Eindhoven University Of Technology

Computer Vision and 3D Data Processing

Fundamentals of Neural Networks

2064472 – R.Regalo

February 4, 2024

I. Point cloud Generation using stereo camera

The process of generating a point cloud using a stereo camera starts by capturing two images from slightly different viewpoints, similar to how the human eyes work. Then, for each pixel (or feature) in the right image, the algorithm tries to find the matching feature in the left image. Then, it computes the disparity for each pixel, which is the distance (in pixels) between the same feature in the two images. Then, the depth measurement is computed by

$$depth = \frac{f \cdot b}{d \cdot p}$$

where:

- f is the focal length
- b is the baseline (physical distance between the centers of the two sensors)
- p is the pixel width

This builds a depth map, associating a depth value for each pixel.

Then, the camera's projection matrix, which depends on the camera's intrinsic and extrinsic parameters, converts the RGBD image into a 3D PointCloud. As more scans of the same area are obtained, the new point clouds are aligned with the old ones (one option for this is the Iterative closest point algorithm). This process is computationally intensive and often will require filtering to remove noise and improve the quality of the point cloud.

II. Causes for noise in the point cloud

The noise in the point cloud generated by a stereo camera can be explained by some factors:

- Errors in feature matching between left and right images will lead to incorrect depth calculations.
- Smooth areas with little texture do not have enough information for accurate depth calculations.
- Reflections can lead to feature mismatch and, therefore, incorrect depth calculations.
- Objects that are either further away from the camera or too close to it may also get their depth estimation wrong.
- If the environment is not static, that is, if there are moving objects, the matching of subsequent scans will not align perfectly with the previous, thus also introducing noise.

III. Filtering node design

For simplicity, we considered only the final message from the cumulative point-cloud in the provided rosbag, which is the one that contained more points. Because of that, instead of the point cloud being obtained from the subscriber, we just load the last message in the bag and process it.

In order to design a filter, we must first analyze the original point cloud to see exactly what we are dealing with. The original cloud which is shown in figure 1.

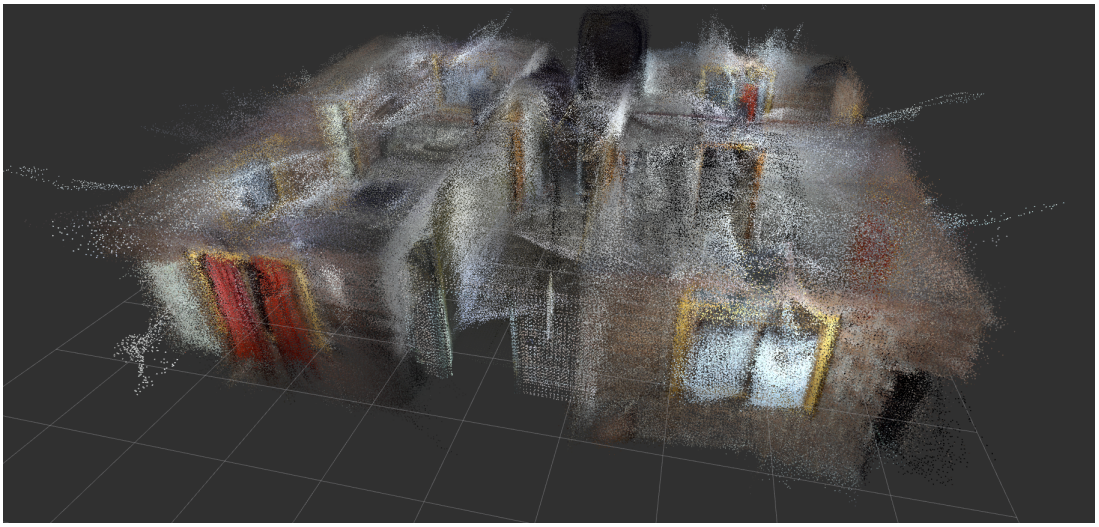

Figure 1: Original Noisy Pointcloud

We can see many points that have a white color. These are points that have ... invalid . Removing these points already makes the point cloud look a lot better, as shown in figure 2.

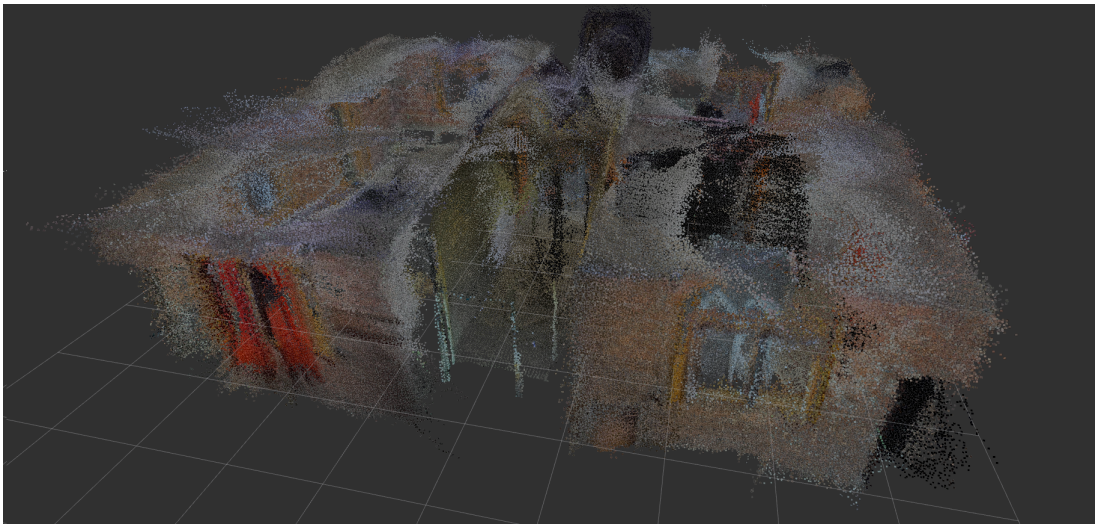

Figure 2: Pointcloud after removing NaN points

We can see that the point cloud has a lot of noise in the floor and ceiling. Because we are only interested in vertical structures like walls, doors and windows, a simple approach is just removing

all points below and above certain thresholds. For this point-cloud, we decided to keep all points for which the z coordinate lies in $[-0.9, 0.9]$.

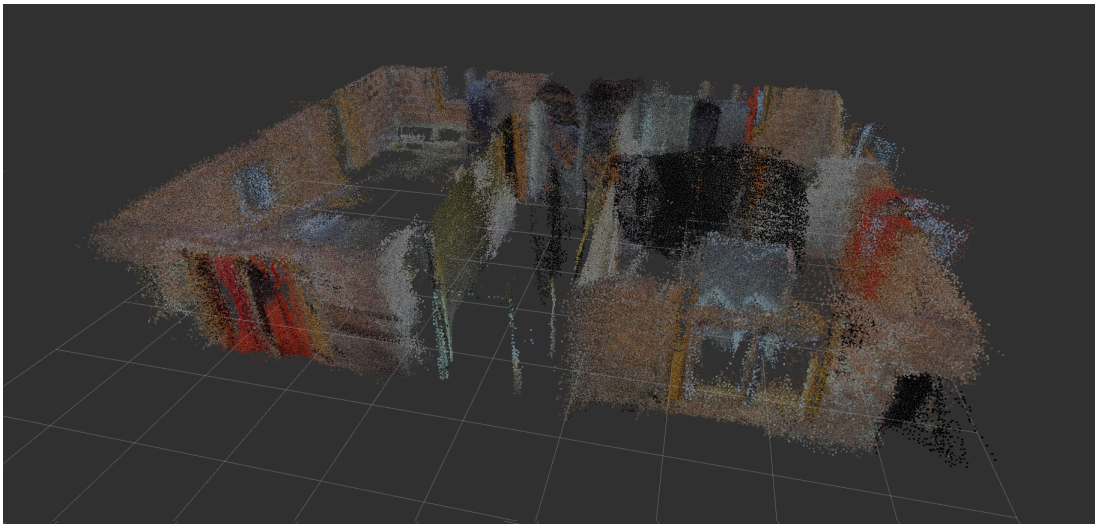

Figure 3: Pointcloud after removing floor and ceiling points

Although a lot of unnecessary points have already been removed at this point, the point cloud still has many outliers. In order to solve this, we will use an algorithm called Statistical Outlier Removal. This method removes noise from a point cloud based on a statistical analysis of each point's neighborhood. The algorithm finds outliers by comparing the distance of each point to its k-nearest neighbors against a global threshold. The global threshold is computed based on the mean and standard deviation of the distances across the full point cloud.

The pseudo code for the algorithm is

- 1. Find k-nearest neighbors for each point.
- 2. Calculate the mean distance to neighbors for each point.
- 3. Compute global mean (μ) and standard deviation (σ) of all mean distances.
- 4. Mark points as outliers if their mean distance falls outside the range $[\mu - \alpha\sigma, \mu + \alpha\sigma]$, where α is a parameter that can be tuned to vary precision

Applying this led to the final filtered point cloud represented in figure4.

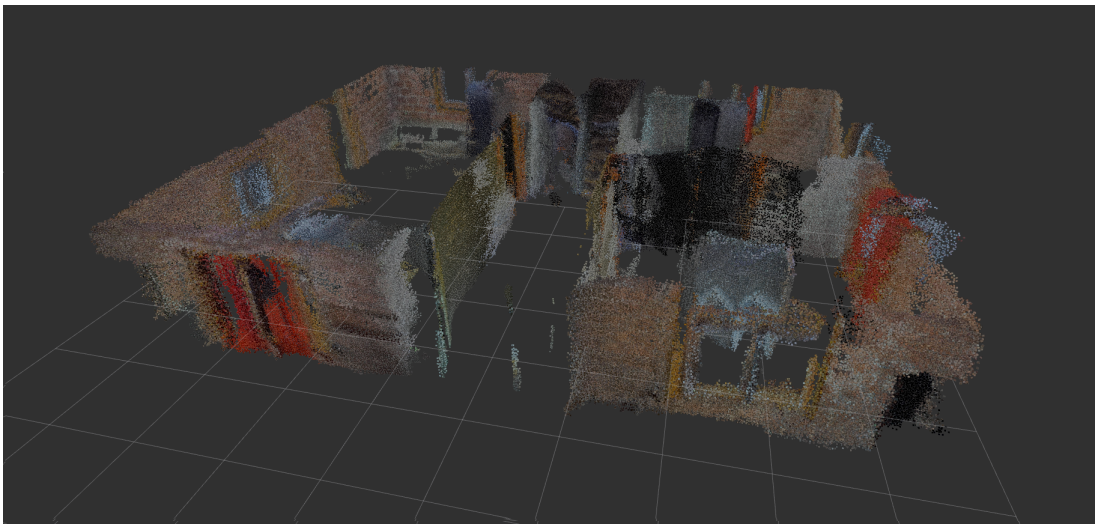

Figure 4: PointCloud after applying Statistical Outlier Removal

IV. Further improvements that could be implemented

One idea that could be very successful is doing feature extraction like planar segmentation and edge detection. Since walls and windows are planar structures, using Random sample consensus (RANSAC) would be a good way to find planar structures and filter out the noise.

Another idea would be clustering-based approaches like Euclidian clustering and Region Growing. Euclidian clustering groups points based on their Euclidean distance, which is useful for separating distinct objects like windows, walls, and furniture. Region Growing segments the point cloud based on the curvature and smoothness of the surface, which helps separate walls.

Finally, more advanced methods based on deep learning, like PointNet architectures, could be used, but we would need to have access to a lot of labeled data.