10xConstruction - 3D Cuboid Rotation Analysis

Approach and Algorithmic Methodology

Sayuj Gupta
3rd Year CSE, IIT Jammu
sayujgupta2005@gmail.com

November 29, 2025

Abstract

This report details the implementation of a simple perception pipeline designed to estimate the geometric properties and rotational dynamics of a spinning cuboid using depth sensor data. The solution employs a "geometry-first" approach, utilizing RANSAC for robust plane segmentation and Singular Value Decomposition (SVD) for rotation axis estimation. This document discusses the rationale behind selecting geometric methods over intensity-based feature matching (such as SIFT/ORB) and provides a mathematical breakdown of the implemented algorithms.

1 Introduction

The problem statement requires the analysis of a sequence of depth images containing a rotating cuboid. The core objectives are to identify the largest visible face at each timestamp, calculate its surface area and orientation relative to the camera, and derive the global axis of rotation.

Input data is provided as a ROS bag file containing depth frames encoded as 16-bit integers. Unlike RGB images, these frames provide direct metric distance information (Z), allowing for the reconstruction of the scene's 3D geometry.

2 Methodology Selection: Geometric vs. Feature-Based

During the design phase, two primary methodologies were considered based on standard Computer Vision coursework principles:

2.1 Option A: Feature-Based Matching (SIFT/ORB)

The traditional approach for tracking object pose involves detecting keypoints using algorithms like SIFT (Scale-Invariant Feature Transform) or ORB (Oriented FAST and Rotated BRIEF). By matching these descriptors across frames, one can recover the Homography or Fundamental Matrix to estimate motion.

2.2 Option B: Geometric Segmentation (Selected Approach)

The second option is to treat the input strictly as 3D spatial data. By converting the depth map into a point cloud, we can fit mathematical models (planes) directly to the structures.

2.3 Justification for Selection

I chose **Option B (Geometric Segmentation)** for this specific task for the following reasons:

- **Texture Dependence:** Feature descriptors like SIFT require surface texture (gradients) to identify keypoints. A standard "cuboid" or box often has uniform color and low texture, which leads to poor feature detection and tracking instability.
- **Direct Access to Depth:** Since we have high-fidelity depth data, inferring geometry through feature tracking (Structure from Motion) is redundant and less accurate than directly measuring it.

- **Robustness:** Plane fitting is inherently more robust to lighting changes and motion blur than pixel-level feature tracking.

3 Algorithmic Approach

The implemented pipeline follows a sequential structure: Data Acquisition $\rightarrow$ Point Cloud Reconstruction $\rightarrow$ Segmentation $\rightarrow$ Metric Extraction $\rightarrow$ Dynamics Estimation.

3.1 Data Pipeline & Preprocessing

The script reads messages from the ROS bag using the 'rosbags' library. The raw 16-bit depth data (d_{raw}) is converted to meters:

$$Z_{meters} = \frac{d_{raw}}{1000.0}$$

A pass-through filter is applied to remove background noise (e.g., walls) and invalid sensor readings (zeros), keeping points within the range $[0.1m, 3.0m]$.

3.2 3D Point Cloud Reconstruction

Using the Pinhole Camera Model, we back-project every valid pixel (u, v) with depth Z into 3D space (X, Y, Z). Assuming standard intrinsics (f_x, f_y, c_x, c_y) for the sensor:

$$X = \frac{(u - c_x) \cdot Z}{f_x}, \quad Y = \frac{(v - c_y) \cdot Z}{f_y}$$

This transforms the 2D image into a sparse cloud of 3D vectors.

3.3 Plane Segmentation via RANSAC

To identify the face of the cuboid, we employ the Random Sample Consensus (RANSAC) algorithm. This is critical because real-world depth data contains "salt-and-pepper" noise and outliers.

1. **Sampling:** Randomly select 3 non-collinear points from the cloud.
2. **Model Generation:** Compute the plane normal $\mathbf{n}$ using the cross product of the vectors formed by these points.
3. **Consensus:** Calculate the perpendicular distance of all other points to this plane. Points within a threshold $\tau = 1.5cm$ are counted as "inliers."
4. **Optimization:** Repeat for $k = 1000$ iterations. The model with the highest inlier count is selected.
5. **Refinement:** To minimize error, we re-compute the final normal $\mathbf{n}$ by performing SVD on the covariance matrix of all inlier points, not just the initial three.

3.4 Metric Computation

Normal Angle (θ): The angle between the face normal $\mathbf{n} = [n_x, n_y, n_z]$ and the camera's optical axis $\mathbf{k} = [0, 0, 1]$ is computed using the dot product:

$$\theta = \arccos\left(\frac{\mathbf{n} \cdot \mathbf{k}}{||\mathbf{n}|| \cdot ||\mathbf{k}||}\right) = \arccos(n_z)$$

Visible Area: Since depth sensors often have "holes" (invalid pixels), simply counting points underestimates the area. We project the 3D inliers onto the 2D plane defined by $\mathbf{n}$ and compute the **Convex Hull**. The area of this 2D polygon represents the physical surface area of the face.

3.5 Rotation Axis Estimation

As the box rotates, the normal vector of its face, $\mathbf{n}_t$ changes over time. These vectors trace a cone (or a great circle) in 3D space. The physical axis of rotation, $\mathbf{r}$, is the vector that stays constant relative to this motion.

Mathematically, $\mathbf{r}$ is the normal to the plane formed by the collection of face normals $\{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_T\}$. We estimate this using SVD on the matrix of normals: 1. Center the normals: $M = [\mathbf{n}_1 - \bar{\mathbf{n}}, \ldots, \mathbf{n}_T - \bar{\mathbf{n}}]^T$. 2. Compute SVD: $M = U\Sigma V^T$. 3. The rotation axis $\mathbf{r}$ corresponds to the column of V associated with the smallest singular value (the direction of least variance).

4 Implementation Details

The solution is implemented in Python 3. Key libraries include:

- **NumPy:** For vectorized linear algebra operations (projections, dot products).
- **SciPy (Spatial):** For the 'ConvexHull' algorithm used in area calculation.
- **Rosbags:** For parsing the '.bag' file without a native ROS environment.

The code is optimized for performance by avoiding Python loops for pixel-level operations, instead using NumPy broadcasting for simultaneous projection of the entire depth frame.

5 Results

The algorithm processes the sequence effectively, generating a stability table for the rotating face.

Frame ID	Normal Angle ($^\circ$)	Visible Area (m^2)
1	67.88	0.7111
2	16.98	0.7707
3	39.02	0.8382
4	58.35	0.7370
5	34.02	0.5922
6	52.62	0.8480
7	53.39	1.0362

Table 1: Output Metrics

The estimated axis of rotation vector was calculated as $[0.9993, -0.0352, -0.0127]$, indicating a compound rotation relative to the camera frame.

6 Conclusion

The proposed geometric approach successfully extracts the required physical parameters from the depth stream. By leveraging RANSAC and Convex Hull algorithms, the system remains robust to sensor noise and data sparsity. The use of SVD for axis estimation provides a mathematically rigorous solution that averages out frame-to-frame jitter, resulting in a stable estimation of the cuboid's dynamics.