Title

Reinforcement-Adaptive Visibility-Graph Planning for Robust Humanoid Navigation

ISBN

9798247985693

Author

Singal, Mehak

Publication Date

2026-06-08

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Reinforcement-Adaptive Visibility-Graph Planning for Robust Humanoid Navigation

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Mehak Singal

2026

ABSTRACT OF THE THESIS

Reinforcement-Adaptive Visibility-Graph Planning
for Robust Humanoid Navigation

by

Mehak Singal
Master of Science in Computer Science
University of California, Los Angeles, 2026
Professor Jens Palsberg, Chair

Achieving reliable humanoid navigation requires simultaneously reasoning over long planning horizons and satisfying short-horizon dynamic and safety constraints. Classical planners based on visibility graphs, when paired with model predictive control (MPC), can produce efficient collision-free trajectories, yet their effectiveness hinges on carefully hand-tuned parameters and accurate system models. On physical hardware, however, actuation delays, state-estimation errors, and imperfect low-level locomotion tracking routinely cause trajectories to deviate from their nominal paths, producing constraint violations even when the planned path is geometrically sound.

This thesis introduces RAVEN (Reinforcement-Adaptive Visibility-Graph Planning with Collision-Free MPC), a hierarchical framework that addresses these limitations by coupling a learning-based global planner with a constrained local controller. Rather than tuning MPC cost weights or substituting the planner with a neural network, RAVEN trains a reinforcement learning meta-policy to adapt the geometric structure of the visibility graph by adjusting per-obstacle inflation radii online. Modifying these radii reshapes the navigable free

space and thereby shifts the topology of the global path, allowing the system to proactively compensate for delay and tracking error before they propagate to the controller. A collision-free MPC layer then follows the adapted path while enforcing velocity and obstacle-avoidance constraints over a receding horizon.

Training under injected control delays and observation noise encourages the meta-policy to discover inflation strategies that remain robust across varying system conditions, while the underlying geometric planner and constrained optimizer are left intact. Experiments compare RAVEN against a fixed-parameter visibility-graph MPC baseline and a pure end-to-end RL policy in both simulation and hardware deployment on a bipedal humanoid. RAVEN achieves substantially lower obstacle penetration depth under delay than the classical MPC baseline, along with shorter average path length and faster task completion than either baseline, and its simulation behavior transfers reliably to the physical platform. These results suggest that learning to adapt the planning geometry, rather than the controller parameters, is an effective and interpretable strategy for robust humanoid navigation.

The thesis of Mehak Singal is approved.

Dennis W. Hong

Yuchen Cui

Jens Palsberg, Committee Chair

University of California, Los Angeles

2026

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This thesis is based on research conducted at the Robot Mechanics Laboratory (RoMeLa) at the University of California, Los Angeles.

I would also like to thank my co-authors on this work, Ruochen Hou, Shiqi Wang, Beom Jun Kim, and Hanzhang Fang, along with Professor Dennis W. Hong, for their collaboration and contributions that made this research possible. This thesis is substantially based on our joint work submitted for publication: "RAVEN: Reinforcement-Adaptive Visibility-Graph Planning for Robust Humanoid Navigation with Collision-Free MPC," submitted to the 2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

I would also like to thank Professor Jens Palsberg, Professor Dennis W. Hong, and Professor Yuchen Cui for taking the time to serve on my thesis committee and for their thoughtful feedback and guidance throughout this process.

I am grateful to my labmates at RoMeLa for their collaboration and support throughout this project.

Lastly, I would like to thank my family and friends for their encouragement and unwavering support throughout my time at UCLA.

VITA

2024 Bachelor of Science in Data Science and Applications, Indian Institute of Technology Madras

2026 Candidate for Master of Science in Computer Science, University of California, Los Angeles

CHAPTER 1

Introduction

Figure 1.1: The RAVEN framework deployed on the T1 Booster biped navigating among obstacles. Hexagons indicate the adaptive obstacle inflation regions whose radii are determined by the RL meta-policy to account for control delays and tracking uncertainties.

The promise of humanoid robotics lies in the potential of these platforms to navigate the cluttered and unstructured environments designed for humans. Compared with wheeled robots,

humanoids can naturally leverage human-designed infrastructure such as stairs, doorways, and narrow hallways without modification. However, realizing robust navigation on these platforms in the real world exposes a fundamental tension between idealized geometric planning and uncertain dynamic execution.

Classical navigation stacks for mobile robots often pair a global planner with a local trajectory-tracking controller. A widely used global planning primitive is the *visibility graph*, which computes shortest collision-free paths through a map of polygonal obstacles. At the local level, *Model Predictive Control* (MPC) can track the global path while enforcing dynamic and safety constraints over a receding horizon. While this hierarchical structure is geometrically sound in theory, its real-world performance on physical humanoids is limited by a key assumption: accurate system modeling.

In practice, deploying these pipelines on bipedal humanoid robots introduces three compounding sources of uncertainty. First, *control delays* arise from the latency between sensing, computation, and actuation, causing the robot to execute commands that were computed for a position it has already left. Second, *state-estimation noise* degrades the accuracy of the robot's self-localization, introducing error into the planning reference frame. Third, *locomotion uncertainties* in the low-level walking policy mean that high-level velocity commands are tracked imperfectly. When a robot inevitably lags behind its nominal path, a trajectory that perfectly grazes an obstacle boundary in simulation can quickly devolve into a collision on hardware.

To address these real-world uncertainties, the field has largely bifurcated into two competing paradigms. On one hand, end-to-end Deep Reinforcement Learning (DRL) approaches train monolithic neural networks to map raw sensor observations directly to motor commands [LZL25, HRS23, ZFW23]. While these methods exhibit remarkable agility and emergent behaviors, they suffer from a lack of interpretability, making it nearly impossible to formally verify safety constraints or diagnose failures in safety-critical scenarios. Furthermore, DRL has been heavily applied to quadrupedal platforms, which are inherently statically

stable and less prone to hardware damage. Bipedal humanoid robots, by contrast, demand stricter, mathematically guaranteed safety margins alongside agile navigation.

On the other hand, classical MPC provides mathematical rigor, explicit constraint satisfaction, and predictable behavior. However, its real-world efficacy is bottlenecked by static, manually tuned geometric heuristics, most notably, *obstacle inflation radii*. A fixed inflation parameter cannot universally account for varying speeds, narrow passages, or fluctuating system delays, forcing a painful compromise between conservative, sluggish navigation and aggressive, collision-prone trajectories. Neither extreme is sustainable for robust deployment.

This thesis argues that the key to bridging this divide is to use learning not to replace the planner, but to *adapt its geometric construction* online. Tuning algebraic cost weights inside the MPC is an indirect and computationally expensive mechanism, while generating arbitrary waypoints reduces to a point-to-point prediction task that ignores global topology. Instead, reinforcement learning can be used to modulate the obstacle inflation parameters of the visibility graph itself. This directly reshapes the free-space geometry and alters the topology of the resulting global path, allowing the system to account for real-world uncertainties before the local controller begins its optimization.

1.1 Thesis Contributions

This thesis presents **RAVEN** (Reinforcement-Adaptive Visibility-Graph Planning with Collision-Free MPC) [HWK26], a hierarchical framework for robust humanoid navigation. The main contributions are as follows:

1. **Reinforcement-adaptive visibility-graph planning via a meta-policy.** A learning-augmented visibility-graph planner in which an RL meta-policy adapts the geometric construction of the graph by modifying per-obstacle inflation radii. Unlike prior approaches that tune MPC cost weights, this directly reshapes the free-space geometry and alters the topology of the resulting global path.

2. **A hierarchical RL–MPC navigation framework.** RAVEN combines RL-adaptive global planning with Collision-Free MPC (CF-MPC) for local trajectory tracking. The MPC layer explicitly enforces dynamic limits and obstacle-avoidance constraints while following the RL-adapted plan, preserving explicit geometric planning and transparent constraint enforcement.

3. **Guided exploration through visibility-graph shortest-path structure.** Because the visibility graph computes shortest paths under the current obstacle geometry, the meta-policy's adaptation effectively searches over a family of shortest-path solutions. This structured prior guides exploration toward near-optimal navigation strategies and reduces convergence to suboptimal behaviors compared with pure reinforcement learning.

4. **Robust navigation under delay and observation noise.** By training under realistic control delays and perception noise, the learned planner adapts obstacle inflation and path geometry to mitigate overshoot and tracking errors. Experiments in simulation and on a physical T1 Booster bipedal humanoid demonstrate improved reliability in challenging scenarios such as narrow passages and actuation delays, with consistent performance across simulation and real-world deployments.

1.2 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 surveys related work in end-to-end learning for locomotion, RL-driven path generation, and synergies of RL and MPC. Chapter 3 describes the RAVEN framework in detail, including the DAVG-CFMPC formulation, the RL meta-policy design, and the training procedure. Chapter 4 presents the experimental setup, including baseline methods, simulation environments, and hardware platform. Chapter 5 reports simulation and hardware results, with analysis of robustness to delay and noise. Chapter 6 concludes and outlines directions for future work.

CHAPTER 2

Related Work

This chapter surveys the literature most relevant to RAVEN, organized into three themes: end-to-end learning for agile locomotion, RL-driven path generation and subgoal planning, and hybrid architectures that combine reinforcement learning with model predictive control.

2.1 End-to-End Learning for Agile Locomotion

The dominant trend in modern legged robotics has been the shift toward end-to-end Deep Reinforcement Learning (DRL) policies that map raw sensor observations directly to joint-level commands. These monolithic architectures have demonstrated remarkable capabilities in traversing unstructured terrain and executing complex locomotion skills.

Zhuang et al. [ZFW23] propose a vision-based policy capable of executing diverse parkour skills, including climbing, vaulting, and leaping, emerging entirely from a unified neural network trained without explicit reference motions. The framework demonstrates that complex locomotion behaviors can emerge from reward shaping and large-scale GPU-accelerated simulation alone.

Liu et al. [LZL25] introduce COMPASS, a cross-embodiment navigation pipeline that distills specialist locomotion policies into a single generalist agent using residual RL to adapt a foundational navigation policy to different robot morphologies. This work highlights the scalability of end-to-end approaches across platform types, though interpretability remains limited.

Hoeller et al. [HRS23] adopt a hierarchical but fully learned architecture, decomposing the problem into a low-level library of specialized locomotion skills and a high-level navigation policy that selects the appropriate skill based on perception. The resulting system achieves impressive agility on quadrupedal platforms in parkour-style scenarios.

While these learning-based methods excel at agility and generalization, they often function as "black boxes." For bipedal humanoids, which possess complex underactuated dynamics, the lack of interpretability presents a significant challenge for formal verification and safety guarantees. Ensuring strict dynamic constraint satisfaction and mathematically bounded safety margins remains difficult with purely neural architectures, motivating the need for hybrid methods that combine learned agility with explicit optimization-based control.

2.2 RL-Driven Path Generation and Subgoal Planning

Addressing long-horizon navigation in complex environments often motivates hierarchical frameworks where RL acts as a high-level waypoint or subgoal generator for a classical low-level controller. This partitioned architecture successfully blends learned spatial intuition with the stable locomotion of model-based control.

Peng et al. [PZG25] employ RL with data bootstrapping to dynamically predict discrete local subgoals for an MPC to track through cluttered or dynamic environments. Liu et al. [LFD25] use Graph Neural Networks to predict local subgoals for an MPC navigating crowds with heterogeneous constraints, achieving real-time performance in dense pedestrian environments. Zhang et al. [ZHW23] utilize a deep Markov model to generate navigation waypoints in unknown environments without prior maps, combining learned path generation with fine-tuned motion control.

While these approaches successfully decompose the navigation problem, they fundamentally treat path generation as a point-to-point prediction task. The global planner's geo-

metric rules remain rigid and inaccessible to the learning agent: the RL policy generates waypoints without awareness of how the underlying planner constructs the space between them.

RAVEN departs from this paradigm. Rather than predicting waypoints in the workspace, its meta-policy systematically alters the geometric construction of the visibility graph itself. By dynamically adapting obstacle inflation, it reshapes the global path topology to account for the execution delays and tracking errors of the downstream MPC.

The two approaches differ not merely in what they output, but in how they interact with the planner. Subgoal-generation policies produce waypoints, which are discrete points in the robot's physical workspace that serve as navigation outputs rather than planner parameters. The geometric rules of the underlying planner remain fixed and inaccessible to the learning agent, which merely injects new targets into the pipeline. RAVEN instead sets the inflation radii that reshape the planner's free-space geometry, directly governing the class of paths the planner can produce. A subgoal generator is therefore not a meta-policy. It maps observations to workspace waypoints that specify where the robot should go rather than to planner configurations that govern how the path is constructed.

2.3 Synergies of Reinforcement Learning and MPC

To combine the adaptability of learning with the mathematical rigor of control theory, recent research has explored various integrations of RL and MPC. These hybrid architectures can be broadly categorized by the structural relationship between the learning and control modules.

2.3.1 Action-Level Integration and Imitation

Several frameworks combine RL and classical control at the execution level rather than integrating the planning processes. Johannink et al. [JBN18] utilize RL to output an additive continuous correction to a nominal hand-crafted controller, retaining the base controller's

stability while extending its dynamic range via residual learning.

The inverse relationship is explored via imitation learning. Carius et al. [CFH20] use a computationally expensive, state-based MPC as an expert demonstrator to train a faster, reactive neural network policy, effectively distilling the MPC's optimization into a sensorimotor mapping. Zhang et al. [ZKL16] apply a similar approach to autonomous aerial vehicles, using MPC-guided policy search to train deep control policies. These methods tend to treat the controller as a fixed module to be patched or imitated, rather than adapting its planning structure.

2.3.2 Gating Mechanisms and Safety Filters

Other architectures maintain distinct classical and learning-based modules, managing their interaction through high-level gating or filtering mechanisms. Sharma et al. [SLA24] employ heuristics to toggle execution between a classical planner for stable, static environments and an RL policy for reactive, dynamic obstacle avoidance.

In safety-critical contexts, Model Predictive Shielding [BLX21] employs the MPC strictly as a safeguard. An explicit model-based safety filter evaluates the action proposed by a primary RL agent, intervening only if the action is predicted to violate strict kinematic or collision constraints. While effective for preventing catastrophic failure, these reactive architectures can result in overly conservative or disjointed behavior because the safety filter and the primary policy are not jointly optimized.

2.3.3 Learning System Dynamics for MPC

A parallel line of work enhances the MPC's internal optimization by using neural networks to approximate complex or highly nonlinear system dynamics. Salzmann et al. [SKA23] integrate a learned neural dynamics model directly into the MPC optimization loop, computing local linear approximations of the network to serve as dynamic constraints for quadrotors

and agile robotic platforms.

Compton et al. [CCJ24] introduce Dynamic Tube MPC, which leverages massively parallel simulation to learn a dynamic error representation. This model characterizes the expected low-level tracking error as a function of the planned trajectory, allowing the MPC to optimize a path while strictly ensuring the learned "tube" of potential tracking deviation remains in collision-free space. While this approach improves localized tracking precision, it still relies on a nominal geometric path that may be topologically suboptimal under system delays. Furthermore, introducing neural-network-based system dynamics results in a nonlinear model that significantly increases computational cost; Dynamic Tube MPC can only run at approximately 10 Hz.

RAVEN shifts this burden to the high-level planner, ensuring the requested path inherently accounts for these uncertainties before the local MPC begins its optimization, while maintaining real-time performance at 100 Hz.

2.3.4 RL-Tuned and Differentiable MPC

Closest to RAVEN are methodologies that use RL to actively tune the internal parameters of the MPC. Mehndiratta et al. [MCK18] demonstrated automated search for optimal, static MPC cost weights via iterative offline simulations. To handle varying environmental conditions, Zarrouki et al. [ZSB24] dynamically select from a discrete set of pre-optimized weight vectors based on current state observations, applied to autonomous vehicle motion control.

The most advanced integration involves end-to-end differentiability. Romero et al. [RAS24] embed a differentiable MPC solver as the final layer of the actor network, allowing the policy to map state observations directly to the MPC's quadratic cost matrices and optimize local behavior online via backpropagation through the solver. While highly expressive, this framework is notoriously computationally intensive and adjusting algebraic cost weights is an indirect mechanism for altering a robot's spatial behavior.

RAVEN bypasses these bottlenecks by tuning the per-obstacle inflation radii of a high-level visibility graph rather than the algebraic weights of the local controller. This directly reshapes the topological free-space to account for execution delays, allowing a standard, non-differentiable MPC to track a dynamically robust global path at full real-time speeds.

These weight-tuning approaches also fall short of the meta-policy concept. Although they adapt parameters of a controller, those parameters are algebraic weights inside a fixed MPC cost function: they modulate how strongly certain behaviors are penalized without changing the topology of the plan itself. RAVEN's meta-policy instead modulates the geometric inputs to the global planner, so that qualitatively different paths emerge, taking different obstacle-avoidance directions and selecting different passages. It is a policy over planner configurations, not a policy over controller weights.

2.4 The DAVG-CFMPC Pipeline

The RAVEN framework builds directly upon the Dynamic Augmented Visibility Graph and Collision-Free MPC (DAVG-CFMPC) pipeline introduced by Hou et al. [HFZ25], which was previously validated for real-time humanoid navigation in adversarial RoboCup scenarios. That work established the core planning and tracking architecture; RAVEN's contribution is to replace the fixed, manually tuned obstacle inflation parameters in DAVG with a learned adaptive policy, enabling the system to respond to real-world uncertainties that static tuning cannot anticipate.

CHAPTER 3

The RAVEN Framework

This chapter presents the proposed RAVEN framework in full detail. As illustrated in Figure 3.1, the system is organized into three hierarchical layers. At the highest level, a reinforcement learning (RL) meta-policy adapts the construction parameters of the visibility-graph planner, enabling the robot to adjust its planning strategy according to environmental conditions and system dynamics. At the intermediate level, a visibility-graph planner generates a global collision-free path, while a Collision-Free MPC (CF-MPC) tracks the planned trajectory under explicit dynamic and safety constraints. At the lowest level, a locomotion policy executes the commanded velocity to produce stable humanoid walking motion.

This hierarchical structure cleanly separates long-horizon planning, trajectory optimization, and low-level control, allowing learning to adapt the planning process while preserving the safety and interpretability of classical planning and control methods.

3.1 DAVG-CFMPC Formulation

RAVEN builds upon the Dynamic Augmented Visibility Graph (DAVG) and Collision-Free MPC (CF-MPC) pipeline [HFZ25], a system previously validated for real-time humanoid navigation. This section focuses specifically on the geometric parameters that the learning framework modulates; the full algorithmic formulation is detailed in prior work.

DAVG computes a kinematically feasible global path by discretizing the environment into a visibility graph, routing the robot around obstacles that are artificially enlarged by a

safety margin. The key geometric parameter is the *obstacle inflation radius* assigned to each obstacle, which defines the effective size of that obstacle in the planning space. By inflating obstacles, the planner produces paths with guaranteed minimum clearance from the true obstacle boundaries.

This trajectory serves as the reference for the local CF-MPC, which tracks the path while enforcing dynamic feasibility. CF-MPC formulates collision avoidance as a linearized soft constraint within a convex Quadratic Program (QP). For each obstacle j, the avoidance constraint is linearized around the reference trajectory:

$$[x_k - x_{\text{obs},j},\ y_k - y_{\text{obs},j}] \cdot V_{k,j} \ \geq\ \|V_{k,j}\|\big(R_{\text{obs},j} - \delta_j\big), \tag{3.1}$$

where $V_{k,j}$ is the relative position vector between the robot and obstacle j at planning step k, $\delta_j \geq 0$ is a slack variable introduced to ensure QP feasibility in tight situations, and $R_{\text{obs},j}$ is the obstacle inflation radius passed down from the visibility-graph layer.

The MPC minimizes a quadratic cost over a finite horizon N:

$$\min_{\substack{X_{1:N} \\ u_{0:N-1}}} J = \sum_{i=1}^{N} (X_i - X_{r,i})^T Q (X_i - X_{r,i}) + \sum_{i=0}^{N-1} u_i^T R\, u_i + \rho \sum_{j=1}^{k} \delta_j^2, \tag{3.2}$$

subject to the linearized collision-avoidance constraints (3.1), velocity bounds $\|v\| \leq v_{\max}$, and acceleration bounds $\|\dot{v}\| \leq a_{\max}$. Here $X_{r,i}$ is the reference state from the DAVG path, Q and R are fixed weight matrices, and ρ is a penalty weight on constraint slack.

Crucially, the inflation radius $R_{\text{obs},j}$ in (3.1) is the geometric parameter exposed to the RL layer. By modifying this value per obstacle and per timestep, the meta-policy can reshape the free-space geometry that both the visibility graph and the MPC operate within.

3.2 RL Meta-Policy

At the highest level of the proposed framework, a reinforcement learning meta-policy is introduced to adapt the global planning behavior online. Concretely, a *meta-policy* is a

learned function

$$\pi_\theta \; : \; \mathcal{S} \; \longrightarrow \; [-1,\, 1]^K, \tag{3.3}$$

that maps the current observation $s_t \in \mathcal{S}$ not to a robot action (velocity or torque) but to a vector of K normalized actions, which are subsequently scaled into per-obstacle inflation radii via an affine transform (line 2 below). This distinguishes a meta-policy from a conventional policy $\pi : \mathcal{S} \to \mathcal{U}$, where $\mathcal{U}$ is the robot's velocity space: the meta-policy's output is consumed by the planner, not by the actuators. In algorithmic terms:

Algorithm 1: RAVEN meta-policy execution at timestep t

Input: observation s_t (robot pose, obstacle positions, goal)

1: $a_t \leftarrow \pi_\theta(s_t)$ *// meta-policy outputs raw action*

2: $r_{t,i} \leftarrow \frac{1}{2}(a_{t,i} + 1)(r_{\max} - r_{\min}) + r_{\min}$ for $i = 1, \ldots, K$ *// scale to radii*

3: $G_t \leftarrow \text{VisibilityGraph}(\text{obstacles}, r_t)$ *// rebuild graph*

4: $\text{path}_t \leftarrow \text{ShortestPath}(G_t)$ *// global plan*

5: $u_t \leftarrow \text{CF-MPC}(\text{path}_t, s_t)$ *// local control*

6: Execute u_t

Unlike end-to-end RL navigation policies that directly output motion commands, the meta-policy in RAVEN does not control the robot velocity. Instead, it outputs normalized actions that are mapped to per-obstacle inflation radii, which in turn modify the geometric construction of the visibility graph.

Adjusting these radii gives the planning layer access to a continuum of path geometries suited to different operating conditions. Under large control delays or significant locomotion uncertainty, higher inflation values steer the robot along wider arcs with greater clearance; under favorable conditions, lower values recover shorter, more direct routes. The planner can therefore tune its own conservativeness in response to actual system characteristics without any changes to the controller below.

Crucially, the learning component never touches the robot's actuators directly. Confining the policy's influence to the planning geometry means the explicit constraint enforcement of the classical MPC remains intact, preserving the interpretability of the overall pipeline while still allowing it to accommodate model mismatches and real-world uncertainties.

3.2.1 Observation Space

The observation space of the meta-policy includes robot state information, goal-related features, obstacle locations, and selected planning-related variables from the previous timestep, as summarized in Table 3.1.

The robot pose in the world frame is represented as $[x, y, \sin\psi, \cos\psi]$, where the sine and cosine encoding avoids discontinuities associated with angular wrap-around. The goal pose is provided both in the world frame and in the robot body frame, where the body-frame representation captures the relative goal direction with respect to the current robot heading.

Obstacle positions are included in both world and body frames. Each obstacle is represented only by its planar position $[o_x, o_y]$ without orientation, since obstacles are modeled as circular regions in the navigation space. Including both coordinate frames allows the policy to reason about the global geometry of the environment as well as the relative spatial relationships between the robot and nearby obstacles.

In addition to the geometric observations, the policy receives planning-related variables from the previous timestep: the previous MPC velocity command u_{t-1}^{cmd} and the previous meta action a_{t-1} that controlled obstacle inflation in the visibility graph. Providing these variables helps the policy maintain temporal consistency when adapting the planning parameters.

The policy is trained using an *asymmetric actor–critic* architecture. The actor receives observations that may contain realistic perception noise and control delay, simulating the conditions of real deployment. The critic receives privileged observations that concatenate both the noisy/delayed state and the clean ground-truth state. This design improves value

Table 3.1: Observation components used by the RAVEN meta RL policy. Actor observations correspond to noisy and delayed robot states, while the critic receives privileged observations including both noisy and clean states. In the default configuration $K = 3$, giving actor dimension $d = 30$ and critic input dimension 60.

Observation component	Dim.	Actor	Critic
Robot pose (world frame)	4	✓	✓
Goal pose (world frame)	4	✓	✓
Goal pose (body frame)	4	✓	✓
Obstacle positions (world)	2K	✓	✓
Obstacle positions (body frame)	2K	✓	✓
Previous MPC command	3	✓	✓
Previous meta action	K	✓	✓
Total actor dimension d	$15 + 5K$		
Privileged critic observation	$2d$ (noisy + clean)		

estimation during training while ensuring that the deployed actor policy relies only on observations available in the real system.

3.2.2 Action Space

The meta-policy outputs a continuous action vector

$$a_t \in [-1, 1]^K, \tag{3.4}$$

where K is the number of obstacles in the environment and each element corresponds to one obstacle. Each action component is interpreted as a parameter controlling the effective obstacle size used during visibility-graph construction. Specifically, the raw action is mapped

to a planning radius for each obstacle through an affine transformation:

$$r_{t,i} = \tfrac{1}{2}(a_{t,i} + 1)(r_{\max} - r_{\min}) + r_{\min}, \quad i = 1, \ldots, K, \tag{3.5}$$

where $r_{\min}$ and $r_{\max}$ define the allowable range of obstacle inflation. These radii are then used to parameterize the obstacle geometry in the visibility-graph planner, effectively modifying the free-space geometry and influencing the resulting path topology.

By increasing the planning radius, the meta-policy enlarges obstacle regions and generates trajectories with larger safety margins, improving robustness to delays and tracking errors. By decreasing the radius, the planner produces shorter, more aggressive paths. The resulting planning parameters are passed to the visibility-graph planner, whose output trajectory is subsequently tracked by the CF-MPC layer.

3.2.3 Reward Function

The reward function is designed to encourage efficient and safe navigation while maintaining smooth planning behavior. The overall reward at each timestep is a sum of several components, detailed in Table 3.2.

The **time penalty** and **path-length penalty** encourage the robot to reach the goal efficiently and avoid unnecessarily long trajectories. These terms discourage the planner from generating overly conservative paths that significantly increase travel distance.

The **collision penalty** and **penetration penalty** promote safety by penalizing configurations where the robot approaches or intersects with obstacle regions. The collision term penalizes entry into a predefined safety radius around obstacles, while the penetration term provides an additional penalty proportional to the depth of overlap. Together, these encourage the meta-policy to select inflation parameters that generate collision-free paths with sufficient clearance.

The **action-rate penalty** regularizes the meta-policy output by penalizing large changes in obstacle inflation parameters between consecutive timesteps, promoting smoother adap-

Table 3.2: Reward components used by the RAVEN meta RL policy. Default parameters: $\epsilon_p = 0.2\,\mathrm{m}$, $\epsilon_\psi = 0.2\,\mathrm{rad}$, $r_{\mathrm{pen}} = 1.0\,\mathrm{m}$.

Component	Equation	Weight
Time penalty	1	-5.0
Path length	$\|p_t - p_{t-1}\|_2$	-20.0
Collision penalty	$\sum_i \mathbf{1}\{\|p_t - o_i\|_2 < r_{\mathrm{pen}}\}$	-8.0
Penetration penalty	$\sum_i \max(0,\, r_{\mathrm{pen}} - \|p_t - o_i\|_2)$	-80.0
Action rate	$\sum_i (r_{t,i} - r_{t-1,i})^2$	-0.5
Success bonus	$\mathbf{1}\{\|p_t - g\|_2 < \epsilon_p \ \wedge\ \|\mathrm{wrap}(\psi_t - \psi_g)\| < \epsilon_\psi\}$	$+5000$
Fall penalty	$\mathbf{1}\{\text{fall detected}\}$	-50000

tation and preventing unstable planning behavior.

Finally, sparse **terminal rewards** represent task success and failure. A large positive reward is granted when the robot reaches the goal pose within specified position and orientation tolerances, while a large negative penalty is applied if the robot falls during execution.

3.2.4 Episode Design

Each training episode consists of a navigation task where the robot starts from a randomly initialized pose and must reach a specified goal pose while avoiding K obstacles. Obstacle positions are randomized within the environment bounds at the start of each episode to ensure broad coverage of navigation scenarios. The episode terminates upon one of three conditions: the robot reaches the goal within the specified tolerances, a failure condition such as a fall or timeout occurs, or a maximum time horizon is exceeded.

Observation noise and control delay are injected during training to match real-world de-

ployment conditions. Gaussian noise is added to the robot pose observations, and actuation delay is simulated by buffering and replaying control commands with a configurable lag. This forces the meta-policy to learn inflation strategies that are robust to the uncertainties the robot will encounter on hardware.

3.3 Guided Exploration via Shortest-Path Structure

A key advantage of the RAVEN architecture over pure RL navigation is the structured exploration it provides through the visibility-graph backbone. Because the visibility graph computes shortest paths under the current obstacle geometry, the RL adaptation effectively searches over a *family of shortest-path solutions* parameterized by the inflation radii.

This structured prior guides exploration toward near-optimal navigation strategies and reduces convergence to suboptimal behaviors compared with pure reinforcement learning. A pure RL navigation policy must discover collision avoidance strategies from scratch through random exploration, which can be slow and lead to conservative local minima. In RAVEN, even a randomly initialized meta-policy produces geometrically valid paths; learning only needs to discover the best member of the shortest-path family for a given system characteristic.

3.4 Locomotion Policy Integration

At the lowest level of the hierarchy, a locomotion policy converts the velocity commands output by the CF-MPC into joint-level motor commands. For the hardware experiments described in Chapter 4, we deploy the open-source Booster Gym locomotion policy, which takes as input a 47-dimensional observation vector including the gravity vector, IMU data, and joint states, and outputs a 12-dimensional command corresponding to the 12 leg actuators.

To ensure consistency between simulation and reality, the PyTorch-based locomotion pol-

icy was converted into a JIT-compilable format and embedded within the RL-MPC training environment. This allows the physics simulation and the locomotion policy to run on the GPU in a fully vectorized manner, enabling efficient parallel training across thousands of environments.

Because low-level motion execution and constraint enforcement are handled by the MPC rather than a learned policy, the proposed framework relies less on precise simulation of robot dynamics than end-to-end RL navigation approaches. This structural choice reduces the sim-to-real gap, a property we validate in the hardware experiments.

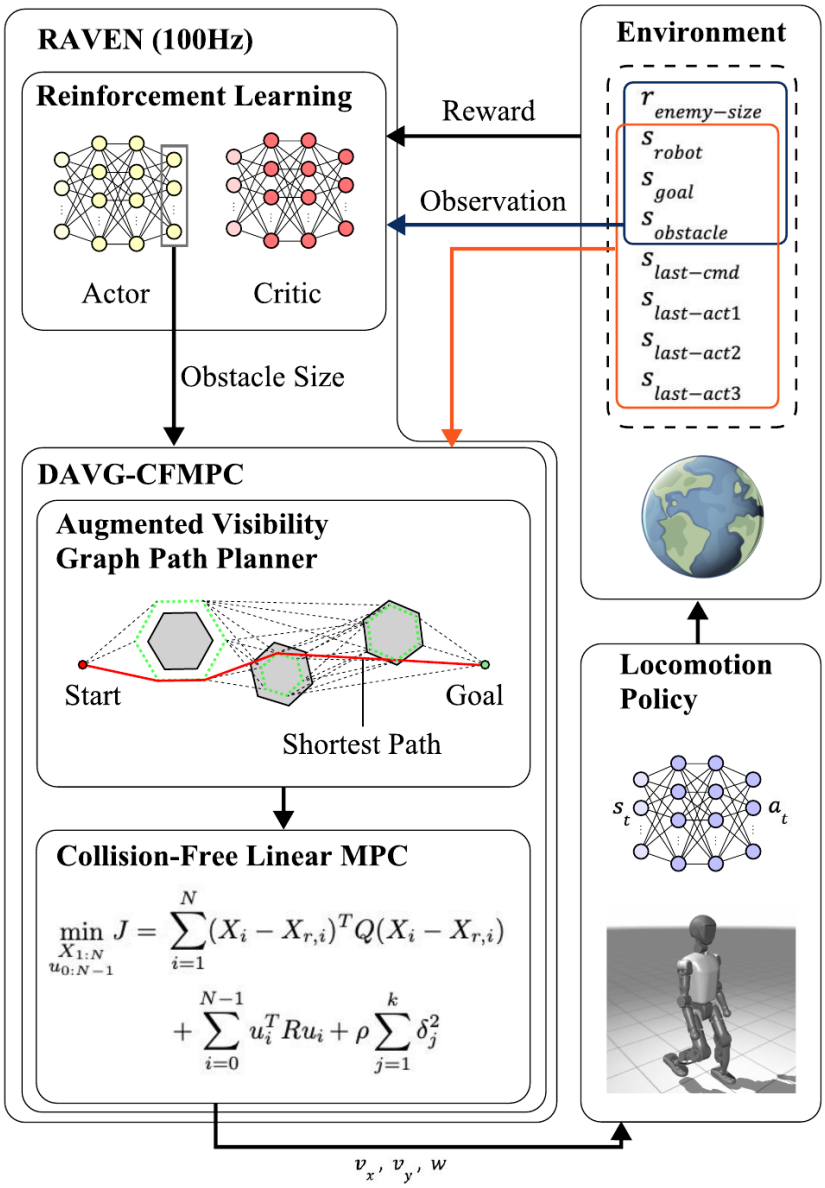
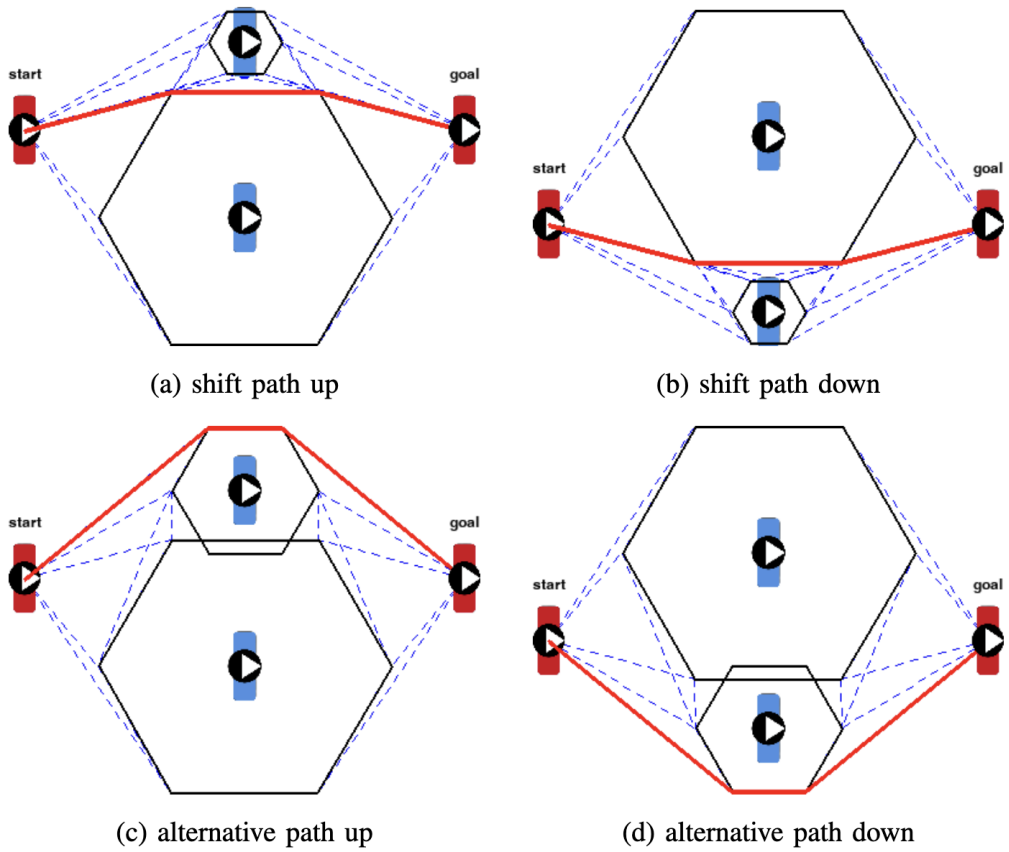


Figure 3.1: Overview of the RAVEN architecture showing the three hierarchical layers: RL meta-policy (top), visibility-graph planner and CF-MPC (middle), and locomotion policy (bottom). The 100 Hz label refers to the visibility-graph and MPC planning loop; the meta-policy inference runs at 120 Hz on hardware (see Section 4.4).



(a) shift path up

(b) shift path down

(c) alternative path up

(d) alternative path down

Figure 3.2: Illustration of how adapting obstacle inflation size changes the visibility graph and alters the resulting path. Increasing the inflation radius shifts the path up or finds an alternative route, while decreasing it allows the planner to pass closer to obstacles.

CHAPTER 4

Experimental Setup

This chapter describes the experimental setup used to evaluate RAVEN, including the baseline methods, RL training details, simulation environment, and hardware platform. All experiments use the same robot model, velocity limits, acceleration limits, and injected delay and observation noise to ensure a fair comparison across methods.

4.1 Baseline Methods

We compare RAVEN against two baseline methods: a classical MPC baseline and a pure end-to-end RL baseline.

4.1.1 MPC Baseline

The MPC baseline uses the DAVG-CFMPC method [HFZ25] with a fixed obstacle inflation radius of $r = 1.0\,\mathrm{m}$ for all obstacles. This radius was manually tuned to represent a reasonable operating point for the nominal zero-delay scenario, and is held constant throughout all experiments. This baseline represents the state-of-the-art classical planning approach for this platform and isolates the contribution of learned adaptive inflation.

4.1.2 End-to-End RL Baseline

The end-to-end RL baseline is a policy that maps observations directly to velocity commands, without any intermediate planning layer. Its observation space is summarized in Table 4.1

and its reward function in Table 4.2.

Table 4.1: Observation components for the end-to-end RL baseline. In the default configuration $K = 3$, giving actor dimension $d = 27$ and critic input dimension 54.

Observation component	Dim.	Actor	Critic
Robot pose (world frame)	4	✓	✓
Goal pose (world frame)	4	✓	✓
Goal pose (body frame)	4	✓	✓
Obstacle positions (world)	2K	✓	✓
Obstacle positions (body frame)	2K	✓	✓
Previous velocity command	3	✓	✓
Total actor dimension d	$15 + 4K$		
Privileged critic observation	$2d$		

The reward design combines a dense progress signal with collision avoidance and smoothness regularization. The success bonus for the RL baseline is intentionally smaller than RAVEN's (100 vs. 5000) to reflect the different reward scales appropriate for each policy's action space.

4.2 RL Training Details

Both RAVEN and the pure RL baseline are trained using Proximal Policy Optimization (PPO) implemented in the Brax training framework. The policy and value networks are multilayer perceptrons with hidden sizes $(512, 256, 128)$ and Swish activation functions. The policy outputs a tanh-squashed diagonal Gaussian distribution over actions. The actor receives noisy robot state observations, while the critic receives privileged observations con-

Table 4.2: Reward function for the end-to-end RL baseline. Default parameters: $\epsilon_p = 0.2\,\mathrm{m}$, $\epsilon_\psi = 0.2\,\mathrm{rad}$, $r_{\mathrm{pen}} = 1.0\,\mathrm{m}$.

Component	Equation	Weight
Position error	$\|p_t - g\|_2$	-2.0
Yaw error	$\|\mathrm{wrap}(\psi_t - \psi_g)\|$	-0.5
Collision penalty	$\sum_i \mathbf{1}\{\|p_t - o_i\|_2 < r_{\mathrm{pen}}\}$	-5.0
Penetration penalty	$\sum_i \max(0,\, r_{\mathrm{pen}} - \|p_t - o_i\|_2)$	-5.0
Progress shaping	$d_{t-1} - d_t,\ d_t = \|p_t - g\|_2$	$+0.5$
Smoothness penalty	$\|u_t^{\mathrm{cmd}} - u_{t-1}^{\mathrm{cmd}}\|_2^2$	-0.1
Forward motion	$(p_t - p_{t-1})^\top \frac{g - p_t}{\|g - p_t\|}$	$+0.5$
Time penalty	1	-5.0
Success bonus	$\mathbf{1}\{\|p_t - g\|_2 < \epsilon_p \ \wedge\ \|\mathrm{wrap}(\psi_t - \psi_g)\| < \epsilon_\psi\}$	$+100.0$
Fall penalty	$\mathbf{1}\{\text{fall detected}\}$	-50000

catenating both noisy and clean states to stabilize value estimation.

Training is performed in JAX using the MuJoCo Playground MJX environments, which enable fully vectorized physics simulation on GPU. For RAVEN, the DAVG-CFMPC layer is implemented in JAX and solved via projected gradient optimization using JAXopt, allowing both the physics simulation and the MPC optimization to run on the GPU in parallel across all environments.

For RL+MPC training, we use $N_{\mathrm{env}} = 1024$ parallel environments with an unroll length of $T = 32$ steps. Due to the embedded MPC optimization, the training throughput is approximately 10,000 simulation steps per second (SPS) on an NVIDIA RTX 5090 GPU. In comparison, the pure RL baseline achieves roughly 100,000 SPS because it does not require solving an optimization problem at each control step.

To ensure a fair comparison, we allocate comparable total training *time* for both approaches. Specifically, RAVEN is trained for 10^8 environment steps, while the pure RL baseline is trained for 10^9 steps due to its higher simulation throughput. This protocol ensures the RL baseline is not disadvantaged by shorter training duration despite its faster simulation speed.

Additional PPO hyperparameters follow standard Brax configurations: generalized advantage estimation ($\lambda = 0.95$), clipping parameter $\epsilon = 0.2$, advantage normalization, observation normalization, and gradient norm clipping. These settings are kept identical across both methods.

4.3 Simulation Environment

All simulation experiments are conducted in a planar 2D navigation environment implemented in MuJoCo MJX. The environment contains $K = 3$ circular obstacles with randomized positions at the start of each episode. The robot must navigate from a randomly initialized start pose to a goal pose while avoiding the obstacles.

Observation noise is modeled as additive Gaussian noise on the robot position and orientation. Control delay is simulated by buffering actuation commands in a FIFO queue and replaying them with a configurable lag. We evaluate all methods under two conditions: (a) a zero-delay, zero-noise ideal scenario, and (b) a realistic scenario with 0.06 s actuation delay and observation noise.

4.4 Hardware Platform

RAVEN was deployed on the T1 Booster bipedal humanoid robot. The T1 Booster has a height of 1.18 m, a mass of 30 kg, and 23 degrees of freedom. At the locomotion layer, we deploy the open-source Booster Gym locomotion policy.

Physical deployment was conducted in a half-sized RoboCup soccer field. Motion capture cameras provided global position and orientation estimates, validating velocity tracking performance and providing localization data for the mid-level policy. Due to physical space constraints, mocap coverage was restricted to $[2.0, 6.0]\,\mathrm{m}$ in the x-axis and $[-2.5, 2.5]\,\mathrm{m}$ in the y-axis; all start positions, goal positions, and physical obstacles were localized within this bounding box.

The high-level RAVEN path planning policies were inferred on an external NVIDIA RTX 4050 GPU and communicated to the T1 Booster's internal computer via ROS2. On the physical robot, the locomotion policy is executed locally on the onboard Express-i7 computer, where the low-level joint control loop runs at $500\,\mathrm{Hz}$ and policy inference is performed at $50\,\mathrm{Hz}$. The visibility graph and MPC layers ran at $100\,\mathrm{Hz}$ and the meta-policy ran at $120\,\mathrm{Hz}$, well within real-time requirements.

CHAPTER 5

Results and Discussion

This chapter presents the evaluation of RAVEN against the two baseline methods described in Chapter 4. We first analyze simulation results under varying levels of control delay and observation noise, then report hardware experiments validating sim-to-real transfer on the T1 Booster bipedal humanoid.

5.1 Simulation Results: Robustness to Delay and Noise

5.1.1 Qualitative Behavior

To illustrate the behavior of RAVEN, we consider a representative navigation scenario with a control delay of 0.06 s. The robot starts at pose $(x, y, \theta) = (3.0, -1.0, -1.57)$ and must reach the goal pose $(6.0, 1.8, 0.0)$. Three obstacles are located at $(5.4, -1.4)$, $(4.4, 1.0)$, and $(3.0, 1.5)$.

As the robot approaches the first obstacle, the meta-policy increases the corresponding inflation radius, causing the visibility graph to route the robot along a wider arc and avoiding the overshoot that a fixed-inflation planner would exhibit under the 0.06 s delay. When the robot enters the narrow passage between the first two obstacles, both inflation radii increase simultaneously, generating a path that points in the tangential direction between the two inflated circles and prevents the robot from running into either obstacle. After the robot clears the narrow passage, the inflation radius of the second obstacle shrinks, enabling the planner to generate a direct, efficient path to the goal.

This adaptive behavior, which expands margins near obstacles and contracts them in open space, cannot be achieved by a fixed inflation radius and is qualitatively distinct from the behavior of the pure RL baseline, which maintains a uniformly conservative standoff from all obstacles.

5.1.2 Quantitative Comparison

Table 5.1 summarizes the performance metrics for the zero-delay ideal scenario and the realistic 0.06 s actuation delay scenario. Each result is averaged over 100 randomized navigation episodes.

Table 5.1: Simulation performance metrics under different delay conditions. Bold indicates best performance per metric.

(a) No Delay, No Noise

Metric	MPC	RL	RAVEN
Average Path Length (m)	9.838	9.645	**9.23**
Average Time to Completion (s)	11.43	11.55	**11.28**
Ave. Max Obstacle Penetration (m)	0.049	**0.016**	0.019

(b) 0.06 s Actuation Delay

Metric	MPC	RL	RAVEN
Average Path Length (m)	11.25	9.801	**9.33**
Average Time to Completion (s)	12.32	12.21	**11.58**
Ave. Max Obstacle Penetration (m)	0.128	**0.0**	0.03

5.1.3 Analysis

Zero-delay scenario. In the ideal, zero-delay zero-noise environment (Table 5.1a), all three methods perform competently. RAVEN already demonstrates superior geometric efficiency, achieving the shortest average path length (9.23 m) and fastest completion time (11.28 s). The pure RL baseline achieves the lowest obstacle penetration depth (0.016 m) in this condition, but RAVEN's 0.019 m is comparably safe.

Delayed scenario. Introducing a 0.06 s delay (Table 5.1b) drastically alters the performance landscape. The classical DAVG-CFMPC baseline degrades severely: its static obstacle inflation cannot account for delayed tracking, causing dangerous trajectory overshoot. The maximum obstacle penetration depth spikes to 0.128 m, and reactive avoidance maneuvers balloon the average path length to 11.25 m.

The pure end-to-end RL baseline proves highly resilient to collisions under delay, achieving 0.0 m penetration depth. However, this safety comes at the cost of efficiency: lacking the structural prior of a shortest-path graph, the RL agent adopts an overly conservative, reactive policy that yields a slower average completion time (12.21 s).

RAVEN successfully bridges this gap. By observing the delayed system state, the meta-policy dynamically increases the obstacle inflation parameters within the visibility graph. This geometric adaptation guides the underlying MPC along a wider, more conservative trajectory around obstacles, explicitly mitigating the overshoot caused by the delay without sacrificing global optimality. RAVEN maintains a safely bounded penetration depth (0.03 m) while achieving the most efficient average path length (9.33 m) and fastest completion time (11.58 s) under delay.

These results demonstrate that RAVEN outperforms both the brittle static MPC and the overly cautious end-to-end RL agent, delivering smooth, geometrically efficient, and safe navigation under realistic real-world uncertainties.

5.2 Hardware Experiments: Sim-to-Real Transfer

To validate the practical applicability of RAVEN, the framework was deployed on the T1 Booster bipedal humanoid in a half-sized RoboCup soccer field, as described in Section 4.4.

The real-world trajectories closely matched the corresponding simulation rollouts. On hardware, the meta-policy showed the same adaptive inflation behavior of expanding near obstacles and contracting in open space, confirming that its learned strategy transfers successfully from simulation.

The results also confirm that the proposed architecture is less sensitive to sim-to-real discrepancies than pure RL policies. Because the MPC layer provides a physics-based control backbone with explicit constraint enforcement, and the RL meta-policy only adapts high-level geometric planning parameters, the framework reduces dependence on precise low-level dynamics simulation compared with end-to-end RL. The MPC's collision-avoidance constraints remain active and enforced regardless of any gap between simulated and real locomotion dynamics.

5.3 Discussion

5.3.1 Why Geometric Adaptation Works

The central insight of RAVEN is that obstacle inflation is a remarkably expressive control knob for compensating real-world navigation uncertainties. A robot suffering from actuation delay effectively behaves as if it is larger than it is: it overshoots corners and penetrates margins that were safe under the nominal model. The meta-policy directly counteracts this effect by increasing the inflation radius for approaching obstacles, reshaping the planner's free-space geometry before the MPC begins its optimization. Because it acts on the planning geometry rather than on the controller internals, it can route the robot through a different gap or along a wider arc, producing qualitatively different path topologies, without any

modification to the MPC formulation or the locomotion policy. It is precisely this separation of concerns, with the meta-policy governing geometry and the classical controller governing dynamics, that makes the adaptation robust and interpretable.

5.3.2 Structured vs. Unstructured Exploration

The comparison with the pure RL baseline also highlights the role of structured exploration. The RL baseline achieves strong collision avoidance but at the cost of efficiency, because it must discover navigation strategies from scratch through unguided exploration. RAVEN's visibility-graph backbone ensures that every meta-policy output, even at initialization, generates geometrically valid shortest paths. The meta-policy only needs to discover the best inflation parameters for a given system characteristic. This is a much smaller search space than discovering collision avoidance from raw observations, which is why RAVEN converges to more efficient strategies than the pure RL agent.

5.3.3 Limitations

The current RAVEN framework has several limitations that motivate future work. First, the visibility graph requires a known, static obstacle map; RAVEN cannot currently adapt to moving or previously unseen obstacles. Second, the circular obstacle model is a simplification; real environments contain non-convex obstacles that a circular inflation model approximates less accurately. Third, the number of obstacles K is fixed at training time; extending to variable numbers of obstacles would require a more flexible observation representation, such as an attention-based architecture. These limitations are addressed in the future work discussion of Chapter 6.

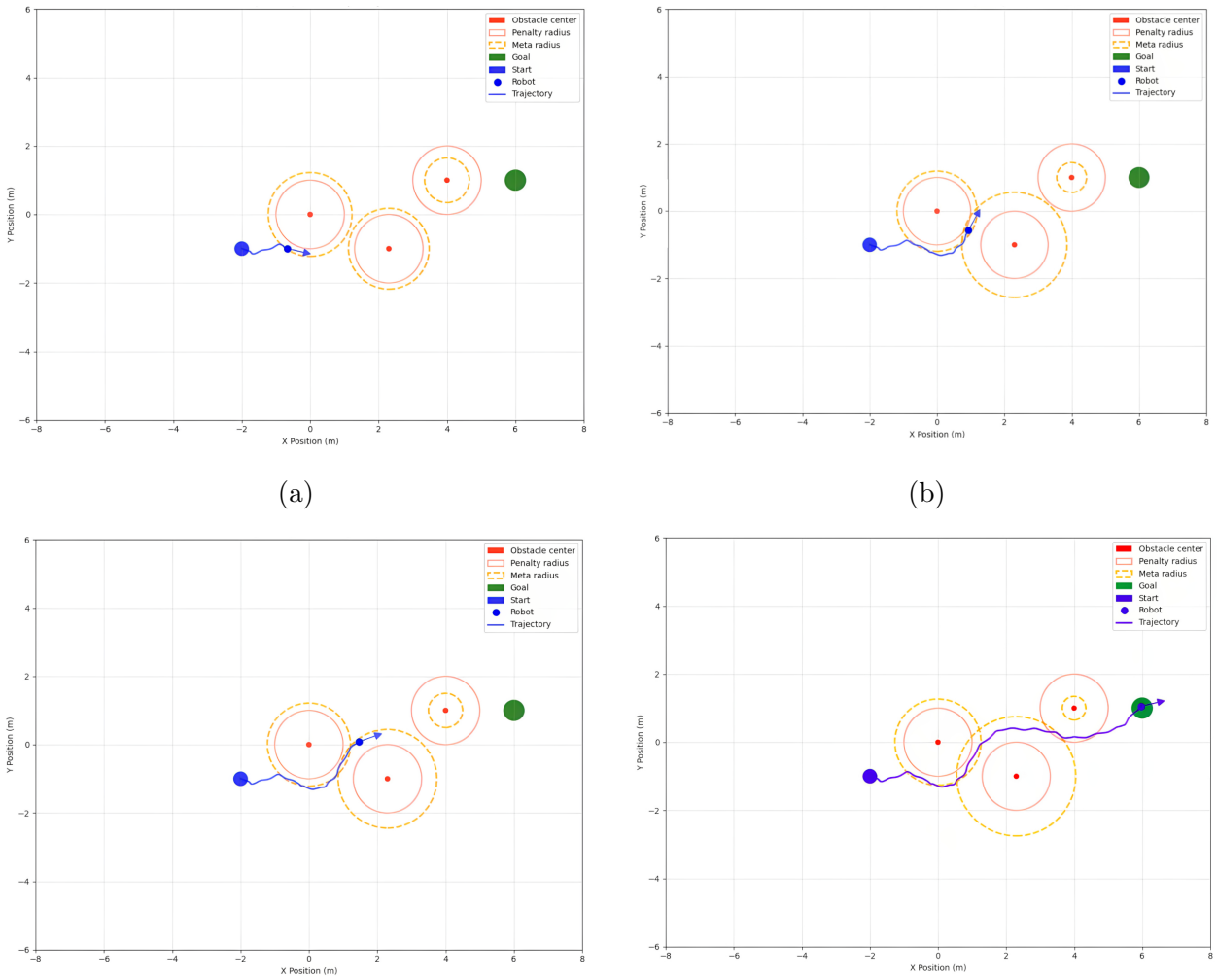
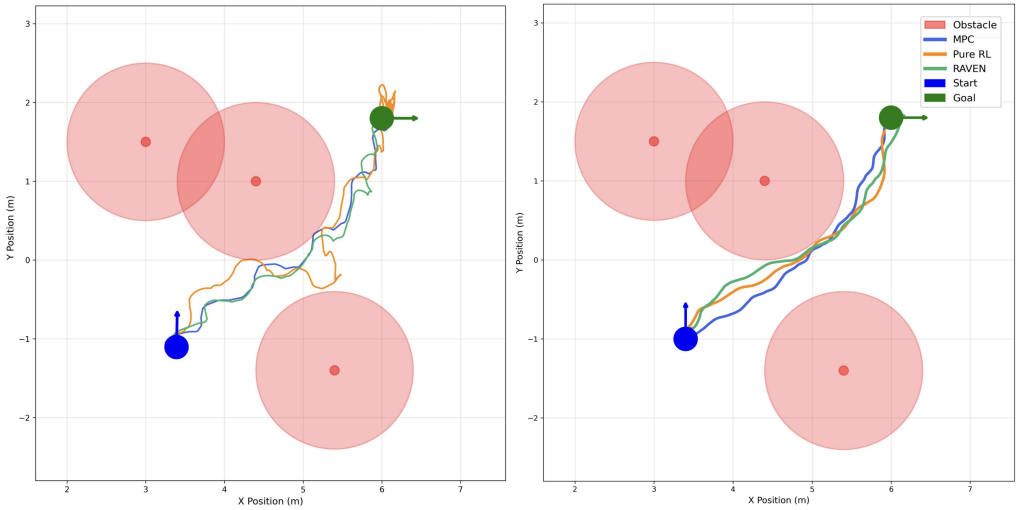


Figure 5.1: Illustration of how RAVEN adapts the planned path under different obstacle configurations. The adaptive inflation radii reshape the free-space geometry, producing qualitatively distinct trajectories compared to a fixed-radius planner.



(a) Real-world experiment trajectories (b) Simulated rollout trajectories

Figure 5.2: Comparison of trajectories between real-world experiments and simulation rollouts, demonstrating successful sim-to-real transfer of the learned meta-policy.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

This thesis presented RAVEN, a hierarchical framework that tightly couples a learning-based global planner with a constrained local controller for robust humanoid navigation. The core idea is to place reinforcement learning at the level of geometric planning rather than motion control: a meta-policy adjusts the obstacle inflation parameters of a visibility graph online, reshaping the free-space topology to account for actuation delays, state-estimation errors, and locomotion imperfections before the CF-MPC layer begins tracking. By keeping the constrained optimizer intact and learning only the planning geometry, the system retains the interpretability and explicit constraint handling of classical optimization-based control while gaining the adaptability of learned policies.

The key insight of this work is that *geometric adaptation at the planning level* is a more direct, computationally efficient, and interpretable mechanism for handling real-world navigation uncertainties than tuning the algebraic cost weights of a local controller or replacing the planner entirely with a neural network. By modifying obstacle inflation radii in the visibility graph, the RL meta-policy directly reshapes the free-space topology, expanding safety margins near obstacles when delays are present and contracting them in open space to maintain efficiency.

Through simulation and hardware experiments on the T1 Booster bipedal humanoid, we compared RAVEN against a manually tuned DAVG-CFMPC baseline and a pure end-to-end

RL navigation policy. The results demonstrate that:

- RAVEN achieves the shortest average path length and fastest completion time across both zero-delay and delayed conditions, outperforming both baselines in navigation efficiency.
- RAVEN maintains safely bounded obstacle penetration (0.03 m) under 0.06 s actuation delay, compared to 0.128 m for the classical MPC baseline, while avoiding the overly conservative behavior of the pure RL policy.
- The learned planning adaptation transfers successfully from simulation to hardware, with real-world trajectories closely matching simulation rollouts, validating the sim-to-real robustness of the hierarchical architecture.

These findings indicate that reinforcement-adaptive graph construction combined with constrained MPC provides an effective and interpretable alternative to end-to-end learning for robust humanoid navigation.

6.2 Future Work

Several directions for future work are motivated by the results and limitations of the current framework.

6.2.1 Dynamic Environments with Moving Obstacles

The current RAVEN framework assumes a known, static obstacle map. Extending it to dynamic environments with moving obstacles is a natural and important next step. One approach would be to augment the observation space with estimated obstacle velocities and train the meta-policy to predict inflation radii that account for both the robot's own tracking delay and the obstacle's predicted future position. The visibility-graph backbone would need

to be replaced or augmented with a dynamic obstacle avoidance layer.

6.2.2 Non-Convex and Unknown Obstacle Geometries

Obstacles are currently modeled as circles with a single inflation radius. Extending the framework to handle non-convex or polygonal obstacles would improve applicability to real-world environments, where objects have complex shapes. One direction is to represent each obstacle as a set of convex components and assign separate inflation parameters to each component, allowing the meta-policy to adapt the effective geometry more precisely.

6.2.3 Variable Numbers of Obstacles

The current architecture fixes the number of obstacles K at training time, limiting flexibility in deployment. Replacing the fixed-size obstacle representation with an attention-based or set-based architecture would allow the meta-policy to generalize to environments with variable numbers of obstacles, including newly appearing obstacles not seen during training.

6.2.4 More Advanced Locomotion Models

The locomotion policy is treated as a fixed black box in the current framework. Incorporating a more detailed model of the locomotion policy's response characteristics, including its step frequency, foot placement variability, and speed-dependent stability margins, into the RL observation space could allow the meta-policy to learn even more precise inflation strategies tailored to the robot's actual locomotion behavior at different speeds.

6.2.5 Multi-Robot and Social Navigation

An exciting long-term direction is extending RAVEN to multi-robot settings and socially aware navigation among pedestrians. In these settings, each "obstacle" has its own dynamics and intentions. The visibility-graph backbone could be augmented with social force models

or predictive pedestrian trajectory models, while the RL meta-policy learns to adapt inflation parameters based on predicted collision risk rather than just geometric proximity.

In summary, RAVEN establishes a principled and practical framework for learning-augmented geometric planning on humanoid robots. The combination of RL-adaptive visibility-graph planning with collision-free MPC offers a path toward navigation systems that are simultaneously agile, safe, interpretable, and robust to the inevitable uncertainties of real-world deployment.

Ethics

The content of this thesis is substantially based on a conference paper submitted to the 2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), of which I am a co-author. Generative AI was used solely for assistance with the language of this work, e.g., paraphrasing, spell-checking, or polishing the original content, without suggesting new content. No AI tools were used to generate any scientific content, experimental results, or conclusions presented in this thesis.

REFERENCES

[BLX21] Osbert Bastani, Shuo Li, and Anton Xue. "Safe Reinforcement Learning via Statistical Model Predictive Shielding." In *Robotics: Science and Systems (RSS)*, 2021.

[CCJ24] William D. Compton, Noel Csomay-Shanklin, Cole Johnson, and Aaron D. Ames. "Dynamic Tube MPC: Learning Tube Dynamics with Massively Parallel Simulation for Robust Safety in Practice." *arXiv preprint arXiv:2411.15350*, 2024.

[CFH20] Jan Carius, Farbod Farshidian, and Marco Hutter. "MPC-Net: A First Principles Guided Policy Search." *IEEE Robotics and Automation Letters*, **5**(2):2897–2904, 2020.

[HFZ25] Ruochen Hou, Gabriel I. Fernandez, Mingzhang Zhu, and Dennis W. Hong. "Model Predictive Control with Visibility Graphs for Humanoid Path Planning and Tracking Against Adversarial Opponents." In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14520–14526. IEEE, 2025.

[HRS23] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. "ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots." *arXiv preprint arXiv:2306.14874*, 2023.

[HWK26] Ruochen Hou, Shiqi Wang, Beom Jun Kim, Hanzhang Fang, Mehak Singal, and Dennis Hong. "RAVEN: Reinforcement-Adaptive Visibility-Graph Planning for Robust Humanoid Navigation with Collision-Free MPC." In *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026. Submitted.

[JBN18] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. "Residual Reinforcement Learning for Robot Control." *arXiv preprint arXiv:1812.03201*, 2018.

[LFD25] Huajian Liu, Yixuan Feng, Wei Dong, Kunpeng Fan, Chao Wang, and Yongzhuo Gao. "Hierarchical Learning-Enhanced MPC for Safe Crowd Navigation with Heterogeneous Constraints." *arXiv preprint arXiv:2506.09859*, 2025.

[LZL25] Wei Liu, Huihua Zhao, Chenran Li, Yuchen Deng, Joydeep Biswas, Soha Pouya, and Yan Chang. "COMPASS: Cross-Embodiment Mobility Policy via Residual RL and Skill Synthesis." *arXiv preprint arXiv:2502.16372*, 2025.

[MCK18] Mohit Mehndiratta, Efe Camci, and Erdal Kayacan. "Automated Tuning of Nonlinear Model Predictive Controller by Reinforcement Learning." In *2018*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3016–3021, 2018.

[PZG25] Chengyang Peng, Zhihao Zhang, Shiting Gong, Sankalp Agrawal, Keith A. Redmill, and Ayonga Hereid. "Reinforcement Learning with Data Bootstrapping for Dynamic Subgoal Pursuit in Humanoid Robot Navigation." *arXiv preprint arXiv:2506.02206*, 2025.

[RAS24] Angel Romero, Elie Aljalbout, Yunlong Song, and Davide Scaramuzza. "Actor-Critic Model Predictive Control: Differentiable Optimization Meets Reinforcement Learning for Agile Flight." In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[SKA23] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. "Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms." *IEEE Robotics and Automation Letters*, **8**(4):2397–2404, 2023.

[SLA24] Vishnu D. Sharma, Jeongran Lee, Matthew Andrews, and Ilija Hadžić. "Hybrid Classical/RL Local Planner for Ground Robot Navigation." *arXiv preprint arXiv:2410.03066*, 2024.

[ZFW23] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Sören Schwertfeger, Chelsea Finn, and Hang Zhao. "Robot Parkour Learning." *arXiv preprint arXiv:2309.05665*, 2023.

[ZHW23] Longyuan Zhang, Ziyue Hou, Ji Wang, Ziang Liu, and Wei Li. "Robot Navigation with Reinforcement Learned Path Generation and Fine-Tuned Motion Control." *IEEE Robotics and Automation Letters*, **8**(8):4489–4496, 2023.

[ZKL16] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. "Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search." *arXiv preprint arXiv:1509.06791*, 2016.

[ZSB24] Baha Zarrouki, Marios Spanakakis, and Johannes Betz. "A Safe Reinforcement Learning Driven Weights-Varying Model Predictive Control for Autonomous Vehicle Motion Control." In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1401–1408, 2024.