Technical Report: Autonomous Cinematic Navigation Agent

1. Introduction

The objective of this assignment was to build an autonomous agent capable of ingesting raw 3D Gaussian Splatting (3DGS) scenes and generating professional, cinematic videos. The core challenge lies in the nature of 3DGS data: unlike meshes, Gaussian Splats are unstructured point clouds with no explicit surface definition, making physics and navigation non-trivial.

2. Problem Solution Strategy

2.1 The "Voxel Map" Abstraction

Navigation directly on millions of Gaussian points is computationally infeasible. My solution abstracts the scene into a **Binary Voxel Grid** (0.5m resolution).

- **Why?** A grid allows for $O(1)$ collision lookups and enables standard graph algorithms like BFS and A*.
- **Robustness:** By using percentile-based bounding boxes (1st-99th) and opacity filtering, the system ignores "floater" artifacts that often appear in raw scans.

2.2 The "Max-Clearance" Path Planner (Novelty)

A standard A* algorithm seeks the shortest path, which often results in the camera hugging walls or slipping through cracks (windows/holes) to take shortcuts outside the building.

The Solution: I implemented a **Euclidean Distance Transform (EDT)** based planner.

1. **Physics Field:** We compute a 3D field where every voxel knows its distance to the nearest wall.
2. **Inverse Cost Function:** The movement cost is defined as $C = 1 + \frac{W}{D^2}$, where D is the distance to the nearest obstacle.
3. **Result:** This creates a "gravity" that pulls the camera toward the center of rooms and corridors. Small gaps (like windows) become mathematically "expensive" to traverse, effectively sealing the building without manual mesh editing.

2.3 Cinematic Smoothing

Raw A* paths are jagged (Manhattan motion). To achieve a "drone-like" feel:

- **Subsampling:** We only sample the A* path every ~2-5 meters.
- **Cubic Splines:** We fit a scipy.interpolate.CubicSpline through these control points.
- **Ease-In/Ease-Out:** Time parameterization is mapped via a Smoothstep function ($3t^2 - 2t^3$) to eliminate robotic constant-velocity starts and stops.

3. Computer Vision Pipeline (Bonus)

I integrated a hybrid 2D-3D object detection system without requiring depth buffers.

- **2D Detection:** YOLOv8 runs on the rendered RGB frames.
- **3D Localization (The Trick):** Instead of complex depth estimation, we reuse the navigation **Voxel Map**. We cast a ray from the camera pose through the detected pixel. The ray marches through the voxel grid until it hits an Occupied voxel. This returns the precise 3D world coordinate of the object.

4. Challenges & Solutions

Challenge	Solution
"Static Noise" Rendering	The raw PLY files stored data in non-standard log/logit scales. I applied exp() and sigmoid() activations and enforced sorted loading of SH coefficients to fix the visual artifacts.
Agent Escaping Building	The agent kept finding "shortcuts" through windows. I implemented the **Max-Clearance** cost function (described above) and a **Vertical Clamp** (ignoring space above the roof) to force interior exploration.
Jerky Movement	A* produces zigzag paths. I implemented path subsampling and SLERP (Spherical Linear Interpolation) for rotations to ensure smooth camera pans.

5. Results and Evaluation

- **Success Rate:** The agent successfully navigates both small object scenes (switching to "Orbit Mode") and large interior scenes (switching to "Explorer Mode").
- **Quality:** The resulting video (trajectory_clean.mp4) demonstrates smooth, collision-free motion that traverses the major axis (Diameter) of the environment.
- **Detection:** The system accurately labels objects (e.g., "chair", "person") and places 3D coordinate labels floating in space.

6. Vision for Future Improvements

Technical Enhancements

1. **Dynamic Depth-Based Collision:** Currently, collision is pre-computed. In the future, I would render a low-res Depth Buffer in real-time to allow the agent to react to dynamic obstacles or higher-detail geometry that the coarse voxel grid missed.
2. **Semantic Navigation:** Instead of exploring random "farthest points," we could use CLIP-based semantic query. The user could type "Go to the kitchen," and the agent would guide the camera toward the semantic center of that area.

Creative Directions

1. **Style Transfer:** Apply a "Cinematic LUT" or Neural Style Transfer to the rendered frames to simulate film stock (e.g., Kodak Portra 400).
2. **Director Mode:** Allow the user to define key "Interest Points" manually, and use the path planner to optimally string them together into a coherent story.