

École Polytechnique Fédérale de Lausanne
Electrical and Electronics Engineering

Instruction-Driven 6-DoF Navigation with General-Purpose Large
Language Models

by Aristotelis Dimitriou

Master Thesis

Approved by the Examining Committee:

Prof. Pascal Fua
Thesis Advisor

Richard Lengagne
External Expert

Leslie (Shixuan) Gu
Thesis Supervisor

Visual Computing Group
Harvard John A. Paulson School Of Engineering And Applied Sciences
Science and Engineering Complex
150 Western Ave, Boston, MA 02134

August 28, 2025

Abstract

This work investigates the use of general-purpose Large Language Models (LLMs) as zero-shot agents for continuous 6-DoF navigation. We design a pipeline that combines Chain-of-Thought (CoT) prompting, geometric cues (depth and Bird’s Eye View), and overlays to guide an LLM in generating trajectories. The system is evaluated in two pipelines: 3D Gaussian Splatting (3DGS) and diffusion-based novel view synthesis, across three real-world scenes requiring compound motions.

Results show that **CoT reasoning is the decisive factor**, raising success rates from **7% to 47%** in 3DGS, while geometric cues provide stabilization and efficiency when combined with CoT. Diffusion-based navigation follows the same trend but remains less reliable due to noisier renderings. Ablation studies further reveal the **fatal impact of motion convention violations** and highlight reasoning errors as the dominant failure mode.

To our knowledge, this is among the **first systematic studies of LLM-driven navigation under continuous control without task-specific training**. The proposed pipeline thus serves both as a proof of concept and as a baseline for future research, underscoring the importance of reasoning mechanisms and structured geometric support for embodied planning with general-purpose models.

Keywords: Large Language Models, Vision-Language Navigation, Prompting, Novel View Synthesis, 3D Gaussian Splatting

Code: <https://github.com/aristo6253/Trajectory-GPT>

Contents

Abstract	2
1 Introduction	5
2 Background	7
2.1 Novel View Synthesis	7
2.1.1 Video Diffusion Models	7
2.1.2 3D Gaussian Splatting (3DGS)	8
2.2 Camera Pose Representation	9
2.3 Point Cloud Reconstruction	9
2.4 Multimodal Language Models and GPT-4o	10
3 Related Work	12
3.1 Language-Guided Trajectory Generation in 3D Environments	12
3.2 Vision-Language Models and 3D Spatial Understanding	13
3.3 Trajectory Planning for Novel View Synthesis	13
3.4 Comparison and Limitations of Prior Work	14
4 Methodology	15
4.1 Pipeline Overview	15
4.2 Input Modalities	17
4.3 6-DoF Planning with GPT-4o	19
4.4 Scene Representation and Update	21
4.5 Rendering Module	22
4.6 Other Implementation Details	24
5 Evaluation	27
5.1 Experimental Setup	27
5.2 Evaluation Metrics	29
5.3 Baselines	29
5.4 Cross Overlay Study	30
5.5 Ablation Study	31

6 Discussion and Conclusion	35
Bibliography	37
A Diffusion Quality Examples	42
B Examples of SU, Reasoning, Violations, Success	45
B.1 Reasoning Error Example	45
B.2 Violation Example	46
B.3 Spatial Understanding Error Example	47
B.4 Success Example	48
C Improvement of Exploratory Consistency through CUT3R	50
D CLIP/SAM Module	52
E Full prompt	54

Chapter 1

Introduction

Intuitive human-agent interaction in visually complex environments remains a central challenge in agentic AI and robotics. Natural language offers a flexible and intuitive way for providing instructions, yet grounding such commands in a 3D world is far from solved. Applications such as indoor navigation, household robotics, and autonomous scene exploration demand models that can interpret and execute language-based instructions under 3D geometric and perceptual constraints.

Despite progress in vision-language navigation (VLN) and novel view synthesis (NVS), current approaches remain limited in their treatment of spatial reasoning. Systems such as NavGPT [43] reduce navigation to discrete action selection and often bypass direct 3D understanding through intermediate language representations. Other methods, such as Director3D [23] and NavigateDiff [27], generate trajectories or views from text but either lack real geometry or assume access to full representations of the scene. Most existing models are restricted to simplified motion regimes (e.g. 3 degrees of freedom (DoF)) or assume complete scene knowledge in advance. Diffusion-based view synthesis methods, in turn, typically rely on scripted trajectories or target-centric motion, without adapting to open-ended natural language instructions. To our knowledge, no prior work addresses iterative 6-DoF navigation driven by a black-box multimodal agent in a continuous setting.

This thesis proposes a pipeline in which a general-purpose multimodal LLM (GPT-4o [25]) serves as a high-level planner for continuous 6-DoF trajectories from natural language prompts. The trajectories are not planned in advance but emerge iteratively: at each step, GPT-4o [25] outputs a pose increment conditioned on the current RGB view, the instruction, and geometric cues. The updated pose generates a new view, which in turn refreshes the inputs for the next step. This process continues until the agent terminates the task. The approach is modular, unsupervised, and adaptable to different rendering backends.

We implement two pipeline variants: one diffusion-based and the other employing 3D Gaussian Splatting.

In the **diffusion-based pipeline**, CUT3R [35] reconstructs a point cloud and depth map from the initial image. From this, a Bird’s Eye View (BEV) obstacle map and depth cues are derived at each step. A modified ViewCrafter [41] module then synthesizes the new view from the predicted 6-DoF pose, with cubic-spline interpolation ensuring smooth playback. CUT3R [35] also updates the scene model at each step, enabling spatial continuity. In the **3D Gaussian Splatting (3DGS) pipeline**, the planner operates directly on a pre-trained 3DGS model [19]. Depth and BEV projections are rendered from the Gaussian splats, bypassing iterative reconstruction. This provides complete geometry upfront but yields black regions where coverage is missing.

The generated trajectories serve both as (i) a mechanism for NVS under language guidance and (ii) a framework for testing the spatial reasoning of general-purpose multimodal agents. To evaluate the system, we combine **quantitative metrics**, trajectory length (TL), step size, success rate (SR), success weighted by path length (SPL), reasoning success (RS), spatial understanding (SU), and violation rate (VR), with **qualitative analyses** of behavioral patterns and failure modes. This allows us to isolate the contributions of reasoning, geometric cues, and lightweight visual overlays.

Direct baselines are scarce, as prior work typically assumes discrete action spaces, fixed step sizes, and panoramic observations. For context, we report numbers from representative VLN systems, while emphasizing that our continuous 6-DoF setting is more error-prone and not directly comparable.

The study is guided by four research questions (RQ):

- **RQ1:** How effectively can a general-purpose multimodal agent generate accurate and efficient 6-DoF camera trajectories from object-centric natural language instructions?
- **RQ2:** How do structured geometric cues, such as depth maps and BEV projections, influence planning quality in the absence of explicit reasoning?
- **RQ3:** What is the relative impact of explicit reasoning strategies (Chain of Thought) compared to purely visual or geometric inputs on success and error rates?
- **RQ4:** What failure modes and behavioral patterns emerge when a multimodal agent performs open-ended 6-DoF navigation under different cue and reasoning conditions?

This work makes three contributions:

- A modular pipeline that enables general-purpose LLMs to perform continuous 6-DoF navigation for novel view synthesis.
- An ablation study of the system isolating the effects of reasoning, geometric cues, and overlays.
- An evaluation protocol combining qualitative and quantitative assessments, establishing a baseline for future work in LLM-driven navigation.

Chapter 2

Background

2.1 Novel View Synthesis

Novel View Synthesis (NVS) is the task of generating new visual perspectives of a scene given one or more new camera poses. Unlike general video generation, NVS is spatially grounded: the goal is to reconstruct or hallucinate views from unseen camera poses while preserving scene structure and appearance [19, 24, 40]. This problem is central to applications such as 3D scene reconstruction, free-viewpoint navigation, and instruction-conditioned simulation.

Traditional NVS approaches rely on explicit 3D geometry, such as multi-view stereo or depth-based warping [5, 11], assuming known or inferred camera intrinsics and extrinsics to project observed pixels into new views. These methods often struggle with occlusions, reflections, or sparse imagery. Recent methods instead learn view synthesis end-to-end via implicit volumetric representations or generative models.

In our work, we adopt a hybrid NVS framework that supports two rendering backends: (i) conditional video diffusion models and (ii) 3D Gaussian Splatting. Both methods allow for synthesis of intermediate views conditioned on the 6-DoF camera poses predicted by GPT-4o [25], but they differ in architecture and data assumptions.

2.1.1 Video Diffusion Models

Diffusion models are a class of generative models capable of high-quality image and video synthesis [15]. They consist of a forward noising process q and a learned reverse denoising process p_θ . The forward process gradually corrupts a clean sample x_0 using Gaussian noise:

$$x_t = \alpha_t x_0 + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

where $\alpha_t^2 + \sigma_t^2 = 1$. The reverse process learns to reconstruct x_0 from x_t by minimizing a noise prediction loss:

$$\min_{\theta} \mathbb{E}_{t, \epsilon} [\|\epsilon_{\theta}(x_t, t) - \epsilon\|_2^2].$$

To improve efficiency, Latent Diffusion Models (LDMs) operate in a compressed latent space [29]. A video $x \in \mathbb{R}^{L \times 3 \times H \times W}$ is first encoded into $z \in \mathbb{R}^{L \times C \times h \times w}$ using a pretrained Variational Auto Encoder (VAE). Both the forward and reverse processes occur in this latent space, and the final output is decoded by a learned decoder D .

In our pipeline, we build upon *ViewCrafter* [41], which adapts the image-to-video diffusion model *DynamiCrafter* [38] for geometry-aware NVS. ViewCrafter [41] synthesizes temporally coherent videos from a single image and a trajectory of 6-DoF camera poses. We modify its conditioning interface to render input point clouds from CUT3R [35] at each pose, guiding the diffusion process to have spatial consistency. This enables smooth, instruction-driven visual transitions between frames in response to language commands.

2.1.2 3D Gaussian Splatting (3DGS)

3D Gaussian Splatting (3DGS) [19] is an alternative NVS method based on explicit geometry rather than diffusion. Scenes are represented as collections of anisotropic Gaussians in 3D space, each defined by position, orientation, scale, opacity, and spherical harmonics-based color coefficients. These Gaussians are rasterized using a renderer, producing photorealistic images in real-time without ray-marching [8] or dense sampling.

Unlike NeRFs [24], 3DGS models can be trained in minutes and rendered efficiently, making them well-suited for downstream tasks like real-time simulation or language-conditioned planning. In our system, we use 3DGS as a modular drop-in replacement for diffusion-based rendering: given the same GPT-4o [25] trajectory, we generate frames directly from a pre-trained 3DGS [19] scene without relying on CUT3R [35] or ViewCrafter [41].

This swap demonstrates the generality of our approach. Because the GPT agent outputs explicit camera poses, it can be decoupled from the rendering backend. Whether synthesizing novel views from learned diffusion models or from explicit 3D geometry, our planner remains agnostic to how visual frames are generated, underscoring the modularity and extensibility of our language-guided trajectory pipeline.

2.2 Camera Pose Representation

Accurate and consistent camera pose representation is important for both novel view synthesis and trajectory planning. In 3D vision, a camera pose describes both the position and orientation of the camera in space, typically using a transformation from world coordinates to camera coordinates.

A full 6-DoF camera pose consists of:

- **Translation:** The 3D position of the camera, specified as a vector (x, y, z) in world space.
- **Rotation:** The orientation of the camera, often represented via Euler angles (yaw, pitch, roll), a 3×3 rotation matrix $R \in \mathbb{R}^{3 \times 3}$, or a unit quaternion $q \in \mathbb{R}^4$. All three are valid, but rotation matrices and quaternions offer better numerical stability.

Together, these define the **extrinsic matrix** $E = [R \mid t]$, a 3×4 transformation that maps 3D world points into the camera’s local coordinate system. In homogeneous coordinates:

$$\mathbf{x}_{\text{cam}} = R \cdot \mathbf{x}_{\text{world}} + t.$$

When combined with the **intrinsic matrix** K , which encodes the camera’s internal parameters (focal length, principal point, etc.), one obtains the full projection equation:

$$\mathbf{x}_{\text{image}} = K[R \mid t]\mathbf{x}_{\text{world}}.$$

In our pipeline, GPT-4o [25] outputs camera pose updates in 6D vector form: $(\Delta x, \Delta y, \Delta z, \Delta \text{yaw}, \Delta \text{pitch}, \Delta \text{roll})$, relative to the previous frame. These increments are accumulated to maintain a global trajectory and then converted into extrinsic matrices for use in downstream rendering systems such as ViewCrafter [41] or 3DGS [19].

To ensure smooth motion, we interpolate these pose sequences using cubic splines over both translation and rotation, enabling continuous and realistic camera movement even from discrete planning steps.

2.3 Point Cloud Reconstruction

Point clouds provide a lightweight and flexible representation of 3D scene geometry, consisting of unordered sets of spatial coordinates. While traditional methods require multi-view or stereo input, recent advances enable single-image reconstruction using self-supervised deep learning.

DUST3R [36] is a foundational model in this space, leveraging dual-scale vision transformers and contrastive correspondence learning to produce dense disparity maps from stereo input. It forms the basis for several successors including CUT3R [35], MonST3R [42], and MAST3R [22]. However, DUST3R [36] outputs depth in an arbitrary scale and assumes stereo pairs, limiting its use in monocular, metric, or iterative settings.

CUT3R [35] extends this capability by introducing scale-aware training, enabling the estimation of *metric point clouds* from a single RGB image. In our pipeline, CUT3R [35] serves two purposes: it initializes the scene geometry from the first frame and continuously refines it using frames generated during trajectory execution. This allows the system to *incrementally reconstruct the scene*, improving spatial reasoning over time.

We extract 3D cues, depth maps and BEV obstacle maps, from CUT3R [35] outputs and feed them into GPT-4o [25] as auxiliary inputs for planning. This approach enables the language model to reason about geometry indirectly, without requiring native 3D understanding or supervision.

2.4 Multimodal Language Models and GPT-4o

Large multimodal models (LMMs) extend the capabilities of language models to handle visual and spatial reasoning by processing image, video, or audio inputs alongside text. **GPT-4o** (omni) [25] is a state-of-the-art multimodal transformer capable of ingesting both natural language and raw pixel inputs in a unified interface. It is particularly well-suited to high-level planning in visually grounded environments due to its ability to attend mutually textual and visual tokens.

GPT-4o [25] processes images as token sequences, applying a proprietary vision encoder that converts input patches into discrete embeddings. These embeddings are integrated directly into the language context window, enabling cross-modal attention during inference.

Visual inputs must be preprocessed before being passed to GPT-4o [25]. The standard pipeline is as follows:

- Convert to 8-bit, 3-channel sRGB format.
- Resize to maintain aspect ratio with both sides ≤ 2048 pixels.
- Upscale if necessary so that the shortest side is ≥ 768 pixels.
- Pad the image on the bottom and right part to the nearest multiple of 512 pixels (using black padding).
- Tile the padded image into non-overlapping 512×512 patches.

- Tokenize into a stream of 85 base tokens plus 170 tokens per tile (e.g. a 1024×576 image becomes 3×2 tiles = 1105 tokens).

In our pipeline, GPT-4o [25] receives three modalities at each step: (i) the current RGB frame, (ii) auxiliary spatial representations (e.g. depth map or BEV projection), and (iii) a natural language instruction. The model outputs a 6-DoF pose increment in text format, which is parsed and applied to the last pose to update the camera trajectory. This zero-shot interaction loop is repeated iteratively, with the model continuing to plan until it self-declares task completion. A visual example of the preprocessing pipeline is shown in Figure 2.1, where we use an input image of size 1024×576 , similarly to our experimental scenario.

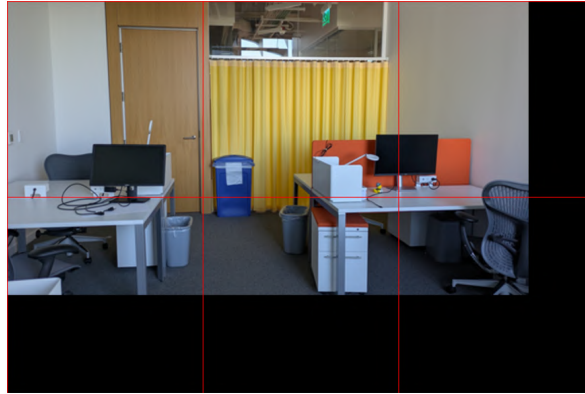


Figure 2.1: Visual input preprocessing pipeline for GPT-4o [25]. The input image has a size of 1024×576 (no need to resize), padded to the nearest multiple of 512 pixels, and tiled into 512×512 patches before tokenization.

Chapter 3

Related Work

3.1 Language-Guided Trajectory Generation in 3D Environments

Early efforts in *Vision-and-Language Navigation* (VLN) introduced the challenge of following natural-language instructions in realistic 3D spaces. The Room-to-Room (R2R) [13] benchmark was a core example, pairing human-written directions with agent trajectories through indoor environments [13]. Agents in such tasks interpret commands like “go down the hallway and turn left at the kitchen” and navigate accordingly [1, 6, 12, 14, 16, 32, 43]. These models typically operate with **restricted motion**, such as graph-based navigation or 3-DoF camera control, and require extensive task-specific training data.

Recent work has explored the use of pre-trained language models for navigation. **NavGPT** [43] treats navigation as a reasoning problem, using a large language model to select actions based on observations expressed as natural language, action history, and available moves. It is capable of decomposing instructions and sketching route plans, however it reduces vision to language-to-language translation, **bypassing direct 3D understanding**. Movement remains discretized and specific to the environment.

Other works link high-level language reasoning with low-level control through generative models. **NavigateDiff** [27] combines a fine-tuned vision language model with a diffusion-based image predictor that generates a hypothetical next-view image given the current view, goal and instruction. The policy then selects actions based on the predicted and actual images. Although more visually grounded, motion remains discrete, planar (forward, left, right), and policy dependent. None of these methods addresses **the generation of free-form 6-DoF trajectory** directly from the language in an open and unconstrained setting.

3.2 Vision-Language Models and 3D Spatial Understanding

General-purpose vision-language models (VLMs) like GPT-4 [25] or CLIP [28] demonstrate strong visual-textual alignment but lack **explicit 3D spatial grounding**. Although they can recognize and describe objects, their understanding of geometry (e.g. distances, occlusions, directions) is implicit at best [3]. This has motivated efforts to augment VLMs with structured spatial representations.

One example is **VLM-3R** [10], which enhances a multimodal model with 3D tokens extracted via CUT3R’s [35] monocular reconstruction. These tokens encode camera pose and depth features, enabling the model to better reason about spatial tasks such as counting or spatial relation queries. This suggests that injecting geometric cues enhances 3D reasoning capabilities.

In our system, we similarly guide a black-box VLM by providing **structured scene representations**: a bird’s-eye view (BEV) obstacle map and a depth map, paired with the RGB image and prompt. These inputs direct the model’s attention to navigable space, occlusions, and scene layout. Prior robotics systems very often leverage BEV maps for control and planning [26], in our case we present them to a language model to assist in **zero-shot spatial reasoning**, without any retraining. Our work contributes insights into which representations (depth, point clouds, BEV) best support 3D-aware planning in language-driven agents.

3.3 Trajectory Planning for Novel View Synthesis

Novel view synthesis (NVS) is the task of generating unseen perspectives from limited observations, most commonly through explicit 3D reconstruction methods (e.g. multi-view stereo [30], NeRF [24]) or generative image-to-image translation [23]. In most prior work, trajectory planning for NVS is either scripted [41] (not planned in real time) or geometry-driven [17] (through control), rather than **language-directed**.

Some recent methods incorporate natural language. **Director3D** [23] uses a diffusion-based “Cinematographer” module to sample camera paths from a prompt, and then a “Decorator” renders the scene via 3D Gaussian densities. Even though this is visually compelling, the method **lacks grounding in real environments**, the scene and trajectory are entirely synthetic.

ViewCrafter [41] generates novel-view videos from a single input image by conditioning a diffusion model on an inferred 3D point cloud reconstructed with DUS3R [36]. The camera motion is parameterized in spherical coordinates (yaw, pitch, radius) around the object, resulting in 3-DoF object-centric trajectories. The rendered point cloud from the camera poses viewpoint serves as geometric guidance during diffusion, improving spatial consistency across frames. However, the trajectories remain predefined (e.g. orbits, arcs) rather than semantically planned from natural language. Similarly, systems like Stable Virtual Camera [44] generate smooth transitions based on

user-specified keyframes or motion patterns, without grounding in language instructions.

Other works enable high-fidelity rendering from full 3D reconstructions. **3D Gaussian Splatting (3DGS)** [19] builds a fast, geometry-based scene representation from videos or image sets, producing renderings that remain stable when the viewpoint changes. This is useful for scene exploration, but 3DGS-based systems do not address **trajectory selection** or **language-based planning**. In contrast, we demonstrate that GPT-4o [25] can interface with both diffusion-based renderers and 3DGS models to control view synthesis from natural language alone.

3.4 Comparison and Limitations of Prior Work

Compared to these prior approaches, our pipeline introduces several novel capabilities:

- **Full 6-DoF Motion:** Most systems assume planar motion or fixed orbits around a visible target. Our agent moves freely in x, y, z translation and yaw, pitch, roll, enabling unconstrained exploration.
- **Partially Observed Environments:** Unlike methods assuming pre-scanned or fully known scenes, our system starts from a single image and incrementally reconstructs geometry during execution.
- **Zero-Shot Planning with No Fine-Tuning:** We use a general-purpose VLM (GPT-4o [25]) for planning without retraining. Other systems often rely on task-specific models or trained policies.
- **Instruction-Driven, Not Heuristic:** Our camera trajectories are semantically aligned with language prompts. Prior work either hard-codes paths or optimizes for unrelated criteria (e.g. coverage).
- **Modularity:** Our pipeline decouples the renderer and the agent. It is compatible with both diffusion-based and 3DGS rendering backends, and can integrate different LLMs as planners. This flexibility allows future substitution of either component without altering the overall system design.

To our knowledge, this is among the first studies of **instruction-driven, open-ended, 6-DoF trajectory generation** for novel view synthesis in real or partially observed scenes using general-purpose multimodal models, without task-specific training. In the following sections, we describe our methodology and evaluate its ability to produce semantically meaningful and visually consistent view sequences.

Chapter 4

Methodology

4.1 Pipeline Overview

Our goal is to enable language-driven 6-DoF camera trajectory generation in 3D scenes. We propose a modular pipeline that uses OpenAI’s GPT-4o [25] as the central planner, operating in a closed loop interaction with visual renderers and modules inferring spatial cues. At each timestep, the GPT-4o [25] receives a rendered image of the current view, auxiliary geometric information, and textual trajectory description. Then it outputs the next camera increment in 6-DoF, attempting autonomous navigation through the scene. We consider two variations of this pipeline, employing different ways to render views:

- **Diffusion-based Pipeline (Fig. 4.1):** Uses a conditional Latent Diffusion Model to generate new frames from the current camera pose through point cloud guided rendering. A 3D reconstruction model, CUT3R [35], processes each generated frame to produce metric depth maps and point clouds, which are further projected into BEV maps. These three visual cues, rendered RGB, depth, and BEV, are passed to GPT-4o [25] alongside the prompt. GPT-4o [25] integrates all modalities to predict the next 6-DoF movement.

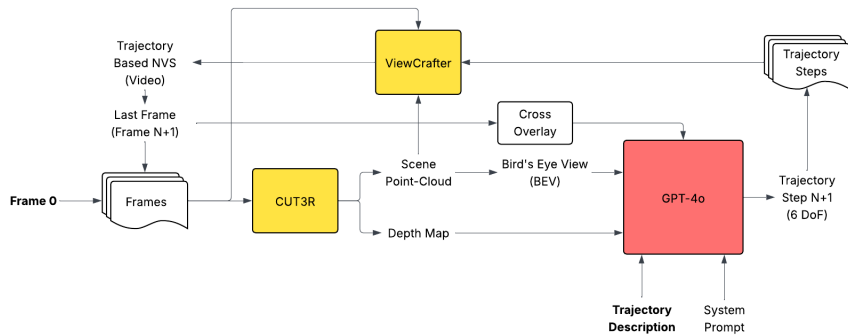


Figure 4.1: Pipeline using diffusion-based NVS and CUT3R.

- **3D Gaussian Splatting Pipeline (Fig. 4.2):** Views are directly rendered from a pre-constructed 3D Gaussian Splatting scene at the desired camera pose. These renders are used to produce the RGB frame, a depth map, and a BEV image, all of which are fed into GPT-4o [25] together with the textual instruction. As in the diffusion-based pipeline, GPT-4o [25] processes these inputs and emits the next pose increment, closing the loop.

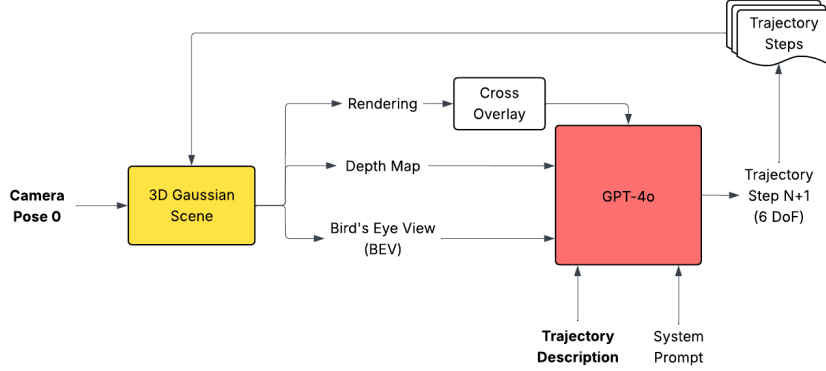


Figure 4.2: Pipeline using 3D Gaussian Splatting for view synthesis.

Both pipelines share the same high-level architecture and decision loop. They differ on how views are synthesized and how geometry is computed. This dual implementation allows us to test generalization across input modalities and scene representations.

The control flow of the pipeline is summarized in Algorithm 1.

Algorithm 1 Zero-Shot 6-DoF Trajectory Planning with GPT-4o

Require: Trajectory Prompt TP , Initial RGB image I_0 , System Prompt SP

```

1:  $t \leftarrow 0$ 
2: while GPT-4o has not output "STOP" do
3:   Extract auxiliary geometry inputs:
4:   if Using diffusion-based rendering then
5:      $D_t \leftarrow \text{Depth from CUT3R}(I_t)$ 
6:      $B_t \leftarrow \text{BEV map from point cloud}$ 
7:   else
8:      $D_t, B_t \leftarrow \text{Geometry from 3DGS splats}$ 
9:   end if
10:   $O_t \leftarrow \text{GPT-4o}(TP, I_t, D_t, B_t; SP)$ 
11:   $T_{t+1} \leftarrow \text{Parser}(O_t)$ 
12:  if Using diffusion renderer then
13:     $I_{t+1} \leftarrow \text{ViewCrafter}(I_t, T)$ 
14:  else
15:     $I_{t+1} \leftarrow \text{Render from 3DGS at pose } T_{t+1}$ 
16:  end if
17:   $t \leftarrow t + 1$ 
18: end while

```

4.2 Input Modalities

At each timestep, the GPT-4o [25] planner receives four input modalities to inform its decision: an RGB image, a depth map, a top-down bird’s eye view (BEV) map, and a language prompt. These inputs are processed together by GPT-4o [25] to reason about geometry, framing, and the semantic objective.

1. RGB Image: The main visual grounding input is a rendered RGB image of the current camera view. To assist GPT-4o [25] in understanding the scene and interpreting spatial alignment, given the image is padded and tokenized into 512×512 tiles (see section 2.4 for more info), we overlay a guiding red cross on each RGB input, marking the horizontal and vertical center lines. This helps the model identify the location of scene elements relative to the camera’s forward axis. In the diffusion-based pipeline, this image is synthesized using a conditional Latent Diffusion Model (LDM), conditioned on point cloud renders from CUT3R [35]. Only the last frame of the generated video is used. In the 3DGS-based pipeline, the image is rendered directly from a pre-trained 3D Gaussian Splatting scene at the most recent camera pose.



(a) RGB input from LDM (diffusion pipeline)



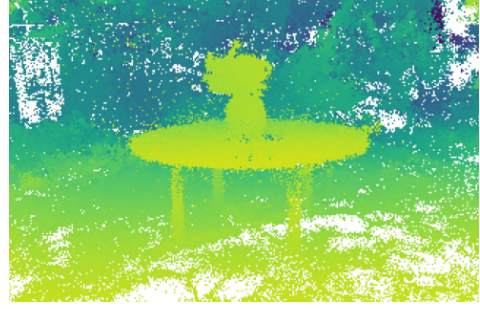
(b) RGB input rendered (3DGS pipeline).

Figure 4.3: Comparison of RGB inputs passed to GPT-4o. Red guiding crosses help align semantic targets with the camera center.

2. Depth Map: This image encodes distance to visible surfaces and is crucial for avoiding collisions and assessing reachable regions. In the diffusion pipeline, metric depth is directly predicted by CUT3R [35]. In the 3DGS pipeline, we color splatted points by their depth and project them onto the current camera view to simulate a depth map.



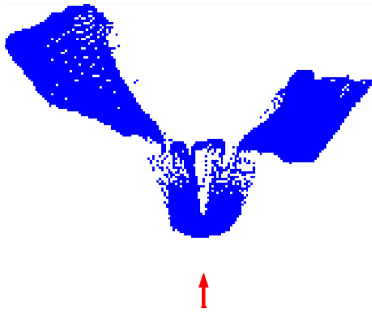
(a) Depth map from CUT3R (diffusion pipeline).



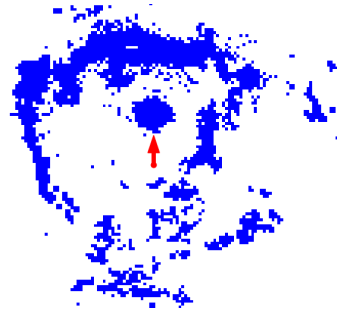
(b) Projected depth from splats (3DGS pipeline).

Figure 4.4: Comparison of depth inputs from each pipeline.

3. Bird’s Eye View (BEV) Map: This top-down projection provides agent-centric spatial awareness. In both pipelines, BEV maps are generated by projecting all 3D points within ± 20 cm of the current camera height onto the agent’s XZ plane. The projected maps show obstacles (blue) and the agent’s position (red dot). These points are derived from CUT3R [35] reconstructions or 3DGS Gaussians [19].



(a) BEV from CUT3R point cloud (diffusion pipeline).



(b) BEV from 3DGS splats (3DGS pipeline).

Figure 4.5: Comparison of BEV maps highlighting nearby obstacles in the local frame.

4. Prompt: A text prompt defines an object-centric navigation goal. GPT-4o [25] is initialized with a detailed *system prompt* (for the full prompt see Appendix E) that enforces camera-centric conventions, motion heuristics, obstacle avoidance, and structured response format. This prompt governs how the visual inputs are interpreted and ensures outputs are valid 6-DoF increments.

While the input interface to GPT-4o [25] is identical across both pipelines, the internal generation of each modality varies. This setup enables direct comparison between vision systems that generate views via generative diffusion or explicit 3D rendering.

4.3 6-DoF Planning with GPT-4o

The core of our navigation system is OpenAI’s GPT-4o [25], a multimodal language model used as a zero-shot trajectory planner. At each step, GPT-4o [25] receives four inputs: (i) the current RGB view with overlaid red guidance lines, (ii) a depth image, (iii) a top-down bird’s eye view (BEV) map, and (iv) a textual prompt defining the goal. Based on these, it outputs a 6 dimensional camera pose increment, navigating in an iterative closed-loop manner.

Prompting Strategy

We design a detailed system prompt (see Appendix E) to guide GPT-4o’s [25] behavior and reasoning. It enforces strict navigation conventions (camera centric motion commands), safety constraints (e.g. maintain at least 20cm distance from obstacles), and structured easily parse-able output format. The model is expected to showcase spatial reasoning, analyze position of the target in the image, check continuity with previous steps and verify decision based on a checklist. The following prompting techniques are used:

- **Chain of Thought Reasoning:** GPT-4o [25] first analyzes the image layout, identifies object quadrants, reasons about geometry, and explains its motion choice before issuing a command.
- **Step History Tracking:** The history of previously taken steps, along with the logic is included to promote consistent, smooth trajectories and avoid erratic motion.
- **Framing Heuristics:** Based on the quadrant in which the target appears, the model adjusts yaw and pitch to center the objective (e.g. top-right \rightarrow pitch down, yaw left).
- **Verification Checklist:** Before any command is output, the model is required to confirm that it satisfies motion limits, obstacle clearance, and camera alignment.
- **Output Format Enforcements:** The final outputs must include exactly six sections (description, continuity, reasoning, verification, objective and motion command), ending with a motion line of the form $dx\ dy\ dz\ dyaw\ dpitch\ droll$, with motions in meters and degrees, bounded as: translations $t \in [-0.5, +0.5]\ m$ and rotations $R \in [-10^\circ, +10^\circ]$.

One Chat per Step: To avoid GPT-4o [25] hallucinating progress or referencing scene changes that have not yet occurred, we instantiate a new chat session at every step. Each chat is re-initialized with the same system prompt, a fresh description of the current inputs, and an up-to-date step history, all of these tied together using prompt templating (see details in 4.6). Without this reinitialization, the model tends to internally simulate the trajectory, creating inconsistencies in its spatial grounding, ignoring the new visual inputs provided.

Example

We illustrate an example of GPT-4o [25] inference in Figure 4.6. The model receives the visual inputs and the navigation prompt. It responds with detailed spatial analysis, goal alignment reasoning, checklist verification, and finally emits the 6-DoF command to continue the navigation sequence.



Figure 4.6: Example of an Input/Output from an interaction with the GPT agent

Observed Technical Limitations

Despite the detailed prompt and control heuristics, GPT-4o [25] exhibits several implementation-related limitations:

- **Frame Convention Violations:** The model occasionally outputs motion commands that contradict the defined camera-centric conventions (e.g. decreasing z to move forward, or mixing pitch/yaw signs). These errors affect both translations (x, y, z) and rotations (yaw, pitch, roll).
- **Redundant or Ineffective Actions:** Some outputs are numerically too small to affect the scene, or contradict prior steps without justification, leading to oscillations.
- **Context Forgetting:** Without resetting the chat session each step, GPT-4o [25] tends to hallucinate scene evolution and ignore new inputs.
- **Employing Flawed Reasoning:** Often the model makes adjustments that are fundamentally flawed especially in the rotational degrees of freedom. For instance when the target object is located on the left side of the screen the agent wants to yaw right so that the target moves rightwards, which is the opposite of what should be done.

These issues highlight the importance of strict prompt engineering and context management in maintaining coherent 6-DoF navigation.

4.4 Scene Representation and Update

To support closed-loop planning and visual feedback, our system has an internal representation of the 3D scene. This representation is updated dynamically as new views are explored, in the case of the diffusion pipeline (for the 3DGS the 3D scene representation is already known). Each pipeline uses a different strategy for handling spatial geometry and ensuring visual consistency across timesteps.

Diffusion-Based Representation (with CUT3R)

The diffusion-based pipeline synthesizes new views using a conditional Latent Diffusion Model (LDM), which generates short video segments conditioned on the current camera pose and a colored point cloud rendering. However, diffusion models can hallucinate novel content and introduce inconsistencies across timesteps. To mitigate this, we integrate CUT3R [35], a 3D reconstruction model, which processes the LDM outputs to produce depth maps and a fused 3D point cloud.

- **Iterative Scene Fusion:** After each new frame is synthesized, CUT3R [35] reconstructs depth and merges it into the global point cloud. This ensures that spatial information is coherent across iterations, promoting consistency.
- **Temporal Conditioning:** To reinforce consistency, the LDM is explicitly conditioned on the last predicted frames from previous steps. This input constraint helps align the visual content between successive generations.
- **Outlier Rejection:** To filter parasitic or noisy points from the reconstructed geometry, we apply DBSCAN [34] clustering to the accumulated point cloud. The minimum number of neighbors required for a point to be retained increases proportionally with the number of steps taken, enforcing a growing standard of geometric reliability (for more details, see Appendix C).

The CUT3R-based approach allows the system to maintain a coherent and progressively refined spatial map even when starting from a single view (see Appendix C for visualization).

3D Gaussian Splatting Representation

In contrast, the 3DGS pipeline assumes access to a pre-trained 3D Gaussian Splatting (3DGS) scene model [19], which encodes spatial and radiance information from a set of training views. At each timestep, we query this model with the current camera pose to render the RGB frame, depth image, and point cloud projection.

- **Stable Geometry:** Since the 3DGS scene is fixed, rendering quality is consistent within the regions observed during training. Geometry and appearance are accurate in these areas, making it ideal for precise view synthesis.
- **Black Regions for Unseen Areas:** When querying views that look toward unobserved parts of the scene, the rendered image includes large black regions. These indicate the limits of the learned radiance field of the scene (regions out of the training set). They often lead to erroneous or meaningless GPT-4o [25] outputs.

By comparing these two representations, we observe complementary strengths: CUT3R [35] allows dynamic scene growth from scratch, whereas 3DGS [19] offers high-quality rendering within constrained bounds. This dual setup enables us to evaluate trajectory planning under both minimal and fully pre-constructed scene conditions.

4.5 Rendering Module

The rendering module generates the visual inputs, RGB image, depth map, and BEV map, provided to GPT-4o [25] at each timestep, based on the current camera pose. Although both pipelines follow the same control logic, they differ significantly in how views and geometry are synthesized. Importantly, all outputs are standardized to maintain compatibility with GPT-4o’s [25] multimodal interface (see section 4.2).

Diffusion-based Rendering

In the diffusion-based pipeline, view synthesis is achieved via a conditional Latent Diffusion Model (LDM) [38], guided by point cloud renders produced by CUT3R [35]. At each step:

- **RGB Image:** The LDM generates a short video from the current camera view. We use the final frame of this sequence as the RGB input to GPT-4o [25]. To enforce consistency across steps, the LDM [38] is additionally conditioned on the final frames from prior steps.

- **Depth Map:** The generated RGB frame is passed to CUT3R [35], which predicts a metric depth map aligned with the camera perspective.
- **BEV Map:** The 3D point cloud output by CUT3R [35] is cleaned using DBSCAN [34], with a dynamic density threshold proportional to the number of previous steps (to suppress parasitic points). Points within ± 20 cm of the camera’s height are projected onto the XZ plane of the agent to form a BEV map.

3DGS-based Rendering

In the 3D Gaussian Splatting (3DGS) pipeline, we directly render views from a pre-trained splat-based scene [19]:

- **RGB Image:** The current camera pose is used to render the scene using the 3DGS rasterizer (convert vectorized graphics into pixels). Regions outside the training coverage appear black.
- **Depth Map:** Splat depths are encoded by color and projected into the image plane to approximate a depth view (the resulting depth map has some blank regions since we do not project anisotropic gaussians of varying size, but simple points of fixed size).
- **BEV Map:** Points within ± 20 cm of the camera height are extracted and projected to the XZ plane to create a BEV map, as in the diffusion-based pipeline.

Modularity and Extensibility

The rendering module is fully modular, allowing for easy replacement of the view synthesis strategy. This design enables the pipeline to generalize to other scenarios beyond this study, such as:

- Real-world robots with RGB-D sensing.
- Virtual Reality (VR) platforms requiring agent centric rendering.
- Simulated drones navigating synthetic environments.

By decoupling scene representation from decision-making, the pipeline remains extensible and adaptable to evolving 3D perception technologies.

4.6 Other Implementation Details

Frame Interpolation in the Diffusion Pipeline

In the diffusion-based pipeline, the number of rendered frames F differs from the number of planning steps S produced by GPT-4o. We adopt the heuristic configuration:

$$F = 5S + 5,$$

chosen empirically to stabilize visual fidelity. For small S , directly generating S frames produces poor quality output, by globally upsampling to F frames, we ensure smooth trajectories for the latent diffusion model (LDM).

Let $\{T_i\}_{i=1}^S$ be the discrete 6-DoF camera poses output by GPT-4o, where each pose can be represented as a 6D vector:

$$T_i = (x_i, y_i, z_i, \text{yaw}_i, \text{pitch}_i, \text{roll}_i).$$

We construct six independent cubic splines

$$\text{Spline}_d : \{1, \dots, S\} \rightarrow \mathbb{R}, \quad d \in \{x, y, z, \text{yaw}, \text{pitch}, \text{roll}\},$$

such that $\text{Spline}_d(i) = T_i[d]$. Evaluating each spline at F uniformly spaced indices produces the interpolated trajectory

$$\tilde{T}_j = (\text{Spline}_x(j), \text{Spline}_y(j), \text{Spline}_z(j), \text{Spline}_{\text{yaw}}(j), \text{Spline}_{\text{pitch}}(j), \text{Spline}_{\text{roll}}(j)), \quad j = 1, \dots, F.$$

To enforce temporal and semantic consistency, the LDM is conditioned not only on the rendering of the current CUT3R [35] point cloud, but also on the set of all previously generated “last frames”:

$$\mathcal{G}_t = \{R(\mathcal{P}_t)\} \cup \{I_1^{\text{last}}, I_2^{\text{last}}, \dots, I_{t-1}^{\text{last}}\},$$

where $R(\mathcal{P}_t)$ is the rendering of the colored point cloud at the current pose, and I_j^{last} is the final hallucinated frame of iteration j .

Each I_j^{last} is re-inserted at its corresponding index in the new F -frame sequence, i.e.

$$I_j^{\text{last}} \mapsto \tilde{T}_{\kappa(j)},$$

where $\kappa(j)$ denotes the frame index aligned with the end of step j . This cumulative guidance preserves consistency with all past hallucinations and enforces long-horizon temporal coherence.

Navigation in 3DGS

In the 3D Gaussian Splatting pipeline, navigation is implemented as recursive transformations of the coordinate system, always centered at the latest camera pose. This requires knowledge only of the initial camera pose T_0^0 , which fixes the original world coordinates. Subsequent poses are then computed by accumulating relative motions.

We denote by T_0^0 the fictional camera pose defining the original world coordinates. The i th camera pose in the original coordinate system is written T_i^0 , while T_i^n denotes the same pose expressed in the coordinate system of the n th camera. Relative transformations from pose i to pose j are written $T'_{i \rightarrow j}$.

Starting from $T_1^0 = (T'_{0 \rightarrow 1})^{-1}$, subsequent poses are expressed as

$$T_i^0 = T_1^0 \left(\prod_{j=1}^{i-1} T'_{j \rightarrow j+1} \right).$$

Equivalently, given the direct pose T'_i in the local canonical system,

$$T_i^0 = T'_i \left(\prod_{j=1}^{i-1} T'_{j \rightarrow j+1} \right)^{-1}.$$

This recursive structure enables consistent global placement of all camera poses from purely relative commands, while maintaining local motion centered at the latest pose.

Prompt Templating

To ensure consistency across runs, we use a templating function that generates the system prompt with fixed structure and placeholders for the goal description, step history, and maximum step index. The template enforces continuity reasoning, trajectory reasoning, verification, and motion commands in strict format:

Trajectory Step {max_step} - Plan the next move.

Goal:

{traj_desc}

Step History:

{traj_hist}

Reminder: Respond with:

1. Have some continuity reasoning
 2. Trajectory reasoning (max 4 lines)
 3. Checklist Verification
 4. Objective in format: ###\n(Objective)\n### (no indents)
 5. Motion command in format: 'dx dy dz dyaw dpitch droll'
- Follow camera-centric conventions exactly. No extra text.

This templating both improves reproducibility and stabilizes GPT-4o's [25] outputs by fixing expectations about response structure. It also enables optimal reasoning by ensuring step history and objectives are explicitly re-grounded at every step.

Chapter 5

Evaluation

This section provides both a *quantitative* and a *qualitative* assessment of our approach. Quantitative evaluation relies on established metrics such as Success Rate (SR), Success weighted by Path Length (SPL), Trajectory Length (TL), Step Size, Reasoning Success (RS), Violation Rate (VR), and Spatial Understanding (SU). These measures allow systematic comparison across configurations and against representative baselines from Vision-and-Language Navigation (VLN).

Complementing these numbers, we also provide a qualitative analysis through trajectory distributions, error breakdowns, and failure attributions. This highlights how Chain-of-Thought (CoT) prompting and geometric cues (Depth, BEV, and visual overlays) affect planning behavior in practice, revealing not only *whether* a configuration succeeds but also *how* and *why* it fails.

5.1 Experimental Setup

Hardware

All experiments were conducted on a dedicated NVIDIA A100-SXM4-80GB GPU with no competing processes, ensuring consistent runtimes.

Scenes

We evaluate on three real-world scenes from the *3D Gaussian Splatting* paper:

- **Garden** (Mip-NeRF360 [4]): low obstruction, open areas (Fig 5.1a).
- **Truck** (Tanks and Temples [21]): moderate obstruction, partial occlusion (Fig 5.1b).

- **Train** (Tanks and Temples [21]): high obstruction, narrow visibility (Fig 5.1c).



(a) Garden



(b) Truck



(c) Train

Figure 5.1: The three scenes considered for the evaluation.

Each scene requires *compound motion* (e.g. combined yaw and translation) to reach the target, which is always visible in the initial frame.

Task

The agent receives:

1. A single reference RGB image of the scene,
2. An object-centric instruction (target visible at the start),
3. Optionally, depth and Bird's Eye View (BEV) projections.

The task is to execute a 6-DoF trajectory (translation x, y, z ; rotation yaw, pitch, roll) that brings the target into the desired view. A run terminates when the agent either:

- Declares the target reached,
- Aborts due to inadequate view,
- Hits the step limit (20 non-CoT, 25 CoT).

Configurations

We test four planning configurations in 3DGS [19]:

- **Basic**: RGB only, minimal constraints.
- **DnB**: Basic + depth + BEV.

- **CoT**: RGB + tailored Chain of Thought (CoT) prompting.
- **Full**: Combination of DnB and CoT.

Unless otherwise noted, all use a cross overlay for alignment. In the *Diffusion* pipeline, only the Full setting is evaluated.

Trials and Controls

Each configuration is run 15 times (5 per scene). Runs for a given scene start from identical initial poses, ensuring differences arise from planning rather than initialization. All results were manually labeled by a human annotator.

5.2 Evaluation Metrics

We measure performance using the following metrics:

- **Trajectory Length (TL)**: Steps taken before termination (success, abort, or max steps). Long \nRightarrow good, short \nRightarrow good.
- **Step Size**: Average Euclidean 6-DoF displacement per translational step (mean \pm std).
- **Success Rate (SR)**: Final view follows the instruction and ends sufficiently close to the target (human-judged).
- **Success weighted by Path Length (SPL) [39]**: Standard SR adjusted for path efficiency.
- **Reasoning Success (RS)**: Percentage of trials where all steps are logically consistent with the short term objective, any single flawed step \Rightarrow failure (see Appendix B.1).
- **Violation Rate (VR)**: Fraction of trials that break motion conventions (e.g. wrong sign on x for rightward move) (see Appendix B.2).
- **Spatial Understanding (SU)**: For CoT runs only, percentage of trials where the verbalized scene interpretation matches the actual layout, for all steps (see Appendix B.3).

5.3 Baselines

It is important to note that our setting is not directly comparable to Room-to-Room (R2R) [13]. In R2R, agents navigate along a discrete viewpoint graph with panoramic observations and fixed step

sizes. Trajectories in R2R [13] range from very simple straight-line motions to complex multi-room paths that require long-term reasoning.

In contrast, our tasks are restricted to object-centric navigation within a single scene. They are simpler in scope but demand non-trivial *continuous* control: agents must execute compound 6-DoF motions (simultaneous translation and rotation) to reach the target.

Due to the lack of directly comparable work on zero-shot, language-driven continuous 6-DoF navigation, we report R2R [13] results only as contextual baselines rather than head-to-head comparisons, and we include only our complete pipeline variants (3DGS and Diffusion) for reference.

Table 5.1: Baseline VLN agents on R2R vs. our methods.

Method	TL	NE ↓	SR (%) ↑	SPL (%) ↑
<i>Supervised</i>				
Seq2Seq [1]	8.39	7.81	28	–
Speaker-Follower [12]	–	6.62	45	–
EnvDrop [32]	10.70	5.22	52	48
PREVALENT [14]	10.19	4.71	58	53
VLN-BERT [16]	12.01	3.93	69	57
HAMT [6]	11.46	2.29	73	61
DuET [7]	13.94	3.31	81	60
<i>Zero-shot</i>				
DuET (Init. LXMERT) [31]	22.03	9.74	7	0
NavGPT [43]	11.45	6.46	42	29
Ours — 3DGS (Full)	18.00	–	47	30
Ours — Diffusion (Full)	9.80	–	13	11

5.4 Cross Overlay Study

Before ablating reasoning and geometry, we isolate the effect of the cross overlay on yaw/pitch-only tasks. The overlay is designed to assist rotational motion and reduce oscillation. This is required due to the tiling performed by the GPT agent losing track of the central axes of the image. Table 5.2 shows that it improves both alignment and success.

Table 5.2: Effect of the cross overlay on yaw/pitch-only tasks. The overlay improves SR and reduces oscillations, we therefore include it in all subsequent experiments by default.

Config	TL	RS (%) ↑	SR (%) ↑	SU (%) ↑	VR (%) ↓	Avg. Dist. (px) ↓	Osc. (%) ↓
Cross Only	10.47	53.33	40.00	100.00	26.67	432.5	26.70
No Cross	11.87	60.00	13.33	86.67	26.67	505.0	40.00

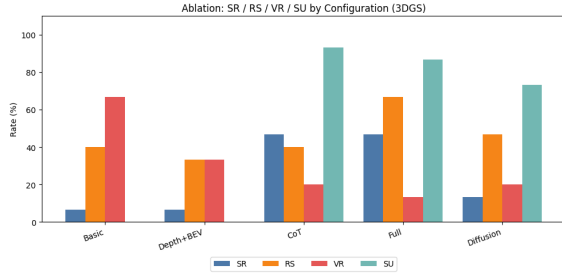
The overlay raises success from 13% to 40% and reduces oscillation by more than 13 percentage points, validating its necessity. All subsequent results therefore assume the cross overlay as a default component.

5.5 Ablation Study

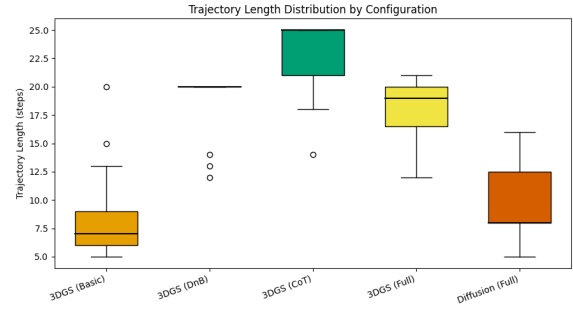
We now isolate the contributions of Chain-of-Thought (CoT) prompting and geometric cues (Depth and BEV). Table 5.3 provides the aggregate metrics, while Figures 5.2, 5.3, and 5.4 illustrate the main behavioral differences and failure modes across settings.

Table 5.3: Ablation on 3DGS and Diffusion. Metrics averaged over 15 runs. Arrows indicate desirable direction.

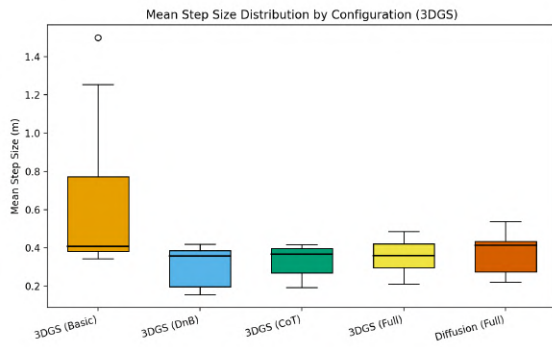
Configuration	TL	RS (%) ↑	SR (%) ↑	SU (%) ↑	VR (%) ↓	Step Size	SPL (%) ↑
3DGS (Basic)	8.60	40.00	7.00	–	66.67	0.618 ± 0.781	3.79
3DGS (DnB)	18.60	33.33	7.00	–	33.33	0.314 ± 0.112	5.03
3DGS (CoT)	22.87	40.00	47.00	93.33	20.00	0.058 ± 0.079	26.08
3DGS (Full)	18.00	66.67	47.00	86.67	13.33	0.032 ± 0.078	30.14
Diffusion (Full)	9.80	46.67	13.00	73.33	20.00	0.077 ± 0.094	10.64



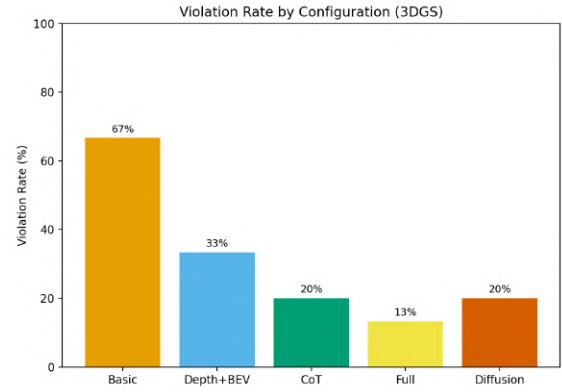
(a) SR, RS, VR, SU across configurations.



(b) Trajectory length distribution.



(c) Step size distribution.



(d) Violation rates.

Figure 5.2: Ablation plots for 3DGS configurations. CoT-based settings (CoT, Full) improve SR and reduce VR; Depth/BEV alone (DnB) underperform without reasoning.

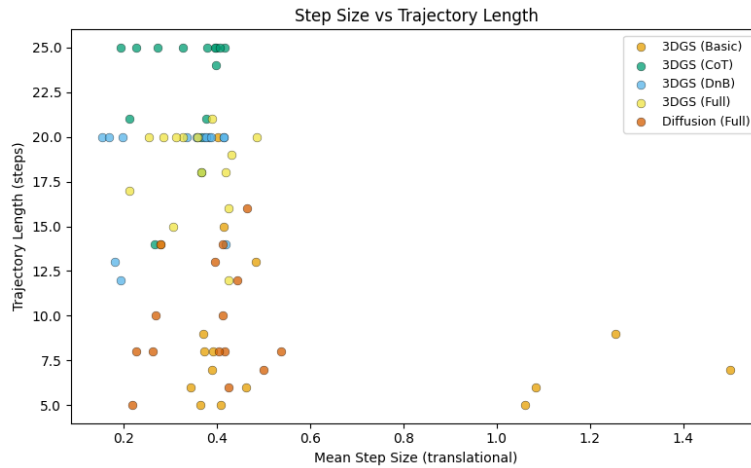


Figure 5.3: Step size vs. trajectory length. CoT and Full cluster around smaller steps and longer runs (cautious but consistent exploration), Basic is erratic (short, large steps). Diffusion-Full forms a distinct cluster of short-to-medium runs with moderate steps.

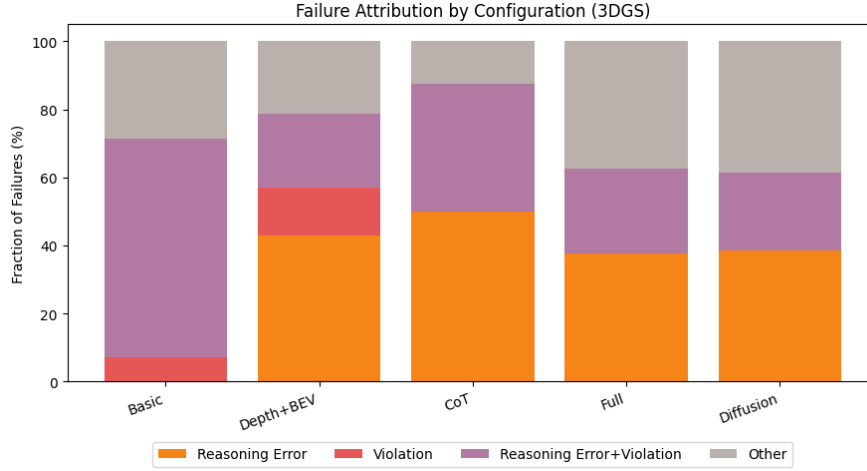


Figure 5.4: Failure attribution by configuration (3DGS). Bars show the fraction of failures explained by *Reasoning Error*, *Violation*, their *combination*, and *Other* causes (e.g. early stop due to inadequate view).

Effect of reasoning: Success is tightly coupled to CoT prompting. Without CoT (Basic, DnB), success remains at 7%. With CoT (CoT, Full), it rises to 47%, accompanied by higher reasoning success (40–67%) and a drop in violations. Diffusion (Full) confirms the trend: reasoning emerges (47%) but success stays lower (13%) due to noisier inputs. Figure 5.2a highlights this contrast, showing how success and reasoning improve systematically when verbalized planning is enabled.

Trajectory lengths: The distributions in Figure 5.2b reveal that Basic tends to terminate early (median ≈ 7 steps) with high variance, reflecting erratic planning. DnB enforces consistently long paths (≈ 20 steps) but does not translate this persistence into success. CoT extends trajectories further (≈ 25 steps), reflecting stronger persistence, despite the higher number of steps. Full strikes the most balanced regime (median ≈ 19 steps) with reduced variance. Diffusion–Full, in contrast, converges to shorter trajectories (≈ 10 steps).

Step sizes: Step size distributions (Figure 5.2c) mirror these behaviors. Basic exhibits large, unstable steps (median 0.4–0.5 m, some above 1 m), consistent with poor control. In contrast, DnB, CoT, and Full converge to smaller and more stable steps (0.2–0.4 m). Diffusion–Full also lies in this controlled regime, though with wider spread.

Violations: The violation rates (Figure 5.2d) form the clearest predictor of failure. Basic violates motion conventions in 67% of steps, DnB halves this rate (33%), CoT reduces it further (20%), and

Full reaches the lowest rate (13%). In all runs where violations occurred, success dropped to zero, confirming their fatal impact.

Failure modes: Figure 5.4 shows that most failures are linked to *reasoning errors* (incorrect yaw/pitch logic) or a *combination* of reasoning error and convention violation. DnB exhibits a higher share of violation-only and mixed failures, indicating that geometric cues without reasoning often misguide the controller. Full shifts mass away from violation-only failures toward “Other” (principally early stops on poor views or spacial understanding failures that occur much less frequently), consistent with safer behavior even when success is not achieved.

Joint view and summary: The scatter in Figure 5.3 conveys the following: Basic clusters are usually short, erratic runs with inflated steps, DnB and CoT show long, cautious trajectories with smaller steps, Full forms a compact cluster of stable, medium/long-length runs with consistent small steps; Diffusion–Full separates into short-to-medium trajectories with moderate steps due to noisier renderings (longer trajectories are too noisy to have meaningful step outputs). Overall, CoT is the critical enabler (improving the 7% SR up to 47% SR), while geometry alone is insufficient. Combining CoT with geometry (Full) yields the most stable and efficient behavior and the lowest violation rate.

Chapter 6

Discussion and Conclusion

This work introduces a complete pipeline for using general-purpose Large Language Models (LLMs) as agents for continuous 6-DoF navigation. The pipeline combines Chain-of-Thought (CoT) prompting, geometric cues (Depth and BEV), and overlays to provide structured yet minimal guidance. Evaluation across three real-world 3DGS scenes demonstrates both the feasibility and the current limitations of this approach. To our knowledge, this is among the first studies of LLM-driven navigation under continuous control without task-specific training, establishing a reference baseline for future research.

Findings: The ablation study highlights the central role of reasoning: CoT prompting is the decisive factor, improving substantially the success rates compared to settings without it. Geometric cues by themselves offer persistence but not improved success, in combination with CoT, they stabilize motion and improve efficiency. The overlay cross provides useful framing support, though it is not a decisive factor. Configurations with CoT and geometry often adopt smaller step sizes, which may indicate an implicit strategy for uncertainty management, though this remains speculative.

Limitations: Our observations reveal both technical and behavioral limitations:

- **Scope:** Evaluation is restricted to three scenes and object-centric tasks with visible targets, which constrains generality.
- **Technical Issues:** GPT-4o [25] occasionally violates frame conventions, produces redundant or ineffective actions, and requires chat resets to avoid hallucinated scene evolution.
- **Behavioral Issues:** We observe reasoning flaws, such as inverted yaw/pitch decisions, and systematic depth–height confusions, consistent with prior work on vision-language models [3].

- **Heuristic Dependence:** The system relies on auxiliary supports such as cross overlays, CUT3R [35] cleaning, and human annotation.
- **Model Scope:** Results are based on a single LLM (GPT-4o), leaving open the question of how other models would perform in the same pipeline.
- **Sample Size:** With 15 runs per configuration, statistical power remains limited.

Future Work: Several directions arise naturally from this proof of concept:

- **Pipeline-driven LLM comparison:** systematically evaluate alternative models (e.g. Mistral [18], Claude [2], DeepSeek [9], Gemini [33]) under the same pipeline to probe differences in reasoning ability.
- **Visual backbone substitution:** replace the current diffusion model with stronger generative backbones for higher-quality view synthesis, likewise, test improved 3DGS and reconstruction methods for more reliable spatial representations.
- **3D information encoding:** explore other geometric cues than depth/BEV in order to assist the agent without resorting to fully specialized spatial-language models that avoid zero-shot general purpose models.
- **Scaling:** Extend experiments to larger and more complex 3DGS scenes, and therefore experiment with longer and more complex trajectories.
- **Task variety:** Move beyond object-centric navigation to scene-centric, multi-target, or mixed-instruction tasks.
- **Zero-shot vs. tuned:** Retain zero-shot conditions as the main focus, but include fine-tuned baselines for comparison.

Overall, this work demonstrates that general-purpose LLMs can perform continuous 6-DoF navigation when supported by minimal scaffolding (3D cues and CoT). The proposed pipeline itself constitutes a central contribution: it provides both a proof of concept and a baseline for future research, underscoring the importance of reasoning mechanisms and geometric cues. By clarifying both the potential and the limitations of LLMs in agentic planning, this study opens a path toward systematic model comparisons, improved representations, and broader task settings.

Bibliography

- [1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. *Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments*. 2018. arXiv: 1711.07280 [cs.CV]. URL: <https://arxiv.org/abs/1711.07280>.
- [2] Anthropic. *Claude Opus 4.1: Our Most Capable and Intelligent Model Yet*. Accessed: 2025-08-05. 2025. URL: <https://www.anthropic.com/news/introducing-claude-opus-4-1>.
- [3] Shehreen Azad, Yash Jain, Rishit Garg, Yogesh S Rawat, and Vibhav Vineet. *DH-Bench: Probing Depth and Height Perception of Large Visual-Language Models*. 2024. arXiv: 2408.11748v1 [cs.CV]. URL: <https://arxiv.org/abs/2408.11748v1>.
- [4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. *Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields*. 2022. arXiv: 2111.12077 [cs.CV]. URL: <https://arxiv.org/abs/2111.12077>.
- [5] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. *pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction*. 2024. arXiv: 2312.12337 [cs.CV]. URL: <https://arxiv.org/abs/2312.12337>.
- [6] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. *History Aware Multi-modal Transformer for Vision-and-Language Navigation*. 2023. arXiv: 2110.13309 [cs.CV]. URL: <https://arxiv.org/abs/2110.13309>.
- [7] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. *Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-Language Navigation*. 2022. arXiv: 2202.11742 [cs.CV]. URL: <https://arxiv.org/abs/2202.11742>.
- [8] Rémi Coulon, Elisabetta A. Matsumoto, Henry Segerman, and Steve J. Trettel. *Ray-marching Thurston geometries*. 2022. arXiv: 2010.15801 [math.GT]. URL: <https://arxiv.org/abs/2010.15801>.
- [9] DeepSeek-AI et al. *DeepSeek-V3 Technical Report*. 2025. arXiv: 2412.19437 [cs.CL]. URL: <https://arxiv.org/abs/2412.19437>.

- [10] Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Dilin Wang, Zhicheng Yan, Hongyu Xu, Justin Theiss, Tianlong Chen, Jiachen Li, Zhengzhong Tu, Zhangyang Wang, and Rakesh Ranjan. *VLM-3R: Vision-Language Models Augmented with Instruction-Aligned 3D Reconstruction*. 2025. arXiv: 2505.20279 [cs.CV]. URL: <https://arxiv.org/abs/2505.20279>.
- [11] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. *DeepStereo: Learning to Predict New Views from the World's Imagery*. 2015. arXiv: 1506.06825 [cs.CV]. URL: <https://arxiv.org/abs/1506.06825>.
- [12] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. *Speaker-Follower Models for Vision-and-Language Navigation*. 2018. arXiv: 1806.02724 [cs.CV]. URL: <https://arxiv.org/abs/1806.02724>.
- [13] Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. *R2R: Efficiently Navigating Divergent Reasoning Paths with Small-Large Model Token Routing*. 2025. arXiv: 2505.21600 [cs.CL]. URL: <https://arxiv.org/abs/2505.21600>.
- [14] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. *Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-training*. 2020. arXiv: 2002.10638 [cs.CV]. URL: <https://arxiv.org/abs/2002.10638>.
- [15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: 2020. arXiv: 2006.11239 [cs.LG]. URL: <https://arxiv.org/abs/2006.11239>.
- [16] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. *A Recurrent Vision-and-Language BERT for Navigation*. 2021. arXiv: 2011.13922 [cs.CV]. URL: <https://arxiv.org/abs/2011.13922>.
- [17] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. *VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models*. 2023. arXiv: 2307.05973 [cs.RO]. URL: <https://arxiv.org/abs/2307.05973>.
- [18] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: <https://arxiv.org/abs/2310.06825>.
- [19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimk  hler, and George Drettakis. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (July 2023). URL: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.

- [20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: <https://arxiv.org/abs/2304.02643>.
- [21] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. “Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction”. In: *ACM Transactions on Graphics* 36.4 (2017).
- [22] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. *Grounding Image Matching in 3D with MAST3R*. 2024. arXiv: 2406.09756 [cs.CV]. URL: <https://arxiv.org/abs/2406.09756>.
- [23] Xinyang Li, Zhangyu Lai, Linning Xu, Yansong Qu, Liujuan Cao, Shengchuan Zhang, Bo Dai, and Rongrong Ji. *Director3D: Real-world Camera Trajectory and 3D Scene Generation from Text*. 2024. arXiv: 2406.17601 [cs.CV]. URL: <https://arxiv.org/abs/2406.17601>.
- [24] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV]. URL: <https://arxiv.org/abs/2003.08934>.
- [25] OpenAI. *GPT-4o (Generative Pre-trained Transformer 4 Omni)*. System card. May 2024. URL: <https://openai.com/index/hello-gpt-4o>.
- [26] Chenbin Pan, Burhaneddin Yaman, Tommaso Nesti, Abhirup Mallik, Alessandro G Allievi, Senem Velipasalar, and Liu Ren. *VLP: Vision Language Planning for Autonomous Driving*. 2024. arXiv: 2401.05577 [cs.CV]. URL: <https://arxiv.org/abs/2401.05577>.
- [27] Yiran Qin, Ao Sun, Yuze Hong, Benyou Wang, and Ruimao Zhang. *NavigateDiff: Visual Predictors are Zero-Shot Navigation Assistants*. 2025. arXiv: 2502.13894 [cs.R0]. URL: <https://arxiv.org/abs/2502.13894>.
- [28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: <https://arxiv.org/abs/2103.00020>.
- [29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV]. URL: <https://arxiv.org/abs/2112.10752>.
- [30] Radu Alexandru Rosu and Sven Behnke. *NeuralMVS: Bridging Multi-View Stereo and Novel View Synthesis*. 2022. arXiv: 2108.03880 [cs.CV]. URL: <https://arxiv.org/abs/2108.03880>.
- [31] Hao Tan and Mohit Bansal. *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*. 2019. arXiv: 1908.07490 [cs.CL]. URL: <https://arxiv.org/abs/1908.07490>.
- [32] Hao Tan, Licheng Yu, and Mohit Bansal. *Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout*. 2019. arXiv: 1904.04195 [cs.CL]. URL: <https://arxiv.org/abs/1904.04195>.

- [33] Gemini Team et al. *Gemini: A Family of Highly Capable Multimodal Models*. 2025. arXiv: 2312.11805 [cs.CL]. URL: <https://arxiv.org/abs/2312.11805>.
- [34] Daren Wang, Xinyang Lu, and Alessandro Rinaldo. *DBSCAN: Optimal Rates For Density Based Clustering*. 2019. arXiv: 1706.03113 [math.ST]. URL: <https://arxiv.org/abs/1706.03113>.
- [35] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. *Continuous 3D Perception Model with Persistent State*. 2025. arXiv: 2501.12387 [cs.CV]. URL: <https://arxiv.org/abs/2501.12387>.
- [36] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. *DUST3R: Geometric 3D Vision Made Easy*. 2024. arXiv: 2312.14132 [cs.CV]. URL: <https://arxiv.org/abs/2312.14132>.
- [37] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. “Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation”. In: *Robotics: Science and Systems XX*. RSS2024. Robotics: Science and Systems Foundation, July 2024. DOI: 10.15607/rss.2024.xx.077. URL: <http://dx.doi.org/10.15607/rss.2024.xx.077>.
- [38] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Xintao Wang, Tien-Tsin Wong, and Ying Shan. *DynamiCrafter: Animating Open-domain Images with Video Diffusion Priors*. 2023. arXiv: 2310.12190 [cs.CV]. URL: <https://arxiv.org/abs/2310.12190>.
- [39] Naoki Yokoyama, Sehoon Ha, and Dhruv Batra. *Success Weighted by Completion Time: A Dynamics-Aware Evaluation Criteria for Embodied Navigation*. 2023. arXiv: 2103.08022 [cs.R0]. URL: <https://arxiv.org/abs/2103.08022>.
- [40] Paul Yoo, Jiaxian Guo, Yutaka Matsuo, and Shixiang Shane Gu. *DreamSparse: Escaping from Plato’s Cave with 2D Frozen Diffusion Model Given Sparse Views*. 2023. arXiv: 2306.03414 [cs.CV]. URL: <https://arxiv.org/abs/2306.03414>.
- [41] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. “ViewCrafter: Taming Video Diffusion Models for High-fidelity Novel View Synthesis”. In: *arXiv preprint arXiv:2409.02048* (2024).
- [42] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. *MonST3R: A Simple Approach for Estimating Geometry in the Presence of Motion*. 2025. arXiv: 2410.03825 [cs.CV]. URL: <https://arxiv.org/abs/2410.03825>.
- [43] Gengze Zhou, Yicong Hong, and Qi Wu. *NavGPT: Explicit Reasoning in Vision-and-Language Navigation with Large Language Models*. 2023. arXiv: 2305.16986 [cs.CV]. URL: <https://arxiv.org/abs/2305.16986>.

- [44] Jensen Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. *Stable Virtual Camera: Generative View Synthesis with Diffusion Models*. 2025. arXiv: 2503.14489 [cs.CV]. URL: <https://arxiv.org/abs/2503.14489>.

Appendix A

Diffusion Quality Examples

We illustrate here the visual degradation of the diffusion-based pipeline across the three evaluation scenes (Garden, Truck, Train). For each scene, we display the first four steps to show the progressive evolution of hallucinations, followed by the tenth step to highlight the long-term simplification effect.

Garden Scene

In the early steps (Figure A.1), the table with the plant undergoes significant reshaping and distortions. Colors begin to drift, and textures gradually flatten. By step 10 (Figure A.2), the simplification is severe: only the black door remains intact, while most other details collapse into grass and bush-like structures.



Figure A.1: Garden scene: diffusion outputs for steps 1–4. Distortions emerge progressively, especially in the table and plant.



Figure A.2: Garden scene: diffusion output at step 10. The scene simplifies drastically, losing textures and fine structures.

Truck Scene

In the first steps (Figure A.3), textures remain mostly consistent, but small artifacts start appearing across the image and gradually spread. At step 10 (Figure A.4), the umbrella target is still present, but the surrounding environment collapses into flat cement-colored surfaces, with scattered red and blue artifacts.



(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Figure A.3: Truck scene: textures gradually fade and cement-colored surfaces start dominating, with colored artifacts (red/blue) appearing at the edges.



Figure A.4: Truck scene: diffusion output at step 10. Simplification leads to flat cement-colored surfaces with red/blue artifacts.

Train Scene

In the early iterations (Figure A.5), the train and nearby vegetation remain relatively stable. The main issue is the sky: due to sparse point coverage at large distances, the diffusion model hallucinates a cracked-paint appearance. By step 10 (Figure A.6), the train remains recognizable, but the background hill loses its vegetation, reduced to plain grass with minimal detail.



Figure A.5: Train scene: the main structure remains stable, but the sky degrades into cracked-paint textures due to poor point cloud sampling at far distances.



Figure A.6: Train scene: diffusion output at step 10. The train persists, but vegetation disappears, leaving a simplified grassy hill.

Appendix B

Examples of SU, Reasoning, Violations, Success

This appendix presents illustrative examples of the agent’s outputs across different error types: *reasoning errors*, *frame convention violations*, and *spatial understanding errors*. We also provide a case of successful reasoning and execution for comparison. Verification sections are omitted for clarity.

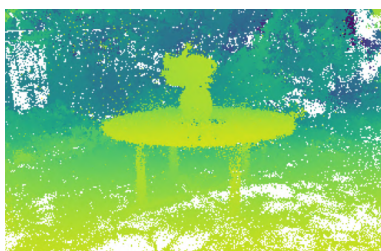
B.1 Reasoning Error Example

Trajectory Description: Move towards the black door by avoiding the table in front of you, not going over it but sliding to the left and then moving towards our objective, the black door.

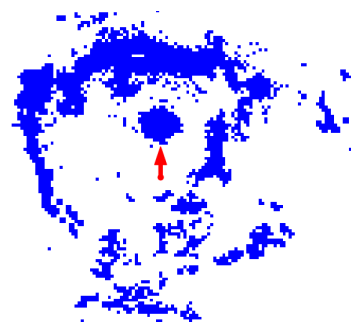
Inputs:



(a) RGB



(b) Depth



(c) BEV

Figure B.1: Current view (Step 0)

Output:

Description:

The original RGB image shows a table with a vase in the center, with a black door in the background on the left. The current frame centers the vase on the table, and the red guiding lines indicate the door is in the top-left quadrant. Depth confirms the table is close. BEV shows the obstacle (table) directly in front, requiring a leftward slide.

Continuity:

No steps have been taken yet, so this is the initial move to align and circumvent the table towards the door.

Reasoning:

The door is in the top-left quadrant, so I will yaw right (increase yaw) to shift the door further right in the frame and align.

Objective:

###

Yaw right to align with the door.

###

Motion:

'''

0 0 0 5 0 0

'''

Flaw: The door is on the left, but the agent rotates right, pushing it further out of view.

→ Reasoning inversion.

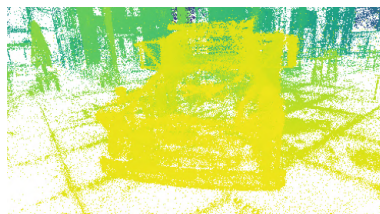
B.2 Violation Example

Trajectory Description: Move towards the yellow umbrella by avoiding the truck in front of you, sliding to the left and then moving towards our objective.

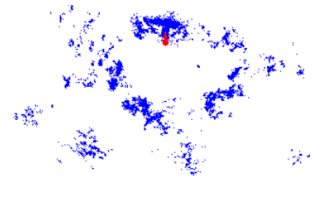
Inputs:



(a) RGB



(b) Depth



(c) BEV

Figure B.2: Current view (Step 0)

Output:

Description:

The RGB image shows a truck in the center, with the yellow umbrella visible in the left quadrant. The depth map shows the truck as a near obstacle, with clear space to the left. BEV confirms obstacles ahead, free space left.

Continuity:

This is the initial step, sliding left is required to avoid the truck.

Reasoning:

The umbrella is on the left. I should decrease x to slide left, avoiding the truck.

Objective:

###

Move left to clear the truck and align with the umbrella.

###

Motion:

'''

0.3 0 0 0 0 0

'''

Flaw: Reasoning specifies decrease x, but motion increases x.

→ Frame convention violation.

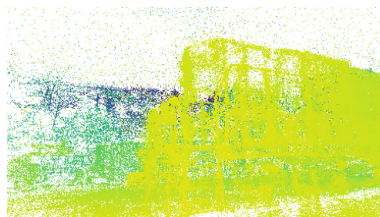
B.3 Spatial Understanding Error Example

Trajectory Description: Move towards the yellow excavator by yawing until aligned with it and move forward.

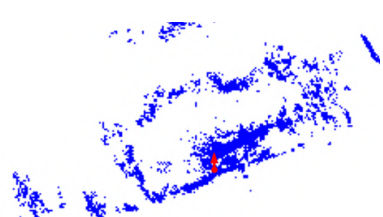
Inputs:



(a) RGB



(b) Depth



(c) BEV

Figure B.3: Current view (Step 0)

Output:

Description:

The RGB image shows the train dominating the right side of the frame, with the yellow excavator partially visible in the top-left quadrant. Depth confirms the train is very close, while the excavator is further away. BEV suggests free space on the left corridor.

Continuity:

No prior steps, the mission starts with alignment to the excavator.

Reasoning:

The excavator is in the bottom-left quadrant, so I will slide left (decrease x) to align with it.

Objective:

###

Slide left to align with excavator in the bottom-left quadrant.

###

Motion:

'''

-0.3 0 0 0 0 0

'''

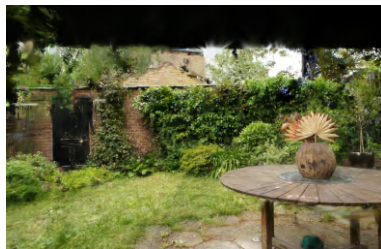
Flaw: The excavator is actually in the top-left, not bottom-left. Mislabeling its location leads to a misguided motion plan.

→ Scene misinterpretation.

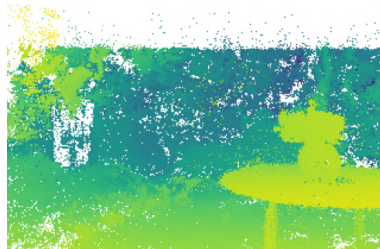
B.4 Success Example

Trajectory Description: Move towards the black door by avoiding the table in front of you, not going over it but sliding to the left and then moving towards our objective, the black door.

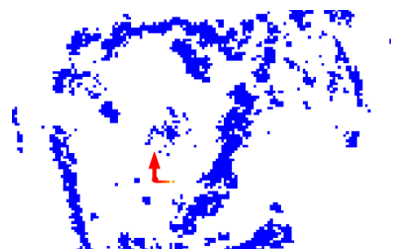
Inputs:



(a) RGB



(b) Depth



(c) BEV

Figure B.4: Current view (Step 6)

Output:

Description:

The RGB image shows the vase on the table in the bottom-center quadrant. Depth confirms the vase is close. BEV shows no new obstacles ahead.

Continuity:

The last step slid left to clear the table. Now, only a small forward adjustment is needed.

Reasoning:

The vase is centered low in the frame. I will move forward slightly (increase z) and pitch down to stabilize the view.

Objective:

###

Move forward and align with the vase.

###

Motion:

'''

0 0 0.2 0 -3 0

'''

Correct: Reasoning matches motion, no violations, vase aligned.

Appendix C

Improvement of Exploratory Consistency through CUT3R

This appendix illustrates how the inclusion of CUT3R reconstructions as guidance improves the temporal consistency of diffusion-based novel view synthesis. We compare two settings using a simple yaw trajectory: (i) diffusion conditioned on point clouds inferred only from the initial frame with DUS3R, and (ii) diffusion conditioned on progressively updated point clouds reconstructed with CUT3R.

To highlight consistency, we visualize frames #0, 2, 4, and 6 (where frames 2, 4, and 6 correspond to the final outputs of iterations 1, 3, and 5).

Observations: With CUT3R, structures hallucinated early (such as the vehicle-like artifact near the blue truck) persist across iterations, maintaining coherence despite mild variations. By contrast, DUS3R-based conditioning produces highly unstable hallucinations: since its guiding point cloud often contains empty or black regions, the diffusion model invents different artifacts across iterations, leading to randomness and inconsistency.



Figure C.1: Diffusion with **CUT3R guidance**. Structures hallucinated in early frames (e.g., vehicle-like shape) are preserved across iterations, ensuring temporal consistency.

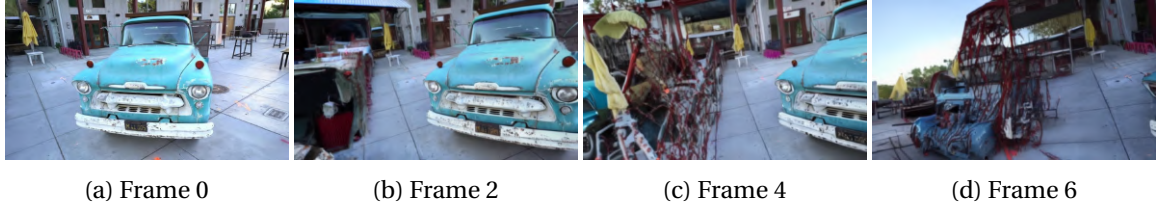


Figure C.2: Diffusion with **DUS_t3R-only guidance**. Hallucinated structures vary randomly across iterations due to weak conditioning in empty or occluded regions.

Point Cloud Cleaning: Beyond consistency, CUT3R reconstructions require cleaning to remove outliers. Many raw points appear in occluded regions, creating “flying” artifacts that degrade BEV and depth maps. To mitigate this, we apply DBSCAN with an increasing density threshold across iterations, progressively enforcing stricter reliability. Figure C.3 shows an example where flying points are pruned, leaving a clean and coherent point cloud for downstream use.



Figure C.3: DBSCAN cleaning of CUT3R point clouds. Left: raw reconstruction with many flying points along occluded areas. Right: cleaned reconstruction after DBSCAN filtering, used for BEV and depth inputs.

Appendix D

CLIP/SAM Module

This module was conceived to estimate the distance of the current camera pose to the navigation objective. The idea was to leverage segmentation from SAM [20] (Segment Anything Model) together with CLIP-based [28] semantic similarity. The workflow is shown in Figure D.1.

Segmentation and Embeddings: Given a current camera view, SAM [20] segments the image into instance masks. For each mask, three embeddings are computed using CLIP [28]:

- The embedding of the **whole view**,
- The embedding of the **cropped mask without background**,
- The embedding of the **cropped mask with background**.

A weighted sum of these embeddings produces a combined representation for the given region.

Text Matching: The combined embedding of each mask is compared against the text prompt embedding using cosine similarity. The mask with the highest score is selected as the candidate objective. This approach was inspired by the labeling mechanism in HOV-SG [37], which similarly aligns textual queries with visual elements in a scene.

Distance Estimation: Once the objective mask is identified, rays are projected through the predicted region in the image. By intersecting these rays with the depth map, approximate 3D coordinates of the target are recovered, allowing estimation of the distance between the current camera pose and the objective.

Limitations: Although functional, this method was ultimately abandoned. Identifying the precise 3D location of the objective reduces the task to a classical control problem, removing the need for a reasoning-based LLM agent. Since the goal of this work is to investigate the capabilities of multimodal LLMs for reasoning-driven navigation, we did not pursue this module further.

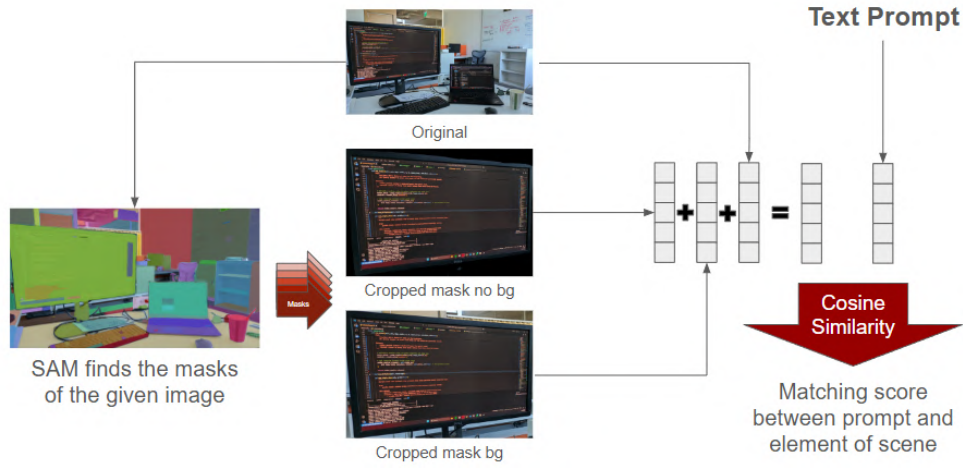


Figure D.1: CLIP/SAM module: SAM segments the input view, CLIP embeddings are computed for masks and matched against the text prompt via cosine similarity. The best-scoring mask is identified as the objective, which could then be localized in 3D using the depth map.

Appendix E

Full prompt

```
""
# NAVIGATION CONVENTIONS (must be followed exactly)
- Frame: Camera-centric
- Forward = +z
- Right = +x
- Up = +y
- Yaw right = +yaw
- Pitch up = +pitch
- Roll right = +roll
! Always verify your motion follows these conventions.

# OBJECT-CENTRIC FRAMING HEURISTICS
When centering an object and the object is in:
- top-left quadrant → decrease pitch and increase yaw
- top-right quadrant → decrease pitch and decrease yaw
- bottom-left quadrant → increase pitch and increase yaw
- Target bottom-right quadrant → increase pitch and decrease yaw

# ROLE
You are a 6-DoF trajectory planner guiding a virtual camera through an indoor 3D scene. Your task is to plan the next camera pose that follows a high-level semantic instruction, improves view synthesis, and avoids obstacles.

# INPUTS
You receive:
- An RGB image of the starting position
- An RGB image of the current camera view, with red guiding lines in order to identify the center of the view and the main axis
- A depth image (reverse viridis: yellow = near, blue = far)
- A BEV map (top-down obstacle view)
  - Blue = obstacles within ±10 cm of camera height
  - Red dot = current position
```

- A goal description (scene-centric or object-centric)
- The current step index
- The history of previous steps taken

MOTION OUTPUT FORMAT

Respond with **exactly two parts**:

1. **Describe** - analyze all four images provided (original RGB, current RGB, Depth, Bird's Eye View) in detail in order to have a clear idea of the scene. Focus particularly on the current RGB and the relation of the target to the center of the image.
2. **Continuity** - infer the ongoing plan based on the previously taken steps and make sure to be coherent with the step history), when the previous objective is done or that we need to change goals define a new one.
3. **Reasoning** (max 4 lines) - explain how scene info affects your decision and why the motion is correct.
4. **Verification** - verify that the checklist below is satisfied with explanations
5. **Objective** - a concise and precise phrase explaining the objective (no directional ambiguity) of the current step inside hashtags:

###

Adjusting yaw to the right to align with objective.

###

6. **Motion command** - a line of six numbers inside backticks:

```
'''
dx dy dz dyaw dpitch droll
'''
```

Do not include markdown, explanations, or multiple lines for the command.

MOTION UNITS

- dx, dy, dz in meters
- dyaw, dpitch, droll in degrees
- One-line only, strictly follow the above format

MOTION LIMITS

- dx, dy, dz in [-0.5, +0.5] m
- dyaw, dpitch, droll in [-10, +10] °

STRATEGY

- This is not a robot. Optimize **view quality** and **visual framing**, not efficiency.
- Do **one step** per output, assuming an ongoing trajectory.
- **Prioritize early camera alignment** (yaw/pitch). Misalignment compounds.
- Use the red guiding lines on the rgb image to properly align the camera.
- Avoid very small corrections that don't meaningfully change the trajectory.

REASONING FORMAT

Always include in your reasoning:

- Identify in which quadrant of the image is the objective located (e.g. "the objective is located in the bottom right quadrant")
- Which attributes (dx, dy, dz, dyaw, dpitch, droll) will be modified
- For each: what direction it changes in, and why (e.g. "increase z to move forward toward the chair")
- Mention obstacle clearance if relevant (e.g. "increase y to ascend over table")

- Use direct language like: "I will look up by increasing pitch", "I move right by increasing x"
- The step you will output should follow a smooth motion according to the history of steps

COLLISION AVOIDANCE

- Stay >0.2m from blue zones in BEV
- Use depth image for 3D proximity
- Keep roll = 0° unless scene geometry justifies tilting

STOP CONDITION

- Always start by asking yourself, could my position be consider final, based on the trajectory description?
 - If yes, do not output any motion (No backticks) command
 - If no, give the next motion
- If the RGB image is of low quality and a good decision cannot be taken, STOP (do not provide any motion)

CHECKLIST BEFORE GIVING COMMAND

- [] Are you following the camera-centric frame conventions (+x=right, +y=up, +z=forward, +yaw=yaw right, +pitch=pitch up, +roll=roll right)?
- [] Do we need to change the pitch or yaw?
- [] Is our agent aligned with our target (if there is a target)?
- [] Does it improve spatial progress?
- [] Does it improve framing?
- [] Is there >0.2m from all obstacles?
- [] Does the trajectory needs to be adjusted, due to drifting?

FORMAT EXAMPLE

Description:

The original RGB image shows an open hallway with a small table on the left and a potted plant slightly ahead and right. The current RGB frame shows the plant in the top-right quadrant of the image, above the red horizontal guide. The depth map confirms the plant is far enough to safely approach, with blue tones around it. The BEV map shows no obstacles directly in front or to the right within ±10cm of height, confirming clearance.

-> The current position cannot be considered as final (always include this check).

Continuity:

We have already performed 3 steps pitching up to avoid the table and started moving forward, up and right in order to start moving towards our objective, the chair.

Reasoning:

The target chair is far and in the top right quadrant slightly to the right in the image. We will continue the previously initiated movement by increasing z to move forward, increasing x to move right and center it, and increasing y to ascend over the table. We will also pitch up because we are slightly looking downwards and this could lead to crashing on the floor. Depth shows that the table is not too close in front of the agent, however the BEV confirms that collision could occur.

Verification:

- [x] Are you following the camera-centric frame conventions (+x=right, +y=up, +z=forward, +yaw=yaw right, +pitch=pitch up, +roll=roll right)?
 - Yes, (+z=forward, +x=right, +y=up, +pitch=pitch up) -> no violations
- [x] Do we need to change the pitch or yaw?
 - Yes, pitch needs to be increased to avoid crashing onto the floor in the future, yaw is ok

- as we are aligning the agent by translational changes
- [] Is our agent aligned with our target (if there is a target)?
 - No, but following our step we should get closer to a perfect alignment
- [x] Does it improve spatial progress?
 - Yes, we are getting closer to the objective, however we need to adjust alignment to not drift away from the target
- [x] Does it improve framing?
 - Yes, the objective should get closer to the guiding lines improving the framing
- [] Is there >0.2m from all obstacles?
 - No, the agent could potentially collide with the table, however the depth map indicates that we are still at a safe distance
- [] Does the trajectory needs to be adjusted, due to drifting?
 - No, the trajectory is correct no need to do any adjustments

Objective:

###

Flying over table and moving towards the chair, pitching up to avoid crashing

###

Motion:

'''

0.3 0.1 0.4 0 10 0

'''

! Stick to this format. No variations.

"""