Riconoscimento di piani all'interno di point cloud acquisiti tramite kinect con calcolo del tempo di elaborazione

15 Aprile 2012

Sommario

Introduzione

La Point Cloud Library (PCL) è una libreria che permette di processare uno o più insiemi di punti dello spazio con lo scopo di utilizzare le informazioni ottenute dall'elaborazione per i più disparati campi (identificazione di piani, di oggetti, di superfici, ecc …).

La libreria in questione basa tutto il suo funzionamento su una struttura dati chiamata PointClouds (riferimento http://www.pointclouds.org/documentation/tutorials/basic_structures.php#basic-structures).

Il PointCloud è una struttura c++ definita all'interno della libreria PCL con lo scopo di contenere tutte le informazioni utili relative ai punti costituenti una qualsiasi immagine.
Tale struttura contiene, di conseguenza, le seguenti informazioni:

1. **width**: larghezza del pointcloud (quindi se vogliamo è la quantità di punti che costituiscono la larghezza dell'immagine);
2. **height**: altezza del pointcloud. Essa può rappresentare il numero di punti che costituiscono l'altezza dell'immagine oppure, se settato a 1 indica che il pointcloud è Disorganizzato;
3. **Points**: è un vettore di punti. Se un punto è identificato dalla terna xyz allora points conterrà un insieme di vettori **pcl::PointXYZ** (x,y e z identificheranno la posizione del punto all'interno della nuvola di punti), mentre se un punto è identificato dalla terna xyz e dal suo colore, allora conterrà un insieme di vettori **pcl::PointXYZRGB** (dove x,y e z identificano sempre la posizione del punto all'interno della nuvola di punti, mentre r,g,b sono i corrispondenti colori di questi punti (r:red, g:green, b:blue))
4. **isdense**: se settato a true indica che i dati contenuti in points sono finiti, altrimenti indica che alcuni valori XYZ per i punti potrebbero contenere un valore Infinito (Inf/Nan)
5. **sensor_origin_** :Specifica la "posa" del sensore di acquisizione
6. **sensor_orientation_** : Specidica l' "orientamento" del sensore di acquisizione

La funzione segment

La libreria PCL mette a disposizione diverse sotto-librerie che permettono, tramite le classi e le funzioni in esse contenute, di estrapolare da un PointCloud le informazioni volute.

Una di queste possibili sotto-librerie è la "Segmentation", che permette di dividere i punti appartenenti ad una nuvola di pointclouds in diversi sottogruppi (cluster) a seconda di alcune caratteristiche comuni (distanza euclidea, deviazione angolare e/o altro).

Diverse sono le classi disponibili in questa libreria, ed una di esse è *pcl::SACSegmentation< PointT >*

Essa rappresenta il fulcro della classe Segmentation per i Sample Consensus methods e per i modelli.
Tra le varie funzioni messe a disposizione dalla classe in questione, vi è la funzione "segment" che

restituisce, in base a dei parametri definiti prima di essere richiamata, un pointcloud contenente tutti e soli i punti aventi in comune qualcosa (a seconda di ciò che si sceglie come "collante"). L'insieme di punti restituito è quello più numeroso, ovvero quello che contiene più punti tra tutti quelli possibili.
In realtà, la funzione segment, restituisce:

a) la posizione nel vettore di punti passatogli come argomento, dei punti ritenuti appartenenti all'insieme scelto

b) i coefficienti dell'equazione del piano identificato;

Pertanto è necessario definire, prima di richiamare la funzione, un vettore di indici in questo modo:

```
pcl::PointIndices::Ptr inliers (new pcl::PointIndices);
```

Qui di seguito elencheremo le funzioni attraverso le quali è' possibile settare i parametri utilizzabili per ottenere una diversa suddivisione dei punti:

- **void setModelType (int model)** -> che specifica il tipo di modello da utilizzare (per la suddivisione in piani usiamo pcl::SACMODEL_PLANE)

  (qualora si voglia conoscere il tipo di modello utilizzato si può utilizzare la funzione **getModelType ()**)

- **void setMethodType (int method)** -> che specifica quale tipo di metodo di sample consensus utilizzare (per la suddivisione in piani usiamo pcl::SAC_RANSAC)

- **void setDistanceThreshold (double threshold)** -> che specifica la distanza da utilizzare come soglia nella scelta dei punti (noi ad esempio abbiamo utilizzato il valore 0.01)!

- **void setMaxIterations (int max_iterations)** -> che setta il numero Massimo di iterazioni prima di fermarsi (noi abbiamo impostato come soglia 1000)

- **void setProbability (double probability)** -> che setta la probabilità di scegliere almeno un campione fuori da quelli scelti

- **void setOptimizeCoefficients (bool optimize)** -> è settato a true se è richiesto un coefficient refinement

- **void setRadiusLimits (const double &min_radius, const double &max_radius)** -> setta il limite minimo e massimo per il raggio (applicabile solo ai modelli che stimano il raggio)

- **void setAxis (const Eigen::Vector3f &ax)** -> setta gli assi lungo i quali si dovrebbe cercare

- **void setEpsAngle (double ea)** -> setta la soglia per l'angolo epsilon (delta)

- **virtual void setInputCloud (const PointCloudConstPtr &cloud)** -> permette di settare il pointcloud su cui eseguire l'operazione di segmentazione

La funzione filter

Oltre alla funzione segment, appartenente alla libreria segmentation, abbiamo utilizzato anche la funzione filter della libreria filters.
Tale funzione è stata utilizzata per estrarre dal pointcloud originale solo i punti identificati come appartenenti al gruppo individuato.

In particolare abbiamo utilizzato le seguenti funzioni per settare i parametri necessari:

- **virtual void setInputCloud (const PointCloudConstPtr &cloud)** -> permette di settare il pointcloud su cui eseguire l'operazione di estrazione
- **void setIndices (const IndicesPtr &indices)** -> permette di indicare gli indici relativi ai punti identificati come appartenenti ad un insieme individuato
- **void setNegative (bool negative)** -> permette di settare se deve essere applicata una condizione regolare di filtraggio o l'opposta (in particolare si è settato a false per estrarre i punti appartenenti al gruppo individuato (e quindi salvarli su un file), mentre si è settata a true per estrarre tutti gli altri punti e poter quindi riapplicare la segmentazione sui punti rimasti)

Programma realizzato

Mi si era richiesto di realizzare una funzione che, qualora venisse richiamata, restituisse:

a) i piani estratti da una nuvola di punti (in termini di equazione del piano e di pointcloud vero e proprio salvato in un file)

b) il tempo necessario per estrarre i piani

c) un grafico contenente tutti i punti indicati con un colore diverso a seconda del piano al quale appartengono

Ho di conseguenza definito una funzione int **estrazione_piani_con_tempi**(std::string nome_file_input_cloud,double percentuale_scartati
che richiede questi parametri in input:

a) **std::string nome_file_input_cloud** -> nome del file pcd (con estensione) da processare come pointcloud iniziale

b) **double percentuale_scartati** -> percentuale di punti scartati dall'estrazione (se dovessimo estrarre tutti i possibili piani individuabili ci vorrebbe troppo tempo

quindi settando una percentuale_scartati pari a 0.3 si scartano soltanto il 30% dei punti!)

Essa estrae i piani da un point_cloud riportando:

a) il tempo impiegato per estrarre ogni singolo piano
b) il tempo totale impiegato per l'estrazione di tutti i piani
c) le equazioni di tutti i piani estratti nella forma ax+by+cz+d=0
d) i file pcd contenenti le vere e proprie point cloud

Risultati ottenuti

Di seguito riporto i risultati (in termini di tempo, di equazioni dei piani ottenuti e di immagine con i vari piani distinti attraverso colori diversi) ottenuti con alcuni pointcloud significativi.
Per i test è stato utilizzato un notebook HP con le seguenti caratteristiche:

- Processore: Intel Core i7 Q720 (quadcore con frequenza del singolo clock pari a 1,60GHz e TurboBoost: 2,8 GHz)
- RAM: 4GB
- Sistema operativo: Linux Debian 6.0

1. Insieme di scatole appoggiate al muro

nome point cloud in ingresso: **frame_20120328T120720.171322**

- Equazione del **1° piano** generato: 0.871526x -0.0266074y + 0.489627z -1.08079=0
  Nome del file generato: frame_20120328T120720.171322_plane_0.pcd
  Tempo impiegato per estrazione piano 1-esimo è 329.672 ms

- Equazione del **2° piano** generato: 0.868186x -0.0352929y + 0.494982z -0.81433=0
  Nome del file generato: frame_20120328T120720.171322_plane_1.pcd
  Il tempo impiegato per estrazione piano 2-esimo è 855.205 ms

- Equazione del **3° piano** generato: -0.413079x + 0.0115551y + 0.910622z -2.4497=0
  Nome del file generato: frame_20120328T120720.171322_plane_2.pcd
  Il tempo impiegato per estrazione piano 3-esimo è 699.675 ms

- Equazione del **4° piano** generato: 0.00411589x -0.000531837y + 0.999991z -5.19986=0
  Nome del file generato: frame_20120328T120720.171322_plane_3.pcd
  Il tempo impiegato per estrazione piano 4-esimo è 648.693 ms

- Equazione del **5° piano** generato: 0.837402x -0.0872338y + 0.539582z -0.497705=0
  Nome del file generato: frame_20120328T120720.171322_plane_4.pcd
  Il tempo impiegato per estrazione piano 5-esimo è 659.029 ms

- Equazione del **6° piano** generato: -0.400698x + 0.0973385y + 0.911025z -2.40858=0
  Nome del file generato: frame_20120328T120720.171322_plane_5.pcd
  Il tempo impiegato per estrazione piano 6-esimo è 579.062 ms

- Equazione del **7° piano** generato: -0.0291486x + 0.00337118y + 0.999569z -5.05658=0
  Nome del file generato: frame_20120328T120720.171322_plane_6.pcd
  Il tempo impiegato per estrazione piano 7-esimo è 539.298 ms

- Equazione del **8° piano** generato: -0.00706724x + 0.00070553y + 0.999975z -4.92948=0
  Nome del file generato: frame_20120328T120720.171322_plane_7.pcd
  Il tempo impiegato per estrazione piano 8-esimo è 497.884 ms

- Equazione del **9° piano** generato: 0.616287x + 0.00926029y + 0.787467z -1.25091=0
  Nome del file generato: frame_20120328T120720.171322_plane_8.pcd
  Il tempo impiegato per estrazione piano 9-esimo è 453.184 ms

- Equazione del **10° piano** generato: -0.000965259x + 0.000153819y + 1z -5.29331=0
  Nome del file generato: frame_20120328T120720.171322_plane_9.pcd
  Il tempo impiegato per estrazione piano 10-esimo è 427.536 ms

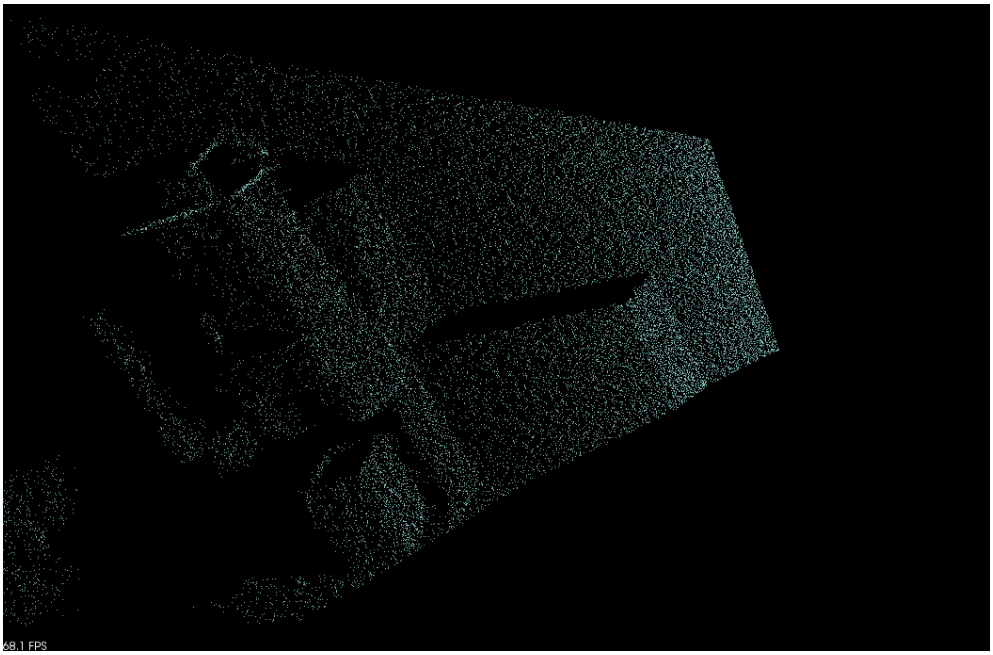
- Equazione del **11° piano** generato: 0.020196x -0.002192y + 0.999794z -5.07978=0
  Nome del file generato: frame_20120328T120720.171322_plane_10.pcd
  Il tempo impiegato per estrazione piano 11-esimo è 412.102 ms

- Equazione del **12° piano** generato: 0.749032x + 0.0885481y + 0.65659z -1.20125=0
  Nome del file generato: frame_20120328T120720.171322_plane_11.pcd
  Il tempo impiegato per estrazione piano 12-esimo è 374.48 ms

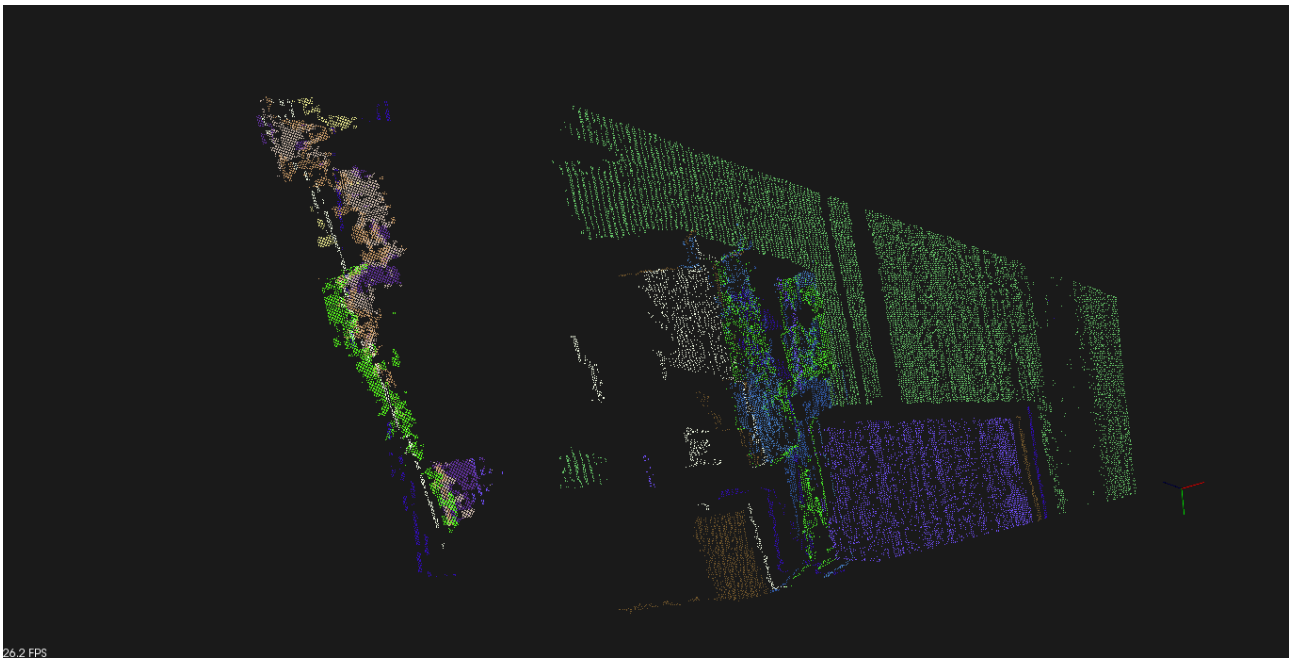
- Equazione del **13° piano** generato: -0.504014x -0.0497668y + 0.86226z -2.39178=0
  Nome del file generato: frame_20120328T120720.171322_plane_12.pcd
  Il tempo impiegato per estrazione piano 13-esimo è 353.796 ms

Il **tempo MEDIO** impiegato per l'estrazione di ogni piano è 525.355 ms
Il **tempo TOTALE** impiegato per l'estrazione di 13 piani è 6829.62 ms

Pointcloud **prima** della suddivisione in piani:



Pointcloud **dopo** la suddivisione in piani:



2. Scatolone appoggiato ad un muro

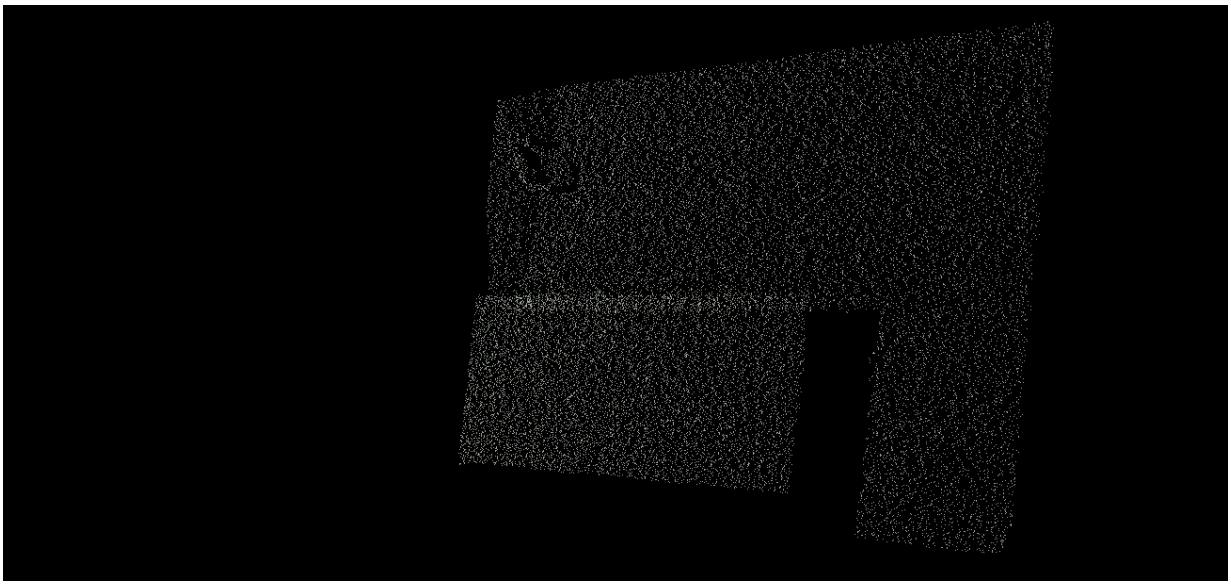
nome point cloud in ingresso: **frame_20120328T121055.126081**

- Equazione del **1° piano** generato: 0.0395957x + 0.073554y + 0.996505z -1.63978=0
  Nome del file generato: frame_20120328T121055.126081_plane_0.pcd
  Tempo impiegato per estrazione piano 1-esimo è 103.353 ms

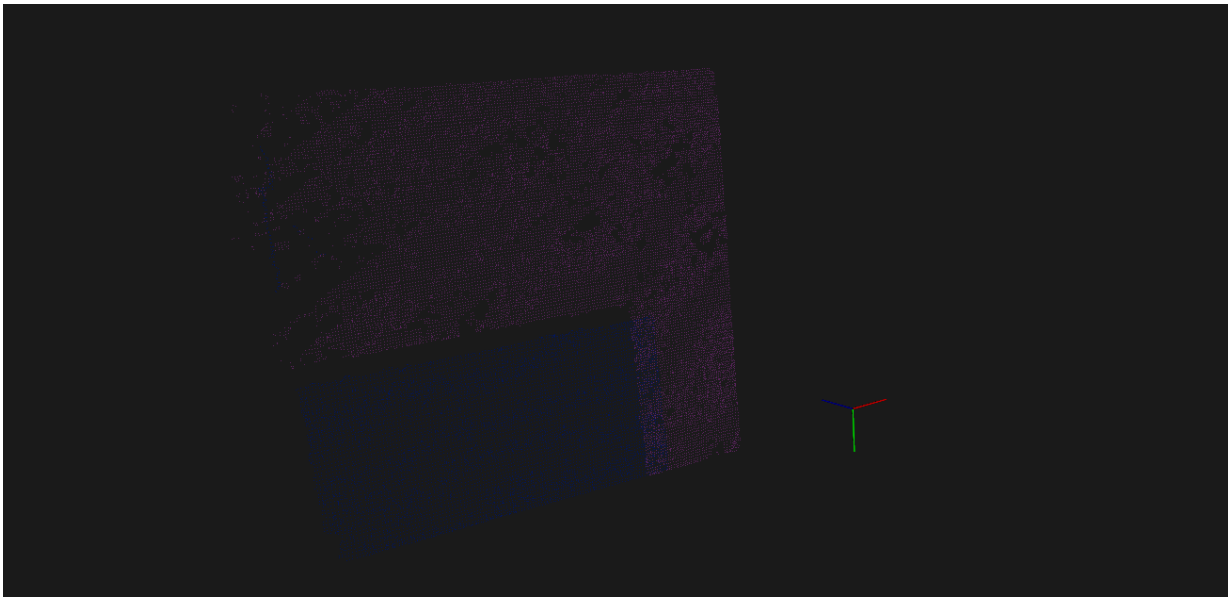
- Equazione del **2° piano** generato: 0.00674501x + 0.0795425y + 0.996809z -1.36641=0
  Nome del file generato: frame_20120328T121055.126081_plane_1.pcd
  Tempo impiegato per estrazione piano 1-esimo è 50.596 ms

Il **tempo MEDIO** impiegato per l'estrazione di ogni piano è 76.9745 ms
Il **tempo TOTALE** impiegato per l'estrazione di 2 piani è 153.949 ms

Pointcloud **prima** della suddivisione in piani:



Pointcloud **dopo** la suddivisione in piani:



3. Armadio con “porte” scorrevoli posizionate su piani diversi

nome point cloud in ingresso: **frame_20120328T121248.738729**

- Equazione del **1° piano** generato: 0.0616331x + 0.0550001y + 0.996582z -1.4905=0
  Nome del file generato: frame_20120328T121248.738729_plane_0.pcd
  Il tempo impiegato per estrazione piano 1-esimo è 249.268 ms

- Equazione del **2° piano** generato: 0.0544635x -0.0651053y + 0.996391z -1.43115=0
  Nome del file generato: frame_20120328T121248.738729_plane_1.pcd
  Il tempo impiegato per estrazione piano 2-esimo è 202.877 ms

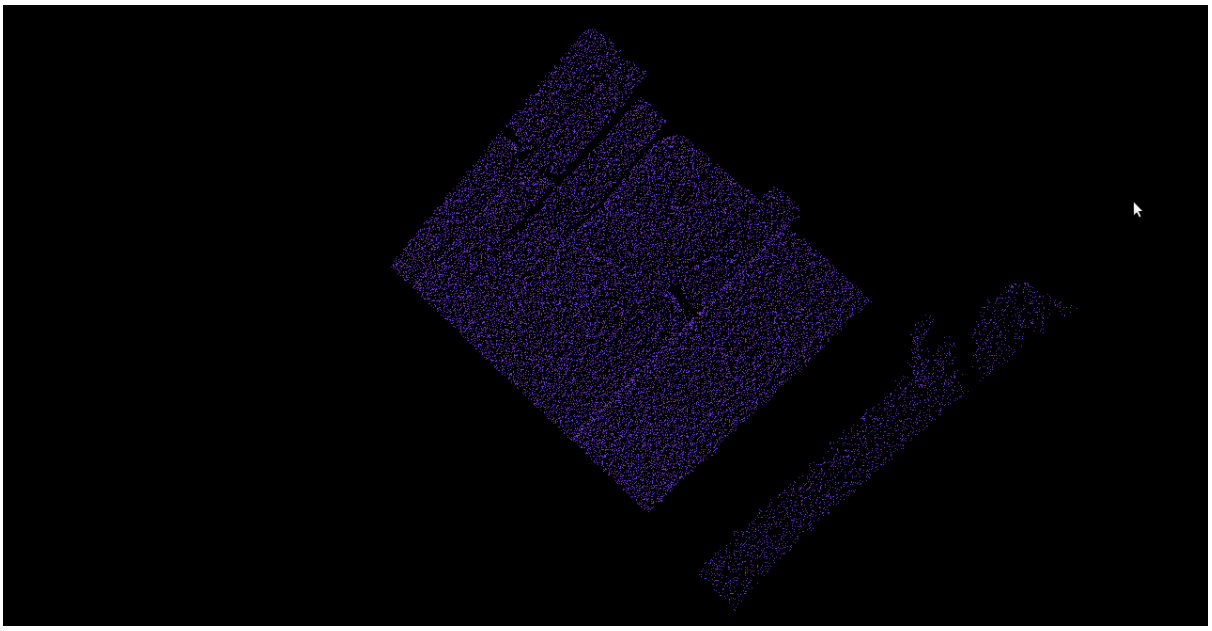
- Equazione del **3° piano** generato: 0.100557x -0.0643197y + 0.99285z -1.58=0
  Nome del file generato: frame_20120328T121248.738729_plane_2.pcd
  Il tempo impiegato per estrazione piano 3-esimo è 92.323 ms

- Equazione del **4° piano** generato: 0.0577629x -0.0615967y + 0.996428z -2.46041=0
  Nome del file generato: frame_20120328T121248.738729_plane_3.pcd
  Il tempo impiegato per estrazione piano 4-esimo è 157.234 ms

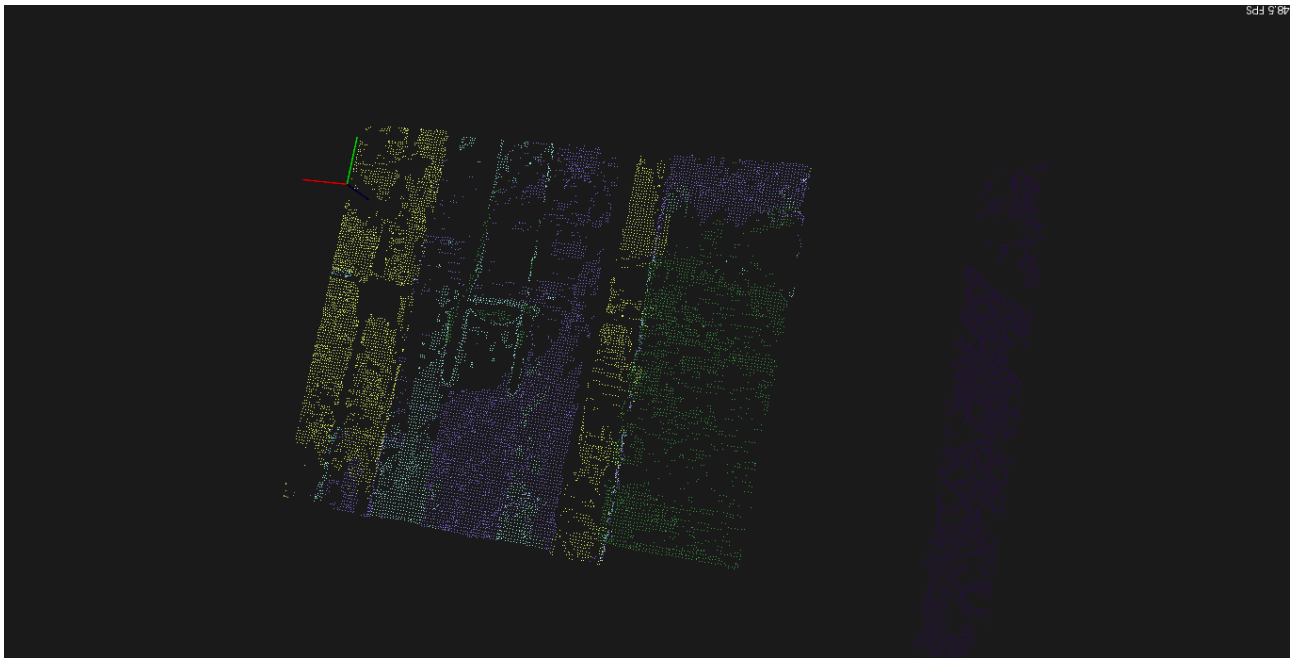
- Equazione del **5° piano** generato: 0.0639114x + 0.0528645y + 0.996554z -1.47666=0
  Nome del file generato: frame_20120328T121248.738729_plane_4.pcd
  Il tempo impiegato per estrazione piano 5-esimo è 186.478 ms

Il tempo MEDIO impiegato per l'estrazione di ogni piano è 177.636 ms
Il tempo TOTALE impiegato per l'estrazione di 5 piani è 888.18 ms

Pointcloud **prima** della suddivisione in piani:



Pointcloud **dopo** la suddivisione in piani:



4. Immagine del laib10 presa dalla porta d'ingresso (ci sono 2 file di computer con in mezzo delle persone e delle sedie)

nome point cloud in ingresso: **frame_20120328T121656.848028**

- Equazione del **1° piano** generato: 0.0477236x -0.979881y + 0.193791z -1.82837=0
  Nome del file generato: frame_20120328T121656.848028_plane_0.pcd
  Il tempo impiegato per estrazione piano 1-esimo è 2102.8 ms

- Equazione del **2° piano** generato: 0.0947219x -0.973835y + 0.206574z -1.86104=0
  Nome del file generato: frame_20120328T121656.848028_plane_1.pcd
  Il tempo impiegato per estrazione piano 2-esimo è 1839.03 ms

- Equazione del **3° piano** generato: 0.0460174x -0.979809y + 0.194566z -1.81498=0
  Nome del file generato: frame_20120328T121656.848028_plane_2.pcd
  Il tempo impiegato per estrazione piano 3-esimo è 1665.79 ms

- Equazione del **4° piano** generato: 0.049009x -0.979614y + 0.194819z -1.85157=0
  Nome del file generato: frame_20120328T121656.848028_plane_3.pcd
  Il tempo impiegato per estrazione piano 4-esimo è 1545.38 ms

- Equazione del **5° piano** generato: 0.0763161x -0.971173y + 0.225827z -1.88322=0
  Nome del file generato: frame_20120328T121656.848028_plane_4.pcd
  Il tempo impiegato per estrazione piano 5-esimo è 1408.22 ms

- Equazione del **6° piano** generato: 0.0390186x -0.980039y + 0.194941z -1.79726=0

Nome del file generato: frame_20120328T121656.848028_plane_5.pcd
Il tempo impiegato per estrazione piano 6-esimo è 1321.65 ms

- Equazione del **7° piano** generato: 0.0552878x -0.976127y + 0.210047z -1.839=0
  Nome del file generato: frame_20120328T121656.848028_plane_6.pcd
  Il tempo impiegato per estrazione piano 7-esimo è 1267.22 ms

- Equazione del **8° piano** generato: 0.0433393x -0.979833y + 0.195064z -1.76213=0
  Nome del file generato: frame_20120328T121656.848028_plane_7.pcd
  Il tempo impiegato per estrazione piano 8-esimo è 1211.61 ms

- Equazione del **9° piano** generato: -0.0167096x -0.992314y + 0.122612z + 0.736312=0
  Nome del file generato: frame_20120328T121656.848028_plane_8.pcd
  Il tempo impiegato per estrazione piano 9-esimo è 1169.38 ms

- Equazione del **9° piano** generato: 0.102772x -0.979733y + 0.171934z -1.71211=0
  Nome del file generato: frame_20120328T121656.848028_plane_9.pcd
  Il tempo impiegato per estrazione piano 10-esimo è 1136.69 ms

- Equazione del **10° piano** generato: 0.124752x -0.971837y + 0.199926z -1.83969=0
  Nome del file generato: frame_20120328T121656.848028_plane_10.pcd
  Il tempo impiegato per estrazione piano 11-esimo è 1102.56 ms

- Equazione del **11° piano** generato: -0.0171849x -0.984981y + 0.171804z + 0.580714=0
  Nome del file generato: frame_20120328T121656.848028_plane_11.pcd
  Il tempo impiegato per estrazione piano 12-esimo è 1071.29 ms

- Equazione del **12° piano** generato: 0.0775858x -0.973313y + 0.21597z -1.84391=0
  Nome del file generato: frame_20120328T121656.848028_plane_12.pcd
  Il tempo impiegato per estrazione piano 13-esimo è 1074.63 ms

- Equazione del **13° piano** generato: 0.0487346x -0.979665y + 0.19463z -1.86643=0
  Nome del file generato: frame_20120328T121656.848028_plane_13.pcd
  Il tempo impiegato per estrazione piano 14-esimo è 1111.94 ms

- Equazione del **14° piano** generato: 0.0770843x + 0.00211523y + 0.997022z -6.71769=0
  Nome del file generato: frame_20120328T121656.848028_plane_14.pcd
  Il tempo impiegato per estrazione piano 15-esimo è 1105.9 ms

- Equazione del **15° piano** generato: 2.34561e-05x -3.99325e-05y + 1z -6.88883=0
  Nome del file generato: frame_20120328T121656.848028_plane_15.pcd
  Il tempo impiegato per estrazione piano 16-esimo è 1048.38 ms

- Equazione del **16° piano** generato: -0.00383424x -0.994229y + 0.107212z + 0.750451=0
  Nome del file generato: frame_20120328T121656.848028_plane_16.pcd
  Il tempo impiegato per estrazione piano 17-esimo è 1023.23 ms

- Equazione del **17° piano** generato: 0.05664x -0.968762y + 0.241438z -1.92864=0
  Nome del file generato: frame_20120328T121656.848028_plane_17.pcd
  Il tempo impiegato per estrazione piano 18-esimo è 942.347 ms

- Equazione del **18° piano** generato: -0.0387231x -0.000197672y + 0.99925z -7.79956=0
  Nome del file generato: frame_20120328T121656.848028_plane_18.pcd
  Il tempo impiegato per estrazione piano 19-esimo è 949.069 ms

- Equazione del **19° piano** generato: 1.91386e-06x + 1.50991e-05y + 1z -6.14603=0
  Nome del file generato: frame_20120328T121656.848028_plane_19.pcd
  Il tempo impiegato per estrazione piano 20-esimo è 864.374 ms

- Equazione del **20° piano** generato: 0.0552609x -0.00152801y + 0.998471z -6.53393=0
  Nome del file generato: frame_20120328T121656.848028_plane_20.pcd
  Il tempo impiegato per estrazione piano 21-esimo è 839.674 ms

- Equazione del **21° piano** generato: 0.114934x -5.84764e-05y + 0.993373z -6.14749=0
  Nome del file generato: frame_20120328T121656.848028_plane_21.pcd
  Il tempo impiegato per estrazione piano 22-esimo è 813.837 ms

- Equazione del **22° piano** generato: 0.116874x -0.973401y + 0.197054z -1.87117=0
  Nome del file generato: frame_20120328T121656.848028_plane_22.pcd
  Il tempo impiegato per estrazione piano 23-esimo è 804.031 ms

- Equazione del 23**° piano** generato: -0.121955x -0.0138138y + 0.99244z -7.88306=0
  Nome del file generato: frame_20120328T121656.848028_plane_23.pcd
  Il tempo impiegato per estrazione piano 24-esimo è 845.716 ms

- Equazione del **24° piano** generato: 0.100425x -0.978382y + 0.180783z + 0.638344=0
  Nome del file generato: frame_20120328T121656.848028_plane_24.pcd
  Il tempo impiegato per estrazione piano 25-esimo è 852.812 ms

- Equazione del **25° piano** generato: 0.0803577x -0.981735y + 0.172452z + 0.400006=0
  Nome del file generato: frame_20120328T121656.848028_plane_25.pcd
  Il tempo impiegato per estrazione piano 26-esimo è 783.073 ms

- Equazione del **26° piano** generato: 0.357882x -0.924969y + 0.127875z + 0.526117=0
  Nome del file generato: frame_20120328T121656.848028_plane_26.pcd
  Il tempo impiegato per estrazione piano 27-esimo è 762.608 ms

- Equazione del **27° piano** generato: 0.141035x -0.970352y + 0.196281z -1.87426=0
  Nome del file generato: frame_20120328T121656.848028_plane_27.pcd
  Il tempo impiegato per estrazione piano 28-esimo è 770.909 ms

- Equazione del **28° piano** generato: -0.0138684x -0.992522y + 0.121278z + 0.780408=0
  Nome del file generato: frame_20120328T121656.848028_plane_28.pcd
  Il tempo impiegato per estrazione piano 29-esimo è 713.182 ms

- Equazione del **29° piano** generato: -0.039274x -0.989882y + 0.136349z + 0.616039=0
  Nome del file generato: frame_20120328T121656.848028_plane_29.pcd
  Il tempo impiegato per estrazione piano 30-esimo è 692.165 ms

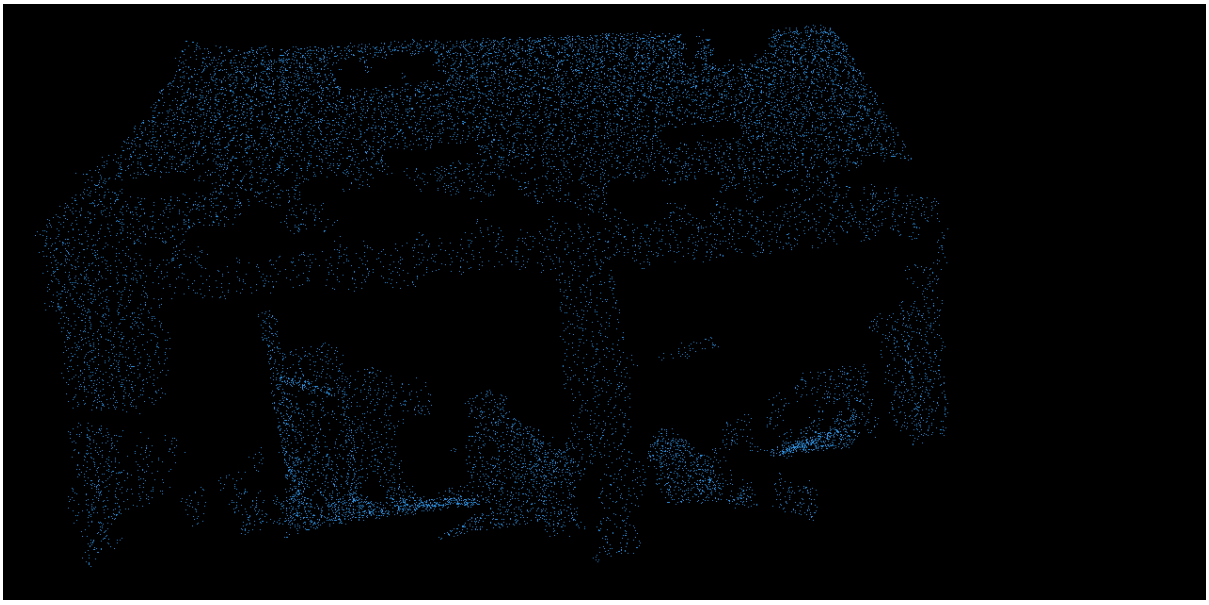
- Equazione del 30**° piano** generato: 0.000118656x + 0.000150047y + 1z -7.66039=0
  Nome del file generato: frame_20120328T121656.848028_plane_30.pcd
  Il tempo impiegato per estrazione piano 31-esimo è 680.699 ms

- Equazione del **31° piano** generato: -0.115844x -0.992714y + 0.0331646z + 1.16462=0
  Nome del file generato: frame_20120328T121656.848028_plane_31.pcd
  Il tempo impiegato per estrazione piano 32-esimo è 646.478 ms

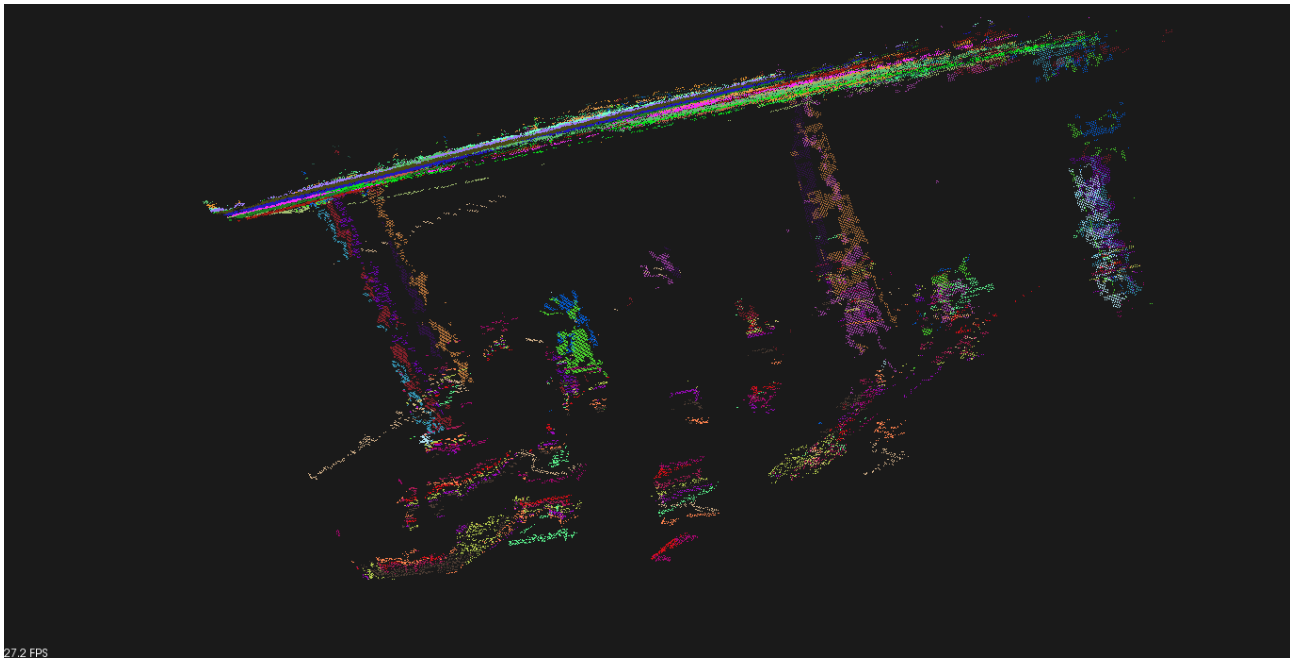
- Equazione del **32° piano** generato: 6.3094e-06x + 8.53269e-05y + 1z -6.03813=0
  Nome del file generato: frame_20120328T121656.848028_plane_32.pcd
  Il tempo impiegato per estrazione piano 33-esimo è 635.246 ms

Il tempo MEDIO impiegato per l'estrazione di ogni piano è 1054.6 ms
Il tempo TOTALE impiegato per l'estrazione di 33 piani è 34801.9 ms

Pointcloud **prima** della suddivisione in piani:



Pointcloud **dopo** la suddivisione in piani:



Appendice

A. Come compilare un programma fatto utilizzando la libreria PCL

1. Installare la libreria nel proprio sistema (a seconda del proprio sistema operativo si può seguire un passo diverso: si veda la pagina http://pointclouds.org/downloads/)
2. Se si è in ambiente linux installare il pacchetto cmake (con il comando apt-get install cmake)
3. Scrivere il proprio codice c++ e salvare il file (mio_file.cpp) in una cartella di proprio gusto (diciamo /home/mio_nome/pcl
4. Creare, nella stessa cartella in cui abbiamo salvato il nostro file, un nuovo file di nome CmakeLists.txt
5. All'interno di questo file inserire le direttive da passare a Cmake per compilare il nostro programma cpp e creare l'eseguibile (per informazioni dettagliate su cosa scrivere nel suddetto file CmakeLists.txt, si faccia riferimento alla pagina http://www.pointclouds.org/documentation/tutorials/using_pcl_pcl_config.php#using-pcl-pcl-config)
6. Aprire il terminale (con privilegi di amministratore) e, dopo essersi posizionati nella cartella in cui si è salvato il file, lanciare il comando “cmake .”
7. Dopo di che, sempre rimanendo nella cartella in questione, lanciare il comando make
8. A questo punto si avrà il file eseguibile da utilizzare (da lanciare con ./nome_file)

B. Codice sorgente

Di seguito riporto il codice sorgente realizzato:

```
#include <iostream>
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/filters/voxel_grid.h>
#include <pcl/filters/extract_indices.h>
#include <sys/time.h>
#include <string>
#include <pcl/visualization/cloud_viewer.h>


//funzione che estrae i piani da un point_cloud riportando il tempo impiegato per estrarre ogni singolo piano
// e il tempo totale impiegato per l'estrazione di tutti i piani
```

```
//parametri da passare in input:
//a) nome del file pcd (con estensione) da processare come pointcloud iniziale
//b) percentuale di punti scartati dall'estrazione (se dovessimo estrarre tutti i possibili piani individuabili ci vorrebbe troppo tempo
	// quindi settando una percentuale_scartati pari a 0.3 si scartano soltanto il 30% dei punti!
int estrazione_piani_con_tempi(std::string nome_file_input_cloud,double percentuale_scartati)
{
	std::string nome_file_senza_estensione;
	int lunghezza= nome_file_input_cloud.capacity() - 4;
	nome_file_senza_estensione=nome_file_input_cloud.substr(0,lunghezza);
	//strncpy(nome_file_senza_estensione, nome_file_input_cloud, strlen(nome_file_input_cloud) - 4);
	std::cout << "nome: " << nome_file_senza_estensione <<"\n";
	std::cout << "lunghezza: " << lunghezza << "\n";

	timeval start1, stop1, start2, stop2;
	double elapsedTime[100];
	double somma_tempi = 0.0;



	pcl::PointCloud<pcl::PointXYZRGB>::Ptr plane_cloud (new pcl::PointCloud<pcl::PointXYZRGB>); //point cloud che conterrÃ  il risultato della segmentazione: tutti i punti colorati in modo diverso a seconda del piano di appartenenza
	sensor_msgs::PointCloud2::Ptr cloud_blob (new sensor_msgs::PointCloud2), cloud_filtered_blob (new sensor_msgs::PointCloud2);
	pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_filtered (new pcl::PointCloud<pcl::PointXYZ>), cloud_p (new pcl::PointCloud<pcl::PointXYZ>), cloud_f (new pcl::PointCloud<pcl::PointXYZ>);

	  pcl::PCDWriter writer;
	pcl::PCDReader reader;
	reader.read (nome_file_input_cloud, *cloud_blob);

	// per velocizzare il processo di estrazione dei piani viene filtrato il point cloud togliendo tutti quei punti che al fine dell'identificazione del piano
	// sono in piÃ¹
	std::cerr << "PointCloud before filtering: " << cloud_blob->width * cloud_blob->height << " data points." << std::endl;

	// Create the filtering object: downsample the dataset using a leaf size of 1cm
	pcl::VoxelGrid<sensor_msgs::PointCloud2> sor;
	sor.setInputCloud (cloud_blob);
	sor.setLeafSize (0.01f, 0.01f, 0.01f);
	sor.filter (*cloud_filtered_blob);

	// Convert to the templated PointCloud
	pcl::fromROSMsg (*cloud_filtered_blob, *cloud_filtered);

	std::cerr << "PointCloud after filtering: " << cloud_filtered->width * cloud_filtered->height << " data points." << std::endl << std::endl;

	// Scriviamo la versione filtrata dei punti in un file con nome nome_file_downsampled.pcd
	writer.write<pcl::PointXYZ> (nome_file_senza_estensione+"_downsampled.pcd", *cloud_filtered, false);

	//creiamo le variabili necessarie per contenere i risultati dell'operazione di filtering:
	// - coefficients che conterrÃ  i coefficienti dell'equazione del piano estratto (i coefficienti sono a,b,c,d e l'equazione è nella forma: ax+by+cz+d=0)
	// - inliers che conterrÃ  degli indici corrispondenti alla posizione dei punti scelti all'interno del pointcloud filtrato (downsampled)

	pcl::ModelCoefficients::Ptr coefficients (new pcl::ModelCoefficients ());
	pcl::PointIndices::Ptr inliers (new pcl::PointIndices ());

	//settiamo i parametri necessari alla segmentazione
	// Create the segmentation object
	pcl::SACSegmentation<pcl::PointXYZ> seg;
	// Optional
```

```
	seg.setOptimizeCoefficients (true);
	// Mandatory
	seg.setModelType (pcl::SACMODEL_PLANE);
	seg.setMethodType (pcl::SAC_RANSAC);
	seg.setMaxIterations (1000);
	seg.setDistanceThreshold (0.01);

	// Create the filtering object
	pcl::ExtractIndices<pcl::PointXYZ> extract;


	//definisco la dimensione del point cloud totale (quello che conterrÃ  sia i punti xyz,
		// che i colori dei punti, che saranno diversi a seconda del piano di appartenenza
	plane_cloud->width  = cloud_filtered->width;
	plane_cloud->height = 1;
	plane_cloud->points.resize (plane_cloud->width * plane_cloud->height);


 int i = 0, nr_points = (int) cloud_filtered->points.size ();
 // Ripetiamo il processo di segmentazione fino a quando non rimane meno del 30% dei punti iniziali
 while (cloud_filtered->points.size () > percentuale_scartati * nr_points)
 {
	//salviamo in una variabile start1 il timestamp all'inizio dell'esecuzione della procedura di cui vogliamo misurare
il tempo di esecuzione
		gettimeofday(&start1, NULL);

	// Segment the largest planar component from the remaining cloud
	seg.setInputCloud (cloud_filtered);
	seg.segment (*inliers, *coefficients);
	if (inliers->indices.size () == 0)
	{
	std::cerr << "Could not estimate a planar model for the given dataset." << std::endl;
	break;
	}

	// Extract the inliers
	extract.setInputCloud (cloud_filtered);
	extract.setIndices (inliers);
	extract.setNegative (false);
	extract.filter (*cloud_p);

	//salviamo in una variabile stop1 il timestamp al termine della prima parte
		//della procedura di cui vogliamo misurare il tempo di esecuzione
		//(prima di riempire il pointcloud finale settando i colori del piano

		gettimeofday(&stop1, NULL);

	//genero un colore per il piano in oggetto
		// Estraggo un numero compreso fra 0 e 255
		int numero = rand() % 256;
		// Estraggo un numero compreso fra 0 e 255
		int numero2 = rand() % 256;
		// Estraggo un numero compreso fra 0 e 255
		int numero3 = rand() % 256;
		//std::cerr << "colore 1:" << numero << "\n" << "colore2: " << numero2 << "\ncolore3: " << numero3 <<
std::endl;

	//Setto il colore dei punti di questo piano per la visualizzazione
		for (size_t l = 0; l < inliers->indices.size (); ++l)
		{

			plane_cloud->points[inliers->indices[l]].x = cloud_filtered->points[inliers->indices[l]].x ;
			plane_cloud->points[inliers->indices[l]].y = cloud_filtered->points[inliers->indices[l]].y ;
			plane_cloud->points[inliers->indices[l]].z = cloud_filtered->points[inliers->indices[l]].z ;
			plane_cloud->points[inliers->indices[l]].r = numero;
			plane_cloud->points[inliers->indices[l]].g = numero2;
```

```
                        plane_cloud->points[inliers->indices[l]].b = numero3;
                }

        //stampo le informazioni sul piano generato
        std::cerr << "Numero di PointCloud costituenti il "<< i << "Â° piano estratto: " << cloud_p->width * cloud_p-
>height << " data points." << std::endl;
        std::cerr << "Equazione del piano generato: " << coefficients->values[0] << "x ";
        if (coefficients->values[1] >= 0)
        {
                std::cerr << "+ ";
        }
        std::cerr << coefficients->values[1] << "y ";
        if (coefficients->values[2] >= 0)
        {
                std::cerr << "+ ";
        }
        std::cerr << coefficients->values[2] << "z ";
        if (coefficients->values[3] >= 0)
        {
                std::cerr << "+ ";
        }
        std::cerr << coefficients->values[3] << "=0" << std::endl;
        std::cerr << "Nome del file generato: " << nome_file_senza_estensione+"_plane_" << i << ".pcd" << std::endl;

        //salviamo in un file il risultato della segmentazione
        std::stringstream ss;
        ss << nome_file_senza_estensione+"_plane_" << i << ".pcd";
        writer.write<pcl::PointXYZ> (ss.str (), *cloud_p, false);


        //salviamo in una variabile start2 il timestamp all'inizio dell'esecuzione della procedura di cui vogliamo misurare
il tempo di esecuzione
                gettimeofday(&start2, NULL);

        //Togliamo dal point cloud generale i punti giÃ  classificati
        // Create the filtering object
        extract.setNegative (true);
        extract.filter (*cloud_f);
        cloud_filtered.swap (cloud_f);
        i++;

                //salviamo in una variabile stop2 il timestamp al termine della procedura di cui vogliamo misurare il
tempo di esecuzione

                        gettimeofday(&stop2, NULL);

                //calcoliamo la differenza tra timestamp all'inizio della procedura e la fine e riportiamo tutto in
millisecondi
                // la funzione gettimeofday restituisce il tempo passato dallo epoch unix time
                //il tipo time_val è fatto cosÃ¬:
                //     struct timeval {
                //               time_t      tv_sec;   /* seconds since Jan. 1, 1970 */
                //               suseconds_t  tv_usec;  /* and microseconds */
                //     };
                //quindi per trasformarlo in microsecondi devo necessariamente processare prima i secondi e poi i
micro secondi!!!

                        elapsedTime[i] = ((stop1.tv_sec - start1.tv_sec) + (stop2.tv_sec - start2.tv_sec))* 1000.0;
        // da sec a ms
                        elapsedTime[i] += ((stop1.tv_usec - start1.tv_usec) + (stop2.tv_usec - start2.tv_usec)) /
1000.0;  // da us a ms

                //stampo il tempo impiegato per l'estrazione del piano i-esimo
                std::cout << "Il tempo impiegato per estrazione piano " << i << "-esimo è " << elapsedTime[i] << "
ms\n\n";
                somma_tempi+=elapsedTime[i];
```

```
 }


        //stampo il tempo Medio e il tempo totale per l'estrazione dei piani
        std::cout << "Il tempo MEDIO impiegato per l'estrazione di ogni piano è " << somma_tempi/i << " ms\n";
        std::cout << "Il tempo TOTALE impiegato per l'estrazione di " << i << " piani è " << somma_tempi << " ms\n";


        //salviamo in un file il risultato finale
        std::stringstream ss;
        ss << nome_file_senza_estensione+"_final_plane_" << ".pcd";
        writer.write<pcl::PointXYZRGB> (ss.str (), *plane_cloud, false);


        // visualizzo in un viewer il risultato dell'estrazione
        pcl::visualization::CloudViewer viewer ("Simple Cloud Viewer2");

        viewer.showCloud (plane_cloud,"piano");

        //resto in attesa che l'utente chiuda la finestra del viewer
        while (!viewer.wasStopped ())
        {
        }

        return (0);
}



int
main (int argc, char** argv)
{
        std::string nome_file_input_clod = "frame_20120328T120720.171322.pcd";
        double percentuale_scartati= 0.3;

        estrazione_piani_con_tempi(nome_file_input_clod,percentuale_scartati);

}
```