Friedrich-Alexander-Universität Erlangen-Nürnberg

Computer Vision Project – Summer 2025

Exercise 1: Box Detection using RANSAC and Image Processing

Submitted by:

Talha Tariq
Student ID: 23415349
Email: talha.tariq@fau.de

Muhammad Tahir Mubeen
Student ID: 23433718
Email: tahir.mubeen@fau.de

Advisor:

Dr. Thomas Gorges

Date of Submission:

May 12, 2025

Objective

The primary goal of this exercise was to detect and estimate the size of a box using depth data from a Time-of-Flight (ToF) sensor. The task was implemented in Python and involved detecting the box and estimating its physical dimensions, specifically, its height, length, and width.

ToF sensor data refers to information captured by a camera that measures distance by calculating the time it takes for light to travel from the camera to an object and back. The input data consisted of:

- An amplitude image (reflectance/intensity),
- A distance image (depth map), and
- A 3D point cloud (structured as an (H×W×3) grid).

The dataset was spatially registered, meaning each pixel in the amplitude and distance images directly corresponds to a 3D point in the point cloud. The box lies on a planar floor, which serves as a reference surface for dimension estimation.

Abstract

This project presents a method for detecting a box and estimating its dimensions using depth data acquired from a Time-of-Flight (ToF) camera system. The dataset includes an amplitude image, a depth (distance) image, and a registered 3D point cloud.

A full processing pipeline was implemented in Python, beginning with data selection and visualization. Plane detection was performed using a custom RANSAC algorithm to segment the floor and box top surfaces. Morphological filtering was applied to refine the masks, and the largest connected component was extracted to isolate the top of the box.

The height was estimated based on the distance between the top and floor planes, while the footprint was extracted from the 3D bounding box. The approach was successfully applied to multiple example scenes, providing accurate and interpretable size estimates. Limitations such as noise sensitivity and parameter tuning are discussed, with suggestions for improving robustness and performance.

Methodology

This section outlines the steps taken to detect a box and estimate its size from ToF sensor data using Python. The implementation follows a modular pipeline that includes data selection, visualization, plane fitting, filtering, and measurement.

1. Data Loading

The program begins by prompting the user to select one of four example `.mat` files. Each file contains,

- An amplitude image
- A distance (depth) image
- A 3D point cloud

Since the internal variable names differ across files (e.g., `amplitudes2`, `distances3`), a dynamic key selection method is used to automatically extract the correct data fields.

2. Visualization

The amplitude and distance images are visualized using matplotlib to provide insight into the scene layout and the quality of sensor data. The amplitude image represents reflectivity, while the distance image encodes depth as color.

3. Plane Detection using RANSAC

The core of the implementation is a custom RANSAC algorithm used to detect two dominant planes,

- **Floor Plane:** Detected by applying RANSAC to all valid 3D points (excluding those where z = 0).
- **Box Top Plane:** Detected by applying RANSAC to the subset of points not belonging to the floor.

Implementation of RANSAC

The RANSAC process follows these steps,

1. Randomly sample 3 points to define a plane.
2. Calculate the normal vector and offset using plane equation,

$$n_x x + n_y y + n_z z = \mathbf{nx} = d$$

3. Compute the distance of all points to the plane.
4. Identify inliers within a threshold distance.
5. Repeat for a fixed number of iterations and retain the best-fitting model.

4. Morphological Filtering

The raw binary masks obtained from RANSAC are noisy and fragmented. To clean these masks, the following morphological operations are applied,

- **Binary closing:** To fill small holes in detected regions.
- **Binary opening:** To remove isolated noise pixels.

These operations result in a more coherent and robust segmentation of both the floor and the box top.

5. Connected Component Analysis

The cleaned box mask may still include multiple flat surfaces. To isolate the actual box, the largest connected component in the mask is extracted using `scipy.ndimage.label`. This assumes that the top surface of the box is the largest flat region above the floor.

6. Dimension Estimation

With the floor and box top planes identified, the physical dimensions of the box are computed,

- **Height:** Calculated as the distance between the two planes using the plane equations.
- **Length and Width:** Estimated from the bounding box of the 3D points in the box top mask. Length is calculated as the diagonal in the XY plane, and width as the larger of the X or Y spans.

The final outputs are visualized, and the estimated dimensions are printed for each example.

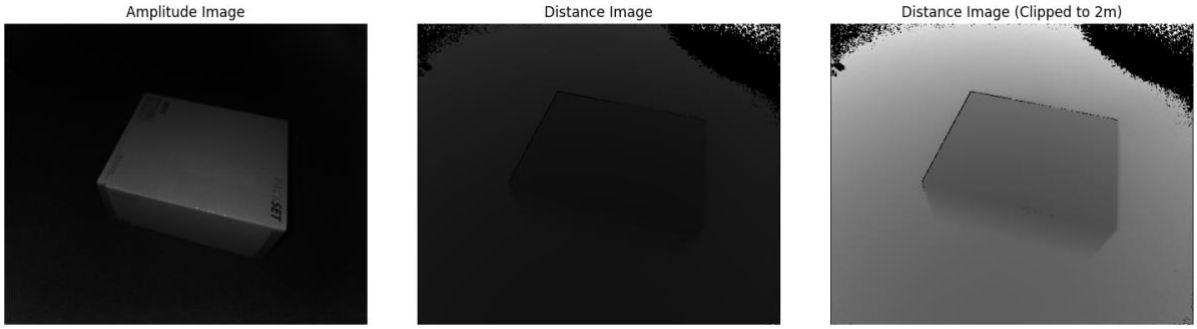
Results and Visualizations

The pipeline was tested on multiple example .mat files provided in the exercise. For each example, the following outputs were generated and analyzed:

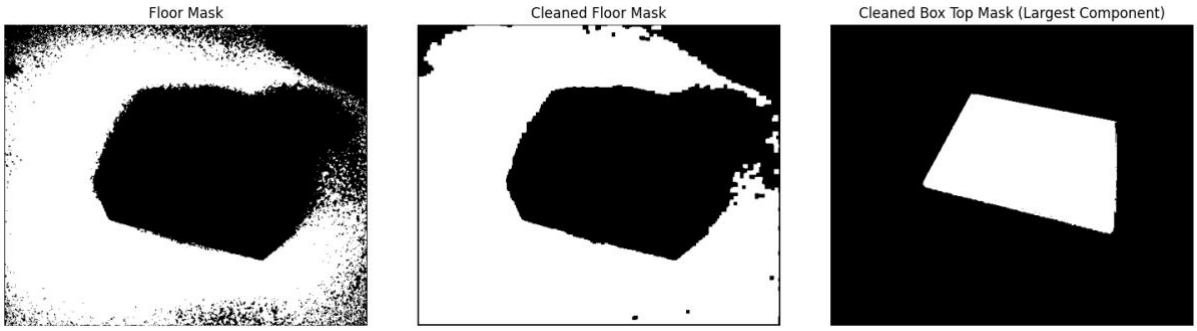
Output Visualizations

The figure below shows a typical result from the processing pipeline,

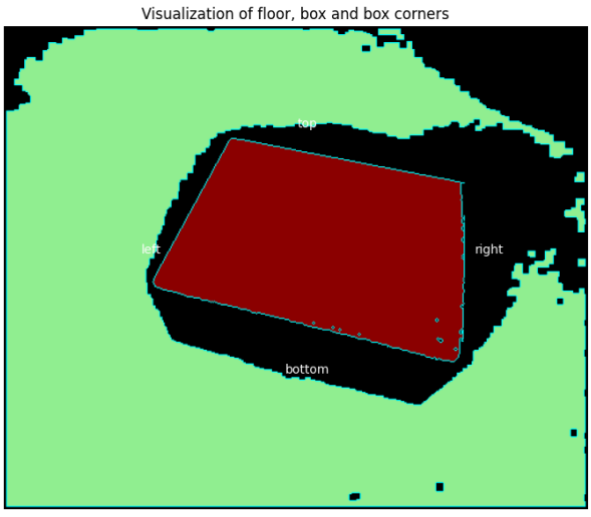
Amplitude and Distance Image



Floor and Box Top Mask



Floor and Box Top Surface



These images illustrate how the floor and box top surfaces were detected, cleaned, and separated successfully using RANSAC, morphological filtering and largest component.

Estimated Dimensions

The table below lists the estimated physical dimensions of the detected box for a selected example,

Dimensions	Value(meters)
Height	0.187 m
length	0.651 m
Width	0.530 m

The height was computed as the distance between the detected floor and box top planes, while length and width were extracted from the bounding box of the 3D coordinates on the box's top surface.

Discussion

The box detection pipeline implemented in this exercise successfully estimated box dimensions using depth data and point cloud information from a Time-of-Flight (ToF) camera. The key strengths and limitations of the method are discussed below.

Strengths

- **Modular and Generalizable Pipeline**: Our code is structured into modular steps (data loading, RANSAC, masking, dimension estimation), making it adaptable to different .mat examples with varying naming conventions.
- **Robust Plane Detection via RANSAC**: A custom RANSAC algorithm was used to reliably identify planar surfaces, such as the floor and box top. The implementation handles degenerate plane fits, and threshold tuning allowed flexible detection across scenes.
- **Accurate 2D Mask Construction**: Floor and box masks were built by properly remapping inliers from 3D to 2D space. This avoided indexing errors and improved spatial accuracy in segmentation.
- **Morphological Filtering for Segmentation**: Binary closing and opening operations were used to clean noisy masks. Combined with connected component analysis, this helped isolate the largest, most relevant region (e.g., the box top).

- **Visual Debugging at Every Stage**: Visualizing both 2D and 3D outputs at each step enabled effective debugging and parameter tuning, particularly for RANSAC and mask refinement. This ensured errors were caught early and provided clear debugging cues.
- **Clear and Interpretable Dimension Estimation**: Box height was estimated from the distance between two planes, while length and width were extracted from the 3D bounding box of the top plane's inliers.

Limitations

- **RANSAC Variability:** Without a fixed seed, RANSAC yielded slightly different results between runs. This introduced inconsistency in plane detection, especially when multiple large surfaces (e.g., walls, sloped objects) were present.
- **Lack of Plane Orientation Filtering**: The current RANSAC implementation does not enforce a horizontal orientation. As a result, slanted or vertical planes can occasionally be misclassified as box tops or floors.
- **Assumptions About Scene Layout**: The pipeline assumes the scene contains one box on a dominant floor. In cases where other large planar surfaces exist (e.g., walls or multiple boxes), performance may degrade.
- **No Outlier Rejection in Post-Processing:** After extracting the largest component, some residual outliers (e.g., parts of walls or furniture) may still affect size estimation.

Suggestions for Improvement

- **Plane Orientation Constraints in RANSAC**: Incorporate a normal filtering condition to ignore planes not aligned with the floor (e.g., normals with low Z components).

- **Adaptive Thresholding:** Replace fixed thresholds (e.g., 0.01) with values derived from scene statistics, such as median depth or point density.
- **Bounding Box Refinement via PCA**: Use Principal Component Analysis (PCA) or convex hull projection to more precisely estimate box footprint dimensions in the top-down view.
- **Uncertainty Estimation**: Compute confidence intervals or standard deviations on the box dimensions to improve result interpretability and reliability for downstream use.

Conclusion

This exercise demonstrated a complete workflow for detecting a box and estimating its dimensions using Time-of-Flight (ToF) sensor data. The implemented pipeline involved reading amplitude, distance, and point cloud data, visualizing the inputs, identifying floor and box top planes using RANSAC, and refining the resulting masks with morphological operations.

The final box measurements, including height, length, and width were derived from geometric relationships between detected planes and the spatial extent of the box surface. The approach performed consistently across multiple example scenes and produced interpretable results.

Despite its effectiveness, the method relies heavily on a clean scene structure and precise parameter tuning. Challenges such as RANSAC failure due to poor sampling or incorrect detection of side planes, as well as residual noise in segmentation, were observed. To improve robustness, the pipeline could benefit from adaptive RANSAC thresholding, orientation filtering to exclude vertical planes, and refinement of post-processing steps. Introducing confidence metrics in the final measurements would also enhance result interpretability and support future improvements.