ASSIGNMENT - 2

Objective

The objective of this project is to process and visualize LiDAR point cloud data obtained from the **NuScenes dataset**. The tasks include:

1. **Aggregating LiDAR Point Clouds**: Transforming and combining LiDAR data from multiple frames into a unified global coordinate system.
2. **Filtering Moving Objects**: Removing points that correspond to dynamic objects, such as vehicles and bicycles, using annotated bounding boxes.
3. **Colorizing the Point Cloud**: Assigning RGB color values to the LiDAR points by projecting them onto corresponding camera images.

Step 1: Aggregating LiDAR Point Clouds

Objective

To combine multiple LiDAR frames into a single global coordinate system.

Step-by-Step Implementation

1. **Load Raw LiDAR Points**:
   - Retrieve LiDAR point cloud data for each frame.
   - Points are initially represented in the sensor coordinate system.
2. **Apply Sensor-to-Vehicle Transformation**:
   - Use the sensor's extrinsic parameters (rotation matrix R_{sensor} and translation vector t_{sensor}).
   - Transform points from the sensor coordinate system to the vehicle coordinate system.
3. Formula:

$$P_{vehicle} = R_{sensor} * P_{sensor} + t_{sensor}$$

   Where:

   - P_{sensor}: Raw LiDAR points.
   - R_{sensor}: Sensor rotation matrix.
   - t_{sensor} : Sensor translation vector.
   - $P_{vehicle}$: Points in the vehicle coordinate system.

4. **Apply Vehicle-to-Global Transformation**:
   - Use the vehicle's ego pose data (rotation matrix R_{ego} and translation vector t_{ego}).
   - Transform points from the vehicle coordinate system to the global coordinate system.

   Formula:

$$P_{global} = R_{ego} * P_{vehicle} + t_{ego}$$

5. **Iterate Through Frames**:
   - Repeat the above steps for each LiDAR frame.
   - Combine all transformed points into a single global point cloud.
6. **Reasoning**:
   - Transforming all frames into a common coordinate system allows for a unified representation of the entire scene.
7. **Output**:
   - A set of point clouds representing all LiDAR frames in the global coordinate system.

Step 2: Filtering Moving Objects

Objective

To remove points corresponding to dynamic objects, such as vehicles, bicycles, and motorcycles, based on annotated bounding boxes.

Step-by-Step Implementation

1. **Load Annotations**:
   - Retrieve bounding box annotations for each dynamic object in the scene.
2. **Identify Points Inside Bounding Boxes**:
   - For each bounding box BBB:
     - Check if a given point PPP lies inside the bounding box using a geometric check.
3. Formula:

$$f(P, B) = \begin{cases} 1 & \text{if } P \text{ lies inside } B \\ 0 & \text{otherwise} \end{cases}$$

Filter Out Points:

- Exclude all points where f(P,B)=1.
- Retain points that fall outside all bounding boxes.

4. **Iterate Through Frames**:
    - For each LiDAR frame:
        - Apply the filtering process to remove points corresponding to dynamic objects.
5. **Reasoning**:
    - Dynamic objects introduce noise when analyzing static features of the environment (e.g., roads, buildings).
    - Removing these points allows focus on the static infrastructure.
6. **Output**:
    - A cleaned point cloud containing only static points.
    - Points are visually highlighted (e.g., colored red) for clarity.

Step 3: Colorizing the Point Cloud

Objective

To assign RGB color values to each LiDAR point by projecting the points onto camera images.

Step-by-Step Implementation

1. **Transform Points to the Camera Frame**:
    - Convert the LiDAR points from the global coordinate system into the camera coordinate system.
    - Use the camera's extrinsic parameters (rotation matrix R_{camera} and translation vector t_{camera}).

    Formula:

    $$P_{camera} = R_{camera} * (P_{global} - t_{camera})$$

    Where:

    - P_{global}: LiDAR points in the global frame.
    - R_{camera}: Points in the camera frame.
2. **Project Points onto the Image Plane**:
    - Use the camera's intrinsic matrix K to project the 3D points onto the 2D image plane.

Formula:

$$P_{image} = K * P_{camera}$$

Where:

- K: Camera intrinsic matrix.
- P_{image}: 2D pixel coordinates in the image.

Normalize the projected coordinates to get the pixel locations:

$$x_{pixel} = x_{image} / 2$$

$$y_{pixel} = y_{image} / 2$$

3. **Extract RGB Values**:
   - For each projected point, check if it lies within the image boundaries.
   - Extract the RGB color of the corresponding pixel from the image.
   - Assign the extracted color to the LiDAR point.
4. **Handle Multiple Cameras**:
   - Repeat the projection process for all six cameras.
   - Prioritize points visible in multiple cameras.
5. **Reasoning**:
   - Adding RGB colors to the LiDAR points combines spatial geometry with visual information, making the point cloud more interpretable and realistic.
6. **Output**:
   - A colorized point cloud where each point has an RGB value derived from the camera images.