Technical Report: Autonomous Cinematic Navigation Agent

1. Introduction

The objective of this assignment was to build an autonomous pipeline capable of ingesting raw 3D Gaussian Splatting (3DGS) data—unstructured point clouds with no explicit mesh topology—and generating smooth, professional cinematic tours. The system needed to solve challenges related to collision avoidance, scene understanding, and photorealistic rendering without human intervention.

2. Problem Solution & Methodology

2.1 The Voxel Abstraction

Gaussian splats are mathematically defined as "soft" clouds. To enable rigid physics simulation, I abstracted the scene into a **Binary Voxel Grid**.

- **Implementation:** The SceneMap class filters points below an opacity threshold (0.2) and maps the remaining "solid" points to discrete spatial bins (default 0.5m size).
- **Benefit:** This converts a probabilistic cloud into a deterministic grid, allowing the use of graph traversal algorithms (O(1) lookups).

2.2 The "Vertical Clamp" Planner (Novelty)

A standard A* algorithm often fails in scanned buildings because it tries to "shortcut" over the roof or under the floor. I solved this with three distinct algorithmic tricks:

1. **Vertical Clamping:** I calculate the 5th and 90th height percentiles of the point cloud. The cost map sets any voxel outside this vertical range to infinity, effectively creating an artificial "lid" and "foundation" for the building.
2. **Vertical Sandwich Start:** To ensure the camera starts indoors, the agent doesn't just pick the geometric center. Instead, it scans the vertical column at the center (x, z) and identifies the largest span of empty voxels (the gap between floor and ceiling) to place the start point.
3. **EDT Cost Field:** I utilized scipy.ndimage.distance_transform_edt to generate a repulsive field from walls. The cost function $Cost = 1 + \frac{10}{Distance}$ ensures the camera prefers the center of hallways rather than hugging walls.

2.3 Cinematic Smoothing

Raw pathfinding yields jagged, Manhattan-style movements. To achieve a cinematic look:

- **Spatial:** I used **Cubic Splines** to interpolate the A* control points.
- **Rotational:** I calculated view tangents and used **SLERP** (Spherical Linear Interpolation)

to ensure smooth camera pans.
- **Temporal:** I applied an ease-in/ease-out timing function to the spline sampling to avoid robotic constant-velocity movement.

3. Challenges Faced & Solutions

Challenge	Solution
"Floater" Artifacts	Scans often have noise floating in the sky. I implemented **Percentile Bounding (1%-99%)** instead of Min/Max to ignore these outliers when calculating the scene scale.
Escaping the Building	The agent kept flying out of windows or roofs. I implemented the **Vertical Clamping** logic (described in 2.2) and a specific check in path_planner.py (_has_ceiling) to penalize nodes that have direct line-of-sight to the sky.
Data Inconsistencies	Different PLY exporters store opacity as either probabilities (0-1) or logits (-inf to +inf). My SceneMap class performs a heuristic check and automatically applies a **Sigmoid activation** if logits are detected.

4. Results and Evaluation

- **Robustness:** The system successfully handles indoor environments (e.g., ConferenceHall) by utilizing the "Explorer" logic, and automatically switches to "Orbit" logic for small, dense object scenes based on the max_dimension check.
- **Video Output:** The generated indoor_original.mp4 shows stable navigation through the conference hall, adhering to the "Vertical Clamp" limits.
- **Detection:** The integrated YOLOv8 module successfully identifies objects (chairs, sofas) in the rendered video, overlaying bounding boxes in indoor_detection.mp4.

5. Vision for Future Improvements

Technical Enhancements

1. **True 3D Localization (Ray-Casting):** Currently, the detection is 2D only. A major improvement would be casting rays from the camera center through the detection box

   center into the voxel grid to determine the precise (x,y,z) coordinate of the object.
2. **Dynamic Octree:** Replacing the fixed-size voxel grid with a Sparse Octree would allow for finer collision resolution near complex objects while saving memory in empty spaces.
3. **Real-Time Preview:** Integrating a lightweight web-based viewer (using WebGL) to preview the path before committing to the heavy CUDA render.

Creative Directions

1. **Semantic Navigation:** Instead of random exploration, allow users to input text prompts (e.g., "Go to the podium"). By integrating CLIP with the rendered frames, the agent could guide the path generation toward semantically relevant areas.
2. **Music Reactivity:** The path speed could be modulated by an audio track, moving faster during high-tempo segments and slowing down for dramatic moments.