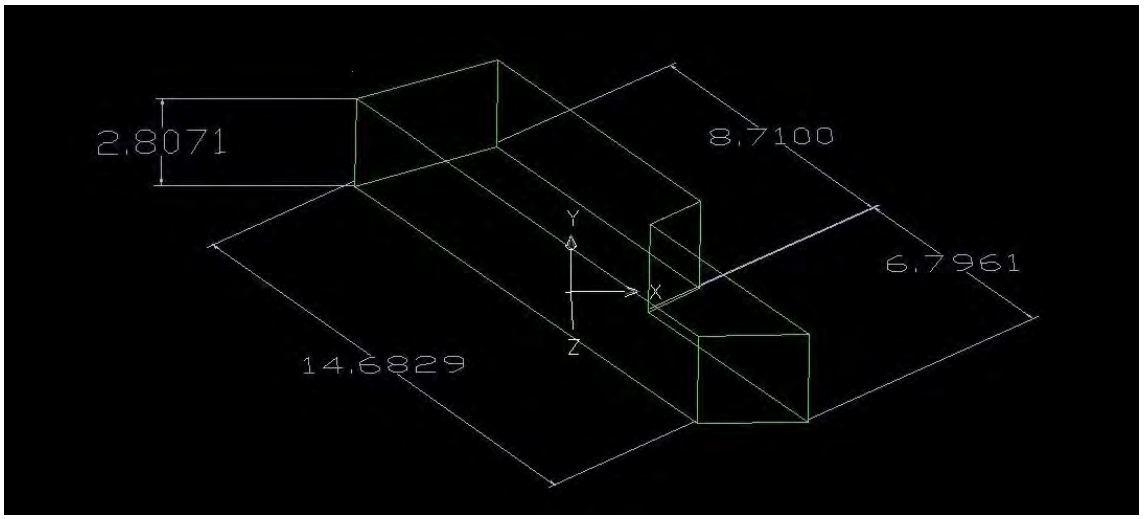


Figura 5.16. Acotaciones del pasillo en AutoCAD.

5.6. Enumeración de objetivos disgregados conseguidos

Tras mostrar los resultados obtenidos se llegan las siguientes conclusiones sobre los objetivos disgregados del proyecto.

- Generando el modelo del recinto formado por una seria de puntos conectados entre sí, se ha logrado reducir drásticamente el número de puntos de la nube de puntos original sin perder información referente al recinto.
- El recinto generado se puede ser reconstruido en varios formatos.
- El recinto generado es un recinto cerrado, lo que facilita el cálculo de volúmenes y superficies del mismo.
- Unos de los formatos de reconstrucción hace que la reconstrucción sea exportable a otros formatos de diseño gráfico.

6. ANÁLISIS ECONÓMICO DEL PROYECTO

El análisis económico de este proyecto es difícil de realizar debido a que se han usado diversas librerías “open source” cuyo valor el difícil de calcular. El análisis realizado se ha basado en el coste de los dispositivos utilizados y en el coste salarial de las horas invertidas por un ingeniero.

Concepto	Coste	Descripción
Láser bidimensional Hokuyo	4.000,00 €	Láser de captura bidimensional utilizado para la captura de las nubes de puntos.
Motor Dynamixel EX	500,00 €	Motor utilizado para hacer pivotar el dispositivo láser. Este pivotamiento transformara las dos dimensiones de captura en tres
Componentes varios	50,00 €	Componentes varios utilizados para hacer funcionar el dispositivo. Estos componentes incluyen la tarjeta de alimentación, el soporte de acomodación del dispositivo, etc.
sistema operativo Linux Ubuntu	-	Sistema operativo utilizado para programar el software. Al ser "open source" su valor no se puede calcular.
Librerías PCL	-	Conjunto de librerías de C++ utilizadas para el manejo de las nubes de puntos. Al tratarse de "open source" su valor no se puede calcular
Librerías Eigen	-	Conjunto de librerías de C++ utilizadas para el manejo y operación de vectores y matrices. Al tratarse de librerías "open source" su valor no se puede calcular
Librerías STL	-	Conjunto de librerías de C++ utilizadas para el manejo de funciones comunes de programación. Al tratarse de librerías "open source" su valor no se puede calcular.
Sistema operativo ROS	-	Sistema operativo que controla la sincronización entre el dispositivo láser y la rotación del motor.
Coste salarial	-	-

Total:	4.550,00 €

7. TRABAJOS FUTUROS

A continuación se describen posibles vías de desarrollo del algoritmo de este proyecto. Estas vías de desarrollo buscan implementar mejoras del algoritmo implementado. Entre las posibles mejoras que se pueden implementar dentro del algoritmo están:

- Implementación de un algoritmo que analice la exactitud del modelo geométrico generado. El análisis se realizaría comparando la superficie generada por el modelo con los puntos de la nube original. Con este análisis se podrá saber por ejemplo cuales de las superficies son reales y cuales se han generado para cerrar el modelo. Otra ventaja proporcionaría este análisis será el poder encontrar vacíos dentro de una superficie generada. Estos vacíos en superficies equivaldrán a las puertas o ventanas del recinto. En las *figura 7.1, 7.2 y 7.3* se muestra un ejemplo de esta implementación.

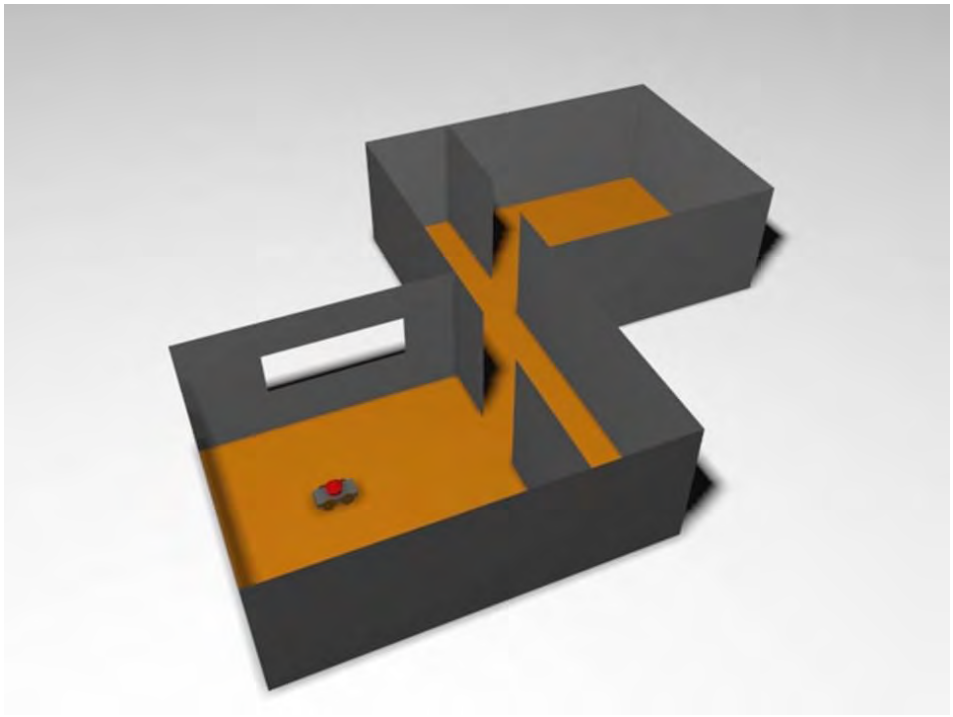


Figura 7.1. Imagen del recinto de ejemplo.

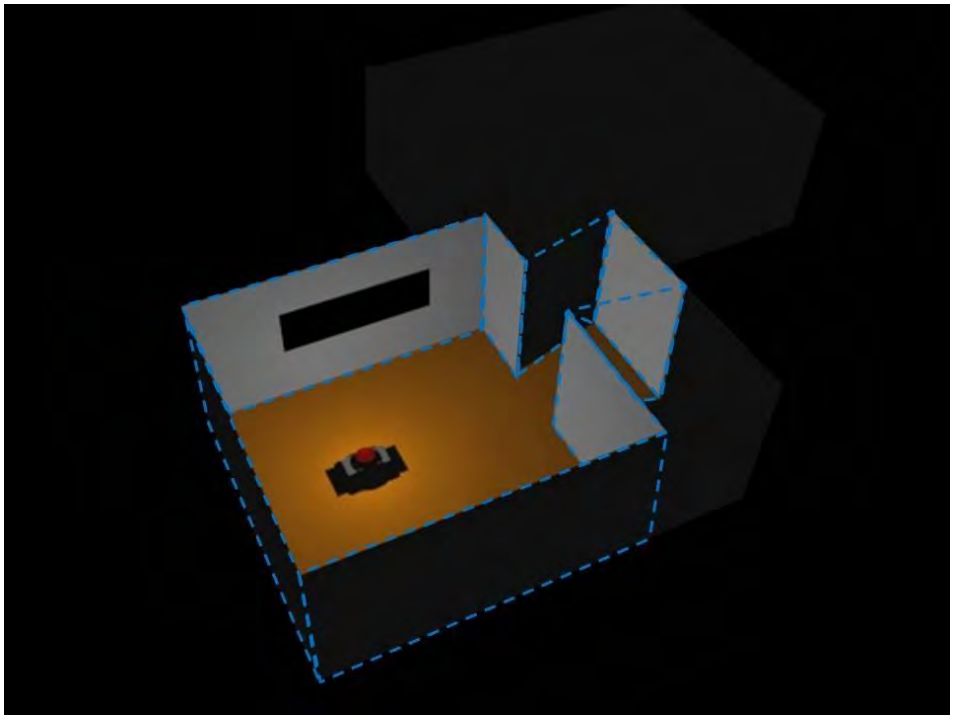


Figura 7.2. Recinto reconstruido tras aplicar el algoritmo implementado.

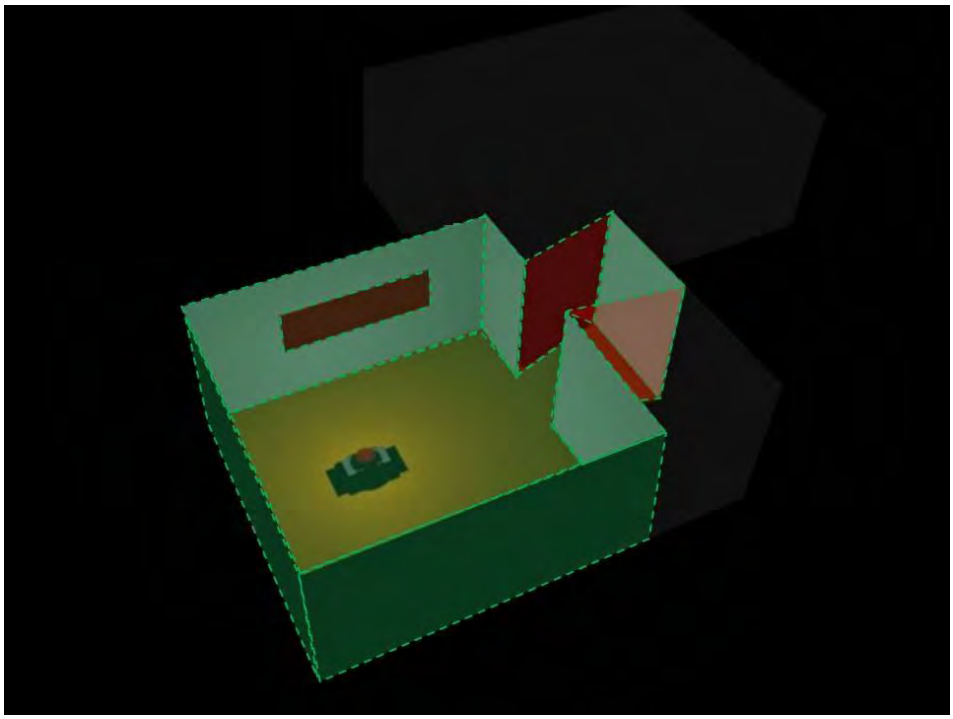


Figura 7.3. Detección de superficies erróneas del recinto y de agujeros en las superficies del modelo (superficies en rojo).

- Mejora del algoritmo para que sea capaz de reconstruir recintos con dos niveles de techo, ya que el algoritmo que se ha desarrollado en este proyecto solo es válido para reconstruir recintos con un único nivel de techo. En la *figura 7.4* se muestra un ejemplo grafico sencillo de los alzados de dos recintos, uno con un nivel de techo y otro con dos niveles.

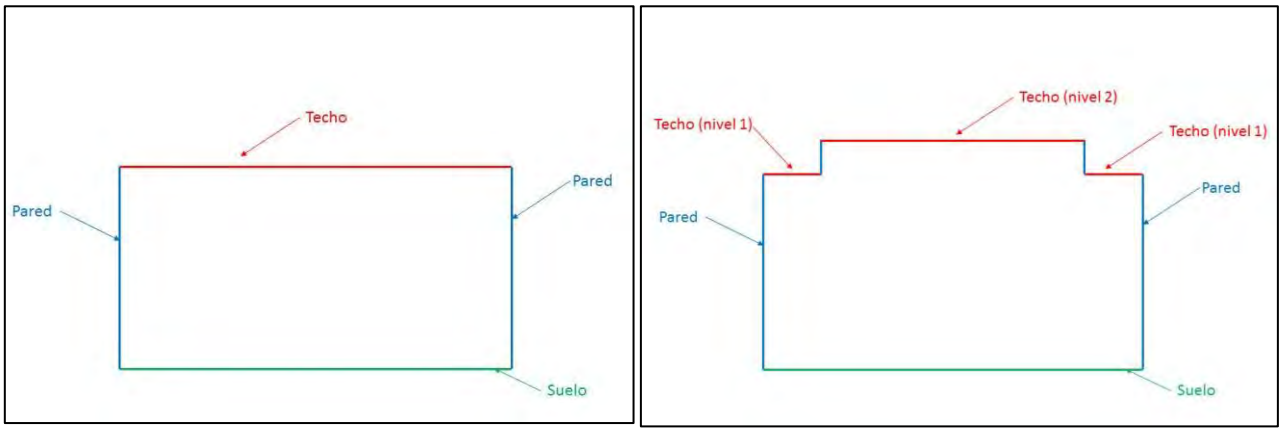


Figura 7.4. Alzado de un recinto con un nivel de techo (izquierda) y alzado de un recinto con dos niveles de techo (derecha).

- Mejora del algoritmo para la reconstrucción de recintos con varios planos tipo H_s o tipo H_i, ya que el algoritmo que se ha desarrollado en este proyecto solo es válido para reconstruir recintos con un plano tipo H_s y un plano tipo H_i, que son los más comunes. En *la figura 7.5* se muestra un ejemplo grafico del alzado de un recinto con dos planos tipo A_U.

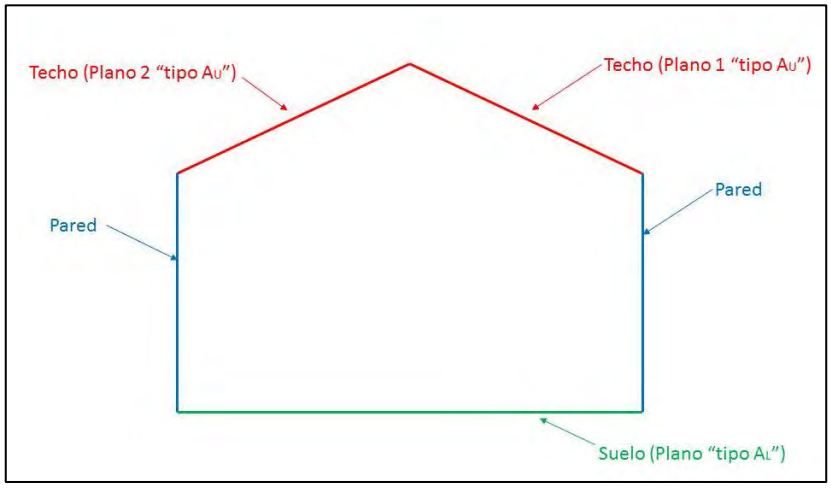


Figura 7.5. Alzado de un recinto con dos planos tipo A_U.

- Mejora del algoritmo para que se pueda comunicarse con el dispositivo laser. Con esto el algoritmo se podría ordenar al dispositivo láser 3D para que realice nuevas capturas de puntos en zonas donde la densidad de puntos capturados es reducida.

- Mejora del algoritmo para que pueda mezclar dos reconstrucciones de un mismo recinto. Las dos reconstrucciones se obtendrían situando el dispositivo láser 3D en dos zonas diferentes dentro del mismo recinto. Gracias a esto se podrían reconstruir recintos de grandes dimensiones tales como pasillos o conjuntos de habitaciones. En la *figura 7.6, 7.7 y 7.8* se muestra un ejemplo de los recintos que se podrían obtener mediante esta mejora y como proceder para ello.

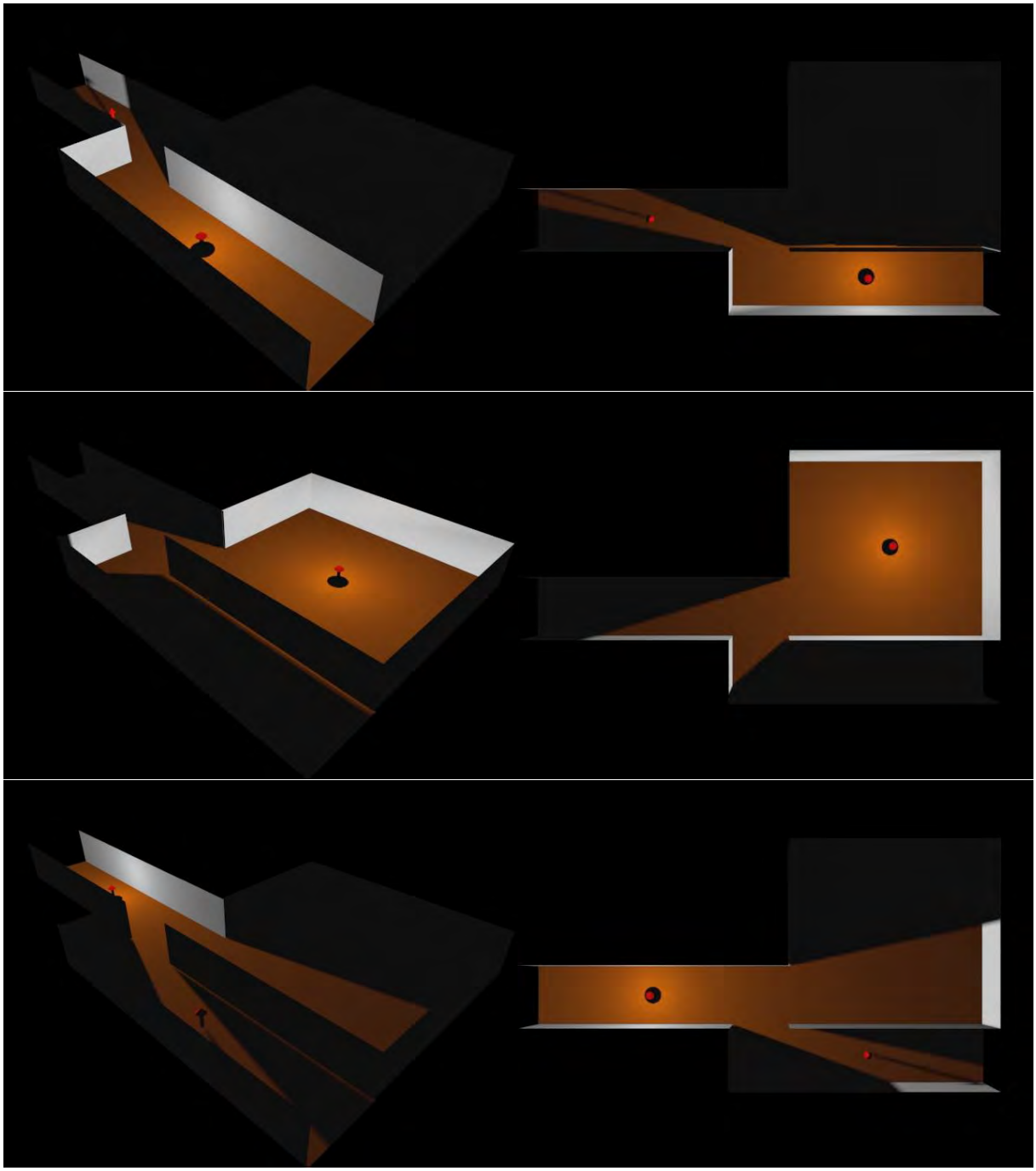


Figura 7.6. Recinto capturado por el dispositivo láser 3D desde el punto de captura 1 (arriba), desde el punto de captura 2 (medio) y desde el punto captura 3 (abajo).

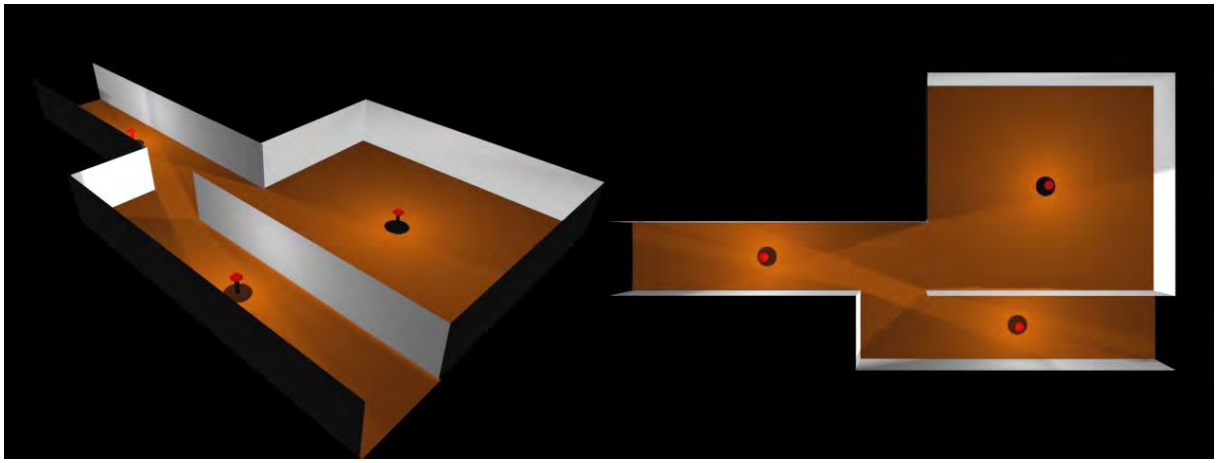


Figura 7.7.Suma de las tres capturas realizadas en el recinto.

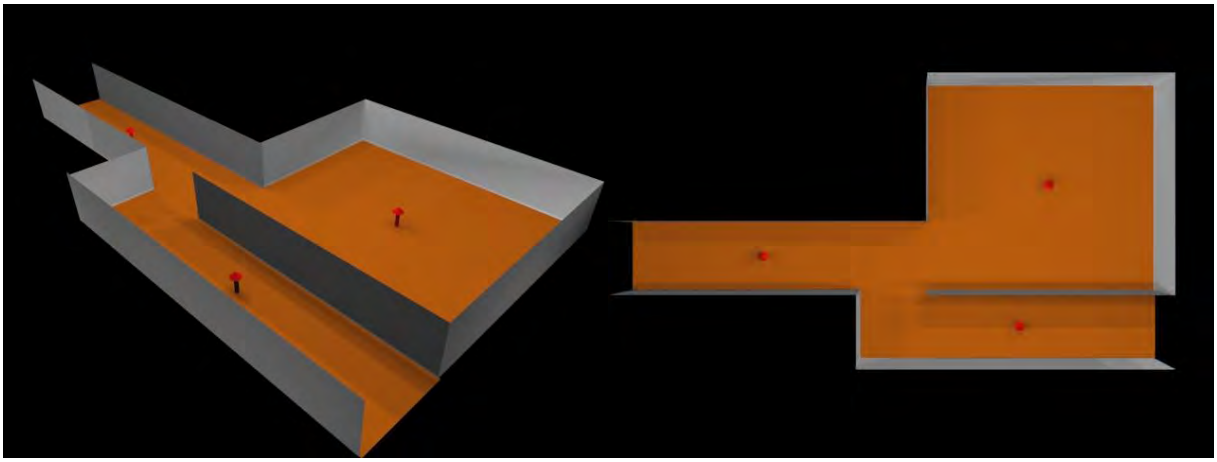


Figura 7.8. Recinto capaz de ser reconstruido con la mejora propuesta.

ANEXO I: DESCRIPCION DEL EQUIPO UTILIZADO EN LA CAPTURA DE DATOS

Este equipo utiliza un sensor láser Hokuyo [14] de dos dimensiones. El láser esta acoplado a un motor que le hace rotar sobre sí mismo para conseguir que se puedan obtener capturas en tres dimensiones.

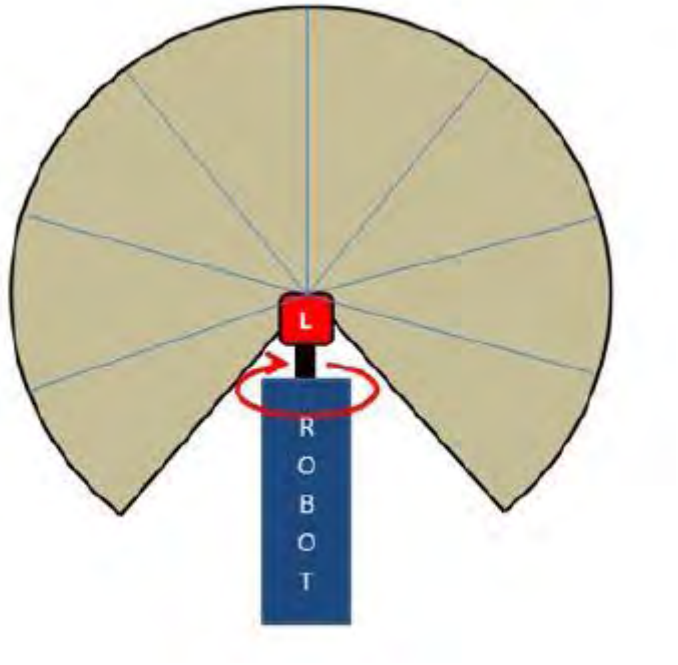


Figura AI.1. Esquema de funcionamiento del equipo laser.

A. Transformaciones geométricas que realiza el equipo

Para obtener los datos en coordenadas cartesianas son necesarias una serie de transformaciones de las medidas obtenidas por el láser. El láser obtiene 1080 medidas radiales de distancia repartidas entre 270º en intervalos de 0.25º (ángulo θ). Conociendo El ángulo ϕ que viene represado por el giro del motor se pueden obtener una nube de puntos en coordenadas esféricas.

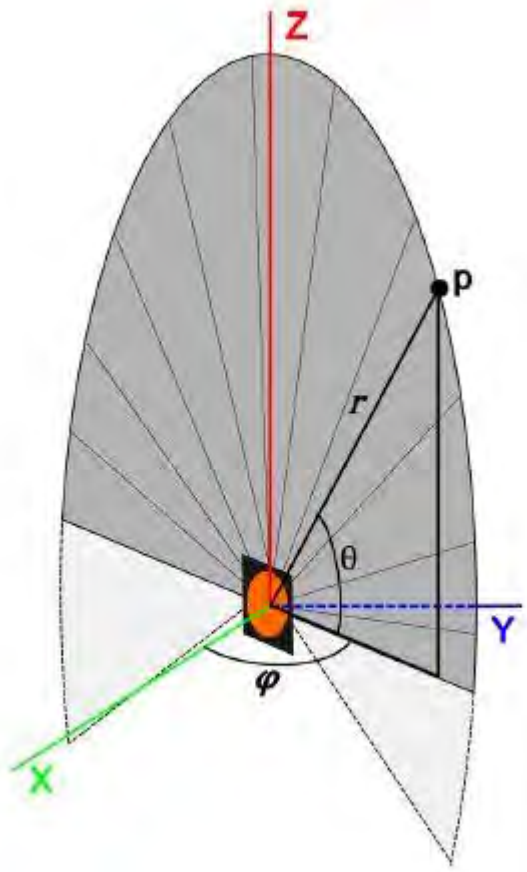


Figura AI.2. Transformaciones de la nube de puntos.

En la captura de una nube de puntos, mientras el motor gira se capturan medidas del láser constantemente. Cada vez que se toman las 1080 medidas del láser correspondientes a un plano, se guarda también el correspondiente ángulo del motor. Cuando el motor termina de girar, y todos los datos están almacenados se realizan las transformaciones a coordenadas cartesianas. Para convertir los datos a una nube de puntos XYZ se usan las siguientes ecuaciones:

$$x = r \cdot cos(\theta) \cdot cos(\varphi)$$
$$y = r \cdot cos(\theta) \cdot sen(\varphi)$$
$$z = r \cdot sen(\theta)$$

Figura AI.3. Ecuaciones de transformación.

B. Descripción del láser Hokuyo UTM-30lx

El láser Hokuyo UTM-30lx [15] es el sensor de mayor gama que tiene la empresa Hokuyo, al ser el que tiene una mayor resolución angular y mayor velocidad de barrido. Es un sensor que tiene un rango de medida desde 0.1 hasta 30 metros, pudiendo llegar en óptimas condiciones hasta 60 metros. Tiene un rango angular de 270º, en el que se pueden tomar 1080 medidas gracias a su precisión angular de 0,25º. Estas medidas se toman en un tiempo de 25 ms gracias a la alta velocidad de rotación del motor del láser, 2400 rpm.

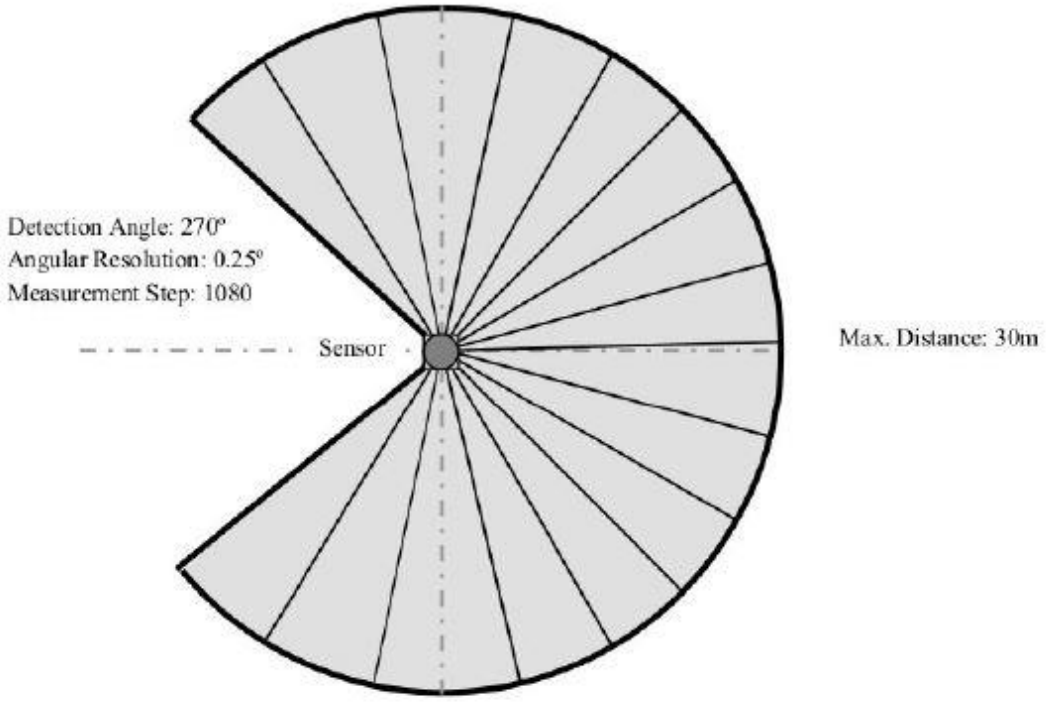


Figura AI.4. Representación del rango de medida del láser.

Figura AI.5 El láser Hokuyo UTM-30lx.

La interfaz de comunicaciones con el PC es USB 2.0. Además, el fabricante facilita las librerías de programación en lenguaje C++, lo que facilita su integración con el sistema operativo ROS.

Tipo de fuente de luz	Semiconductor láser λ=905nm Clase 1
Voltaje de alimentación	12V $\pm$ 10 %
Consumo de corriente	0.7A nominal, 1A máximo
Potencia de consumo	Menos de 8W
Rango de detección	0.1m a 30m (hasta 60m óptimas condiciones)
Precisión con 3000lx	$\pm$30mm (0.1m a 10m)
Precisión con 10000lx	$\pm$50mm (0.1m a 10m)
Angulo de escaneo	270°C
Resolución angular	0,25° (270/1080)
Velocidad escaneo	25ms (velocidad de motor: 2400rpm)
Interfaz comunicaciones	USB Ver2.0 Full Speed (12Mbps)
Salida	Salida síncrona 1-point
Condiciones ambientales	-10° a +50° / menos de 85 % humedad (sin niebla)
Efecto del entorno	Distancia medida será menor bajo lluvia, nieve y sol directo
Resistencia vibraciones	10Hz a 55Hz (doble amplitud 1.5mm en cada eje) durante 2h 15Hz a 200Hz (98m/s2, barrilo 2min cada eje) durante 2h
Resisntencia impacto	196m/s2 en cada eje, 10 veces.
Estructura de protección	Optica: IP64
Aislamiento	10MΩ 500VDC
Peso	210g (sin cable)
Material envoltura	Policarbonato
Dimensiones (WxDxH)	60mmx60mmx85mm

Figura AIV.6. Especificaciones del sensor de forma detallada.

C. Motor Dynamixel EX-106+

El Dynamixel EX-106+ [16] es el motor utilizado para hacer rotar al sensor y así conseguir capturar datos del espacio tridimensional. Es un motor robusto, de velocidad variable, interfaz sencilla de comunicaciones, funcionalidad y amplio uso en la comunidad "open source", y por ello están disponibles una gran variedad de librerías.

Figura AI.7 Motor Dynamixel.

Los requerimientos mecánicos están cubiertos con su par de 107 Kg por cm y su velocidad de 91 rpm, suficientes para mover el láser. La velocidad variable del motor permite tomar nubes de puntos más o menos densas en función de las necesidades, ahorrando tiempo de adquisición si fuera necesario. Su interfaz USB 2.0 lo hacen manejable, fácil de conectar en cualquier PC y comunicación sencilla con las librerías. El motor ofrece una gran amplitud de funcionalidades entre las que destacan:

- Obtener posición en escala 0 a 4095 o en escala gradual 0º a 250º.
- Obtener velocidad en escala 0 a 1023, con 0 como máxima velocidad.
- Mover a una posición con velocidad dada o mover incrementos.
- Detectar errores de comunicaciones y de hardware.

Peso	154g
Dimensiones	40.1mm x 65.3mm x 50.1mm
Resolución	0.06°
Relación de reducción	184:1
Torque	107Kg x cm
Velocidad sin carga	91 rpm (a 18.5V)
Rango de movimiento	0° a 251°
Temperatura de funcinamiento	-5°C a +80°C
Voltaje de funcionamiento	12V a 18.5V (14.8V recomendado)
Protocolo de comunicación	RS485 Asíncrono (8bit, 1stop, No paridad)
Enlace físico	RS485 Multi Drop Bus
ID	254 ID (0 a 253)
Velocidad de comunicación	7343 bps a 1 Mbps
Feedback	Posición, temperatura, carga, voltaje de entrada, etc
Corriente de standby	55 mA

Figura AI.8. Especificaciones del motor de forma detallada.

La velocidad del motor se controla mediante el uso interno de 10 bits, lo que hace que el motor se controle entre 0 y 1023 (velocidad relativa). La velocidad máxima del motor es 91 rpm.

Velocidad relativa	Velocidad real	Velocidad relativa	Velocidad real
10	1,11	50	5,55
20	2,22	60	6,66
30	3,33	70	7,77
40	4,44	80	8,88

Figura AI.9. Tabla con los valores de velocidad del motor para determinados valores de los 10 bits.

D. Tarjeta de alimentación

Esta tarjeta de alimentación está diseñada para unificar y facilitar las conexiones de alimentación del láser Hokuyo y el motor Dynamixel. Ambos dispositivos tienen que alimentarse con el mismo voltaje por lo que las dos alimentaciones cuelgan del mismo punto. Está compuesta por conectores específicos para cada dispositivo por lo que no se pueden confundir las posiciones. La tarjeta también dispone de funcionalidades de seguridad. Solo se permite una polaridad de alimentación mediante un diodo en serie y la alimentación está limitada mediante un diodo zener. Tiene dos posibilidades de alimentación que se seleccionan mediante un switch. Una desde un adaptador de alimentación de red de 220V a 12V, y otra desde la alimentación que proporciona el robot manfred [17] ubicado en el laboratorio de 12V.

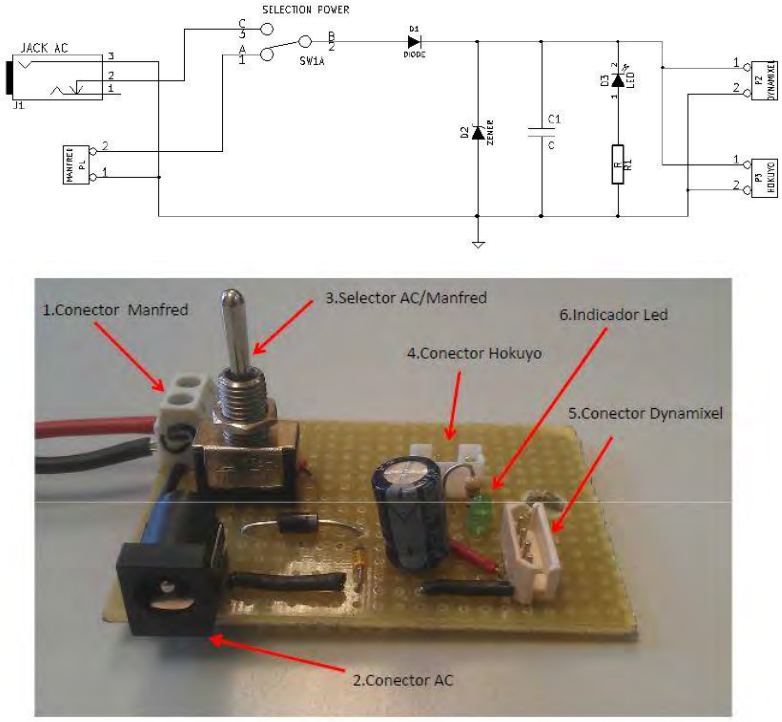


Figura AI.10. Tarjeta de alimentación del equipo.

ANEXO II: INSTALACIÓN DE DEPENDENCIAS Y DEL PROGRAMA DE DEMOSTRACIÓN DE LA CLASE "ROOM_INSIDE_DETECT"

Con el objetivo de enseñar y demostrar todas las características de la clase implementada, se ha desarrollado un programa que, a través de una serie de menús de opciones, ejecutará todas las opciones que ofrece la clase.

A. Archivos y dependencias necesarias

Para la compilación del software de demostración es necesario un PC con un sistema operativo Linux. En concreto se ha utilizado la versión de Linux Ubuntu 12.04 LTS. Esta versión se puede descargar directamente desde la página oficial de Ubuntu. Más abajo se especificaran los links de descarga.

Los archivos de compilación del código se pueden descargar desde internet, desde un directorio de dropbox. Se encontraran los siguientes archivos:

- **CMakeList.txt**: Archivo con información relevante para la compilación.
- **definitions.h**: Definiciones de las diferentes estructuras utilizadas en la clase.
- **launcher.h y launcher.hpp**: Clase implementa para gestionar los menús del programa de demostración.
- **main.cpp**: código principal de la demostración.
- **room_inside_detect.h y room_inside_detect.hpp**: Clase desarrollada en el proyecto.
- **viewer_rt.h y viewer_rt.hpp**: Clase desarrollada para la visualización en tiempo real.
- **Build**: En esta carpeta se compilara todo el código, además, contiene varias nubes de ejemplo.

Para compilarlos será necesario instalar las siguientes dependencias:

- **Compilador CMAKE.** Para instalarlo se ejecuta desde una terminal la siguiente instrucción:

  $ "sudo apt-get install cmake"

- **Librerías PCL.** Para instalarlo se ejecuta la siguiente secuencia de instrucciones:

  $ sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl

  $ sudo apt-get update

  $ sudo apt-get install libpcl-all

- **Compilador C/CXX.** Para instalarlo se ejecuta la siguiente instrucción desde un terminal de Linux:

  $ sudo apt-get install build-essential

B. Compilación del código

La compilación se realiza desde una terminal de Linux, desde esta se accederá a al directorio "build" situada dentro del directorio donde se encuentran los archivos del código. Desde esta carpeta se ejecutaran las siguientes instrucciones:

$ DIRECTORIO/build/ cmake ..

$ DIRECTORIO/build/make

Tras ejecutar estas dos instrucciones el código quedara compilado.

C. Descripción y funcionamiento del programa de ejemplo

Una vez compilado con éxito el código, habrá que dirigirse a la carpeta donde se encuentre el ejecutable compilado. Este se encontrara en la carpeta "build". En esta carpeta también se encontraran las nubes de ejemplo.

Para procesar una nube de ejemplo, desde la terminar ejecutamos la siguiente instrucción:

$ ./detecion_paredes NOMBRE_DEL_ARCHIVO.pcd

El programa ejecutara en la terminal el menú del programa y en una ventana paralela el visualizador en tiempo real de las nubes.

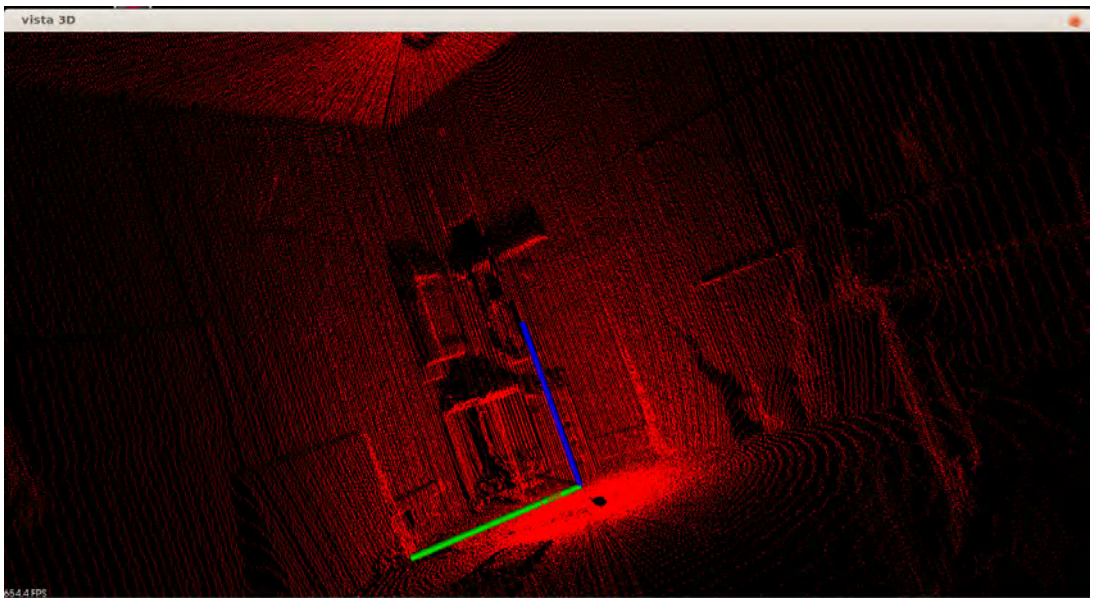


Figura Anexo II.1. Visualizador en tiempo real.

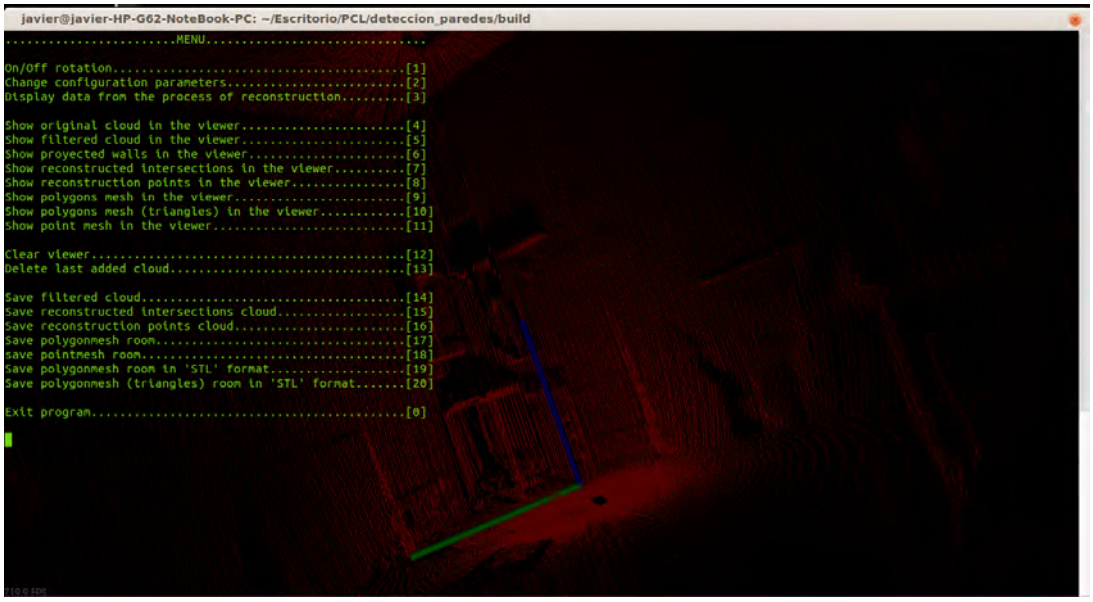


*Figura Anexo II.2. M*enú principal del programa de demostración.

En el menú se encontraran las siguientes opciones:

- **On/Off rotation**: Enciende o apaga la rotación del visualizador
- **Change configuration parameters:** Despliega el menú que ofrece el cambio de todos los paramentos de la clase "room_inside_detect". En este menú también se podrán activar la visualización en tiempo real de cada proceso (viene desactivada por defecto).
- **Display data from the process of reconstruction:** Despliega el menú donde se muestra todos los datos tales como coeficientes de planos, puntos de conexión, vector de proceso, etc.
- **Show original cloud in the viewer:** Muestra en el visualizador la nube original con la que se trabajara.
- **Show filtered cloud in the viewer:** Muestra en el visualizador la nube tras el proceso de filtrado.
- **Show proyected walls in the viewer:** Muestra en el visualizador la nube obtenida tras las proyecciones.
- **Show reconstructed intersections in the viewer:** Muestra en el visualizador todas las intersecciones reconstruidas tras el proceso.

- **Show reconstruction points in the viewer:** Muestra en el visualizador todos los puntos de reconstrucción, con los que se realizarán las conexiones.
- **Show polygons mesh in the viewer:** Muestra la reconstrucción del habitáculo mediante un mallado de polígonos "PolygonMesh".
- **Show polygons mesh (triangles) in the viewer:** Muestra la reconstrucción del habitáculo mediante un mallado de polígonos triangúlales "PolygonMesh". Este formato será capaz de leerlo programas de diseño tridimensional, tales como el 3D studio.
- **Show point mesh in the viewer:** Muestra en el visualizador la reconstrucción del habitáculo mediante un mallado de puntos.
- **Clear viewer:** Elimina todos los elementos del visualizador.
- **Delete last added cloud:** Elimina el último elemento añadido al visualizador.
- **Save filtered cloud:** Guarda la nube filtrada como un archivo "cloud_filtered.pcd" dentro de la carpeta de ejecución.
- **Save reconstructed intersections cloud:** Guarda las intersecciones reconstruidas como un archivo "reconstructed_intersections_cloud.pcd" dentro de la carpeta de ejecución.
- **Save reconstruction points cloud:** Guarda los puntos de reconstrucción como un archivo "reconstruction_points_cloud.pcd dentro de la carpeta de ejecución.
- **Save polygonmesh room:** Guarda el habitáculo reconstruido mediante un archivo de mallado de polígonos (VTK) como un archivo "polygonmesh_room.vtk" dentro de la carpeta de ejecución.
- **save pointmesh room:** Guarda el habitáculo reconstruido mediante un mallado de puntos como un archivo "pointmesh_room.pcd" dentro de la carpeta de ejecución.
- **Save polygonmesh room in 'STL' format:** Guarda el habitáculo reconstruido mediante un mallado de polígonos (STL) como un archivo " STL_room.stl" dentro de la carpeta de ejecución.

- **Save polygonmesh (triangles) room in 'STL' format:** Guarda el espacio reconstruido mediante un mallado de polígonos triangulares (STL) como un archivo “STL_room(triengles).stl” dentro de la carpeta de ejecución.
- **Exit program:** Detiene la ejecución del programa.

D. Link de descarga

- Sistema operativo Ubuntu:

  http://www.ubuntu.com/download/

- Archivos de compilación:

  https://github.com/jvgomez/RoomReconstruction

REFEFENCIAS

[1] Sensor LIDAR Velodyne.

- Online: http://velodynelidar.com/lidar/lidar.aspx (Última visita: abril 2014).
- Online: http://www.uav-lidar.com/?page_id=157 (Última visita: abril 2014).

[2] Sensor Kinect.

- Online: http://es.wikipedia.org/wiki/Kinect (Última visita: abril 2014).

[3] PrimeSense.

- Online: http://www.primesense.com/ (Última visita: abril 2014).

[4] Barcelona Robot Lab Dataset.

- Online: http://www.iri.upc.edu/research/webprojects/pau/datasets/BRL/php/dataset.php (Última visita: abril 2014).

[5] KITTI Vision Benchmark.

- Online: http://www.cvlibs.net/datasets/kitti/index.php (Última visita: abril 2014).
- Online: http://www.computervisiononline.com/dataset/kitti-vision-benchmark-suite (Última visita: abril 2014).

[6] Mapeado tridimensional con escáner láser integrado en ROS.

- Online: http://sgpsproject.sourceforge.net/JavierVGomez/index.php/Ra%C3%BAl_Villajos (Última visita: abril 2014).

[7] RANSAC.

- Martin A. Fischler and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated".
- David A. Forsyth and Jean Ponce (2003). Computer Vision, a modern approach. Prentice Hall. ISBN 0-13-085198-1.
- Richard Hartley and Andrew Zisserman (2003). Multiple View Geometry in Computer Vision (2nd ed.). Cambridge University Press.
- P.H.S. Torr and D.W. Murray (1997). "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix". International Journal of Computer Vision24.

[8] Percentil.

- Galton, F. (1885a). Some results of the Anthropometric Laboratory. J. Anthrop. Inst., 14, 275-287.

[9] STL

- Online: https://www.sgi.com/tech/stl/ (Última visita: mayo 2014).

[10] PCL

- Online: http://pointclouds.org/ (Última visita: mayo 2014).

[11] Eigen

- Online: http://eigen.tuxfamily.org/index.php?title=Main_Page (Última visita mayo 2014).

[13] **Polygonmesh**.

- Colin Smith, On Vertex-Vertex Meshes and Their Use in Geometric and Biological Modeling.
- Online PDF: http://algorithmicbotany.org/papers/smithco.dis2006.pdf (Última visita: abril 2014).
- Bruce Baumgart, Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975.
- Online: http://www.baumgart.org/winged-edge/winged-edge.html (Última visita: abril 2014).
- Tobler & Maierhofer, A Mesh Data Structure for Rendering and Subdivision. 2006. Online PDF: http://wscg.zcu.cz/wscg2006/Papers_2006/Short/E17-full.pdf (Última visita: abril 2014).

[14] **Láser Hokuyo**.

- Online: https://www.hokuyo-aut.jp/ (Última visita: abril 2014).

[15] **Láser Hokuyo UTM-30lx**.

- Online: https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html (Última visita: abril 2014).

[16] **Motor Dynamixel EX-106+**.

- Online: http://www.trossenrobotics.com/dynamixel-ex-106-robot-actuator.aspx (Última visita: abril 2014).

[17] **Robot manfred**.

- Online: http://roboticslab.uc3m.es/roboticslab/robot.php?id_robot=5 (Última visita: abril 2014).