3D Point Cloud Segmentation

Nitin Deshpande · Saad Abdullah · Sheikh Radiah Rivu

Abstract This report reviews 3D point cloud segmentation. Segmentation is the process of partitioning point cloud data into multiple homogeneous regions essentially segmentation of regions with the same properties. We have used the point cloud library for segmenting 3D point cloud images which are obtained using 3D reconstruction from monocular camera.

Keywords point cloud · segmentation · robotic vision · autonomous navigation

1 Introduction

With the increase in availability of three dimensional scanners everywhere, there have been an increase in the need to process 3D point clouds. A point cloud is a set of data points in a given coordinate system and to process the point cloud data obtained we need a library which provides the necessary tools to analyze and visualize the point cloud data. To achieve tasks in the project the chosen library was Point Cloud Library (PCL) [1]. PCL provides state of the art algorithms for 3D perception. With the development of PCL, point cloud processing and segmentation have gained popularity in technical disciplines - to mention a few that are of relevance; robotics and autonomous navigation.

N. Deshpande
Technische Universität München
E-mail: deshpand@in.tum.de

S. Abdullah
Technische Universität München
E-mail: saad.abdullah@tum.de

S. Radiah Rivu
Technische Universität München
E-mail: radiah.rivu@tum.de

This report aims to successfully partition homologous cloud clusters using PCL. Segmentation is the initiatory step in 3D image analysis because isolating similar clusters help to understand various aspects of the scene and helps in object recognition, classification, and feature extraction. The report is organized as follows - the second section deals with the literature review in the subject area of point cloud segmentation. The third section provides an overview and introduction to the PCL and describes about the features that it has to offer the users and software engineers. The penultimate and the ultimate section deal with the segmentation problems which is the crux of the report, and visualization of point clouds respectively. The report also offers an auxiliary section to advise users to setup an environment. Setting up an environment includes the installation of PCL on a computer, the directory structure of the project and instructions to run the binaries. This report offers a conclusive statement which describes in brief about possible enhancements, changes and challenges faced whilst completing this project.

2 Related Research

2.1 3D Point Cloud Segmentation: A survey

This study [2] aims to carry out a survey on all the techniques available for 3D segmentation. The main focus of this paper was to highlight the challenges faced in 3D point cloud segmentation. Segmenting a point cloud is not a trivial task because of the noise associated with it along with the fact that the data provided are almost always sparse and unorganized. Moreover the segmentation becomes demanding because of uneven sampling density and lack of statistical distribution pattern in the data. Though there are 3D sensors available worldwide it has its limitations and this causes the foreground to be highly entangled with the background. The paper also reviews the methods. The methods mainly discussed in this literature are edge based methods, region based methods, attributes based methods, model based models and graph based models. The edge based methods aim to detect the edges or boundaries of regions in the point cloud data in order to obtain segmented regions. The approach is similar to that of a 2D image, where edges are detected at locations which have a rapid change in intensity. Region based segmentation methods employ the neighborhood information to combine points. The criteria to merge points are limited by a threshold value, or by combining points that have same or similar properties to find isolated regions. Unlike, edge based methods, some region based methods are robust to noise. However, they are susceptible to over-segmentation or under-segmentation. The attributes based methods are based on clustering attributes of point cloud data. In this method, the attributes of the point clouds are computed, and they are clustered based on these computed attributes. However, these approaches rely heavily on the quality of computed attributes. Hence, it is essential to precisely compute these attributes to produce quality segmentation results. On the brighter side, these

methods offer flexibility in terms of incorporating different cues into process of segmentation. The model based methods on the other hand use the geometric primitive shapes for merging and grouping points. A pre-computed mathematical representation is stored. The points which have a similar mathematical representation as the pre-computed ones are grouped together. State-of-the-art robust models such as RANSAC (RANdom SAmple Consensus) are used to detect geometric primitives. Yet another class of point cloud segmentation is the graph based segmentation. This method conceptualizes the point clouds in terms of a graph - each point in the cloud data is viewed as a vertex, and pairs of neighboring points are viewed as edges. Graph based methods introduce a regularization function to enforce smooth segmentation. The method is modeled in a way that the foreground information is loosely connected to the background information. The segmentation is achieved with a min-cut method. A popular application of graph based methods is the interactive division. However, prior knowledge is required to specify the object of interest. The report mainly aims at region based segmentation owing to its ease of implementation, variants to handle both monochrome and color point clouds, and faster running times.

2.2 Object Recognition in 3D Point Cloud of Urban Street Scene

This paper [3] aims at creation of a novel framework to segment and recognize objects from point clouds. The point clouds are obtained from a high-definition LiDar (Light Detection And Ranging) laser scanner. The method proposed in this study first segments ground point cloud data. The segmentation of ground is important as it connects other objects. The segmentation used in this research is the region growing method. As mentioned earlier, this method is simple to implement and faster running times. To avoid over-segmentation problem, connectivity based methods are used. The main use of region growing method is in creation of facades. Once the point cloud of the scene is obtained, they are segmented and bundled. The clustering is achieved by applying a geometrical distance threshold. Hence creating voxels which provide a good perspective of the 3D cloud data. The aim of segmentation in this case is to ease the task of object recognition. The task of point cloud segmentation reduces the scene complexity, and thereby reducing the need of manual labeled data for training the classifier.

2.3 Segmentation Of Point Clouds Using Smoothness Constraint

The segmentation method presented in this study [4] is based on smoothness constraint. The smoothness constraint (over space) refers to an assumption that intensity values of pixels will change slowly in any given direction, and also the value of any given pixel will likely be close to its neighboring pixels. The method presented in this paper segments the point clouds by finding smoothly

connected regions. To cluster the points in the cloud, local surface normals and connectivity of points are used. The local surface normals are enforced by using the K-Nearest neighbors, and connectivity of points is eenforced by using the distance neighbors. The presented methodology is similar to the approach discussed by Nguyen et al [2]. The method of segmentation used in [4] describes in brief about region growing segmentation, its trade-offs and its advantages. The region growing method can render the point cloud over-segmented. However, the method requires few parameters to implement, and the inclusion of a residual threshold controls the over-segmentation of the point cloud at the cost of under-segmentation.

3 Point Cloud Library

Point cloud library (PCL) is a large scale open project for the processing of 2D or 3D images [1]. As briefly mentioned in the introduction, PCL contains state of the art algorithms that provide feature estimation, model fitting, segmentation, visualization etc. PCL is a cross platform that provides small code libraries so that compilation can be done separately. This modularity provides reduced computational or size constraints on platforms.From the libraries available in PCL we have focused on segmentation and visualization library.

The segmentation library was our prime focus because it provides algorithms to segment a point cloud data into distinct clusters. The use of visualization library was to prototype and visualize the results of segmentation algorithm operating on 3D point cloud data.

Though pcl mainly uses point cloud data it also supports various other forms of data and in this paper we have worked with both point cloud data (pcd) and polygon file format (ply).

3.1 Point Cloud Data

Point cloud Data (pcd) is a cluster of points in any given coordinate axis. Since the existing file formats do not support PCL extensions, point cloud data has been used to overcome the challenges. The use of point cloud data is emerging as a popular file format because of the utilization achieved using PCL.

3.2 Polygon File Format

Polygon File Format (ply), developed at Stanford University has been designed to store 3D data. The ply file stores graphical objects as a collection of polygons.

4 Segmentation

Segmentation is a process to partition an image into regions of related content. Each region is homogeneous corresponding to individual surfaces and objects. In other words, segmentation is a method to organize the image content into semantically related groups [5]. According to Sithole et al [11], segmentation can be mathematically written as $\Theta P = \{\theta p \mid \forall p \in P\}$. Here ΘP denotes the segmentation operation on point clouds P and θp is the assignment of labels for a point p in the point cloud P. The output of the operation are segments s, such that S = { s | s ⊂ P } [11] [12].

Advanced systems that are based on computer vision methods rely highly on real time processing of scenes. The scenes may be 2-dimensional or 3-dimensional in nature. Systems such as autonomous driving and mobile robot navigation require automatic processing of point clouds. One such way of achieving automatic processing is by employing segmentation. The method of segmentation solves a range of ambiguities. One such ambiguity being segmentation of parts of the scene which are not required for achieving the navigation and maneuvering tasks. Also, point clouds and their segmentation can complement images when they are ambiguous.

This report focuses mainly on region growing segmentation for reasons mentioned in section 2. Region based methods mainly rely on the smoothness constraint of the neighboring points in the cloud. The common process is to compare one point with its neighbors in the cloud. If a certain similarity criterion is satisfied, the point is marked and assigned one particular cluster of points. The region based segmentation can be categorized into two categories - seeded and unseeded region growing algorithms. The former is a bottom-up approach, and the latter is a top-down approach. The seeded approach involves two essential steps. The first being choosing of seed points. The seed points are chosen based on the curvature values. The seed points are chosen such that they have minimum curvature value. The reason for this is that the points with minimum curvature values are located in the flat area. Next, the points are sorted according to their curvature values. The second step involves growing these seed points based on distance in terms of the smoothness constraint and planarity of surfaces. It is noteworthy to know that the growth of regions from the flattest area helps to reduce the number of segments [1]. A step-by-step short description algorithm is as follows:

1. add the chosen points to the seeds set
2. for all seed points, find the neighboring points
   - for every neighboring point in the set check the angle between the normal of the point and the considered seed point. If the angle is less than the threshold value θ_{th}. If true, then add the current neighboring point into the region
   - for every neighboring point in the set, check the curvature value. If the curvature value is less than the threshold value c_{th}. If true, then add this neighboring point into the seed point set
   - finally remove the considered seed point from the seed point set

Algorithm 1 Region Growing Algorithm

inputs:
point cloud = {P}
point normals = {N}
points curvatures = {c}
finding neighbors function $\Omega(.)$
curvature threshold c_{th}
angle threshold θ_{th}
---
intialization steps:
region list R $\leftarrow \emptyset$
point list $\leftarrow \{1, ..., |P|\}$
---
while $\{A\} \neq \emptyset$ **do**
    existing region - $\{R_c\} \leftarrow \emptyset$
    existing seed - $\{S_c\} \leftarrow \emptyset$
    get points with minimum curvature in $\{A\} \rightarrow r$
    $\{S_c\} \leftarrow \{S_c\} \cup P_{min}$
    $\{R_c\} \leftarrow \{R_c\} \cup P_{min}$
    $\{A\} \leftarrow \{A\} \setminus P_{min}$
    for `i = 0 to size (`$\{S_c\}$`)` **do**
        `find the nearest neighbors of the present seed point` $\{B_c\} \leftarrow \Omega(S_c\{i\})$
        for `j = 0 to size (`$\{B_c\}$`)` **do**
            present considered neighbour point $P_j \leftarrow B_c\{j\}$
            if $\{A\}$ contains P_j and $cos^{-1}(|(N\{S_c\{i\}\}, N\{S_c\{j\}\})|) < \theta_{th}$ **then**
                $\{R_c\} \leftarrow \{R_c\} \cup P_j$
                $\{A\} \leftarrow \{A\} \setminus P_j$
                if $c\{P_j\} < c_{th}$ **then** then
                $\{S_c\} \leftarrow \{S_c\} \cup P_j$
                end if
            end if
        end for
    end for
    include the region into the global segment list $\{R\} \leftarrow \{R\} \cup \{R_c\}$
end while
return $\{R\}$

The implementation tasks in this project mainly involved seeded region growing segmentation. The algorithm can be summarized as described [1] as follows:

Yet another method that is presented in this report is the color-based region growing segmentation. Like region growing segmentation, this method also uses the seeded (bottom-up) approach. There exists two differences in this approach - (1) usage of colors instead of normals (2) the merging algorithm controls the problem of over-segmentation and under-segmentation [1]. The merging of individual clusters after the segmentation is achieved through close colors. A colorimetrical threshold is chosen, and two nearby clusters are merged if and only if the average color of both the clusters is smaller than the colorimetrical threshold. In the second step of merging, a check is performed on the number of points a particular cluster may contain. If the number of points are less than a user-defined value, then the current cluster is merged with the nearby cluster [6].

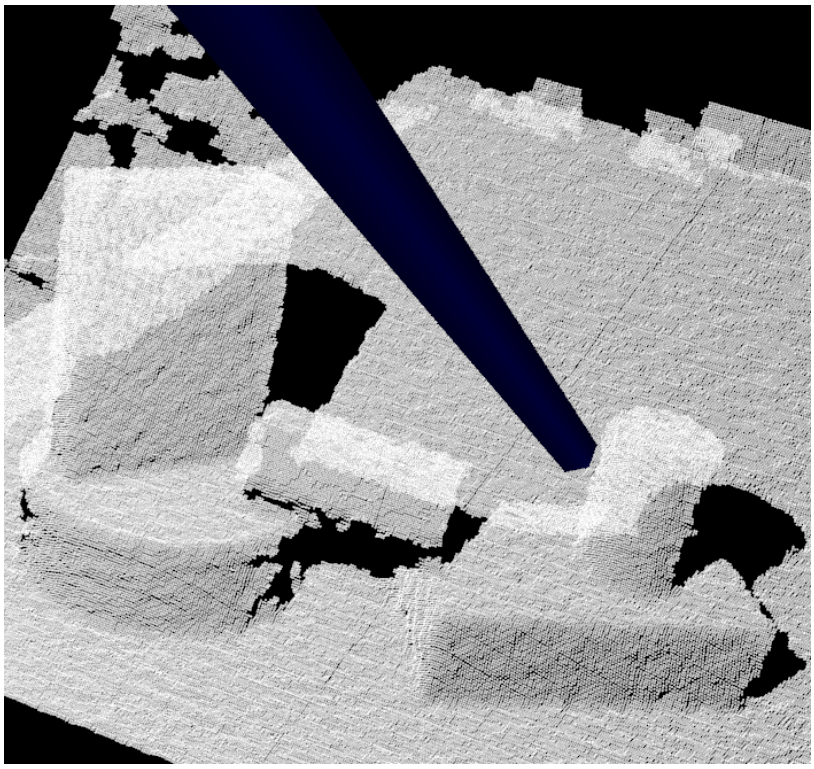

Fig. 1 Image describing the whole point cloud of the scene, before applying any kind of segmentation.

4.1 Results of Region Growing Segmentation

The intended dataset was supposed to be a dense reconstructed point cloud using work by Pizzoli et al [7]. However, the project task results were evaluated using datasets from The Object Segmentation Database [8]. A plain version of the point cloud can be seen in figure 1. The results of the implementation are presented in figure 1. The segmented point cloud in figure 2 describes a scene where boxes and cans are stacked on top of each other. The whole scene is set up on a table. Also, the result of color-based region growing segmentation is displayed in figure 3. The algorithm was tested on the same scene as the previous case. It is noteworthy to observe how clusters are assigned the same color information. This is also due to the fact that a colorimetrical threshold is used instead of normals.

5 Recognition

Although this was not the objective of the project task. The report will address the problem of recognition. This was done in order to achieve completeness of

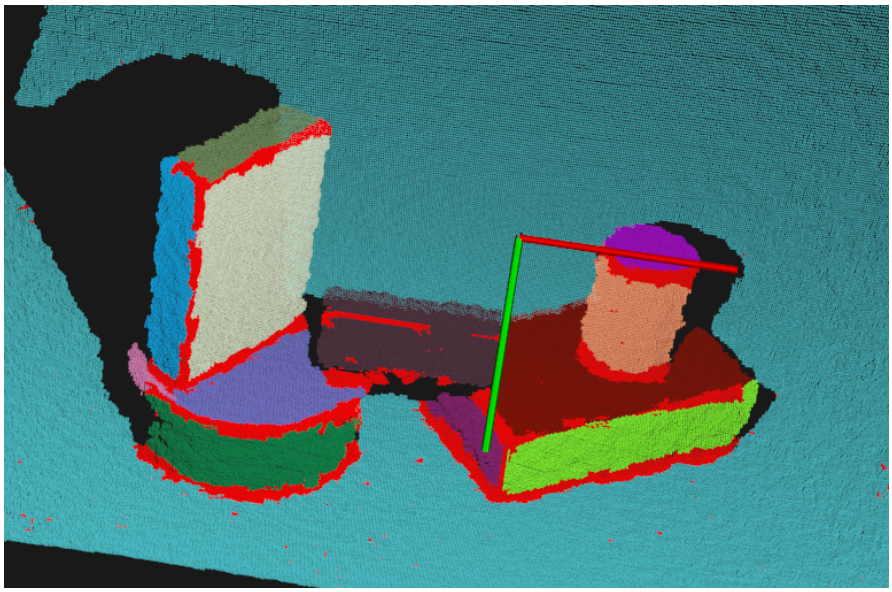

Fig. 2 Image describing the result of region growing segmentation. Observe how the homogeneous regions are assigned the same color.

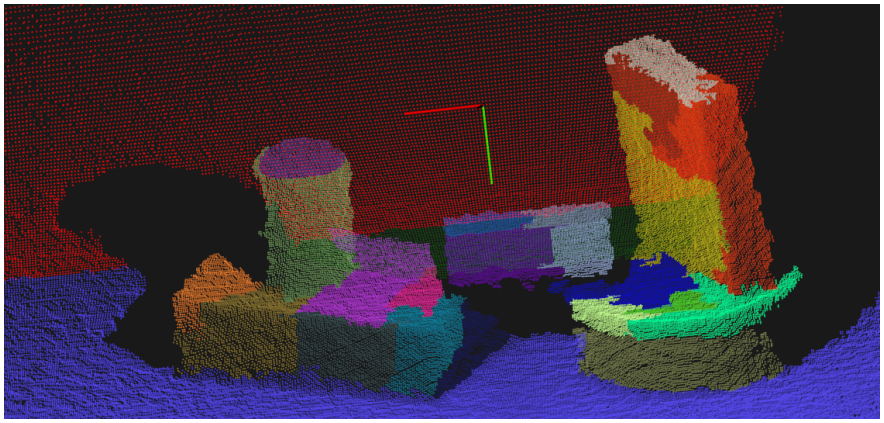

Fig. 3 Image describing the result of color-based region growing segmentation. Observe how the clusters or patches are assigned the same color.

the task. The following task after segmentation will be that of recognition. As mentioned earlier, segmentation reduces the complexity for recognition tasks. A number of works and studies revolving around recognition exist in the scientific community. However, a novel method of group correspondences was chosen - the reason being it reduces the search space to Hough (3D) space compared to a traditional 6D space which is complex and time consuming [1]. The following section will contain methods, advantages and implementation results alone. The detailed explanation of the group correspondences method

is outside the theme of this project task, and hence will not be explained. In the work proposed by Tombari et al [9], a novel Hough voting approach for the identification of free-form shapes in a 3D space, to be utilized for object recognition as a part of 3D scenes with a noteworthy level of impediment and clutter. The proposed strategy depends on coordinating 3D elements to gather evidence of the nearness of the items being looked for in a 3D Hough space. The proposition is validated by exhibiting a quantitative exploratory examination with best in class strategies too as by indicating how the technique enables 3D object recognition from real-time stereo information. The Hough Transform (HT) is a well known computer vision strategy to detect lines in 2D images. Progressive alterations permitted the HT to distinguish basic shapes, for example, circles and ovals. In general, the key thought is to use a voting scheme of the image elements (such as edges and corners) in the parameter space of the shape to be recognized. Votes are aggregated in a accumulator whose dimensionality meets the number of obscure parameters of the considered shape class. Therefore, albeit general in principle, this system can not be applied to shapes described by an excessive number of parameters, since this would bring about a sparse, high-dimensional accumulator driving to poor execution and high memory prerequisites. By method for a coordinating edge, peaks in the collector highlight the nearness of a specific shape in the picture. The Generalized Hough Transform (GHT) augments the HT to discovery of items with subjective shapes, with each feature voting in favor of a particular position, orientation and scale component of the shape being looked for. To lessen the intricacy, the slope heading is generally registered at every component position to rapidly file the gatherer.

The HT scheme extended to 3D case is simple and permits identification of planes inside the point cloud.. As in 2D case, the 3D HT has been altered to deal with extra 3D analytical shapes characterized by fewer number of parameters, such as spheres and cubes. A more broad class of items [9], i.e. polyhedra, is considered, with a Hough Voting technique in two separate 3D spaces accounting for rotation and translation permitting to identify objects based on correspondences between vertex points built up by coordinating straight edges. However, this [9] strategy can't give an unique pose to each correspondences and, all the more essentially, can't be connected to non specific free-form objects.

The Hough voting scheme is intuitive and uses the features of the local reference frame [9]. To start with, interest point are separated from both the model and the scene. it is achieved by either picking them arbitrarily or by means of a feature detection algorithm. At that point, every feature point is enhanced with a bit of information that represents a depiction of its neighborhood, i.e. a 3D feature descriptor. Normally, distinguishing and depicting features of the model(s) can be performed offline. Given an arrangement of depicted features both in the model and in the scene, an arrangement of feature correspondences can be resolved by thresholding. Because of the presence of "nuisance factor" [9] such as noise, clutter and occlusions or impediments of the item being looked for, generally this set incorporates likewise off-base cor-

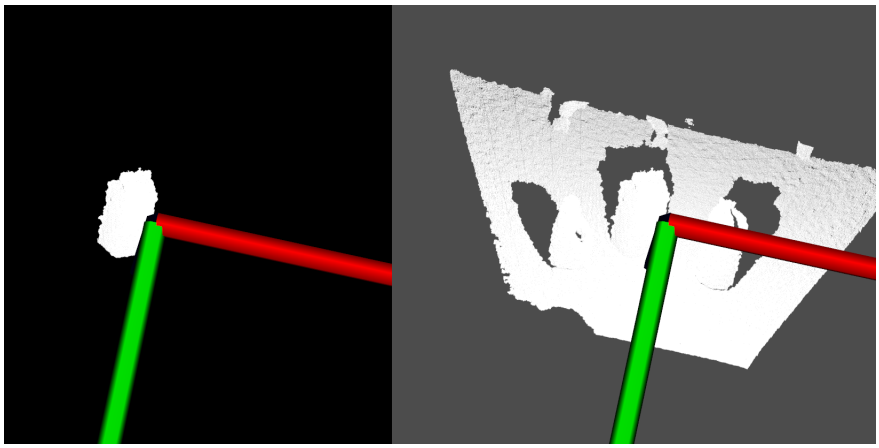

Fig. 4 Split screen view of the model and the scene. The left image which contains only a milk carton is the model that is sought. The right image contains the scene in which the milk carton has to be searched

respondences or wring correspondences. To solve this problem Hough Voting Scheme is used. It aims at gathering evidence for the presence of an desired object by computed correspondences. If enough features vote for the presence of an object in a given pose, then, the object is detected.

5.1 Results of Group Correspondences

The datasets were obtained from Technische Universität Wien [8] - like used previously in the case of region growing segmentation. The plain point cloud can be seen in figure 4. The image screenshots obtained from the recognition tool points the object present in both the model and the scene. The result is displayed in figure 5.

6 Visualization

The visualization library provided by PCL serves the purpose of prototyping and visualizing the results of algorithms applied on the 3D point cloud data. PCL provides a rich visualization library which offers methods for rendering and setting visual properties, drawing basic 3D shapes, geometry and color handler and makes use of Visualization Toolkit.

The visualization library provides various functions to visualize the segmented data. Though there is the option to use CloudViewer we have opted to use the PCLVisualizer. Though PCLVisualizer is more complex to use, it is more powerful and and has the features to display normals, shapes and multiple viewports. In our program we use segmentation on 3D point cloud data and visualize the result using RGB color visualization. We have used

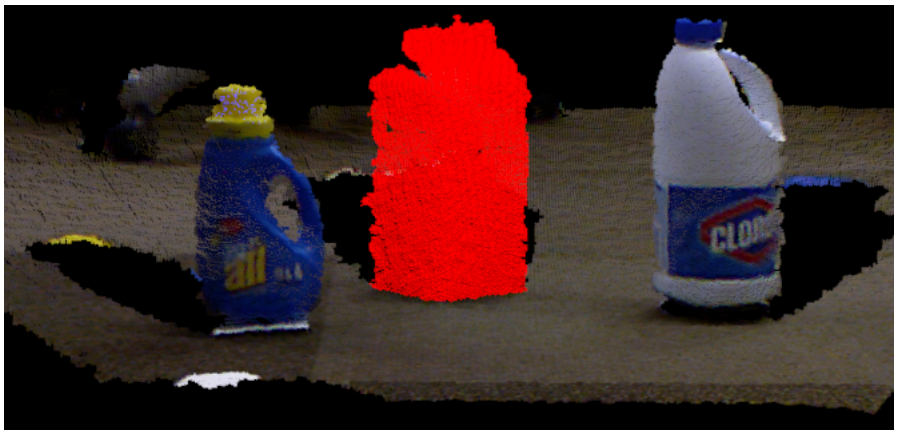

Fig. 5 Image describing the result of group correspondences for recognition. Observe how the object is highlighted in red to indicate the presence of the object.

PointXYZRGB which contains color data. PCLVisualizer also provides facilities to display point clouds with the color data stored within. The challenges faced in this section of the code was the poor documentation available for implementing ply reader in the PCLVisualizer. The documentation or manual lacked examples of the function usage which made it a challenging task to understand and implement in the system. The lack of resources and poor documentation made it difficult for to understand and evaluate the underlying implementation of the libraries available. We have used both CloudViewer and PCLVisualizer to analyze which gives a more accurate result on the segmented 3D data and the results are shown in figures 6 and 7 respectively.

7 Conclusion

The proposed methods and algorithms in this reports provide satisfactory results. However, if the scene becomes too complex, it is best to use machine learning segmentation techniques or the graph-based segmentation techniques as pointed out in [2] by Nguyen et al. The region growing segmentation methods are used in a lot of novel based segmentation methods. However, a supplement method as used in [3] to enhance this algorithm is the need of the hour.

7.1 Challenges faced

The project tasks provided an opportunity to learn rigorous mathematical techniques. However, a few trivial problems existed in the Point Cloud Library (PCL) - an essential tool in completion of the task. The point cloud data files were not read and loaded into the environment all the time. The

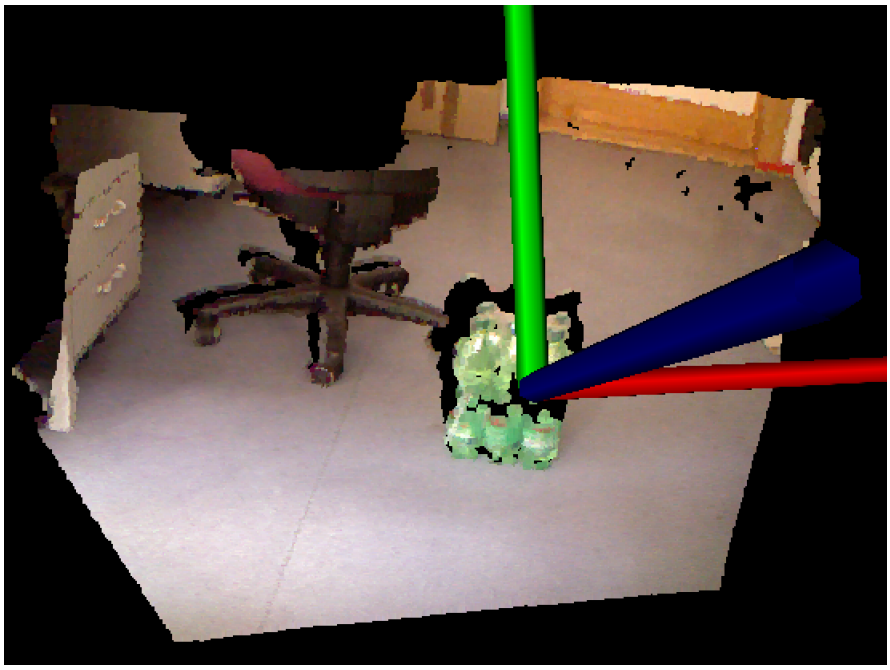

Fig. 6 Point cloud visualization using the PCLVisualizer in PCL

PCL input - output module turned out be faulty and little help was available. This served as a motivation to implementation a custom point cloud data parser using Boost C++ library. However, due to limitations of time and resources, the task could not achieved. This can serve as a red flag to readers to muse upon before proceeding with PCL. At later stages of the tasks, a workaround was employed to solve the problem of input - output module in the PCL. This involved simply placing the point cloud data in the build directory of a particular algorithm that was implemented. Although, not a convincing workaround, solving the problem of faulty input-output module will remain an open issue. The project also motivated in the direction of using a better visualization techniques available in the PCL itself. As mentioned in section 6, the usage of two different classes of visualization produced remarkable difference in results. However, the type of visualization classes used depends on the class of application.

8 Auxiliaries

The present section offers an explanation of regarding the libraries and how to setup an environment suitable to run the project on a computer. The operating system platform used was Linux distribution of Ubuntu 14.04 (x64).

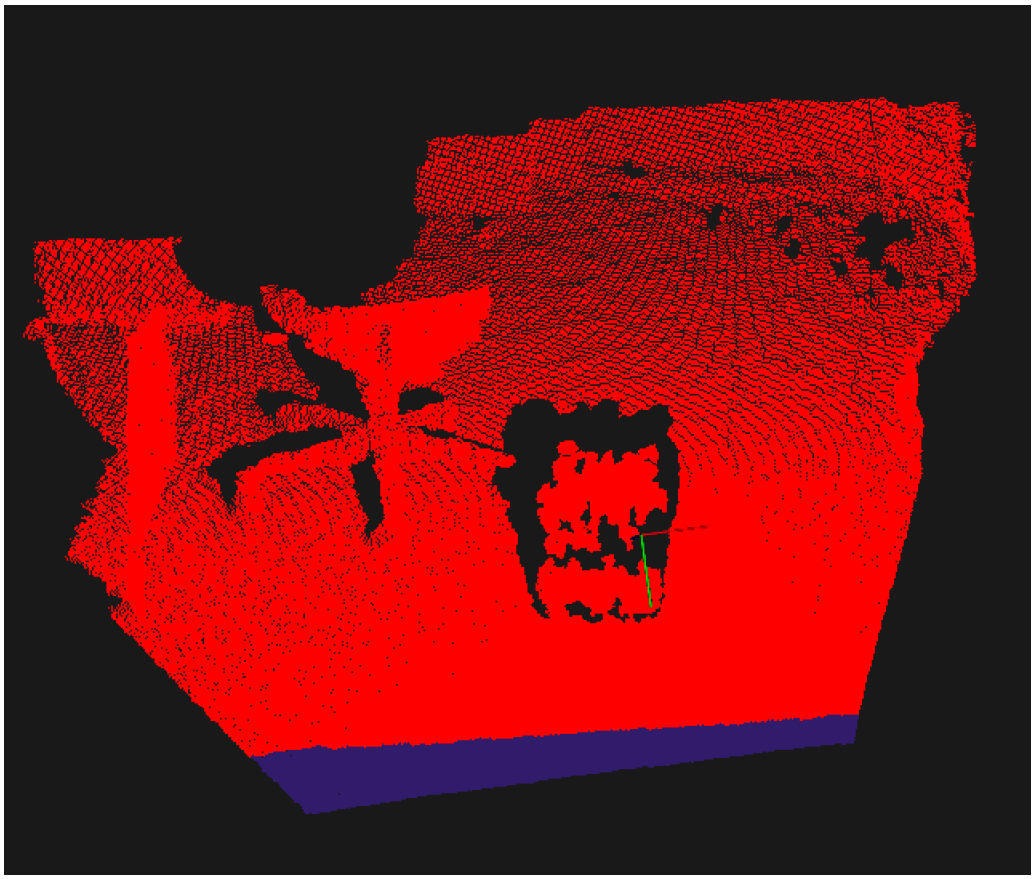

Fig. 7 Point cloud visualization using the CloudViewer in PCL

8.1 Point Cloud Library

The version of Point Cloud Library that was used to achieve the project tasks was Point Cloud Library Version 1.7. A prerequisite package of cmake is required to compile PCL. Use a packaging software to install cmake. To install PCL, obtain the PCL v. 1.7 source code from this location [10]. Once the source code is obtained - follow the steps to install the library on a Ubuntu 14.04 computer.

1. Uncompress the compressed file by issuing the command - "tar xvfj pcl-pcl-1.7.2.tar.gz"
2. Once uncompressed, access the directory, and create a directory, name it build, and access the build directory successively - "cd pcl-pcl-1.7.2 && mkdir build && cd build"
3. When inside the build directory, issue the following command - "cmake .."
4. Now, compile the code by issuing the following command - "make -j2"

5. After the previous step has been concluded, install the library - "make -j2 install". It might be necessary that you remain a superuser or a root user to run the command above

8.2 Organization of the coding environment

The coding environment has been organized into a directory structure as follows. The osr directory has 6 sub-directories, namely (1) kb - contains essential literature pertaining the project (2) Presentation - contains the final presentation slides (3) PRJ_VIDS - contains demo of the applications or project tasks (4) report - contains this report (5) src - contains the source code. It is advisable not to mis-skew the arrangement of the directory, as it might render the environment useless. The most essential sub-directory would be that of src directory. The src directory further contains two more dub-directories, namely (1) 3dor - contains code segment for group correspondences (2) modules - contains code and class implementations for region growing algorithm and visualization routines.

8.3 Modifying the CMakeLists.txt file and running the binaries

Both the directories in **src** directory have **build** directories containing the point cloud data and binaries. All of the generated binaries are run by accepting point cloud data on-the-fly from the command line terminal. To generate a binary: (1) edit the CMakeLists.txt file, and (2) access the **build** directory to run "cmake .." and "make" commands sequentially. The CMakeLists.txt file has to be modified in order to implement, produce the required binaries and finally to run the tool. In order to run the "test_visualisation.cpp" file source in modules directory, a set of instructions need to be followed. First, modify the CMakeLists.txt file as follows:

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(test_visualisation)
find_package(PCL 1.2 REQUIRED)
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})
add_executable (test_visualisation visualisation.cpp
                test_visualisation.cpp)
target_link_libraries (test_visualisation ${PCL_LIBRARIES})
```

Next, access the **build** directory, and run the following command sequentially - "cmake .."; once that command has concluded without errors, run the "make" command. You will now see "test_visualisation" binary generated in

the **build** directory. Provide necessary execution rights to the binary, and run the binrary by issuing the command "./test_visualisation rgb2.pcd". Likewise, a series of different versions of CMakeLists.txt file will be provided in this section. The procedure to run the binary remains the same. The following section of CMakeLists.txt file will cater to the need of running the plain monochrome region growing segmentation algorithm.

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(reg_grow_segmentation)
find_package(PCL 1.2 REQUIRED)
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})
add_executable (reg_grow_segmentation reg_grow_segmentation.cpp
                                test_rgseg_monochrome.cpp)
target_link_libraries (reg_grow_segmentation ${PCL_LIBRARIES})
```

In the **build** directory, run the binary named reg_grow_segmentation by providing the necessary point cloud data. Best results were achieved with test44.pcd. The following CMakeLists.txt is used to run the color-based region growing segmentation algorithm. The difference between region growing segmentation and color-based region growing segmentation methods can be observed.

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(reg_grow_segmentation_rgb)
find_package(PCL 1.2 REQUIRED)
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})
add_executable (reg_grow_segmentation_rgb reg_grow_segmentation.cpp
                                test_rgseg_rgb.cpp)
target_link_libraries (reg_grow_segmentation_rgb ${PCL_LIBRARIES})
```

Run the binary named reg_grow_segmentation_rgb by supplying it with a point cloud data - it is worthwhile providing the test44.pcd to observe the difference between the region growing segmentation method and the current method. Accessing the parent directory **src** will provide access to **3dor** directory which contains code segment to implement group correspondences algorithm. This directory like others are arranged in a similar way. The directory contains a **build** sub-directory and code segments written in C++. The following is the version of CMakeLists.txt file to run the binary of the algorithm.

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
```

```
project(cg_recognition)

find_package(PCL 1.5 REQUIRED)

include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})

add_executable (cg_recognition cg_recognition.cpp)
target_link_libraries (cg_recognition ${PCL_LIBRARIES})
```

Once the binary named "cg_recognition" has been generated, run it by providing two point clouds namely "milk.pcd" and "milk_cartoon_all_small_clorox.pcd". The former is the model and the latter is the scene point cloud models respectively. Note that changing the order of the point clouds will result in adverse recognition output. The following command will produce the required result - "./cg_recognition milk.pcd milk_cartoon_all_small_clorox.pcd"

8.4 Running the video demonstration

In the **src** directory, there is an efficient way to run all the video demonstration files. The bash script file "video_list.sh" file has all a list of videos containing implementation of algorithm in this project. Provide the necessary rights to the file, and run the shell file as follows in the terminal - "./video_list.sh". Note that the latest version of VLC Multimedia player is required to run this video demonstration.

9 Acknowledgement

The project group involved in 3D Point Cloud Segmentation is grateful to the entire faculty of Robotics and Embedded Systems involved in this project. We would like to specially thank Mr. Gao Xiang, Mr. Biao Hu, Mr. Xiebing Wang and Dr. Kai Huang for their valuable advice in the subject area, and mainly for their feedback for our project.

References

1. Point Cloud Library (PCL). In: Point Cloud Library (PCL). http://pointclouds.org/. Accessed 19 Jun 2016
2. A. Nguyen and B. Le, 3D Point Cloud Segmentation: A survey, 2013 6th IEEE Conference on Robotics Automation and Mechatronics (RAM), pp. 225 - 230, Philippines (12 - 15 Nov. 2013)
3. P. Babahajiani, L. Fan and M. Gabbouj, Object Recognition in 3D Point Cloud of Urban Street Scene, ACCV 2014 Workshops, Part I, pp. 177 - 190, Singapore (1 - 5 Nov. 2014)
4. T. Rabbani, F. A. van den Heuvel and G. Vosselman, Segmentation Of Point Clouds Using Smoothness Constraint, ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'- Volume XXXVI, pp. 248 - 253, Germany (25 - 27 Sept. 2006)

5. Prof. Dr. G. D. Hager, Computer Vision - Segmentation, CS-600.461 Computer Vision - Lecture Notes, Johns Hopkins University, USA (12 May. 2002)
6. Q. Zhan, Y. Liang and Y. Xiao, COLOR-BASED SEGMENTATION OF POINT CLOUDS, IAPRS, Vol. XXXVIII, Part 3/W8, France (1 - 2 Sept. 2009)
7. M. Pizzoli, C. Forster and D. Scaramuzza, REMODE: Probabilistic, Monocular Dense, Reconstruction in Real Time, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong (2014)
8. Object Segmentation Database, http://www.acin.tuwien.ac.at/forschung/v4r/software-tools/osd/, Technical University of Vienna, Austria, Accessed 06 Jun 2016
9. F. Tombari and L. Di Stefano, Object recognition in 3D scenes with occlusions and clutter by Hough voting, pp. 349 - 355, Fourth Pacific-Rim Symposium on Image and Video Technology (2010)
10. Github Point Cloud Repository, https://github.com/PointCloudLibrary/pcl/tree/pcl-1.7.2, Accessed 12 May. 2016
11. G. Sithole, Segmentation and classification of airborne laser scanning data, PhD Thesis, pp. 327, Technical University of Delft, Netherlands (2005)
12. P. P. Sapkota, Segmentation of Coloured Point Cloud Data, Master Thesis, pp. 79, INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION, Netherlands (2008)