Depth-Based Rotation Estimation

Sankalp Arora
Perception Engineer – 10xConstruction

November 2025

1 Introduction

This report describes the methodology and algorithm used to estimate the **rotation axis of a cuboid** using depth data from a ROS2 SQLite bag file (`depth.db3`). The project implements a complete depth-processing pipeline — from extracting frames to computing plane normals — using geometric modeling and the RANSAC algorithm.

2 Problem Understanding

The objective of this project is to estimate the **rotation axis** of a cuboid using depth data recorded in a ROS2 SQLite bag file (`depth.db3`). Each frame represents the cuboid at a different orientation. The goal is to identify the dominant planar face visible in each frame and compute its geometric properties.

Given:

- A ROS2 bag file containing depth images as binary BLOBs.
- Camera intrinsics (focal lengths and principal point).

We aim to:

1. Extract all depth frames from the ROS bag.
2. Detect the largest planar face in each frame.
3. Compute each plane's normal vector, visible area, and angle with respect to the camera.
4. Estimate the global rotation axis by aggregating all detected plane normals.

3 Approach Overview

The proposed pipeline consists of several modular stages, illustrated in Table 1. Each stage performs a specific geometric or computational task, from raw depth decoding to high-level rotation axis estimation.

Stage	**Function**	**Description**
1. Data Extraction	`read_images_from_rosbag_sqlite()`	Reads serialized ROS Image messages (BLOBs) from the SQLite database.
2. Depth Conversion	`depth_to_pointcloud()`	Converts 2D depth maps into 3D point clouds using the pinhole camera model.
3. Plane Detection	`ransac_plane()`	Fits the largest planar surface using the RANSAC algorithm.
4. Plane Analysis	`process_images()`	Computes per-frame area, normal vector, and tilt angle.
5. Axis Estimation	Aggregation	Averages all plane normals to estimate the overall rotation axis.

Table 1: **Overview of the main pipeline stages**

The overall workflow is as follows:

$$\begin{aligned} \text{Depth Image} &\rightarrow \text{Point Cloud} \\ &\rightarrow \text{Plane Detection (RANSAC)} \\ &\rightarrow \text{Feature Extraction} \\ &\rightarrow \text{Rotation Axis Estimation} \end{aligned}$$

This pipeline enables a structured approach to processing depth data. By transforming depth frames into 3D representations, it becomes possible to detect geometric primitives such as planes, extract their orientation, and estimate the object's overall rotation behavior. Each stage builds upon the previous one, ensuring that noise and irrelevant features are filtered out progressively. Such modular decomposition enhances interpretability, debugging capability, and scalability of the perception system.

4 Algorithmic Details

4.1 Reading Depth Frames

Each ROS2 `sensor_msgs/Image` message is stored as a binary large object (BLOB) in the SQLite database. The serialized structure includes metadata (height, width, encoding) followed by raw pixel data.

1. Query the database: `SELECT data FROM messages ORDER BY id;`
2. Parse the byte stream to extract image height and width.
3. Convert the pixel data from unsigned 16-bit integers to a NumPy array.
4. Convert depth units from millimeters to meters.

Mathematically:

$$D_m(i,j) = \frac{D_{raw}(i,j)}{1000.0}$$

where D_m represents depth in meters and D_{raw} is the original depth in millimeters.

4.2 Depth to 3D Point Cloud Conversion

The depth image is projected into 3D space using the **pinhole camera model**. Each pixel (i,j) with depth Z corresponds to a 3D point (X,Y,Z) in the camera coordinate frame:

$$X = \frac{(j - c_x) \cdot Z}{f_x}, \qquad Y = \frac{(i - c_y) \cdot Z}{f_y}, \qquad Z = Z$$

Here, (f_x, f_y) are the focal lengths in pixels and (c_x, c_y) is the optical center (principal point). This results in a dense 3D point cloud represented as an $N \times 3$ matrix of coordinates.

In this project, the intrinsic parameters were not explicitly available from the recorded ROS bag. Hence, typical camera intrinsics were assumed as $f_x = f_y = 525.0$ pixels, which represent the approximate focal lengths of standard RGB-D sensors such as the Kinect v1 or Intel RealSense series. The principal point (c_x, c_y) was set to the image center, assuming symmetric lens calibration and negligible optical distortion. These values are sufficient to achieve geometrically accurate reconstruction for planar scenes at short to medium depth ranges.

4.3 Plane Fitting using RANSAC

To identify the dominant plane within a noisy point cloud, the **RANSAC (Random Sample Consensus)** algorithm is applied.

1. Randomly select 3 non-collinear points from the point cloud.

2. Compute the plane normal $\mathbf{n}$ using the cross product:

$$\mathbf{n} = (p_2 - p_1) \times (p_3 - p_1)$$

3. Derive the plane equation:

$$n_x x + n_y y + n_z z + d = 0$$

   where $d = -\mathbf{n} \cdot p_1$.

4. For every point p_i, compute its perpendicular distance to the plane:

$$\text{dist}(p_i) = |\mathbf{n} \cdot p_i + d|$$

5. Mark all points within a distance threshold ϵ (e.g., 0.01 m) as inliers.

6. Retain the plane that maximizes the number of inliers.

The output is the plane parameters $(\mathbf{n}, d)$ and the corresponding set of inliers.

4.4 Plane Area and Orientation

Inlier points form the visible region of the detected plane. To compute its physical area:

1. Compute the centroid of the plane:

$$\mathbf{o} = \frac{1}{N} \sum_{i=1}^{N} p_i$$

2. Project inlier points onto a local 2D coordinate system defined by orthonormal vectors $(\mathbf{u}, \mathbf{v})$ on the plane.

3. Apply the **Convex Hull** algorithm to the 2D projected points to estimate the visible surface area:

$$A = \text{ConvexHull}(\mathbf{u}, \mathbf{v}).\text{area}$$

4. Compute the plane's tilt angle relative to the camera's optical axis (Z-axis):

$$\theta = \cos^{-1}\left(\frac{\mathbf{n} \cdot \mathbf{z}}{\|\mathbf{n}\|\,\|\mathbf{z}\|}\right)$$

4.5 Rotation Axis Estimation

For each frame i, the plane normal vector $\mathbf{n}_i$ is computed. The global rotation axis is obtained by taking the normalized mean of all normal vectors:

$$\mathbf{r} = \frac{\sum_i \mathbf{n}_i}{\|\sum_i \mathbf{n}_i\|}$$

This unit vector $\mathbf{r}$ represents the estimated 3D rotation axis of the cuboid relative to the camera.

5 Implementation Details

- **Language:** Python 3.12
- **Core Libraries:** NumPy, SciPy, Matplotlib, ImageIO, Pillow
- **Input:** ROS2 SQLite Bag (`depth.db3`)
- **Output Files:**
  - `results.csv` — Per-frame plane normals, angles, and areas
  - `rotation_axis.txt` — Final estimated rotation axis vector
- **Parameters:**
  - Focal lengths: $f_x = f_y = 525.0$ (pixels)
  - Distance threshold: $\epsilon = 0.01$ m
  - RANSAC iterations: 1500