3D Point Cloud Filtering
Final Project 3

Shao Hsuan Hung (1723219)
s.hung@student.tue.nl

January 22, 2023

Bag files of the filtered point cloud is upload to the address: https://drive.google.com/file/d/1LwV-nSG1lJHePZdD9HDx3EgI6G90ZfQx/view?usp=sharing
Please log in your tue account to get the access of the OneDrive.
If you want to run the filter code, please make a topic in name: "/cloud_pcd", the filter is subscribe to this topic as shown in the figure 3.
I attached the whole directory of the test_ws, the self-implemented code is at .\code \test_ws \src \test_package\src

1 Point Cloud importing

(1.) The video that used to generate this point cloud was recorded from ZED Mini (a stereo camera). Recall what you learned in the lecture, could you briefly describe the process of point cloud generation using a stereo camera?

The process that the point cloud generation is divided into 2 steps. The first step is to get 3D data from stereo camera sensor. The second step is to combine the 3D data with different place using SLAM technique. The detailed steps are describe below:
(1.) After camera calibration, start to using the stereo camera to get image with depth information. The depth can be calculated based on parameters of the stereo camera. Map a RGB-D data type image to a 3D point cloud data type by the back projection.
(2.) Initialized the origin of coordinate system by the first scan of the camera.
(3.) Move the stereo camera around with slow and little movement, obtain a new RGB-D image. Then convert the image in to a 3D point cloud.
(4.) Observe the landmarks, find the position that had been scan in the current scanning image.
(5). Compute the changes in position and orientation of the stereo camera based on the change of the landmarks. ([r t] matrix.)
(6.) Estimate the current camera position and orientation. If loop closure detected, correct the current and previous estimated positions and orientations in order to reduce the drift.
(7.) Update the 3D model with the sample data
(8.) Repeat the step (3) to step (7) until getting the full point cloud.

(2.) As you can see in RVIZ, this is a very noisy point cloud. Could you explain the reason why there are so many noises?

The noises are arouse by the restriction of the CMOS sensor in the stereo camera, fast changes of position of camera, and the change of illumination in the environment.

First, the small noise around walls may be the result of limitation of the sensor in the devices. Since the digital sensor cannot perfectly represent the continuous world. So there might be small random noise.

Second, fast changes of the position of the camera sensor can also be the reason of the noise. When the position and orientation of the camera change fast, even though the loop closure can fix the drift problem, but the fast movement make the estimated position and orientation inaccurate, and accumulated to become big noise in the position of points.

Third, as shown in the figure, the most significant noise happen at windows part. The most possible for the noise is that the changes of illumination and the landmarks of the outside.When computing the depth information of windows part, since the windows are transparent, it may lead to a fault depth because the algorithm detect the outside landmark, which is greater than its actual depth. Also, the light background of the window may also interfere the precision of the image. When using the same aperture in a place with

variant brightness, it is very possible that the sensor get inaccurate noisy information.

2 Design of the point cloud filtering node in ROS

(4.) The complete raw point cloud.

The complete raw point cloud are shown in the figure 1 and 2. To see the raw data more clear, I show the raw point cloud data from top view, and from 4 corners.

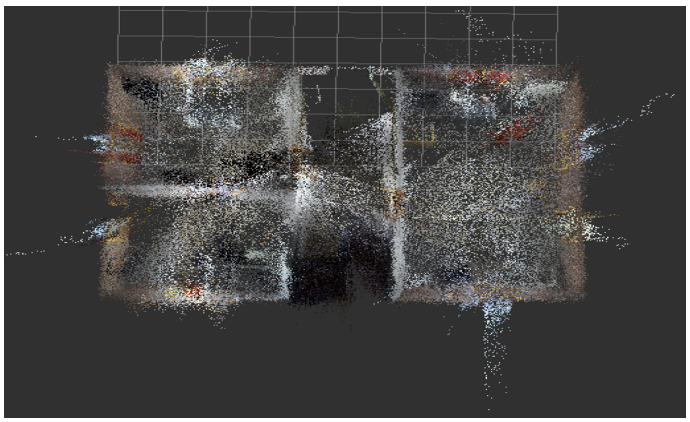
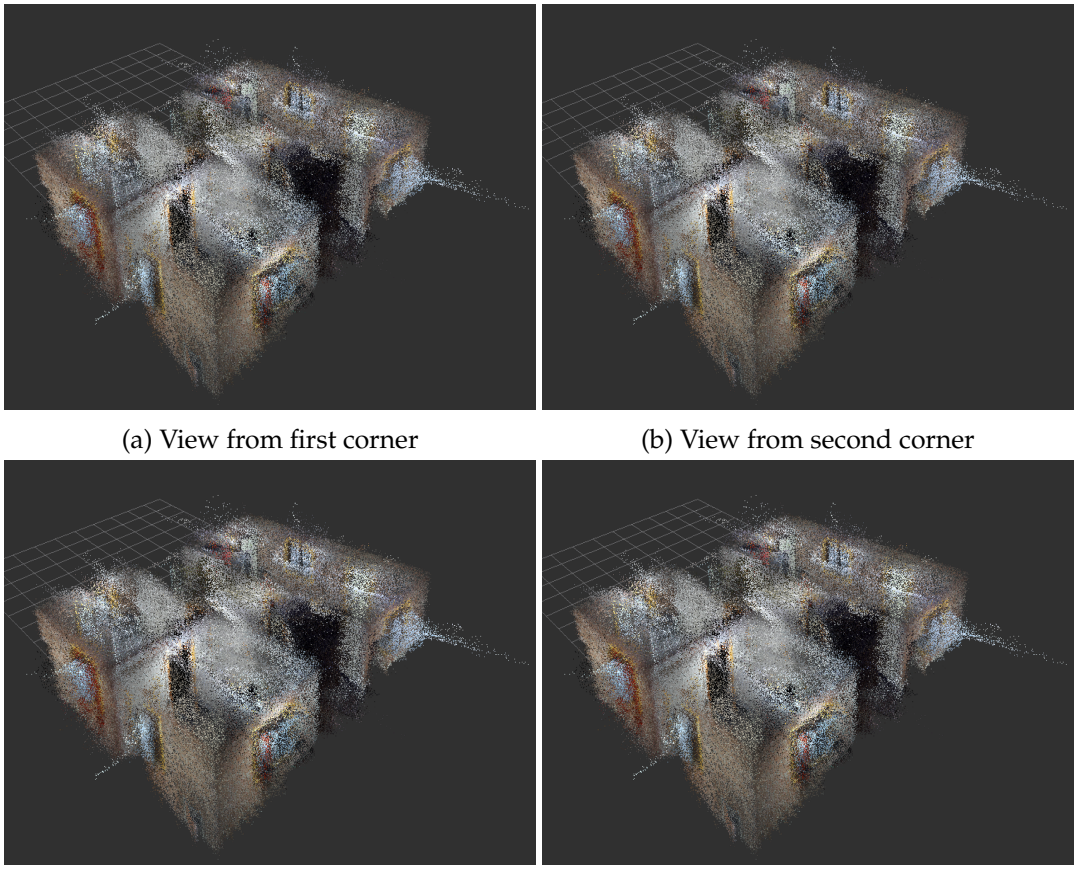


Figure 1: Top view of the raw point cloud



(a) View from first corner

(b) View from second corner

(c) View from third corner

(d) View from fourth corner

Figure 2: Raw point cloud viewed from 4 corners

(5.) Explain details of your filtering procedure. You may also draw a flow chart to support your text.

In this section, I use C++ programming language and the PCL library to design the filtering pipline.

The whole filtering process contains passthrough sampling, down-sampling, outliner-removing and smoothing. As shown in the following figure 3, the raw data is processed by 4 different filters. In the figure, each

Figure 3: Process of denoising

node is a filter (except to the /pcd_to_pointcloud, which is only sent out the point cloud message), and after filtering, each node would send out corresponding topic, then the following node subscribe to that topic. First, the node /pcd_to_pointcloud started to extract the last message in our point cloud bag file. It help us to only send the messages of the last frame in the bag file to reduce time consumption. The topic "/cloud_pcd" is published. then the following node (/Passthrough_Filter) is cascaded to process the point cloud message, and published the step by step filtered point cloud message to responding topics. The whole filtering mindset and correponding reasons are described below:

- First, in the "/Passthrough_Filter" node. Because the main goal of this project is to let the structure in this building become clear and less noisy, that means the vertical structure (wall, door, window) is important than the floor and ceiling. I use the pass-through filter to extract the region of interest. After filtering, the result is published to topic "/passthroughfilter".
- Secondly, in the "/Down_Sample" node. A voxelgrid filter [1] is firstly applied to down sample the whole point cloud in order to make the density of all points in the point cloud are similar which is helpful for the future filtering process. The result is published to topic "/downsample_filter".
- Third, in the "/Outline_Filter" node. A the statistical-outlier-removal filter is applied to remove outliers which are far away from the main object, the process would be helpful before apply the k-nearest neighbor algorithm (smooth filter), since too many outliner point cloud would affect the result of the algorithm. The result is published to topic "/outliner_filter".
- Fourth, in the /Smoother_Filter node. The nearest neighbor searches algorithm are the core operation when working with point cloud data. By setting the radius, we can find the neighbor point that is in the specific radius. By doing smoothing, the drifted points are eliminated. The result is published to the topic "/smoother_filter".

Besides the filtering, some parameter tuning is necessary to get the best result. As shown in following table 1, there are some parameters setting in the corresponding filter. After the filtering , the best result of the filtered point cloud is shown in the next page, figure 4 and 5. Comparing to the raw point cloud data (figure 1, 2), and filtered cloud data (figure 4, 5), the most obvious part is that the noise floating in the air is almost disappear from the top view. For the noise at windows, the noisy point cloud reduce a lot, but there are still many noise point on one window that view from the fourth corner.

Table 1: Parameters setting in each node

Node	**Parameters**	**function of parameter**
/Passthrough_Filter	low_bound: -1.1 high_bound: 0.95	Setting the z coordinate in world coordinate to accepted interval values to (low_bound, high_bound).
/Down_Sample	voxel_size :0.01	Create pcl::VoxelGrid filter with a leaf size of the given voxel_size (m).
/Outline_Filter	mean_k = 30 std_threshold = 1.0	Assume that the resulted distribution is Gaussian with a mean and a standard deviation, all points whose mean distances are outside an interval defined by the global distances mean and standard deviation can be considered as outliers and trimmed from the dataset.
/Smoother_Filter	search_radious = 0.1	A radius that each neighbors have to within a specified radius to remain in the PointCloud.

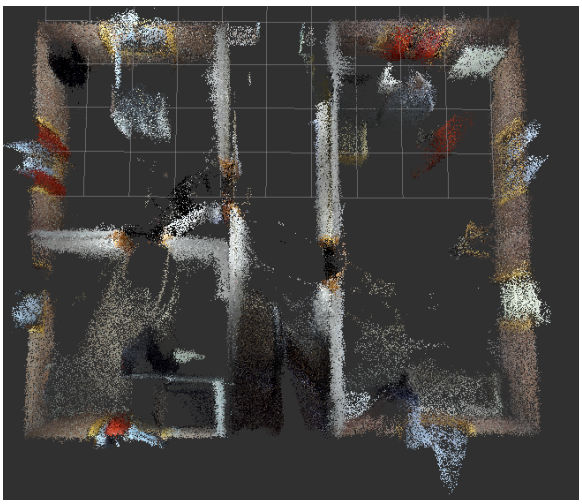
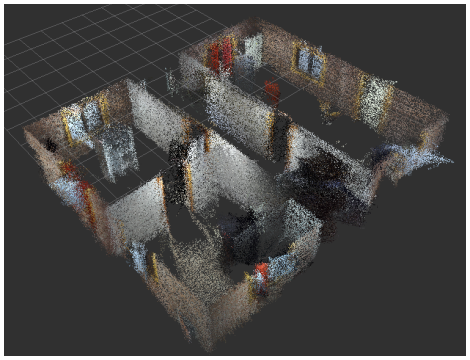
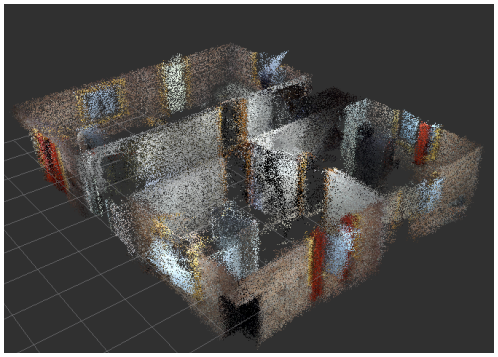


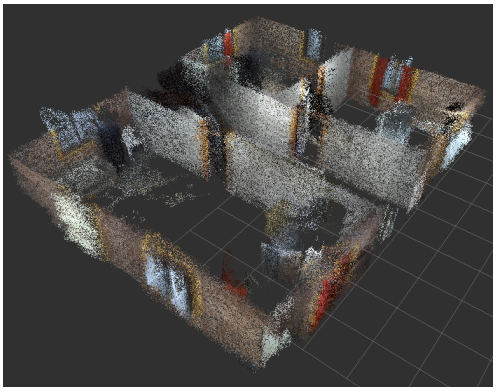
Figure 4: Top view of the filtered point cloud



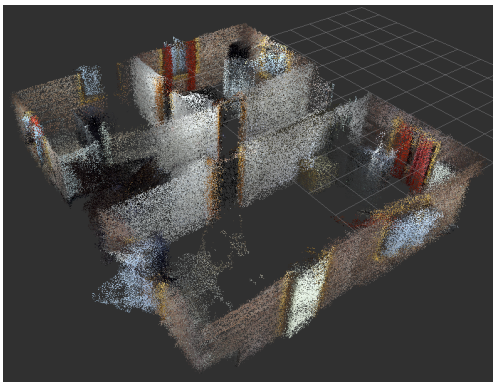
(a) View from first corner



(b) View from second corner



(c) View from third corner



(d) View from fourth corner

Figure 5: Filtered point cloud viewed from 4 corners

(6.) Do you have any idea on how to further improve your filtering node in terms of speed and performance? Explain your idea here.

The filtering performance can be improved by clustering, region growing, segmentation and interpolation.

- **Planar/ vertical detection**: we can use planar/vertical detection to find the ceiling, floor and wall automatically. It is much better than just using a pass through filter since the user has to tune a specific parameters of the pass-through filter for each case. A adaptive method would save a lot of time.
- **Region growing**: since many of the noise happen at windows part. we can use region growing with colors to help dealing with the glasses of windows. By region growing with colors, we can select points belongs to a window and then we can set their coordinates by the coordinates of the corresponding wall. By doing so, we can avoid the holes in windows which is appeared after we remove outliers of the point cloud.

- **Interpolation**: after filtering, since many outliner points are removed and condense. There would be a lot of 'hole' and 'gap' in the filtered point cloud. The interpolation can help to fill these part. By discretizing the cluster, we can generate points which the position and color are interpolated form their nearest neighbor, to make the 3D model looks consecutively.
- **Point density based algorithm**: Some research [2]uses the kernel density estimation technique to approximate the density of noisy point clouds. Then, it removes outlying points in low-density regions. To finally obtain a clean point cloud, it relies on the bilateral filter [3] to reduce the noise of the outlier-free point cloud. Furthermore, since the point density based algorithm in 3 dimension would have high computational complexity, one of the effective method is to apply PCA to transform the point cloud from 3 dimension into 2 dimension [4], and do the adaptive clustering filtering in the two dimensional space.

References

[1] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.

[2] Faisal Zaman, Ya-Ping Wong, and Boon-Yian Ng. "Density-based Denoising of Point Cloud". In: (Feb. 2016).

[3] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. "Bilateral Mesh Denoising". In: *ACM Transactions on Graphics* 22 (May 2003). DOI: 10.1145/1201775.882368.

[4] Yao Duan et al. "Low-complexity point cloud denoising for LiDAR by PCA-based dimension reduction". In: *Optics Communications* 482 (2021), p. 126567. ISSN: 0030-4018. DOI: https://doi.org/10.1016/j.optcom.2020.126567. URL: https://www.sciencedirect.com/science/article/pii/S0030401820309858.