



## 1. Enunciado

En esta tarea vas a implementar un algoritmo que segmenta una nube de puntos, identificando aquellos puntos que pueden pertenecer a un plano e identificando todos los planos posibles dentro de un objeto. Para lograr este objetivo, vas a implementar el siguiente algoritmo:

- Sea  $P$  una nube de puntos 3D.
- Repetir lo siguiente durante un número de iteraciones  $T$ :
  - Escoger tres puntos aleatorios  $p_1$ ,  $p_2$ , y  $p_3$  y calcular el vector normal del plano que forman estos puntos.
  - Calcular la distancia de todos los puntos en  $P$  hacia el plano calculado en el paso previo. Si la distancia de un punto al plano es menor que un threshold  $\rho$ , ese punto se considera un *inlier*. Si el número de *inliers* es mayor que un número  $\lambda$ , todos estos *inliers* representan con mucha probabilidad un plano en la nube de puntos. Marcar los inliers como representando un plano y eliminarlos del conjunto  $P$ .
  - Si el número de *inliers* es menor que  $\lambda$ , no pasa nada.

En la figura 1, se muestra un cubo representado como nube de puntos. Al usar valores adecuados para  $\rho$  y  $\lambda$ , se deberían agrupar todos los puntos dependiendo del plano al que pertenecen.

El objetivo de esta tarea es implementar el algoritmo propuesto arriba y experimentar con los valores de  $\rho$  y  $\lambda$  que se deben usar para identificar los planos de dos objetos: el cubo (figura 1) y la iglesia (figura 2), ambos proporcionados como parte del enunciado. Además de implementar el algoritmo, su programa de Python debería mostrar como resultado cada plano detectado en un color distinto (haciendo uso de Polyscope).

**Nota:** Como la nube de puntos de la iglesia contiene más de un millón de puntos, se sugiere samplear la nube de puntos para experimentar y encontrar los valores de  $\rho$  y  $\lambda$ . Una vez que se identifiquen estos parámetros, hacer correr el algoritmo sobre la nube de puntos entera. Cualquier optimización a este proceso es bienvenido. Documentar en el mismo código (en forma de comentarios) las optimizaciones realizadas.

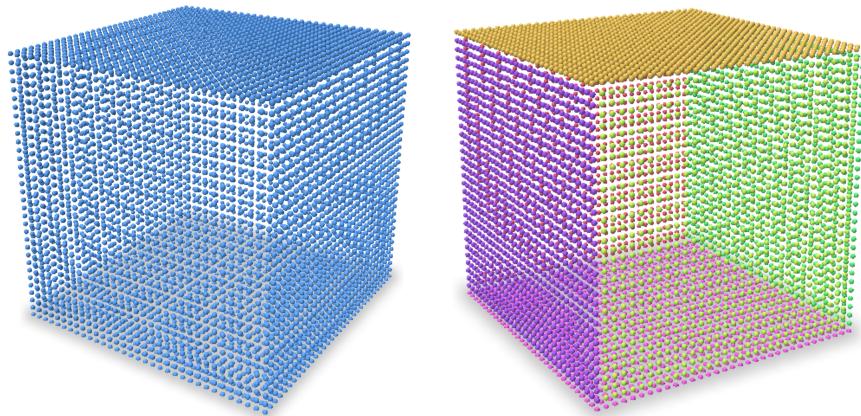


Figura 1: Ejemplo de detección de planos

## 2. Entregable

Para esta tarea se debe entregar el código de Python que implementa sus algoritmos.

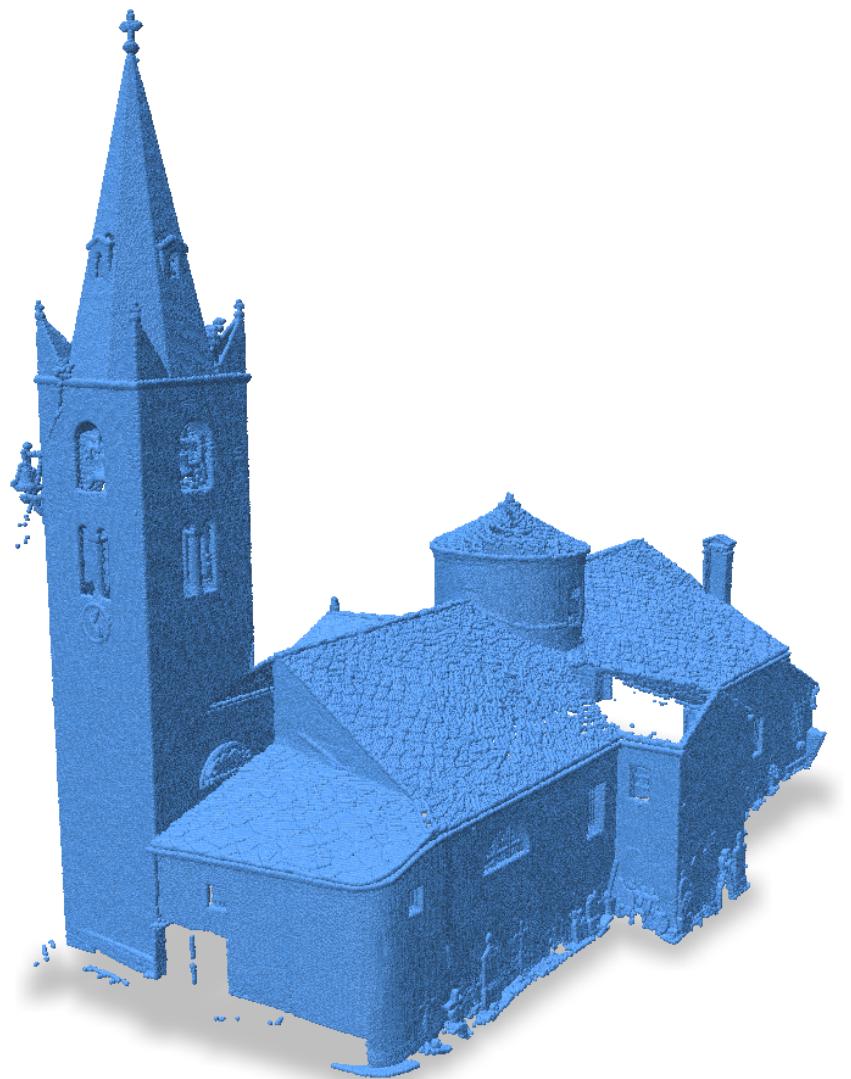


Figura 2: Nube de puntos de iglesia escaneada.