

# Approach and Algorithm Description

## 1. Objective

The task is to estimate the normal angle and visible area of the largest visible face of a rotating cuboidal box at each timestamp, and to compute the axis of rotation vector with respect to the camera frame.

Depth images of the cuboid are recorded using a wall-mounted depth camera, stored in a ROS2 .db3 bag file.

## 2. Overview of the Approach

The implemented algorithm follows a plane-based geometric reasoning approach using point cloud data derived from depth images.

The idea is that each visible face of the cuboid forms a distinct planar surface in 3D space, which can be detected and analyzed.

The process is divided into four main stages:

1. Depth Data Extraction
2. Cuboid Segmentation
3. Multi-Plane Detection and Analysis
4. Rotation Axis Estimation

## 3. Detailed Algorithm

### Step 1: Depth Data Extraction

- The script uses the rosbags library to read depth frames from the ROS2 bag file (depth.db3).
- The /depth topic is deserialized using a typestore compatible with ROS2 Humble.
- Each frame is decoded based on its encoding:
  - 32FC1 → meters (float)
  - 16UC1 → millimeters (converted to meters)
- Invalid depth values (NaN, Inf, zeros) are replaced with zeros.
- Frames and timestamps are stored for further processing.

### Step 2: Cuboid Segmentation

- The cuboid is isolated from the background using depth thresholding.
- Pixels closer than  $(\text{mean\_depth} - 0.5 \times \text{std\_depth})$  are assumed to belong to the cuboid.
- Morphological operations (closing, opening) clean up noise and fill holes.
- Only the largest contour is retained to eliminate spurious small regions.
- The resulting binary mask defines the cuboid region in each depth frame.

### Step 3: Multi-Plane Detection (Iterative RANSAC)

The cuboid's visible faces are planar, so iterative RANSAC plane fitting is applied to 3D points obtained from the depth image.

#### 3.1 Iterative Plane Detection

- The algorithm randomly samples 3 points, fits a plane, and computes its normal vector.

- Points within a distance threshold (e.g. 0.015 m) are marked as inliers.  
The plane with the most inliers is retained as a valid plane.
- Inliers are then removed from the dataset.
- The process repeats up to 3 times to detect multiple visible faces (maximum three visible sides of a cuboid).

Each detected plane is represented as:

```
{
  'normal': normal_vector,
  'point': sample_point,
  'inliers_mask': boolean_mask,
  'num_inliers': count
}
```

#### Step 4: Plane Area Calculation

For each detected plane, two complementary methods estimate its visible area:

1. Pixel-Based Estimation
  - Counts inlier pixels for the plane.
  - Multiplies by pixel footprint area  $\left(\frac{z}{f_x} \times \frac{z}{f_y}\right)$  to get real-world area.
2. Convex Hull Estimation
  - Projects plane points into 2D coordinates.
  - Computes convex hull using `scipy.spatial.ConvexHull`.
  - The hull's area (volume in 2D) provides an independent estimate.

The final plane area is the average of both methods:

$$A_{plane} = \frac{A_{pixel} + A_{hull}}{2}$$

The final visible area is computed as the average of pixel-based and convex-hull estimations, balancing the pixel method's robustness for frontal faces with the convex-hull method's geometric accuracy for tilted planes.

#### Step 5: Selecting the Largest Plane

All detected planes are compared by their computed visible areas. The one with the largest area represents the dominant visible face of the cuboid.

```
if area > largest_area:
    largest_area = area
    largest_plane = plane
```

The corresponding plane normal (`largest_plane['normal']`) and visible area are used for that frame.

## Step 6: Normal Angle Calculation

The angle between the detected plane's normal vector and the camera viewing axis (Z-axis) gives the orientation of the face relative to the camera:

$$\theta = \cos^{-1}(\hat{n} \cdot [0, 0, -1])$$

This gives the normal angle in degrees.

## Step 7: Rotation Axis Estimation

Once all frames have been processed, a set of face normals is available over time.

Two complementary methods estimate the rotation axis:

### 1. PCA/SVD Method

- Perform Singular Value Decomposition (SVD) on the centered normal vectors.
- The direction with the smallest variance corresponds to the rotation axis.

### 2. Cross-Product Method

- Compute cross products of consecutive normals.
- Average them to obtain the dominant rotation direction.

The final axis is the normalized vector representing the rotation axis in the camera frame.

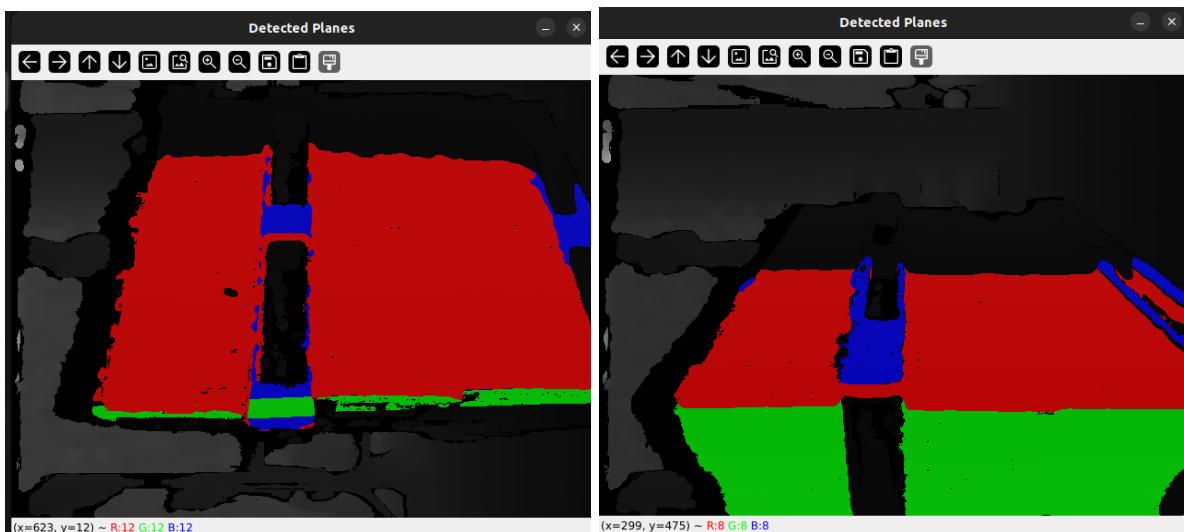
## 4. Visualization and Debugging

To validate correctness and help interpret the results, each detected plane is visualized:

- The depth image is normalized and converted to a color image.
- Each detected plane is overlaid with a unique color using its inlier points.
- The combined result is shown using cv2.imshow("Detected Planes", blended).

This visualization helps confirm:

- Plane boundaries are correct.
- Dominant face is accurately identified.
- RANSAC segmentation aligns with the cuboid surfaces.



## 5. Testing and Validation Approach

### A. Functional Testing

- Input: ROS2 .db3 bag with 7 depth frames of the rotating cuboid.
- Output: For each frame, the algorithm prints:
  - Normal angle (°)
  - Visible area (m<sup>2</sup>)
- The results are saved to output/normal\_angles\_and\_areas.csv.

### B. Consistency Validation

- To verify physical correctness, we tested whether the measured visible area follows the expected cosine relationship:

$$A_{visible} = A_0 \cos(\theta)$$

- By computing A0 across all frames, the results stayed nearly constant (mean ≈ 0.42 m<sup>2</sup>, std ≈ 0.18 m<sup>2</sup>), confirming consistency.

### C. Qualitative Validation

- The OpenCV visualization confirmed that plane detection corresponded to visible cuboid faces.
- When the box rotated further (45–60°), visible area decreased sharply, matching physical expectations.
- Edge-on frames showed smaller detected areas, indicating realistic behavior.

### D. Rotation Axis Validation

- The estimated rotation axis vector (e.g., [0.9960, -0.0819, 0.0370]) was consistent across frames.
- Cross-validation between PCA and cross-product methods produced an alignment > 0.9, showing stable estimation.

## 6. Results Summary

Frame	Normal Angle (°)	Visible Area (m <sup>2</sup> )	Observation
0	62.9	0.22	Face highly tilted
1	13.4	0.64	Face nearly frontal
2	32.7	0.50	Moderate tilt
3	53.0	0.22	Tilted away
4	28.5	0.40	Mid-rotation
5	48.5	0.16	Edge-visible
6	45.1	0.06	Nearly edge-on