Assignment Report: Autonomous Vehicles LiDAR Filtering

Rohit Jagannath Dhote **Email:** rohitdhote111@gmail.com

September 7, 2024 **Github**

Overview: This report includes my approach toward the tasks given in the assignment. I have explained my thought process while solving the tasks and added visual results, advantages, and limitations of the work. All the files related to installation, implementation and analysis can be found on github(`https://github.com/rohitdhote111/TII_Assignment`).

---

1. Data Understanding

The .bag file appears to have been collected using a combination of sensors, particularly those used in autonomous vehicles or robotics. The complete information about Topics and Messages obtained using command `rosbag info LiDARfile.bag` and Based on the topics and message types listed, the sensors involved are likely:

1. **LiDAR Sensors**:
   Topics such as `/mbuggy/os1/points`, `/mbuggy/os2/points`, and `/mbuggy/os3/points` with message types `sensor_msgs/PointCloud2` indicate that LiDAR data was collected.
   The specific message type `ouster_ros/PacketMsg` suggests that the LiDARs used could be from the Ouster brand, which are common in autonomous vehicle applications.
2. **Cameras**:
   Multiple topics such as `/mbuggy/camera_front/image_raw`, `/mbuggy/camera_back/image_raw`, and similar topics for left and right cameras with message types `sensor_msgs/Image` and `sensor_msgs/CompressedImage` indicate that there were at least four cameras (front, back, left, and right) capturing images.
3. **IMU (Inertial Measurement Unit)**:
   Topics like `/mbuggy/os1/imu`, `/mbuggy/os2/imu`, and `/mbuggy/os3/imu` with the message type `sensor_msgs/Imu` indicate that there were IMUs used to capture orientation, acceleration, and angular velocity data.

1.1. Rviz Configuration and Use

1. After the Rviz window is open, select `Fixed Frame > mbuggy/base_link` from the left-side display panel.
2. Add the required point clouds by navigating to `Add > By Topic > select the desired point cloud (PC)`.
   - Increase the size of the points (e.g., 0.03 is a good value).
   - Change the color of the point cloud if required. Navigate to `Color Transform > Flat`, then select the desired color.

2. Noise Filtering

2.1. Statistical Outlier Removal (SOR)

Statistical Outlier Removal (SOR) is a commonly used filtering technique in 3D point cloud processing to remove noise and outliers from the data. The basic idea of SOR is to analyze the distribution of points in a local neighborhood and remove points that are too far away from their neighbors based on statistical analysis.

2.1.1. SOR Algorithm Steps

1. **Neighborhood Search:**
   For each point in the point cloud, the algorithm finds its k nearest neighbors.
2. **Distance Calculation:**
   For each point, the distances between that point and its k nearest neighbors are computed.

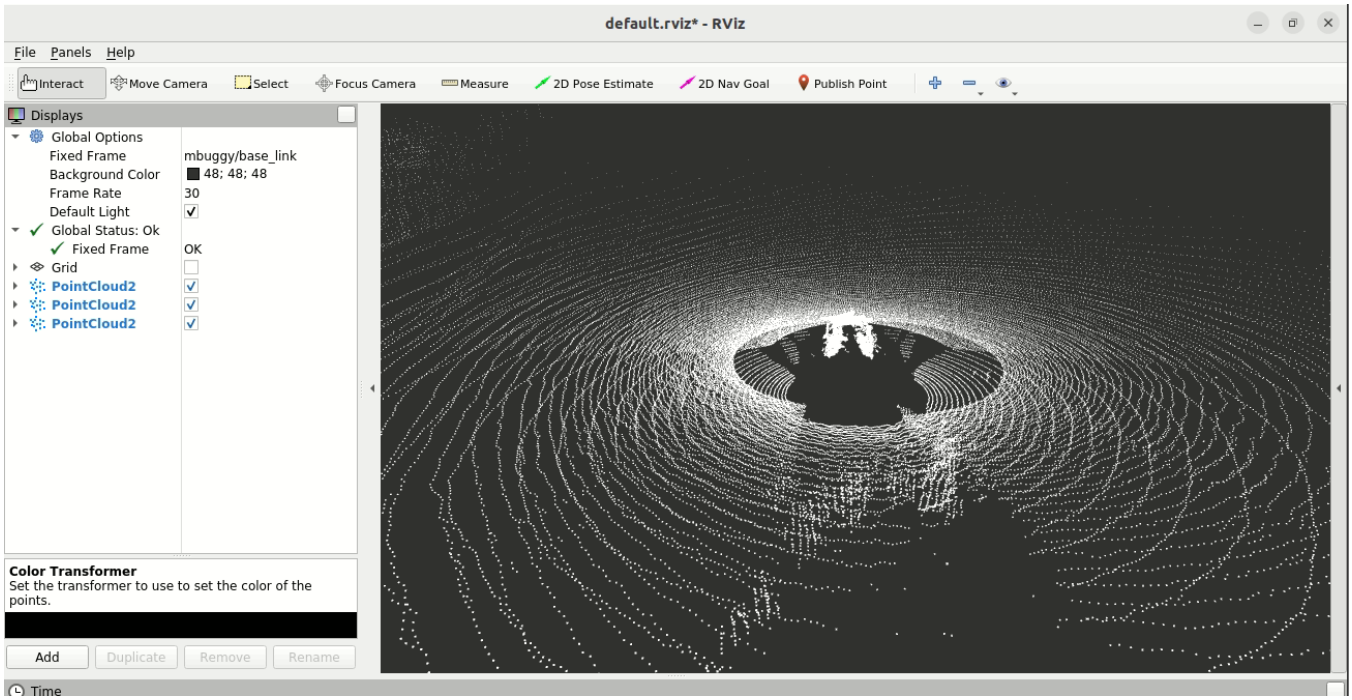


Figure 1: Visualizing 3 LiDAR sensor (os1, os2, os3) data together

3. **Mean Distance Calculation:**
   For each point, the mean distance to its neighbors is calculated. This mean distance serves as an indication of how close or far the point is from its neighbors.
4. **Statistical Thresholding:**
   The mean and standard deviation of the distances for all points are calculated. Points that have a mean distance larger than a specified multiple of the standard deviation (a user-defined threshold) are considered statistical outliers.
5. **Removal of Outliers:**
   Points that are flagged as outliers (i.e., those whose mean distance exceeds the threshold) are removed from the point cloud.

2.1.2. SOR Algorithm Parameters

The SOR filter has two key parameters that need to be set:

1. k – Number of Nearest Neighbors

This parameter defines how many neighbors are considered for each point during the distance computation.

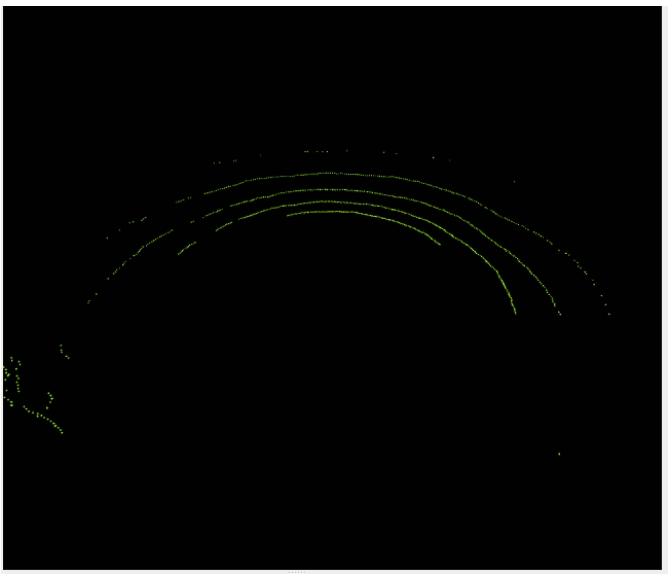
- Selection Criteria:
  - If the point cloud is dense, a larger value for k is generally better, as it will result in more stable outlier detection.
  - If the point cloud is sparse, a smaller k may be more appropriate, as large values might cause valid points to be misclassified as outliers.

2. `std_dev_mul_thresh` – Standard Deviation Multiplier

This parameter controls how strict the outlier removal process is. It defines the threshold distance as a multiple of the standard deviation of all the distances between each point and its neighbors.

- Points that lie further away from their neighbors than this threshold are considered outliers and removed.
- Selection Criteria:
  - A lower value will be more aggressive and remove more points, which might include valid points in areas with varying point density.
  - A higher value will be more conservative, removing fewer points, but it may leave some outliers in the data.

(a) SOR filter: Green color shows noise and white color shows filtered point cloud



(b) SOR Filter: Green color shows noise points

Figure 2: Green color points shows noise points which concludes that SOR filter NOT worked well

2.1.3. Steps to Select SOR Parameters

1. **Visualize the Point Cloud:**
   Start by visualizing the raw point cloud to get an idea of its density and noise level.
2. **Set Initial Parameters:**
   - Set k to a reasonable value based on the density of your point cloud (e.g., start with $k = 20$ for a dense point cloud).
   - Set the standard deviation multiplier to `std_dev_mul_thresh` = 1.0 initially.
3. **Adjust k:**
   - If the point cloud is very noisy, you might want to increase the value of k to stabilize the outlier detection.
   - If you notice that valid points are being removed, decrease k.
4. **Adjust the Standard Deviation Multiplier:**
   - If too many valid points are being removed, increase the value of `std_dev_mul_thresh` (e.g., to 2.0 or higher).
   - If there are still a lot of outliers in the result, reduce the threshold to make the filter stricter (e.g., 1.0 or lower).
5. **Iterate:**
   Continue adjusting these parameters based on the results, visualizing the filtered point cloud to ensure a balance between preserving valid points and removing noise.

2.2. Radius Outlier Removal (ROR)

Overview: Radius Outlier Removal is another common method, especially in cases where noise is highly isolated. It removes points that do not have a sufficient number of neighbors within a given radius.

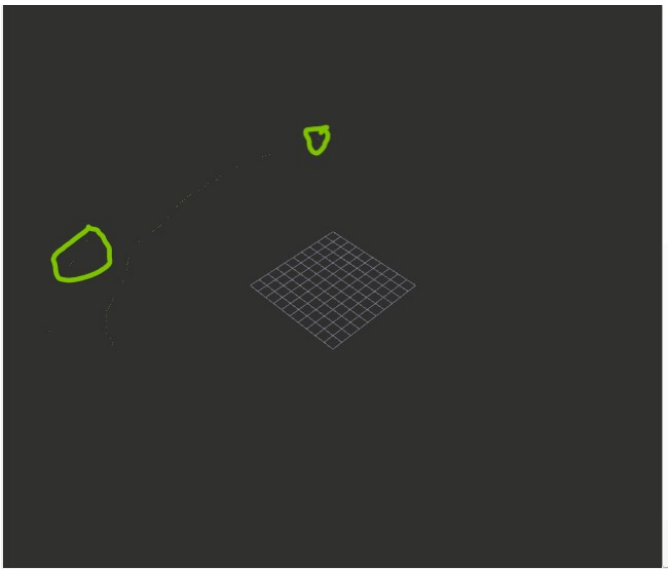
2.2.1. Key Parameters:

- **Radius_search:** This defines the radius around each point in which neighbors are searched.
  - **Suggested value:** 0.1 to 0.5 meters (depending on the sensor's range and the density of the point cloud).
- **min_neighbors_in_radius:** This defines the minimum number of neighbors within the radius for a point to be considered valid. If a point has fewer neighbors than this threshold, it is considered noise.
  - **Suggested value:** 3-5.

2.2.2. Observation:

As very few noise points (maybe dust points) which are highly isolated those are classified correctly as noise points.

(a) ROR filter: Green color shows noise and white color shows filtered point cloud



(b) ROR Filter: Isolated points are marked using pencil for better understanding

Figure 3: Green color points shows noise points which concludes that ROR filter worked well for isolated points

2.3. Voxel Grid Filter (Downsampling)

(Not used as point cloud density is already low)

Overview: The Voxel Grid Filter is used for downsampling the point cloud and removing noisy, redundant points. This is particularly useful for reducing the resolution of the point cloud while preserving the overall structure.

Key Parameters:

- **leaf_size:** This parameter controls the size of the 3D voxel grid. The point cloud will be divided into voxels of this size, and only one point will be kept from each voxel.

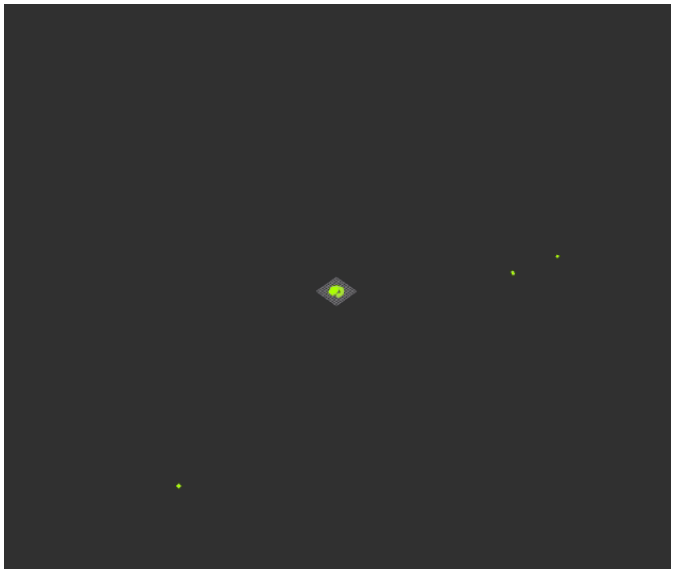
2.4. Conditional Removal Filter

Overview: This algorithm removes points based on a predefined condition. For example, points that fall outside certain ranges in the X, Y, or Z axis, which are likely to be noise, can be removed.

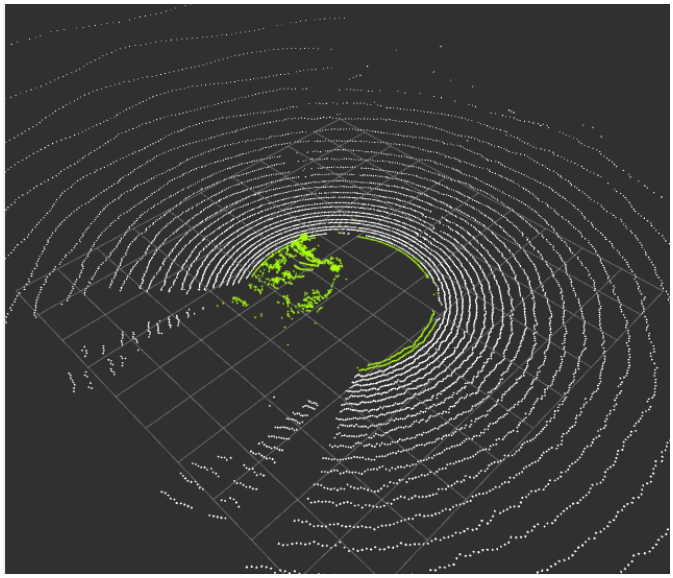
2.4.1. Key Parameters:

Condition Function: We can define custom conditions for removing points, such as thresholds on coordinates or intensity values.

- **Min range:** 3 meters.
- **Max range:** 50 meters.



(a) Zoom out view: Green color shows noise and white color shows filtered point cloud



(b) Zoom in view: Green color points belonging to vehicle are detected as noise

Figure 4: Conditional Filter: Green color points shows noise points

3. Ground Points Removal

3.1. RANSAC Plane Fitting Algorithm

The RANSAC (Random Sample Consensus) plane fitting algorithm is commonly used to fit a plane to 3D LiDAR point cloud data and is particularly useful for ground removal in autonomous driving or robotics applications. The algorithm works by iteratively selecting a subset of points, fitting a plane, and then determining how well the rest of the points fit that plane. It is robust to noise and outliers, making it ideal for point clouds that may include significant amounts of noise or clutter.

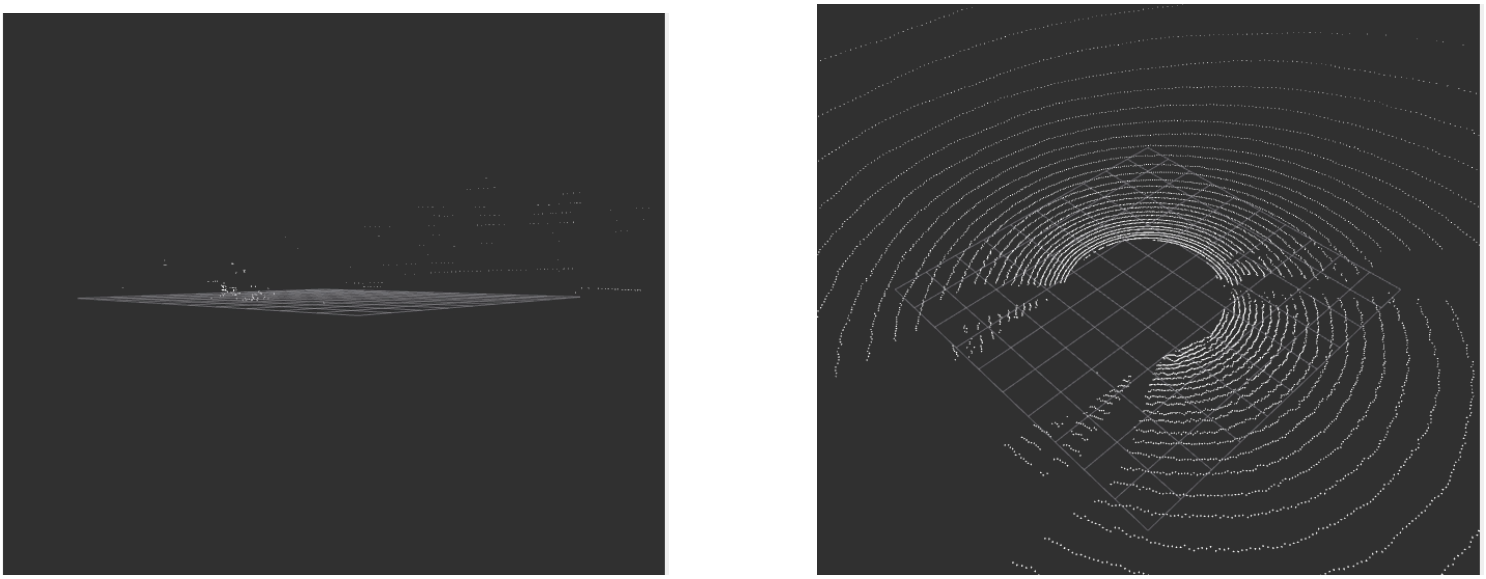
3.1.1. Steps of the RANSAC Plane Fitting Algorithm

1. **Random Sampling:**
   Randomly select a minimal set of points (usually 3 points in 3D) from the point cloud. These points are used to define a candidate plane.
2. **Plane Model Calculation:**
   Using the selected points, calculate the parameters of a plane in the form:

$$ax + by + cz + d = 0$$

   where (a, b, c) is the normal vector of the plane, and d is the distance of the plane from the origin.
3. **Consensus Set Identification:**
   For each remaining point in the point cloud, calculate its distance to the plane. If the distance is less than a specified threshold (known as the inlier threshold), the point is considered an inlier, meaning it fits the plane.
4. **Inlier Count:**
   Count the number of inliers (points that lie within the threshold distance from the plane). The more inliers, the better the plane fits the data.
5. **Iteration:**
   Repeat the process for a predefined number of iterations, randomly sampling different sets of points each time to find the best possible plane.
6. **Best Plane Selection:**
   After the specified number of iterations, the plane model with the most inliers is selected as the best-fit plane.
7. **Ground Removal:**
   The points that are part of the consensus set (i.e., inliers that belong to the plane) are considered part of the

ground. These points can be removed or filtered out, leaving the rest of the point cloud for further analysis.



(a) Non Ground Points **(b)** Ground Points

Figure 5: RANSAC: White color points shows ground points and Green color shows non-ground points

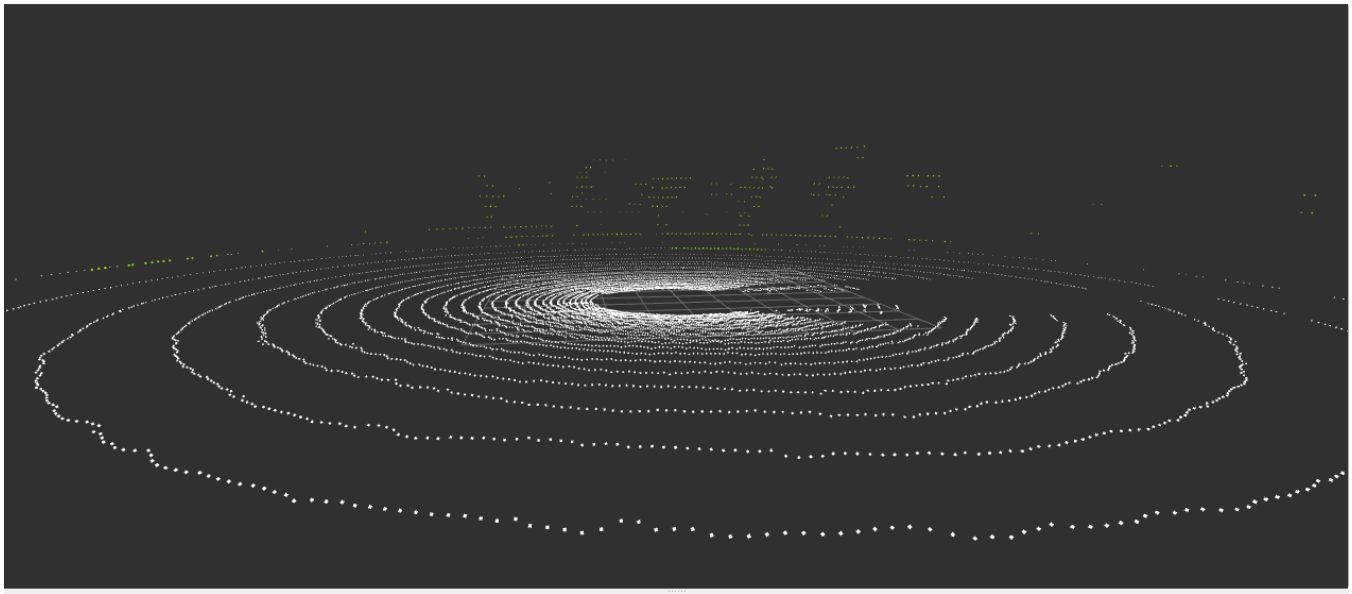


Figure 6: RANSAC: White color points shows ground points and Green color shows non-ground points

3.1.2. Parameter Selection for RANSAC Plane Fitting

Choosing the right parameters for the RANSAC algorithm is critical for successfully fitting a plane and removing the ground points. The key parameters to consider are:

- **Distance Threshold:**
  This parameter defines how close a point needs to be to the estimated plane to be considered an inlier. If the distance between the point and the plane is less than this threshold, the point is classified as an inlier.
  - **Selection:** In practice, this value depends on the noise level and resolution of your LiDAR sensor. For Ouster LiDAR, a typical value might range between 0.05 and 0.2 meters.
  - A smaller threshold makes the algorithm more sensitive and only considers points that are very close to the plane as inliers, which might lead to some valid ground points being excluded.

- A larger threshold is more tolerant of noise but could cause non-ground points (like low-lying objects) to be included as part of the ground.

- **Max Iterations:**
  This parameter defines the maximum number of iterations the algorithm will run to find the best plane. A higher number of iterations increases the likelihood of finding the correct plane, but also increases the computational time.
  - **Selection:** For typical point cloud data, values between 100 and 1000 iterations are usually sufficient. If the point cloud has a lot of noise or outliers, you may need to increase this number.
- **Inlier Ratio:**
  Some implementations of RANSAC use an inlier ratio to stop the iterations early if a plane with a certain percentage of inliers has been found.
  - **Selection:** A ratio of 0.5 to 0.8 is usually effective. This means that if 50% to 80% of the points are classified as inliers, the algorithm may stop early and return the current best plane.
- **Initial Point Selection:**
  The number of points randomly selected to define the initial plane. In 3D, three points are required to define a plane.
  - This is generally fixed in most RANSAC implementations.
- **Confidence Level:**
  This parameter indicates how confident you want the algorithm to be in finding the correct model. A higher confidence level (e.g., 99%) means more iterations will be run to increase the chances of finding the correct plane.
  - **Selection:** Typically set to 95% or 99% depending on the noise level of the data. Higher confidence leads to more iterations but improves the likelihood of finding the best-fit plane.

3.1.3. Key Considerations for Ground Removal

- **Point Cloud Density:**
  If your LiDAR point cloud is dense, smaller distance thresholds may yield better results, as they will allow more precise detection of ground planes.
- **Surface Curvature:**
  In highly uneven or hilly terrain, the ground may not perfectly fit a plane. In these cases, a smaller distance threshold may remove valid ground points, so adjusting the threshold or using more advanced fitting techniques (such as using more complex models like cylindrical or parabolic surfaces) may be necessary.
- **Outliers:**
  Ensure that significant noise or outliers have been removed before applying RANSAC, as they may impact the plane-fitting process.

3.1.4. Experiment for Processing Time Improvement in RANSAC

Frequency value is introduced in the standard implementation. Instead of running the computationally expensive RANSAC plane-fitting algorithm on every incoming point cloud, the plane parameters are calculated once and reused for a specified number of subsequent point clouds. This reduces the frequency of RANSAC computations, which is beneficial in scenarios where consecutive point clouds exhibit similar ground planes. As a result, the processing time for each point cloud decreases, leading to a more efficient system with better real-time performance, while still maintaining the accuracy of ground segmentation over a series of point clouds.

Observation: As the vehicle moves, the elevation of the LiDAR sensor changes, which means the origin of the point cloud (where the LiDAR is located) is also shifting. Since the ground remains stationary, the plane parameters computed at one vehicle position become invalid for the next position if the sensor's movement exceeds a threshold. In this case, the threshold is set to 10 cm. If the change in the sensor's position is greater than 10 cm, the previously computed ground plane will no longer accurately detect the ground.

To address this, we can either reduce the frequency of recomputing the plane parameters or increase the threshold distance to allow for more movement of the sensor.

We can observe from the log images below that processing time is improved.

```
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
[ INFO] [1725693509.790507762]: Point cloud received
[ INFO] [1725693509.811162971]: Point cloud processed in 0.0208 seconds
[ INFO] [1725693509.844315727]: Point cloud received
[ INFO] [1725693509.857752169]: Point cloud processed in 0.0135 seconds
[ INFO] [1725693509.942565017]: Point cloud received
[ INFO] [1725693509.960639758]: Point cloud processed in 0.0181 seconds
[ INFO] [1725693510.043223558]: Point cloud received
[ INFO] [1725693510.059453342]: Point cloud processed in 0.0162 seconds
[ INFO] [1725693510.144529820]: Point cloud received
[ INFO] [1725693510.164292534]: Point cloud processed in 0.0198 seconds
[ INFO] [1725693510.246556108]: Point cloud received
[ INFO] [1725693510.261055042]: Point cloud processed in 0.0145 seconds
[ INFO] [1725693510.344058306]: Point cloud received
[ INFO] [1725693510.350953417]: Point cloud processed in 0.0069 seconds
[ INFO] [1725693510.444571914]: Point cloud received
[ INFO] [1725693510.462219483]: Point cloud processed in 0.0176 seconds
[ INFO] [1725693510.546657318]: Point cloud received
[ INFO] [1725693510.566315184]: Point cloud processed in 0.0197 seconds
[ INFO] [1725693510.645066137]: Point cloud received
[ INFO] [1725693510.656303971]: Point cloud processed in 0.0113 seconds
[ INFO] [1725693510.747890683]: Point cloud received
[ INFO] [1725693510.764359402]: Point cloud processed in 0.0165 seconds
[ INFO] [1725693510.842209924]: Point cloud received
[ INFO] [1725693510.859717845]: Point cloud processed in 0.0175 seconds
[ INFO] [1725693510.941358189]: Point cloud received
[ INFO] [1725693510.959051544]: Point cloud processed in 0.0177 seconds
[ INFO] [1725693511.039494350]: Point cloud received
[ INFO] [1725693511.055290140]: Point cloud processed in 0.0158 seconds
[ INFO] [1725693511.155642988]: Point cloud received
[ INFO] [1725693511.167506495]: Point cloud processed in 0.0119 seconds
[ INFO] [1725693511.238561171]: Point cloud received
[ INFO] [1725693511.253252236]: Point cloud processed in 0.0147 seconds
[ INFO] [1725693511.337906691]: Point cloud received
```

(a) Log for ransac applied for every PC

```
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
[ INFO] [1725693355.780088117]: Point cloud received
[ INFO] [1725693355.798790698]: Computing RANSAC plane.
[ INFO] [1725693355.803109642]: RANSAC plane computed and cached.
[ INFO] [1725693355.804119188]: Processing time: 0.024179 seconds
[ INFO] [1725693355.833550554]: Point cloud received
[ INFO] [1725693355.845046041]: Processing time: 0.011526 seconds
[ INFO] [1725693355.931951020]: Point cloud received
[ INFO] [1725693355.945169143]: Processing time: 0.013215 seconds
[ INFO] [1725693356.031582194]: Point cloud received
[ INFO] [1725693356.043822329]: Processing time: 0.012244 seconds
[ INFO] [1725693356.132617422]: Point cloud received
[ INFO] [1725693356.143824706]: Processing time: 0.011237 seconds
[ INFO] [1725693356.233669053]: Point cloud received
[ INFO] [1725693356.243435609]: Computing RANSAC plane.
[ INFO] [1725693356.244767281]: RANSAC plane computed and cached.
[ INFO] [1725693356.245082877]: Processing time: 0.011449 seconds
[ INFO] [1725693356.332441949]: Point cloud received
[ INFO] [1725693356.342063703]: Processing time: 0.009662 seconds
[ INFO] [1725693356.432683324]: Point cloud received
[ INFO] [1725693356.442968799]: Processing time: 0.010318 seconds
[ INFO] [1725693356.534302503]: Point cloud received
[ INFO] [1725693356.545549493]: Processing time: 0.011264 seconds
[ INFO] [1725693356.632333141]: Point cloud received
[ INFO] [1725693356.642886073]: Processing time: 0.010583 seconds
[ INFO] [1725693356.732866747]: Point cloud received
[ INFO] [1725693356.744093114]: Computing RANSAC plane.
[ INFO] [1725693356.746699395]: RANSAC plane computed and cached.
[ INFO] [1725693356.747032434]: Processing time: 0.014210 seconds
[ INFO] [1725693356.829945236]: Point cloud received
[ INFO] [1725693356.841799380]: Processing time: 0.011875 seconds
[ INFO] [1725693356.929479954]: Point cloud received
[ INFO] [1725693356.940909661]: Processing time: 0.011427 seconds
[ INFO] [1725693357.029134323]: Point cloud received
```

(b) Log for fast ransac experiment

Figure 7: Log to analyse processing time experiment results

4. Issues and Limitation

The provided point cloud processing code is a solid base, but it has a few potential issues and limitations that could arise during operation, especially in real-time applications or complex environments.

4.1. Noise Removal

Static Distance Thresholds:

- The noise filtering is based on two static thresholds (min distance = 3.0 meters and max distance = 50.0 meters). Points closer than 3 meters or farther than 50 meters are classified as noise. This approach assumes a fixed environment, but in reality, the ideal thresholds might vary depending on the environment or sensor settings.
- Limitation: This approach lacks adaptability and might incorrectly classify valid points as noise or miss actual noise points if the sensor's range or the environment changes (e.g., in urban vs. rural settings).

4.2. RANSAC-Based Ground Segmentation

RANSAC is a robust technique for plane fitting, but it has its own limitations:

- Execution Time: It can be slow for large point clouds due to its iterative nature.
- Ground Variability: In complex environments, the ground might not always be perfectly flat, and RANSAC may struggle with uneven or rough terrain.
- **Multiple plane case** If multiple planes are present in the point cloud then ransac may detect any plane other than ground plane. To avoid this we can use selection criteria based on plane orientation. For example, Select planes having some specified plane normal orientation.
- Parameter Tuning: The threshold (distanceThreshold = 0.1) for identifying ground points is hardcoded, which may not be optimal for all environments. It might miss ground points if set too low or include noise if set too high.

Limitation: The method works well for flat ground but may fail in environments with slopes, uneven ground, or multiple planes (e.g., roads with curbs or hills).

4.3. No Error or Exception Handling

- Issue: There is no error handling for cases where the input point cloud might be empty, corrupted, or invalid.
- Limitation: In real-world applications, LiDAR sensors might occasionally produce invalid or incomplete point clouds. The current code does not handle such cases gracefully, which could lead to inconsistent behavior.

5. Conclusion

In conclusion, this work presents an efficient method for processing LiDAR point clouds by leveraging a combination of noise filtering and ground plane segmentation using the RANSAC algorithm. The implementation addresses the dynamic nature of vehicle movement by caching RANSAC plane parameters and reusing them for subsequent frames, thereby reducing computational overhead. While the approach offers a basic distance-based noise filtering technique, it highlights the limitations of static thresholds and their sensitivity to environmental changes. By incorporating more advanced noise filtering methods and adaptive mechanisms, the system can be further enhanced for greater robustness and accuracy in diverse settings.

6. References

1. A Survey on Ground Segmentation Methods for Automotive LiDAR Sensors (Paper link)
2. Wiki ROS Tutorials