Learning Safe Humanoid Navigation from Reduced Order Models

William D. Compton[1], Zachary Olkin[1], Ryan Bena[2], Aaron D. Ames[1,2]

***Abstract*— Research in humanoid robotics has achieved rapid progress in locomotion, and recent results have pushed the boundary on autonomous navigation. We demonstrate that a standard single-stage RL navigation pipeline struggles to scale to multi-level and multi-story terrain, limited by the difficulty of complex humanoid terrain interactions such as stairs. To overcome this challenge, we decompose the navigation problem into two pieces. First, we train a policy operating on the reduced order dynamics but with full 3D LiDAR observations to navigate complex, multi-story terrain. We then utilize this navigation knowledge to kickstart a policy operating on the full-order humanoid dynamics, with a frozen locomotion policy in the loop. Additionally, we demonstrate that applying a Poisson safety filter to the navigation policy output recovers safety in the presence of out-of-distribution obstacles, without dropping navigation success rate. We demonstrate the resulting RoM-Nav policy on a Unitree G1, accomplishing mapless multi-floor navigation covering trials with over 10 m of vertical displacement and over 100 m of path length. Project page with videos https://wdc3iii.github.io/rom-nav/.**

I. Introduction

Safe navigation of complex, real-world environments is a fundamental capability for humanoid robots. Any task requiring a robot to have legs almost certainly contains a navigation component, and will likely challenge the robot to handle terrain that is not flat, such as sidewalk curbs or staircases. Recent advances in perceptive humanoid locomotion have unlocked impressive new capabilities, including running [1], stairs [2], [3], stepping stones [4], parkour [5], and loco-manipulation [6]. As locomotion controllers become more agile, navigation planning becomes increasingly nontrivial; success becomes intertwined with the capabilities of the locomotion controller, which can be difficult to represent in a useful way for the navigation planner.

Reinforcement learning (RL) based navigation has provided a way to meet this challenge. RL provides many benefits; its lightweight runtime computation removes the complexity of real-time optimization. It can integrate directly with sensor data, removing the need for occupancy or traversability estimation. It learns directly to handle complex and terrain-dependent system dynamics, rather than requiring an explicit model or capability envelope. However, it comes with its own set of challenges. Sparse rewards and long-horizon tasks can lead to training instability or exploration collapse. Limitations in useful memory can handicap long-horizon performance, and sim-to-real gaps can lead to safety violation, especially if training geometry is simplistic.

[1]The authors are with the Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA.

[2]The authors are with the Amazon Safe Autonomy Frontiers (SAF) Lab.

This research is supported by Technology Innovation Institute (TII).

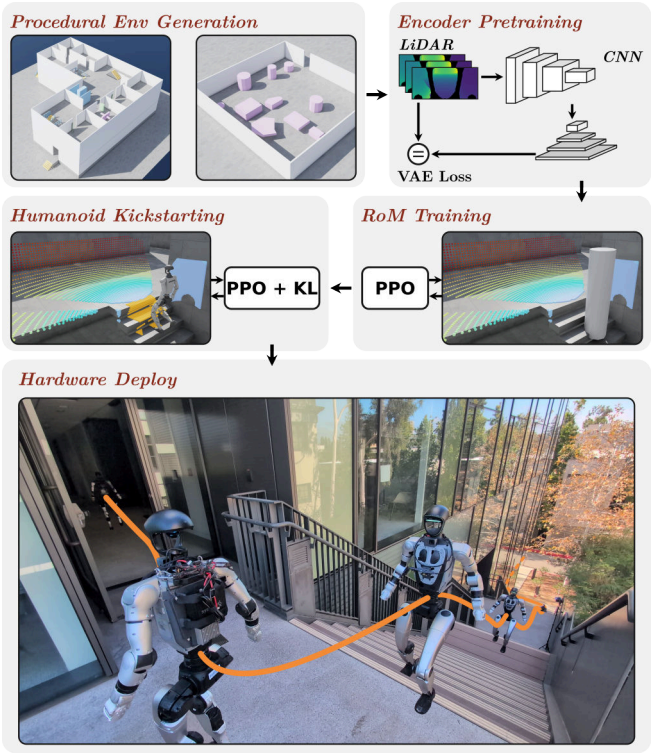

Fig. 1. Training and deployment of the RoM-Nav policy for mapless multi-floor humanoid navigation. The humanoid autonomously traverses stairs covering multiple floors (over 10 m of vertical displacement), and long-horizon, highly non-convex paths, exceeding 100 m in length.

To address these limitations, we propose a two-stage training paradigm, where a navigation policy is first trained on a reduced order model (RoM) of the humanoid. By trivializing the complexity of the environment interaction, the policy trained on the reduced order model quickly achieves superior performance to single-stage navigation policies. Then, with a combined objective of matching the RoM policy's action distribution and a standard PPO objective, we use the navigation capabilities of the RoM to kickstart a policy which handles the complexities of the terrain interaction and full-order dynamics of the humanoid under its frozen locomotion policy. We demonstrate improvement over single-stage training and show the gap surfaces almost entirely in problems requiring significant z-height conditioning, for instance cross-floor goals in multi-story buildings.

Finally, we demonstrate these navigation behaviors on hardware to navigate complex real-world terrain, both outdoors and in multi-floor buildings. Critically, we leverage Poisson safety filters [7], [8] online to bridge environmental sim-to-real gaps, such as complex obstacle geometries, which are difficult to capture during training for an unknown

deployment environment. We show that Poisson safety filters ensure safety when encountering out-of-distribution (OOD) obstacle geometry, while maintaining navigation success rate.

Concretely, our contributions are as follows:

- **Multi-Story Mapless Humanoid Navigation.** We deploy a mapless RL navigation policy on hardware to safely accomplish challenging navigation problems in real-world environments, including outdoor deployment of 100 m, and indoor tasks spanning multiple floors. To our knowledge, this is the first published humanoid navigation work to demonstrate learned mapless navigation for cross-floor goals in multi-story buildings.
- **Reduced Order Model Kickstarting for Navigation.** We propose a two-stage navigation reinforcement learning pipeline (Fig. 2): a policy is first trained to navigate a RoM, which kickstarts a policy operating on the humanoid. We demonstrate effective spawn/goal sampling and LiDAR encoder pretraining which unlock multi-story navigation capabilities.
- **Online Poisson Safety Filter for OOD Obstacles.** We demonstrate degradation of the policy when faced with out-of-distribution obstacles, and leverage the Poisson safety filter to cheaply ensure safety on hardware while maintaining navigation success rate.

II. Related Work

We review navigation for humanoids, learned navigation for legged robots, the choice of dynamics to train against, and runtime safety for learned policies.

Humanoid Navigation: Early work in humanoid navigation relied on model-based methods, be it kinematic [9], dynamic [10], reduced order [11], or sampling-based [12]. These methods typically perform heavy preprocessing of the environment, such as floor and obstacle height maps [13] or convex steppable regions to optimize over [14]. Modern extensions of these ideas have injected learned elements such as traversability estimates [15], [16], [17] or even forward dynamics models [18] to relieve the perception bottleneck, while maintaining the model-based backbone. These methods address local navigation, limited by planning horizon and the convexity of the regions they optimize over, unless paired with waypoints or a global map [12], [15], as in recent work navigating stairs on a humanoid [3]. Rather than optimize over a preprocessed environment, a large body of work learns navigation commands directly from sensor data. For humanoids, these efforts have targeted local navigation, such as goal-conditioned locomotion across constrained height-varying terrain [19], waypoint-guided obstacle avoidance on stairs and slopes [20], or teacher-distilled obstacle avoidance [21]. Vision-language models have also reached humanoids, but interface through parameterized motion primitives [22] or waypoints for a local planner over a SLAM map [23], and so inherit the same local or map-based structure. Humanoid navigation to date, model-based or learned, has therefore been primarily local or reliant on a map.

Learned Navigation: Learned navigation is considerably more mature on quadruped and wheeled-legged robots, with significant work spanning from training navigation and locomotion simultaneously and end-to-end [24], to hierarchical policies commanding a separately trained locomotion controller [25], [26], [27], to long-horizon deployments with explicit memory [28] or specialized recurrent structures [29]. Our single-stage pipeline follows this last line of work. Across all of these learned approaches, humanoid and quadruped alike, the training environments contain a single traversable level, so that the height of the goal carries no information its planar position does not; where stairs and slopes appear, they are terrain features to be crossed en route. Spawns and goals are sampled uniformly over the traversable area, and curricula, where present, scale terrain difficulty rather than goal placement [24]. To our knowledge, no learned, mapless humanoid navigation policy has been commanded to a goal on a different floor of a building.

Navigation Dynamics Model: These works also differ in the fidelity of the dynamics the navigation policy is trained against. Navigation at scale was first achieved on a point robot with discrete actions [30], and imperative planners learn paths over a cost map without simulating the robot's dynamics [31], [32]. Truong et al. make the case explicitly, showing that lower fidelity simulation yields higher sim-to-real transfer for navigation on a quadruped [33], and kinematically trained navigation policies have since been paired zero-shot with learned locomotion controllers [34]. While these works transfer zero-shot, they are evaluated in environments without complex terrain or multiple stories; we find that the low-fidelity approach works for same-level goals but breaks for cross-floor goals that require non-trivial terrain traversal (Section IV), and so add a second training stage on the full humanoid dynamics. NavRL++ also trains in multiple stages, but its secondary stages introduce sensing and actuation perturbations for sim-to-real transfer rather than higher-fidelity dynamics [35]. Teacher-student training in legged robotics is typically observation-privileged, distilling a teacher with privileged terrain and state information into a student with onboard sensing [36]; we instead kickstart [37] from a dynamics-privileged teacher, whose observations are a subset of the student's but which was trained on a simpler system.

Safety Filters: Regardless of the training regime, learned navigation policies carry no collision guarantee, and runtime safety filters are commonly layered on top of them. For legged navigation, learned reach-avoid value functions have filtered high-speed quadruped policies [38], reachability-based filters with disturbance estimation have been deployed from LiDAR in unknown environments [39], and velocity-obstacle shields have filtered RL policy outputs on aerial and legged platforms given tracked obstacles [35]. Each depends on a learned value function, a disturbance model, or an obstacle detector. Poisson safety functions synthesize a control barrier function directly from raw occupancy, requiring none of these, and have been paired with model-based planners on humanoids [7], [8]. We evaluate this filter on a learned humanoid navigation policy against obstacles chosen to lie outside its training distribution.

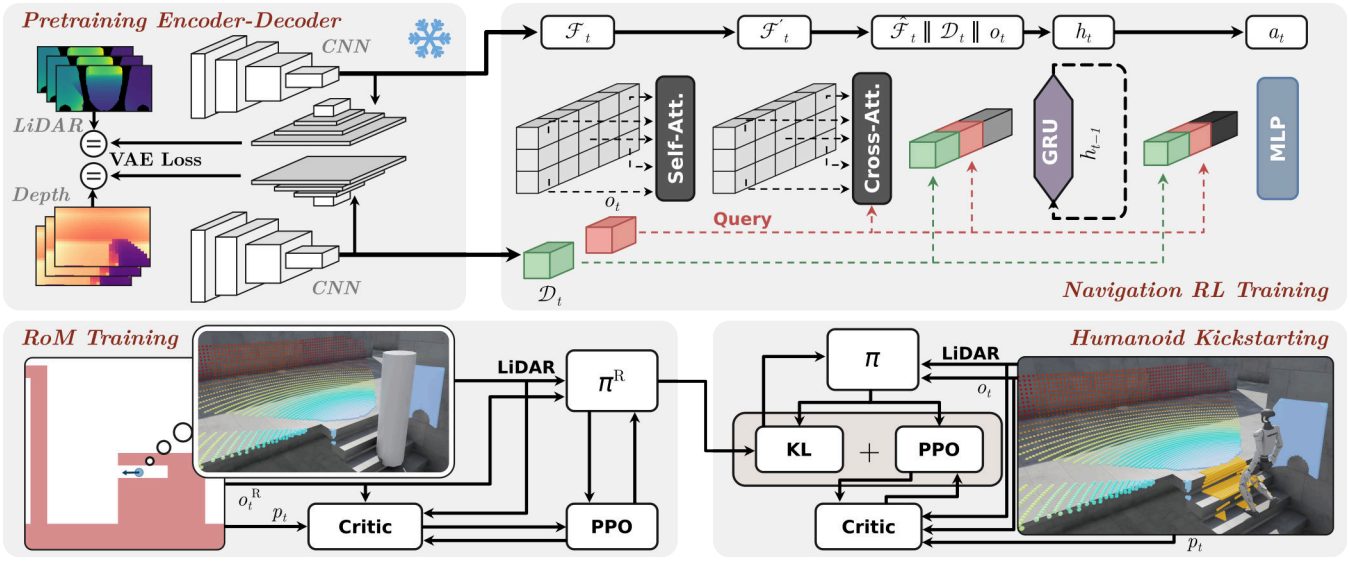

Fig. 2. RoM-Nav architecture. LiDAR and Depth inputs are processed via a pretrained CNN encoder. LiDAR passes through self- then cross-attention. The resulting embeddings are processed by a GRU unit and mapped to actions via an MLP head. A RoM policy is trained first, which is then used to kickstart the RoM-Nav policy by balancing KL divergence with a PPO objective on the humanoid with frozen locomotion controller.

III. Methods

The navigation policy is trained for deployment on a Unitree G1 equipped with a Mid-360 LiDAR and a downward-facing ZED Mini depth camera. The frozen locomotion policy for the robot is trained using LIP-CLF RL [3].

A. Pretraining Vision Encoders

The LiDAR range image and depth image are both high-dimensional inputs which require significant visual processing to extract useful information. Rather than extract these features during the RL training, we pretrain a CNN encoder for both the range image and depth image using a standard VAE approach [40]. Encoder pretraining for depth has been used in navigation RL [25], [29]; the LiDAR case differs in FoV and noise structure and has not been ablated in a humanoid navigation context. The VAE contains a set of convolution heads to project the image down to a Gaussian latent space. A set of deconvolution layers maps back to a full resolution image. We train heads to accomplish depth, XYZ, and valid mask reconstruction, and ground/obstacle/stair/ramp segmentation, to ensure retention of critical features in the VAE latent. Depth and XYZ reconstruction use an MSE loss; masking and segmentation use BCE. The VAEs are trained with synthetic images sampled from randomized robot poses. Stairs and ramps are over-sampled due to their spatial sparsity in the tiles. During VAE training, the inputs are noised, pixel dropout between 0–80% is applied, and variations on the robot's head occlusion mask are applied. VAEs are trained for 10 epochs on 1M uniform and 1M stair, ramp oversampled images. After training the encoder is frozen in navigation policy, as depicted in Fig. 2.

B. Policy Architecture

The observations passed to the navigation policy are a LiDAR range image, a depth image, and a nonvisual observation. The range image bins the LiDAR into 1° angular pixels containing distance, x, y, z in the LiDAR frame, and validity channels. The depth image is down-sampled from the ZED Mini depth to 26×30. The goal observation is encoded as $o_g = [d_x, d_y, \log \|d\| / \log d_{\max}, d_z, \sin(\gamma), \cos(\gamma)]$ where $d \in \mathbb{R}^3$ is the relative location of the goal and γ is the relative heading between the goal heading and the robot heading. The non-visual observation is $o_t = [o_g, \tau, \omega, p_g, a_{t-1}]$, with τ the locomotion phase, ω and p_g the angular rate and projected gravity of the pelvis link, and a_{t-1}, the previous action.

The policy architecture, pictured in Fig. 2, leverages two vision encoders for the LiDAR range image and depth image. The LiDAR passes through a frozen CNN encoder, a layer of self-attention, and a layer of cross-attention with four learned queries and one query encoding the nonvisual observations. The depth passes through a pretrained CNN encoder and is flattened. The vision embeddings are concatenated with the nonvisual encoding, and fed into a two-layer GRU, whose recurrent structure gives the policy memory to compensate for partial observability and to support the exploration required for navigation problems. The resulting recurrent embedding h_t is concatenated again with the visual and nonvisual embeddings, to pass through a dense layer mapping to velocity outputs, parameterized as Beta distributions, a common bounded distribution for navigation RL [28].

We claim no contribution in architecture; ours follows [29]. We use a GRU in place of the SRU, as the GRU is more common and preliminary experiments found no practical difference, likely due to the wider FoV of the LiDAR.

C. RoM Kickstarting

The proposed method kickstarts [37] the final navigation policy from one trained on a single integrator with heading, depicted in the bottom half of Fig. 2, with dynamics:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + (v_{x,t} \cos\theta_t - v_{y,t} \sin\theta_t)\Delta t \\ y_t + (v_{y,t} \cos\theta_t + v_{x,t} \sin\theta_t)\Delta t \\ \theta_t + \omega_{z,t}\Delta t \end{bmatrix}.$$

TABLE I: Navigation reward terms. † indicates geodesic gating.

Term	w	Per-step value r
Goal tracking (dense)		
position†	0.2	$1+\exp(-d_{xy}^2/0.5^2)+\exp(-d_{xy}^2/0.1^2)$
heading†	0.2	$\exp(-e_{yp}^2/0.5^2)$
height	0.2	$\exp(-z_e^2/2.5^2)+\exp(-z_e^2/0.5^2)$
stand†	0.2	$\exp(-\|a_t\|^2/0.1^2)$
Progress shaping (CLF, best-gated)		
geodesic	0.1	$\left[\frac{\Delta d_{\text{geo}}}{\min\{0.8 d_{\text{geo}},\, \overline{v}_{xy}\Delta t\}}\right]_0^1$
height	0.1	$\left[\frac{\Delta z_e}{\overline{v}_z \Delta t}\right]_0^1$
Penalties		
collision	-1	# bodies in contact
action rate	$-0.01 \to -0.1$	$\|a_t - a_{t-1}\|^2$

The environment for the single integrator is an occupancy grid derived from the 3D environment for the humanoid (see Section III-D). When a command leads to an occupied cell, the penetrating component is projected out, sliding along the occupied boundary. The RoM's LiDAR is placed at the humanoid's standing pose above the terrain.

The RoM trains with only LiDAR, goal, and previous action, the signals relevant to RoM navigation. Including the depth observation made no difference in performance, likely as the RoM dynamics are not terrain-dependent. The RoM policy is trained for 2000 iterations using PPO, with 4096 environments on one H100 GPU in 12 hours.

Once the RoM policy is trained, it kickstarts a policy using the architecture in Fig. 2. Kickstarting uses a weighted combination of the PPO objective and the KL divergence between the RoM policy and the RoM-Nav policy [37],

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda \mathcal{L}_{\text{KL}}(\pi, \pi^R).$$

The weight λ is 1 for the first 100 iterations, then decays to 0.05 by 1100 iterations, where it holds until training is finished at 2000 iterations. The RoM-Nav policy is trained with 4096 environments on a single H100 GPU, taking 32 hours. The combined training is under 45 hours on one GPU.

D. RL Environment Configuration

In this section, we provide the details of the training environments. The navigation policies run at 5 Hz. The frozen locomotion policy runs at 50 Hz, with velocity limits of 1 m/s forward, 0.25 m/s lateral, and 1 rad/s angular. The LiDAR FoV is set to a forward-centered 220°, to facilitate human support during testing which occludes the LiDAR. Ultimately, all hardware experiments are run with no support.

Rewards: The rewards follow the structure of established literature [29], [35], [41], consisting of goal tracking rewards, progress shaping rewards, and penalties, shown in Table I, where d_{xy} is the planar distance to goal, e_ψ is the heading error, z_e the vertical error, $\overline{v}_{xy}$ is the speed required for maximum geodesic progress reward, and $\overline{v}_z$ is the vertical speed required for maximum height progress reward. The two primary additions to a standard navigation reward set are the height goal tracking reward and height progress reward, since the policy is trained on multi-floor buildings. The gated rewards only fire when within 2 m of geodesic distance to the goal. The progress shaping rewards are input-limited CLF rewards [42], which incentivize progress to the goal along the geodesic (in x, y) and along height. These rewards only fire when at the episode best, to avoid circling behaviors accumulating free progress. While the CLF reward provides a modest training stabilization influence, the height rewards are critical to the success in multi-floor environments. The action rate penalty changes at iteration 1000, allowing exploration early but encouraging smoothness late.

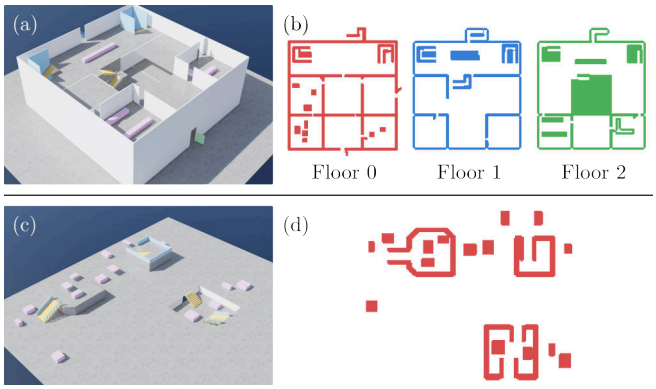

Fig. 3. Environment tiles. (a) multi-story building and (b) its occupancy grid. (c) multi-level outdoor environment and (d) its occupancy grid.

Environments: Where existing works utilize single-level layouts, our deployment in complex multi-level real-world environments requires a different approach. We enumerate several classes of procedurally generated tiles, including outdoor tiles, single room tiles, and multi-story building tiles. A tile is considered to be "multi-story" if there exist x, y coordinates which contain multiple disjoint z locations which are coherent goals, i.e., the same x, y position on different floors. A few sample tiles are shown in the left panes of Fig. 3. As the tiles are generated programmatically, and since the locomotion policy was trained to locomote over specific terrain, ground truth traversability is easily recorded during terrain generation. Obstacle and traversability information is rasterized into a 0.2 m occupancy grid during terrain generation, seen in the right panes of Fig. 3. This occupancy map is then used as the collision model for the RoM during its training. The occupancy map is maintained per floor for the multi-floor environments, where the robot transitions between floors based on a height threshold. The occupancy grids are locally consistent in floor transition regions.

Spawn/Goal Sampling: Most existing works sample spawns and goals uniformly in their environments, perhaps under a difficulty curriculum on the environment. As discussed in Section IV-B, we found naive sampling of the spawns and goals made z-height dependencies in multi-story environments difficult to learn. To address this, we adopt a specialized reset distribution over uniformly distributed, guaranteed free space spawn and goal locations. First, with probability $p = 0.3$ the spawn is selected to be on or near the mouth of a stair or ramp. To place initial conditions on the stair or ramp, where a nominal standing initial condition does not work well, a gait library is collected of the locomotion policy traversing stairs or ramps under standard commands.

TABLE II: Navigation performance across training methods. SR@45 / SR@120 are success within 45 s (the training budget) and 120 s; T2G is mean time to reach the goal over successful episodes; SPL is success weighted by path length against the geodesic route. **Bold** marks the best humanoid arm per column. * RoM (cyl) is evaluated on RoM dynamics.

Arm	SR@45 (%)	SR@120 (%)	T2G (s)	SPL
RoM (cyl)*	84.1	93.6	25.6	0.776
Single-Stage	62.2	81.7	32.1	0.690
RoM (hum)	74.2	88.1	29.8	0.713
RoM-Nav	**82.3**	**92.8**	**28.8**	**0.769**

The closest stair or ramp in the gait library is selected, and the robot is placed at a random time along the gait from the library. When this specialized reset is not sampled, the fallback is uniform sampling over the valid spawns.

The goal sampling is geodesically capped and cross-level biased. A wavefront algorithm starting at the goal computes free-space geodesic distance to each initial condition. Goals beyond 30 m geodesic distance are masked out, ensuring feasibility during the 45 s episode with 1 m/s max speed. This goal mask computation is parallelized via a CUDA kernel, taking 14 minutes to precompute for the entire environment. Second, goals are biased to cross height levels. Three story buildings tiles sample cross-level goals 100% of the time; other tiles, 50%. This bias helps prevent under-conditioning on z, as cross-level goals require traversal of difficult terrain and longer horizons.

E. Poisson Safety Filter

The Poisson safety filter modifies velocity commands coming from the navigation policy before passing them to the locomotion policy, to avoid colliding with OOD obstacles. We adopt the Real-Time CBF-QP layer deployed in [8], which we summarize here. The Poisson safety filter uses a single integrator approximation, with state $p = [x, y]$ and input $v = [v_x, v_y]$ satisfying dynamics $\dot{x} = v_x$, $\dot{y} = v_y$.

According to Control Barrier Function (CBF) theory [43], a smooth function $h(p)$, positive in free space and negative in occupied space, is a control barrier function if, for $\alpha > 0$:

$$\sup_v \dot{h}(p, v) \geq -\alpha h(p).$$

Furthermore, selecting v such that $\dot{h}(p, v) \geq -\alpha h(p)$ ensures that $h(p) \geq 0$ for all time. To construct h, the point cloud is rasterized into a 2D occupancy grid (0.05 m), after removing the ground. The occupancy is convolved with a circle of the robot's radius, and the boundary of occupied space is identified. Then Poisson's equation is solved over this grid, enforcing $h(p) = 0$ on the boundary and positive divergence in free space via a forcing function; see [7], [8] for details.

Once the Poisson safety function h has been synthesized, the Real-Time CBF-QP finds the minimum deviation to the desired input v_{des}, output by the RoM-Nav policy, which satisfies the barrier constraint, with $\alpha = 0.75$:

$$v_{\text{safe}} = \arg\min_v \|v - v_{\text{des}}\|^2 \quad \text{s.t.} \quad \frac{dh}{dp}\Big|_p v \geq -\alpha h(p).$$

This safe planar velocity is then passed to the locomotion policy, avoiding collision in the presence of OOD obstacles.

TABLE III: Success rate paired contrasts by same-level vs. cross-level goals. Positive favors RoM-Nav; † marks a 95% confidence interval excluding zero.

	45 s		120 s	
Contrast (%)	same	cross	same	cross
RoM-Nav − RoM (hum)	$+1.5$	$+15.2^{\dagger}$	$+0.6$	$+9.1^{\dagger}$
RoM-Nav − Single-Stage	$+6.8^{\dagger}$	$+34.6^{\dagger}$	$+3.0^{\dagger}$	$+19.7^{\dagger}$

IV. Results

In this section we examine the RoM-Nav method, showing its improvement over the single-stage training approach, and justify the pretrained encoders and spawn/goal sampling. We demonstrate the effectiveness of the Poisson safety filter for safety around OOD obstacles on hardware. Finally, we deploy the method in real-world multi-floor environments.

A. RoM Kickstarting

First, we evaluate the RoM-Nav method. We compare a few methods: Single-Stage, trained directly on the humanoid via PPO for 4000 iterations. RoM (cyl) is the RoM policy evaluated *on the RoM*, and provides an upper bound for the RoM-Nav policy. RoM (hum) is the RoM policy evaluated *on the humanoid*. RoM-Nav is the proposed method, where the RoM policy kickstarts a policy trained on the humanoid, using the combined KL and PPO loss.

Table II compares the performance of these policies on a paired test. Each policy is spawned from the same 1024 initial conditions, in a randomized terrain generated with a different seed from training. The metrics recorded are success rate, where success is reaching within 0.5 m of the goal, in the training budget of 45 s (SR@45), or within 120 s (SR@120). The longer evaluation allows the policy to take and recover from more wrong turns, reasonable in a mapless context. We also report mean time to goal (T2G) over successful trials, and 120 s success weighted by inverse path length (SPL), a metric introduced in [44] for evaluating navigation efficiency:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{\max(p_i,\, \ell_i)},$$

where S_i is the success of the ith trial, p_i is the length of the traversed path, and ℓ_i the geodesic length.

Evaluating Table II, we see Single-Stage underperforming, while RoM-Nav performs within 1–2% of the RoM (cyl). The RoM deploys reasonably well onto the humanoid without the kickstarting step, but kickstarting recovers an additional 8% success at 45 s, and an additional 5% at 120 s. Table III breaks these success results down by spawn/goal level, relative to the RoM-Nav policy. RoM-Nav makes all of its progress back in the cross-level trials, where stairs or ramps must be traversed to change z level. Furthermore, the cross-level improvement primarily comes from fewer falls. On the 120 s cross-level trials, the timeout rates are 4.9% for RoM (hum) and 3.7% for RoM-Nav, but the fall rate drops from 16.3% on RoM (hum) to 8.3% for RoM-Nav. The kickstarting leverages the RoM navigation ability, and robustifies it against the complex terrain interactions necessary to cross floors.

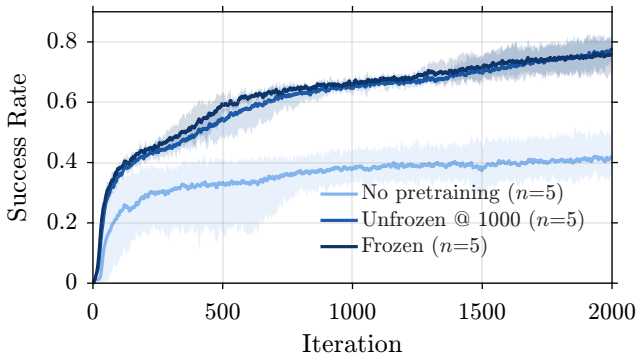

Fig. 4. An evaluation of the LiDAR encoder pretraining. Success rate vs. training iteration is plotted for three pretraining variations; mean over five random seeds, min-max shaded.

TABLE IV: Spawn/goal-distribution ablation on the RoM. Metrics identical to Table II. **Bold** marks the best arm per column (T2G excluded, as this metric skews when only short episodes are completed).

Arm	SR@45 (%)	SR@120 (%)	T2G (s)	SPL
RoM	**84.1**	**93.6**	25.6	**0.776**
Uniform	70.9	77.4	23.3	0.640
NoCap	69.8	75.8	22.9	0.624
UniformNoCap	65.7	75.1	24.9	0.601

B. RL Method Choices

Pretrained Encoders: To demonstrate the impact of the encoder pretraining, we train three variants of the RoM policy, using different LiDAR encoder pretrainings. No Pretraining randomly initializes the CNN and trains it with PPO. Unfrozen @ 1000 takes the pretrained CNN weights and unfreezes them only after the first 1000 iterations of PPO. To train the CNN weights on the H100 GPU, we reduce the environment count to 3072 and we double the PPO minibatches from 8 to 16. Frozen is a control arm, with the CNN weights frozen for all of training (same as deployed variant), but with the reduced environments and increased minibatches. Fig. 4 shows the success rate training curves. Without pretraining, success is dramatically reduced. Frozen and Unfrozen @ 1000 reach equivalent performance. Additionally, we note that training with the unfrozen CNN weights increases the iteration time by 15%. Furthermore, the RoM policy from Table II has a success rate 4% higher than the best curve on this plot, due to its higher environment count and larger PPO batch. These results justify the frozen pretrained encoder.

Spawn/Goal Sampling: Similarly, we demonstrate that our spawn/goal sampling, outlined in Section III-D, improves performance. We compare training without stair oversampling (Uniform), dropping the geodesic cap in favor of a Euclidean cap (NoCap) and both (UniformNoCap). Table IV shows significant and approximately equal drops from removing either, and a larger drop when both components are removed. Table V breaks this down further by splitting the contrast into same-level and cross-level goals. The lost performance comes from the cross-level goals. The spawn/goal sampling procedures dramatically help the learning of these complex cross floor navigation behaviors, distinguishing this work from existing legged RL navigation literature.

TABLE V: Spawn/goal pair success rate by same-level vs. cross-level goals. Positive favors RoM; † marks a 95% confidence interval excluding zero.

	45 s		120 s	
Contrast (%)	same	cross	same	cross
RoM − Uniform	+0.0	+27.8†	+0.6	+33.5†
RoM − NoCap	−0.6	+30.7†	−0.2	+37.7†
RoM − UniformNoCap	+1.0	+37.7†	+0.0	+39.0†

TABLE VI: Hardware CBF comparison in three environments. Initial conditions and goals are shared across arms and environments.

Obstacle Type	Arm	Success	Collision(s)	Time to goal (s)
In-Distribution	no CBF	10/10	0/10	5.9 ± 1.5
	CBF	10/10	0/10	6.7 ± 2.1
Out-of-Distribution	no CBF	10/10	2/10	7.5 ± 1.7
	CBF	10/10	**0/10**	11.0 ± 5.8
Adversarial	no CBF	10/10	4/10	5.5 ± 1.0
	CBF	10/10	**0/10**	6.9 ± 1.7

C. Safety under Out-of-Distribution Obstacles

To demonstrate the sensitivity of the navigation policy to OOD obstacles, we construct challenging environments on hardware. Training obstacles are geometric, both on the ground and floating, including railings and thin features.

In-distribution obstacles (Fig. 5, left) are chosen to be solid and convex. The middle pane shows two OOD obstacles, ladders with cardboard tubes sticking out. Lastly, the right pane contains hanging cardboard tubes, chosen adversarially; thin tubes hung at head height present exceptionally small LiDAR cross-sections.

Ten spawn/goal points are selected, shared across all environments. Each trial is conducted with the CBF both on and off. To facilitate precise initial condition and goal alignment, the room is mapped using GLIM [45], and the robot is continuously relocalized using a relocalization package built on top of Fast-LIO2 [46]. The map is not available to the RoM-Nav policy in any way, which remains strictly mapless. This method is also used in Section IV-D.

The results of this experiment are summarized in Table VI. To avoid cluttering the figure, four of the ten trials (including all collisions) are visualized in Fig. 5. On in-distribution obstacles, we see no collisions in either case. On OOD obstacles, the CBF makes no collisions, while RoM-Nav makes two on the ladders and four on the hanging tubes. The CBF pays for its success in collision avoidance by increasing the mean time to goal in the OOD environments.

D. Hardware Deployment

We deploy the RoM-Nav policy on a Unitree G1 robot to perform mapless multi-story long-horizon navigation in real-world environments. Fig. 6 shows four experimental trials on hardware. These trials include vertical climbs of up to 10 m, which is beyond the training distribution extreme of 8 m (two 4 m floors), as well as path lengths of up to 100 m, which is well beyond the training distribution of 30 m. None of the hardware trials included a collision. The policy successfully navigates stairwells with thin railings, long outdoor stair climbs, clutter, tables and chairs, and ramps.

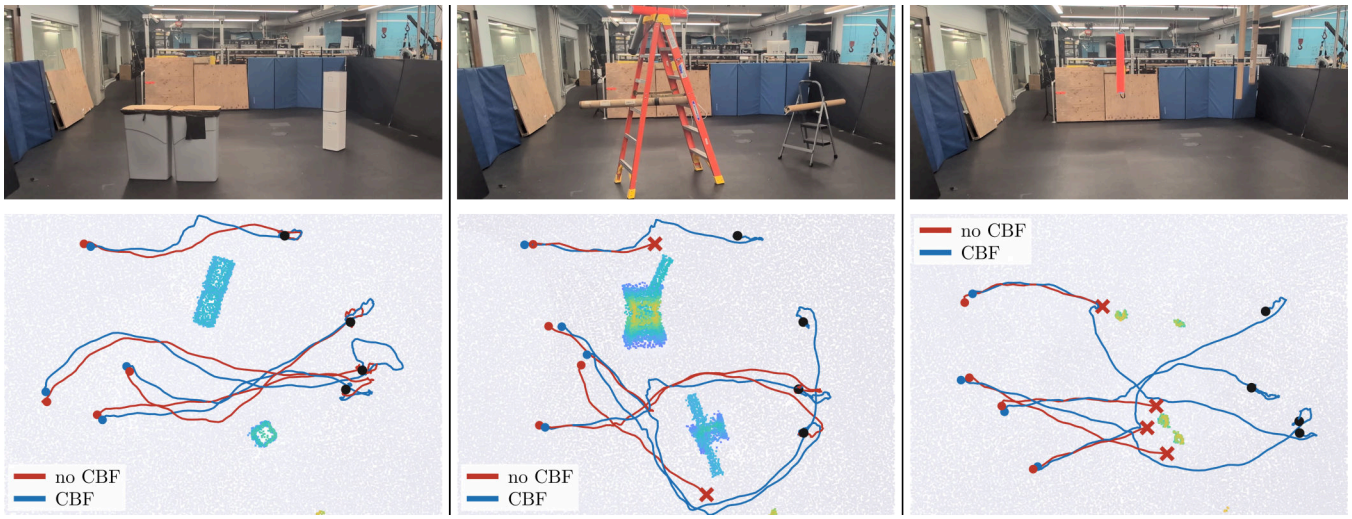

Fig. 5. Investigation on performance with OOD obstacles. Four of ten trials are plotted (collisions marked with ×). (Left) Environment with in-distribution obstacles. (Middle) Environment with OOD obstacles. (Right) Environment with obstacles chosen adversarially, hanging with small LiDAR cross-section.

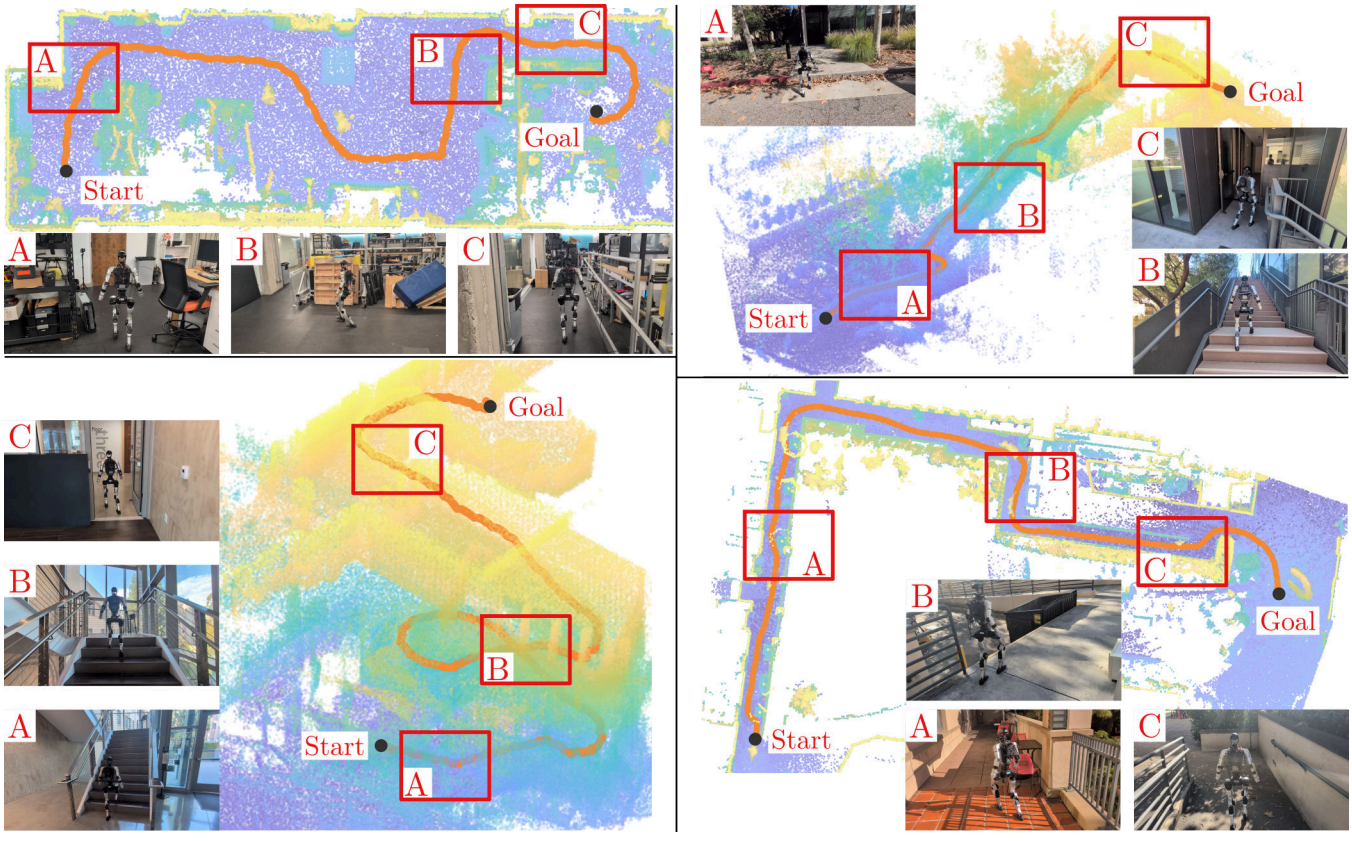

Fig. 6. Four example deployments in complex, real-world environments. (Top Left) Navigation of a cluttered lab environment, 38 m path length. (Top Right) 2-story climb (7 m ascent) on a wire-railed stairwell, 51 m path length. (Bottom Left) 2-story climb (10 m ascent) to enter a building from the outside, 51 m path length. (Bottom Right) A long-horizon outdoor navigation task, avoiding tables, cliff edges and railings, 100 m path length (1 m descent).

One significant drawback of this policy for real-world deployment is navigating near glass or transparent obstacles. A cardboard sheet is used to block a glass door and window in one trial, which otherwise the robot may have tried to traverse. LiDAR is not a sufficient sensor for navigating transparent obstacles. Other than specific instances of blocking glass, no intentional environmental modifications were required to achieve this navigation performance. Another limitation is the requirement of an accurate goal: placement of an accurate goal in a coherent location, expressed in the body frame, is a strong assumption. Lastly, the planar CBF used in this work could block or close routes the RoM-Nav policy deems traversable, although this is not observed experimentally. Whole-body humanoid safety for navigation and ensuring agreement between the safety filter and navigation policy are potential avenues for future work.

V. Conclusion

We investigate autonomous multi-level mapless navigation for humanoid robots. Our method first trains a navigation policy on a reduced order model, and uses this policy to kickstart training on a humanoid robot with a frozen locomotion controller. Nearly all of the gains from the kickstarting process, and also improvements over the single-stage pipeline, occur on trials where the robot must traverse multi-level environments, e.g., to the next story of a building. This is consistent with the literature; the single-stage approach in this paper is heavily based upon prior works that show strong results in single-level navigation. Our extensions show significant benefits for complex multi-story navigation applications on a humanoid robot.

We also demonstrate the use of a Poisson safety filter to ensure collision avoidance with OOD obstacles on hardware, without reducing navigation success rate. LiDAR encoder pretraining and non-uniform spawn/goal sampling procedures were both demonstrated to be critical for cross-level capabilities. Finally, the policy is deployed on a Unitree G1 to accomplish safe autonomous real-world multi-floor mapless navigation, covering path lengths over 100 m and vertical displacements over 10 m.

References

[1] Z. Olkin *et al.*, "Chasing autonomy: Dynamic retargeting and control guided RL for performant and controllable humanoid running," *arXiv preprint arXiv:2603.25902*, 2026.

[2] Y. Zhang *et al.*, "RPL: Learning robust humanoid perceptive locomotion on challenging terrains," *arXiv preprint arXiv:2602.03002*, 2026.

[3] W. D. Compton, Z. Olkin, and A. D. Ames, "Terrain consistent reference-guided RL for humanoid navigation autonomy," *arXiv preprint arXiv:2605.15517*, 2026.

[4] H. Wang *et al.*, "BeamDojo: Learning agile humanoid locomotion on sparse footholds," in *RSS*, 2025.

[5] Z. Wu *et al.*, "Perceptive humanoid parkour: Chaining dynamic human skills via motion matching," *arXiv preprint arXiv:2602.15827*, 2026.

[6] F. Liu *et al.*, "Opt2Skill: Imitating dynamically-feasible whole-body trajectories for versatile humanoid loco-manipulation," *IEEE Robotics and Automation Letters*, 2025.

[7] R. M. Bena *et al.*, "Geometry-aware predictive safety filters on humanoids: From Poisson safety functions to CBF constrained MPC," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2025.

[8] E. Yamaguchi *et al.*, "Layered safety: Enhancing autonomous collision avoidance via multistage CBF safety filters," *arXiv preprint arXiv:2603.00338*, 2026.

[9] R. J. Griffin *et al.*, "Footstep planning for autonomous walking over rough terrain," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2019.

[10] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014.

[11] K. S. Narkhede *et al.*, "A sequential MPC approach to reactive planning for bipedal robots using safe corridors in highly cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, 2022.

[12] J.-K. Huang and J. W. Grizzle, "Efficient anytime CLF reactive planning system for a bipedal robot on undulating terrain," *IEEE Transactions on Robotics*, vol. 39, no. 3, 2023.

[13] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3D perception and environment map generation for humanoid robot navigation," *The International Journal of Robotics Research*, vol. 27, no. 10, 2008.

[14] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014.

[15] Y.-C. Lin and D. Berenson, "Long-horizon humanoid navigation planning using traversability estimates and previous experience," *Autonomous Robots*, vol. 45, no. 6, 2021.

[16] J. Frey *et al.*, "Fast traversability estimation for wild visual navigation," in *RSS*, 2023.

[17] Z. Yoon *et al.*, "STATE-NAV: Stability-aware traversability estimation for bipedal navigation on rough terrain," *IEEE Robotics and Automation Letters*, vol. 11, no. 2, 2025.

[18] P. Roth *et al.*, "Learned perceptive forward dynamics model for safe and platform-aware robotic navigation," in *RSS*, 2025.

[19] Q. Ben *et al.*, "Gallant: Voxel grid-based humanoid locomotion and local-navigation across 3D constrained terrains," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[20] Y. Zhang *et al.*, "FocusNav: Spatial selective attention with waypoint guidance for humanoid local navigation," *arXiv preprint arXiv:2601.12790*, 2026.

[21] H. Han *et al.*, "GuideWalk: Learning unified autonomous navigation and locomotion for humanoid robots across versatile terrains," *arXiv preprint arXiv:2606.10449*, 2026.

[22] A.-C. Cheng *et al.*, "NaVILA: Legged robot vision-language-action model for navigation," in *RSS*, 2025.

[23] Y. Du *et al.*, "VL-Nav: Neuro-symbolic reasoning-based vision-language navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

[24] N. Rudin *et al.*, "Advanced skills by learning locomotion and local navigation end-to-end," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[25] D. Hoeller *et al.*, "Learning a state representation and navigation in cluttered and dynamic environments," *IEEE RAL*, vol. 6, no. 3, 2021.

[26] J. Truong *et al.*, "Learning navigation skills for legged robots with learned robot embeddings," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[27] Z. Fu *et al.*, "Coupling vision and proprioception for navigation of legged robots," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] J. Lee *et al.*, "Learning robust autonomous navigation and locomotion for wheeled-legged robots," *Science Robotics*, vol. 9, no. 89, 2024.

[29] F. Yang *et al.*, "Spatially-enhanced recurrent memory for long-range mapless navigation via end-to-end reinforcement learning," *The International Journal of Robotics Research*, 2025.

[30] E. Wijmans *et al.*, "DD-PPO: Learning near-perfect PointGoal navigators from 2.5 billion frames," in *International Conference on Learning Representations (ICLR)*, 2020.

[31] F. Yang *et al.*, "iPlanner: Imperative path planning," in *RSS*, 2023.

[32] P. Roth *et al.*, "ViPlanner: Visual semantic imperative learning for local navigation," in *IEEE ICRA*, 2024.

[33] J. Truong *et al.*, "Rethinking Sim2Real: Lower fidelity simulation leads to higher Sim2Real transfer in navigation," in *Conference on Robot Learning (CoRL)*, ser. PMLR, vol. 205, 2023.

[34] S. Kareer *et al.*, "ViNL: Visual navigation and locomotion over obstacles," in *IEEE ICRA*, 2023.

[35] Z. Xu, H. Jin, and K. Shimada, "NavRL++: A system-level framework for improving sim-to-real transfer in reinforcement learning-based robot navigation," *arXiv preprint arXiv:2605.15559*, 2026.

[36] T. Miki *et al.*, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, 2022.

[37] S. Schmitt *et al.*, "Kickstarting deep reinforcement learning," *arXiv preprint arXiv:1803.03835*, 2018.

[38] T. He *et al.*, "Agile but safe: Learning collision-free high-speed legged locomotion," in *RSS*, 2024.

[39] A. Lin, S. Peng, and S. Bansal, "One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments," *IEEE Transactions on Robotics*, vol. 42, 2025.

[40] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[41] L. Wang *et al.*, "GUIDE: Goal-initialized directional understanding for end-to-end visual navigation," *arXiv preprit arXiv:2606.10832*, 2026.

[42] K. Li *et al.*, "CLF-RL: Control Lyapunov function guided reinforcement learning," *IEEE Robotics and Automation Letters*, 2026.

[43] A. D. Ames *et al.*, "Control barrier function based quadratic programs for safety critical systems," *IEEE TAC*, 2016.

[44] P. Anderson *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[45] K. Koide *et al.*, "GLIM: 3D range-inertial localization and mapping with GPU-accelerated scan matching factors," *Robotics and Autonomous Systems*, vol. 179, 2024.

[46] W. Xu *et al.*, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, 2022.