

GeniNav: Generative Model Driven Image-Goal Navigation via Imagination-Guided Consistency Flow Matching

Supplementary Material

1. Details of LGM

1.1. Architecture

LGM takes the current observation image, goal image, and a brief navigation prompt as input. We employ Qwen2.5-VL-7B as the multimodal backbone. After the images and prompt are jointly encoded by the model’s vision-language transformer, we extract the final-layer hidden states corresponding to visual tokens, forming a cross-modal representation $H_{\text{vis}} \in \mathbb{R}^{B \times N \times 3584}$. These tokens capture spatial visual cues enriched by goal- and prompt-induced attention. We then apply a linear projection layer (3584→512) to reduce the VLM embedding dimension while preserving cross-modal structure, yielding $H_{\text{proj}} \in \mathbb{R}^{B \times N \times 512}$. To further enhance spatial semantic alignment, the projected tokens are refined by a lightweight transformer encoder-decoder module with a hidden size of 512. This produces the refined representation $H_{\text{ref}} \in \mathbb{R}^{B \times N \times 512}$. Finally, we mean-pool the refined tokens along the token dimension and pass the resulting vector through a two-layer MLP to obtain the subgoal latent $z_s \in \mathbb{R}^{512}$, which conditions the GeniPolicy for continuous navigation control.

1.2. Loss Functions

LGM is trained with three complementary auxiliary objectives that explicitly shape the latent space and encourage it to capture subgoal-relevant semantics and spatial structure.

Semantic alignment. To endow the latent with predictive semantic structure, we adopt a temporal self-distillation scheme. We construct a teacher branch that is a frozen copy of the entire LGM at initialization, including the projection layer, the refinement transformer, and the MLP. For each training sample, the teacher replaces the current observation with a future observation from the same trajectory, while sharing the same goal image and navigation prompt. The frozen VLM and the frozen teacher LGM process this triplet to produce a future-oriented latent representation $h_T \in \mathbb{R}^{512}$. The student branch, which shares the same architecture but remains trainable, produces the current-view latent $z_s \in \mathbb{R}^{512}$. We regress z_s toward the

teacher latent using an ℓ_2 loss:

$$\mathcal{L}_{\text{sem}} = \|z_s - h_T\|_2^2 \quad (1)$$

This predictive self-distillation encourages z_s to encode the semantics of future scene context and aligns the student latent space with temporally informative visual cues.

Directional embedding. To impose spatial consistency, we discretize the relative heading angle ϕ into N_θ bins. An MLP classifier predicts $p_\theta(\phi | z_s)$, supervised with cross-entropy:

$$\mathcal{L}_{\text{dir}} = \text{CE}(p_\theta(\phi | z_s), \phi^*) \quad (2)$$

Distance embedding. Similarly, the radial distance r between the current and goal viewpoints is discretized into N_r bins. An MLP classifier predicts $p_\theta(r | z_s)$:

$$\mathcal{L}_{\text{dist}} = \text{CE}(p_\theta(r | z_s), r^*) \quad (3)$$

Contrastive regularization. To prevent collapse and encourage instance-level discriminability in the latent space, we apply an InfoNCE loss over the latent z_s . For each sample we generate a positive pair (z_s, z_s^+) using independent dropout perturbations, and treat latents from other samples in the batch as negatives:

$$\mathcal{L}_{\text{NCE}} = -\log \frac{\exp(\text{sim}(z_s, z_s^+)/\tau)}{\sum_j \exp(\text{sim}(z_s, z_s^{(j)})/\tau)} \quad (4)$$

Overall objective. Following the notation in the main paper, the full LGM loss is:

$$\mathcal{L}_{\text{LGM}} = \lambda_{\text{sem}} \mathcal{L}_{\text{sem}} + \lambda_{\text{dir}} \mathcal{L}_{\text{dir}} + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}} + \lambda_{\text{ncc}} \mathcal{L}_{\text{NCE}} \quad (5)$$

We set $(\lambda_{\text{sem}}, \lambda_{\text{dir}}, \lambda_{\text{dist}}, \lambda_{\text{NCE}}) = (1.0, 0.5, 0.5, 0.1)$, which balances the scale of the loss terms and follows common practice in predictive self-distillation and contrastive regularization.

2. Training Procedure of GeniNav

Following the two-stage training schedule described in the main paper, we first pretrain the LGM in isolation and then

Table 1. Training configuration of the GeniPolicy module used in joint optimization.

Parameter	Value
Number of flow segments K	4
EMA decay β_{EMA}	0.999
Noise samples	5
Optimizer	AdamW
Learning rate	5×10^{-5}
Batch size	32
Gradient clipping	1.0
Training epochs	50
Weight decay	1×10^{-4}
Warmup ratio	0.03

jointly optimize it together with GeniPolicy. Below we provide the detailed implementation corresponding to these two stages.

2.1. Stage 1: LGM Pretraining

We first pretrain the LGM with the Qwen2.5-VL backbone kept fully frozen. Only the projection layer, refinement transformer, and auxiliary prediction heads are updated. The model is trained with the complete latent loss \mathcal{L}_{LGM} for 10 epochs.

2.2. Stage 2: Joint Optimization with GeniPolicy

After LGM pretraining, we jointly optimize the LGM and the policy module following the GeniPolicy objective. To stabilize training, We fine-tune Qwen2.5-VL using LoRA adapters applied to its cross-attention layers, while keeping all backbone weights frozen. On the LGM side, we update only the trainable components including the LoRA adapters, the projection layer, the refinement transformer, and the auxiliary heads. During joint optimization, the policy is trained with the standard GeniPolicy objective, while the LGM continues to receive its auxiliary supervision. The overall training loss is:

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{GeniPolicy}} + \mathcal{L}_{\text{LGM}} \quad (6)$$

GeniPolicy Training Configuration. For clarity and reproducibility, we summarize all hyperparameters used for optimizing the GeniPolicy module during joint training. These include flow segmentation settings, EMA updates, temporal sampling strategy, and action horizon specifications. The full training configuration is provided in Table 1.

2.3. HRM Hyperparameters

The HRM determines the final trajectory by jointly evaluating geometric safety, semantic alignment, and visibility. For geometric safety, each 3D point on a candidate trajectory

Table 2. Hyperparameters used in the HRM.

Parameter	Value
Safety threshold δ	0.15 m
Maximum depth D_{max}	5.0 m
Number of rays M	60
Semantic weight λ_1	0.7
Visibility weight λ_2	30
View sampling range	$\pm 30^\circ$

is projected into the current depth map, and any trajectory that presents a positive depth violation is removed. Trajectories that pass this check are further evaluated by GPT-4V api, which receives the current observation, the goal image, and the rendered trajectory overlay, and returns a semantic alignment score in the range $[0, 100]$. Visibility is assessed at the terminal pose by sampling rays around the final heading and aggregating their valid depth values relative to the sensor’s maximum range. The final ranking score is obtained by a weighted combination of the semantic score and visibility score, and the trajectory with the highest combined score is selected for execution. All hyperparameters used by the HRM are summarized in Table 2.

3. Additional Details for GeniBench

This section provides additional implementation details and visualizations for GeniBench beyond the main paper. We describe the full trajectory generation pipeline, filtering rules, simulator configurations, and extended dataset statistics, and we provide additional qualitative examples of the generated trajectories.

Trajectory Generation Pipeline. Algorithm 1 presents the complete procedure used to generate trajectories in GeniBench. For each scene, we load the precomputed NavMesh and apply obstacle inflation based on the robot size. Randomly sampled start and goal viewpoint pairs are then passed to Hybrid A* to obtain candidate geometric paths. Each path is simulated under a differential-drive kinematic model and validated through collision checking, retaining only those that remain collision-free and exhibit smooth curvature. Finally, the selected path undergoes cubic spline smoothing and uniform temporal resampling to produce the final trajectory used in the benchmark.

Filtering, Validation, and Simulator Configuration. During trajectory generation, we first perform strict filtering and validation on all candidate paths. Trajectories shorter than 2 m, containing turning radii below the robot’s physical minimum, or intersecting any inflated obstacle boundary are discarded. We further compute the local clearance along each path and remove trajectories whose minimum clearance falls below 0.08 m. All retained trajectories are paired



Figure 1. Additional qualitative visualizations of GeniBench trajectories across diverse indoor layouts, including narrow corridors, multi-room apartments, and cluttered living spaces.

Algorithm 1: Trajectory Generation Pipeline

Input: Scene S , robot footprint F , safety margin m

Output: Valid trajectory τ

1. construct 2D NavMesh N from scene S ;
 2. inflate all obstacle regions in N by margin m to match F ;
 3. sample a pair of viewpoints $(v_{\text{start}}, v_{\text{goal}})$;
 4. $P \leftarrow \text{HybridA}^*(N, v_{\text{start}}, v_{\text{goal}})$;
 5. $C \leftarrow \emptyset$;
 6. **for each** path p in P **do**
 7. $\tau_{\text{sim}} \leftarrow \text{forward_simulate}(p, \text{robot_kinematics})$;
 8. **if** $\text{has_collision}(\tau_{\text{sim}}, \text{depth_map})$ **then continue**;
 9. **if** $\text{not curvature_continuous}(\tau_{\text{sim}})$ **then continue**;
 10. $C \leftarrow C \cup \{\tau_{\text{sim}}\}$;
 11. **end for**
 12. select τ_{raw} from C ;
 13. $\tau_{\text{smooth}} \leftarrow \text{cubic_spline_smooth}(\tau_{\text{raw}})$;
 14. $\tau \leftarrow \text{temporal_resample}(\tau_{\text{smooth}})$;
-

return τ

with synchronized RGB-D frames sampled at 10 Hz, ensuring precise alignment between visual context and geometric structure. All data are generated using Habitat-Sim with physics disabled but realistic sensor noise enabled. RGB-D observations are rendered at a resolution of 256×256 with a 90° horizontal field of view, and depth values are clipped to 5 m. Control is applied at 1 Hz with a maximum linear velocity of 0.25 m/s and an angular velocity limit of $60^\circ/\text{s}$. To ensure full reproducibility, we fix all random seeds throughout the data generation process.

Dataset Statistics and Additional Visualizations.

GeniBench contains 491.6 km of collision-free trajectories across 176 indoor scenes, with an average length of 2.79 m

and a distribution where short paths are common but longer navigation routes also appear frequently enough to capture realistic variability. Most trajectories exhibit smooth curvature transitions due to the kinematic constraints imposed by Hybrid A*. Figure 1 provides additional qualitative visualizations of trajectories across diverse layouts, including narrow corridors, multiroom apartments, and cluttered living spaces. These examples further highlight the geometric diversity, scene complexity, and kinematic consistency of trajectories in the benchmark.

Dataset Structure. GeniBench stores data on a per-trajectory basis, where each trajectory corresponds to a JSON file containing a sequence of step-level records. Each step includes the visual observations, robot states, and kinematic information required for navigation. The dataset directory is organized into five subfolders: `rgb_images/`, `depth_images/`, `goal_images/`, and `episodes/`.

For each step, we store the current RGB image, depth image, and goal image; the robot’s 3D position, orientation, and yaw; the next-step position and yaw generated by Hybrid A* rollout; as well as kinematic quantities such as velocity, acceleration, turning radius, and centripetal acceleration. Additional navigation indicators, including distance to goal, path progress, and floor index, are also recorded. This structured design provides a complete visual, geometric, and kinematic context for multimodal learning and policy supervision. We will publicly release the full GeniBench dataset and preprocessing code after publication to support reproducibility and future research in generative navigation.