Technical Report

Yasmina

Fall Semester 2025

1 Problem Solving Approach

The goal of this project was to develop a cinematic navigation system for 3D environments represented as point clouds. My solution followed a pipeline approach:

1.1 Point Cloud Loading

I used Open3D to read .ply files and analyze the point distribution.

1.2 Floor Plane Detection

I applied RANSAC plane segmentation to detect the floor. I specifically checked that the detected plane was mostly horizontal, rejecting planes that did not match the expected orientation.

1.3 Occupancy Grid Construction

The 3D point cloud was projected onto a 2D occupancy grid. Cells were marked as free or blocked based on floor support points and vertical obstacle thresholds. Small morphological operations were applied to smooth walkable areas.

1.4 Path Planning

- **Search:** I implemented a modified A* algorithm that takes distance to obstacles into account using a distance transform, which encourages paths to stay further from walls.
- **Path Pruning & Smoothing:** Collinear points were removed and a Laplacian smoothing was applied to reduce zigzag motions.

1.5 Trajectory Generation

Cubic spline interpolation generated smooth 3D trajectories for camera movement. Look-at points were computed slightly ahead along the path for cinematic effect.

1.6 Rendering

- Sequential rendering via Open3D for testing or fallback.
- Distributed rendering using Apache Spark for faster processing of long trajectories.

2 Highlights and Novel Tricks

- **Automatic Start/Goal Selection:** The system automatically chooses two distant points in the largest free area, avoiding manual selection.
- **Obstacle-aware A*:** The algorithm penalizes proximity to walls using a distance transform, resulting in more "human-like" navigation.
- **Axis Auto-Alignment:** The point cloud axes are automatically permuted so the vertical axis aligns with the real-world Z-axis.
- **Distributed Rendering with Spark:** Each frame is rendered independently, allowing massive parallelization and significantly reducing rendering time.

3 Challenges Faced and Solutions

- **Irregular Point Clouds:** Some point clouds were rotated or scaled strangely. *Solution:* Implemented `auto_fix_axes` to detect and align axes automatically.
- **Sparse or Noisy Floor Points:** Floor detection sometimes failed with noisy data. *Solution:* Added thresholds for RANSAC inliers and a verification step on plane orientation.
- **Path Outside Occupancy Grid:** Start or goal points occasionally fell outside free space.
  Solution: Implemented `find_nearest_free` to safely move points to the nearest walkable cell.
- **Memory Usage in Spark:** Large point clouds caused memory spikes.
  Solution: Optimized frame serialization using pickle + base64 to reduce Spark data transfer overhead.

4 Results and Evaluation

- **Accuracy:** The pipeline reliably finds walkable paths in various indoor scenes.
- **Smoothness:** Camera trajectories are smooth and visually appealing, thanks to cubic spline interpolation.
- **Performance:**
  - Sequential rendering: suitable for small point clouds.
  - Spark distributed rendering: significantly faster for hundreds of frames (speedup ~5–10x depending on CPU cores).

- **Sample output:** `outputs/scene_1/spark_tour.mp4` demonstrates the cinematic navigation along a hall with minimal jitter and natural camera movement.

5 Vision for Future Improvements

5.1 Technical Enhancements

- **Multi-level Navigation:** Extend occupancy grids to support multiple floors or ramps.
- **Dynamic Obstacle Avoidance:** Integrate 3D semantic segmentation to identify movable obstacles.
- **GPU-Accelerated Path Planning:** Offload A* and smoothing computations to GPU.
- **Advanced Camera Behavior:** Introduce easing curves for camera speed and add cinematic effects like roll, tilt, or dolly zoom.
- **Interactive Start/Goal Adjustment:** Add a lightweight GUI for tweaking automatically selected points or previewing paths.

5.2 Creative Directions

- **Story-driven Camera Tours:** Generate narrative-driven sequences.
- **Integration with VR/AR:** Render real-time navigable environments for immersive exploration.
- **Procedural Scene Analysis:** Automatically identify furniture, doors, or artwork and plan a path that highlights these features.
- **AI-driven Look-at Target Selection:** Dynamically focus the camera on points of interest.