Approach Document: 3D Cuboid Rotation Estimation

1. Objective

The objective of this assignment is to process a sequence of depth images to estimate the properties of a rotating 3D cuboidal box. The specific deliverables include:

1. The normal angle and visible area of the largest visible face at each timestamp.
2. The axis of rotation vector of the box.
3. A Python script, a results table, and a text file containing the axis vector.

This document details the algorithm and implementation employed to solve this task. The solution is implemented as a ROS 2 package, designed for real-time data processing and subsequent storage of final results upon completion. ROS based implementation also makes it easier to visualize the problem and the solution.

Video Link:
https://drive.google.com/file/d/10PMTo_A2A3om7rLbrXHAcSpz88P2nquy/view?usp=sharing

2. Methodology and Algorithm

The core of the solution is a ROS 2 node (depth_processor.py) that subscribes to raw depth data, performs 3D geometric analysis on each frame, and publishes the resultant information.

2.1. Data Handling (ROS 2 Node)

A ROS 2 node, developed in Python3 (rclpy), manages the data stream.

- **Subscription:** The node subscribes to the `/depth` topic, which provides `sensor_msgs/msg/Image` messages.
- **Publication:** For visualization and real-time analysis, the node publishes the following:
  - `/perception/vis_cloud`: A `sensor_msgs/msg/PointCloud2` message for RVIZ, displaying the full point cloud with the detected face highlighted in red.
  - `/perception/normal_angle`: A `std_msgs/msg/Float64` representing the calculated angle.
  - `/perception/visible_area`: A `std_msgs/msg/Float64` representing the calculated area.
- **Data Saving:** The node aggregates results from all processed frames. A shutdown hook, activated by Ctrl+C, saves the comprehensive `results_table.txt` and `rotation_axis.txt` files to the designated directory.

2.2. 3D Point Cloud Reconstruction

Each 2D depth image requires conversion into a 3D point cloud for geometric analysis.

1. **Read Image:** The raw `msg.data` (a byte buffer) is read into a NumPy array, respecting its encoding (e.g., 16UC1) and scaled to meters.
2. **Un-projection:** Utilizing the pinhole camera model, each pixel (u, v) with its corresponding depth Z is "un-projected" into a 3D point (X, Y, Z) within the camera's coordinate frame. This process necessitates the camera's intrinsic matrix K.
   X = (u - cx) * Z / fx
   Y = (v - cy) * Z / fy
3. **Filtering:** The resultant point cloud undergoes filtering using a simple depth range (e.g., 0.2m to 3.0m) to eliminate background elements (wall) and any near-field sensor noise.

2.3. Task 1: Normal Angle & Visible Area

With a refined 3D point cloud of the cuboid, the largest face is identified using the RANSAC (Random Sample Consensus) algorithm.

1. **Plane Fitting:** `open3d.segment_plane()` is employed to fit a plane model (Ax + By + Cz + D = 0) to the point cloud. The plane exhibiting the highest number of inlier points is considered the largest visible face.
2. **Normal Angle:**
   - The coefficients of the plane equation yield the face normal vector, n_face = [A, B, C].
   - The camera's viewing direction (its "normal") is assumed to be the positive Z-axis, n_cam = [0, 0, 1].
   - The angle theta between the two vectors is computed using the dot product: theta = arccos((n_face . n_cam) / (|n_face| |n_cam|))
3. **Visible Area:**
   - The 3D points identified as inliers by RANSAC (which constitute the face) are isolated.
   - An Oriented Bounding Box (OBB) is fitted to these inlier points.
   - The OBB's extent (its dimensions) provides the length, width, and thickness of the planar patch.
   - The area is calculated by sorting the extents and multiplying the two largest dimensions, thereby effectively disregarding the (near-zero) thickness of the planar patch.

2.4. Task 2: Axis of Rotation Estimation

The axis of rotation is defined by a point on the axis and a direction vector.

1. **Point on Axis:** The centroid (average X, Y, Z coordinates) of the entire filtered point cloud is calculated for each frame. The point on the axis is estimated by averaging all these centroids across all 7 frames, providing a stable center point that exhibits robustness to noise.
2. **Direction Vector:** Based on the "Top View" diagram provided in the assignment PDF, the box is depicted rotating about a vertical axis. Consequently, the axis of rotation vector (its direction) is assumed to be parallel to the camera's Y-axis: v_rot = [0, 1, 0].

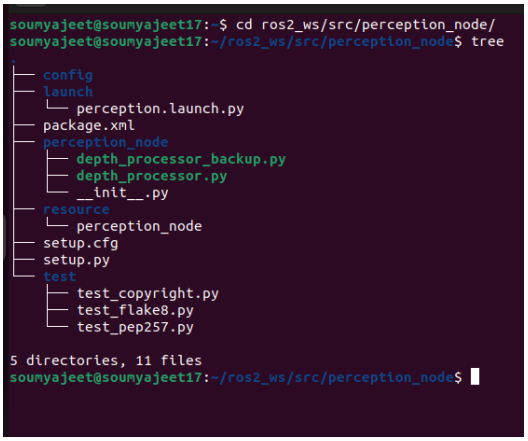
3. Key Assumptions

This solution relies upon two critical assumptions, as they were not explicitly provided within the assignment data:

1. **Camera Intrinsic Matrix (K):** The intrinsic parameters are fundamental for 3D reconstruction. Since no reference to the camera's intrinsic matrix are given, a standard, typical matrix for a 640x480 camera was assumed:
    - Focal Lengths: fx = 525.0, fy = 525.0
    - Principal Point: cx = 320.0, cy = 240.0
      The final results for Area and Centroid are contingent upon this assumption.
2. **Axis of Rotation Direction:** The direction of the rotation axis was assumed to be [0, 1, 0] based on the problem's diagram.

4. Implementation Details

- **Language:** Python 3
- **Framework:** ROS 2 (Humble Hawksbill)
- `depth_processor.py` is used to process the data, it subscribes to the ros bag and then outputs the required details after computation.
- `perception.launch.py` combines the launch for RViz and the processor node simultaneously to avoid using multiple terminals.
- Tree structure of the node is as follows:



- **Config:** folder contains the config file for rviz.
- **Key Libraries:**
    - `rclpy`: For creating the ROS 2 node, publisher, and subscriber functionalities.
    - `rosbags`: (Utilized in the initial script) for reading the .db3 file.
    - `open3d`: For all 3D processing, including RANSAC plane fitting and OBB calculation.
    - `numpy`: For all numerical and vector operations.
    - `pandas`: For formatting and saving the final results table.

5. Results

This section presents the results obtained from processing the depth image sequence, including the normal angle, visible area, and the estimated axis of rotation. RViz was used for visualizations. Visualizations are provided to illustrate the cuboid detection and rotation.

5.1. Normal Angle and Visible Area

The following table summarizes the calculated normal angle and visible area of the largest visible face at each timestamp.

Frame No.	Normal Angle (degrees)	Visible Area (m²)
Frame 1	125.85	2.4188
Frame 2	149.79	1.1631
Frame 3	130.94	2.2229
Frame 4	131.18	2.1985
Frame 5	115.32	6.1289
Frame 6	165.59	1.7276
Frame 7	145.86	1.8955

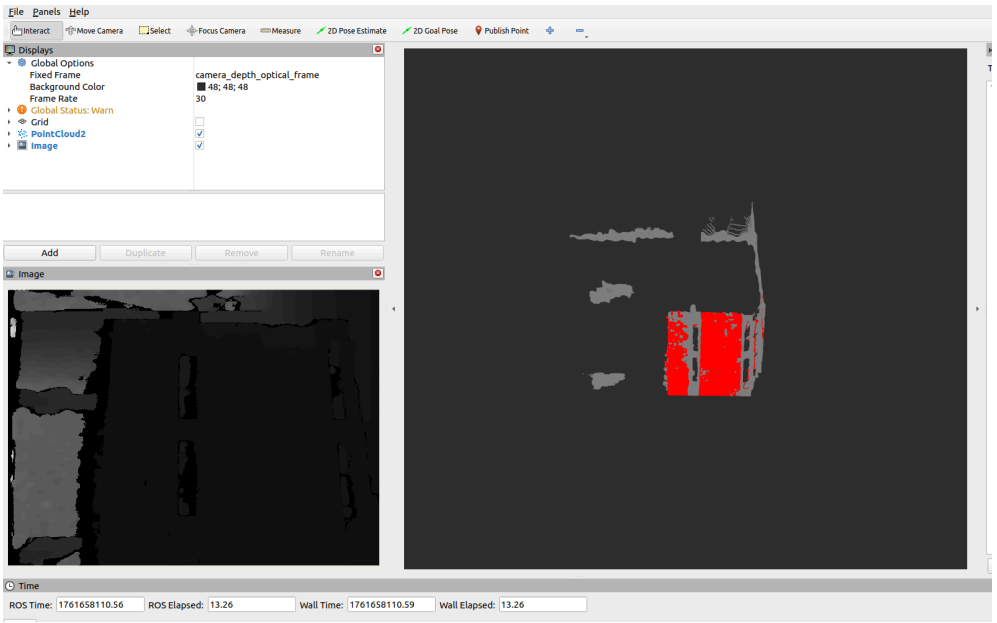


Image 1: RVIZ visualization of the detected cuboid face (highlighted in red) at a sample timestamp.

5.2. Axis of Rotation

Task 2 Result: Axis of Rotation
Axis of Rotation Vector (Direction) [X, Y, Z] = [0. 1. 0.]
Point on the Axis (Average Centroid) [X, Y, Z] = [0.0976 -0.0258 1.1951]

Note: Direction vector [0, 1, 0] is assumed based on the problem diagram (Top View) showing vertical rotation.