UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title

Learning to Perceive and Control for Robust Autonomy Across Aerial and Legged Platforms

Permalink

https://escholarship.org/uc/item/1413m1b2

ISBN

9798276025650

Author

Chen, Shuxiao

Publication Date

2025-12-06

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Learning to Perceive and Control for Robust Autonomy Across Aerial and Legged Platforms

By

Shuxiao Chen

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Koushil Sreenath, Chair
Professor Kameshwar Poolla
Professor Pieter Abbeel

Fall 2025

Learning to Perceive and Control
for Robust Autonomy Across Aerial and Legged Platforms

Copyright 2025
by
Shuxiao Chen

Abstract

Learning to Perceive and Control
for Robust Autonomy Across Aerial and Legged Platforms

by

Shuxiao Chen

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Professor Koushil Sreenath, Chair

The deployment of mobile robots in unstructured, real-world environments demands robust solutions for both perception and control. Conventional approaches often rely on precise, expensive sensors and rigid control structures, which can be brittle under noise and constraints. This dissertation explores a shift toward learning-based frameworks to enable robust, efficient, and self-reliant autonomy across aerial and legged platforms.

To address the challenge of aerial navigation in global positioning system-denied environments, we first introduce a real-time, image-based geo-localization framework. By utilizing a database of rendered synthetic views from satellite imagery and topography, this two-stage pipeline performs global and local descriptor matching to recover the 6-degree-of-freedom vehicle pose. The system achieves onboard inference at approximately 1.1 Hz with a position root-mean-square error of 2.8 m, demonstrating robustness to camera pitch variations, lens flare, and motion blur.

Transitioning to legged platforms, we address the limitations of stiffness in position-based control by proposing an end-to-end learning torque control framework for quadrupedal robots. This reinforcement learning (RL) approach outputs joint torques directly at high frequency, bypassing conventional low-level proportional-derivative controllers. Validated on a quadrupedal robot, the framework demonstrates superior tracking accuracy and dynamic compliance, enabling the robot to recover from significant external disturbances and traverse compliant terrains such as air mattresses and pebbles.

To bridge the gap between control and perception, we present a modular legged state estimation framework. In contrast to concurrent training or teacher-student distillation, which can hinder modularity and tracking performance, we decouple training by optimizing an RL-based estimator against a fixed controller within a unified environment. This approach ensures effective sim-to-real transfer and achieves superior base linear velocity command

tracking compared to concurrent baselines.

Finally, we propose a cross-modality distillation pipeline to achieve cost-effective perceptive locomotion. We distill the geometric reasoning of an elevation map-based teacher policy into a student policy that relies solely on a monocular color camera using a diffusion transformer architecture. The resulting image-based policy enables a humanoid robot to traverse complex terrain, such as stairs, matching the performance of the teacher without requiring explicit depth sensors or map fusion. Collectively, these frameworks support robot autonomy by integrating learning-based perception and control.

To my family and friends.

Contents

Contents **ii**

List of Figures **iv**

List of Tables **viii**

1 Introduction **1**
1.1 Motivation 1
1.2 Contributions 2
1.3 Organization 3

2 Literature Survey **5**
2.1 Overview 5
2.2 Visual Geo-localization in GPS-denied Environments 6
2.3 Control Paradigms for Legged Locomotion 8
2.4 State Estimation Strategies 9
2.5 Perceptive Locomotion and Sensor Modalities 11
2.6 Summary 12

3 Real-time Geo-localization Using Satellite Imagery **13**
3.1 Introduction 13
3.2 Methodology 18
3.3 Experimental Setup and Results 21
3.4 Summary 27

4 Learning Torque Control for Quadrupedal Locomotion **29**
4.1 Introduction 29
4.2 Framework 32
4.3 Results and Analysis 37
4.4 Summary 42

5 Modular Legged State Estimation via Reinforcement Learning **44**
5.1 Introduction 44

5.2 Methodology 46
5.3 Experimental Validation 49
5.4 Summary 57

6 Cross-modality Distillation for Perceptive Locomotion 58
6.1 Introduction 58
6.2 Distillation Pipeline 61
6.3 Results 66
6.4 Summary 68

7 Conclusion and Future Work 69
7.1 Summary of Contributions 69
7.2 Future Work 70
7.3 Summary 72

Bibliography 73

List of Figures

3.1 The proposed geo-localization pipeline. The steps numbered in Roman numerals (i-iv) introduce the procedure of the offline database preparation (Section 3.2.1) of the descriptor database and those in Arabic numerals (1-6) define the onboard inference (Section 3.2.2) based localization strategy. The pose and field of view of the quadrotor along with a captured query image from one of our experiments are shown. 15
3.2 (Left) Front view of our customized quadrotor used at Location 1. Our camera is mounted and encapsulated by the blue cage. A GPS receiver is installed on the top to record ground truth trajectories. A Jetson AGX Xavier is attached to the back of the cage. (Right) The commercial DJI Mavic Air drone used to collect data at Location 2. The equipped gimballed camera can change the pitch angle resulting in a diversity of query images. 22
3.3 The top view of the trajectory generated by our geo-localization pipeline is shown overlaid on a satellite map of Location 1. The red crosses are the positions of the initial matched candidates at which database images were generated. The yellow markers stand for the vehicle positions after the PnP refinement. The orange line segments show the correspondences between the initial matched positions and those after the refinement. The blue trajectory represents the ground truth positions from GPS. The length of the flight path is 360 m. 24
3.4 A comparison between the inferred and ground truth altitudes versus the distance traveled by the vehicle. Although the vehicle was able to localize itself from 45 m to 73 m in elevation, all database images were generated at a single altitude of 70 m 24
3.5 The top row shows the captured query images from the UAV and the bottom row presents the corresponding matched database images. In the two columns in the middle, there are significant discrepancies between the current query images and the dated satellite maps, e.g., the missing buildings in the scene. 25
3.6 Top view of inferred and ground truth trajectories overlaid on a map of Location 2. The visualization format shares the same fashion as in Figure 3.3. The length of the flight path is 220 m. 26

3.7 A comparison between the inferred and ground truth altitudes versus the distance traveled by the vehicle. Query images captured from 18 m to 70 m in elevation were successfully localized despite all database images being generated at a single altitude of 60 m. 26
3.8 Images collected from the experiment conducted at Location 2. 27

4.1 A quadrupedal robot walks on a hill with a torque-based RL policy. The experimental video can be found at https://youtu.be/18v3Z9BD-W8. 30
4.2 The proposed learning torque control framework. The common position-based method (top left) has two main modules where a neural network (blue) runs at a low frequency and a PD controller (green) executes at a high frequency. The proposed torque-based method (top right) merges the two modules in the position-based method into a high-frequency neural network module (red), as the RL policy is queried at a high frequency to directly output torques. A detailed illustration of the high-frequency neural network module is shown at the bottom, consisting of training in the simulation environment and test on the real quadruped. During the training, we add noise and disturbance to facilitate sim-to-real transfer. During the test, the state information is collected through proprioceptive sensors with appropriate preprocessing. The comprehensive explanation of the framework is presented in Section 4.2. 33
4.3 The proposed end-to-end learning torque control framework allows the A1 robot to traverse various terrains, such as (a) inclined soft lawn, (b) pebble path, (c) compliant terrain, and resist external disturbances such as (d) recovering from a kick. All the tests use the same RL policy queried at 200 Hz. 36
4.4 **Action tracking comparison** between learning position control (top) and learning torque control (bottom) on a real quadruped robot for the rear left calf joint. The green lines depict the target values queried from RL policies with post-processing, e.g., scales. The orange lines illustrate the actual values measured from the real quadruped robot. 39
4.5 **Learning curve comparison** between learning position control and learning torque control. We choose three different proportional gains, i.e., 40, 60, and 80, for the position-based RL, which are denoted as orange, green, and red, respectively, and the torque-based RL is depicted as blue. 40
4.6 **Robustness comparison** between learning position control and learning torque control, where the robot base height is illustrated under successive downward kicks while it is standing. The green lines show the measured base heights, and the blue lines depict the nominal base heights. The red lines illustrate the minimal base heights where the base is on the ground. Note that these vertical kicks are not present during training. 41

5.1 Block diagrams of three architectures for legged locomotion and state estimation are illustrated here: (a) the proposed method, (b) a recently devised concurrent training approach for controller-estimator pairs that has gained popularity [47], and (c) the established, widely-used teacher-student architecture featuring an implicit and inseparable estimator [57]. (a) trains an estimator via RL in a unified simulator-controller environment, while keeping the existing controller fixed. (b) jointly trains a supervised-learned estimator and an RL-based controller tandem from scratch, promoting mutual adaptation. The controller inherently avoids states where the estimator exhibits uncertainty, enhancing system robustness. (c) trains a teacher policy using privileged states, then distills a student policy with practical states to replicate the teacher's performance. During distillation, the student not only adheres to the teacher's control objectives but also implicitly encodes an estimator component alongside. 47
5.2 Root-mean-square (RMS) errors of both the base linear velocity output from the RL-based estimator and the command tracking performance achieved by the concatenated system against critical design aspects are presented in the above group of plots. The RMS errors are calculated using the metric mentioned in Section 5.2 from [47] for faithful comparison. Points surrounded by the dashed green box are the results from an identical baseline estimator with the following configuration: an MLP having a $(1024, 512, 256)$ structure activated by the leaky rectified linear unit (ReLU) with a negative slope of 0.01, a history length of 5 appended to the current observation and a discount factor γ of 0. All the estimators are produced against an identical baseline locomotion controller trained with the parameters mentioned in Section 5.2. 50
5.3 The quadruped is following a velocity command in the motion capture room. . . 56

6.1 Architecture of the proposed cross-modality distillation method. The system consists of two stages: (1) an elevation map-based teacher perceptive locomotion policy trained via RL, which takes a local height map and a proprioceptive state as the input to an MLP, and (2) an image-based student that employs a diffusion transformer to encode the policy. At each time step, the student processes an RGB image through a ResNet-18 encoder, concatenates the latent space to a proprioceptive state, feed the combined observation to a diffusion transformer, and attempts to match the teacher's action through supervised learning. The distillation effectively eliminates the need for expensive depth hardware and computationally intensive fusion algorithms. 60

6.2 Spatial coverage comparison among sensing modalities. This diagram contrasts the field of view and ground coverage between the sensor inputs used for the elevation map-based and the image-based policies. The side view illustrates the vertical mounting and pitch of the sensors on the Unitree G1 robot; the lidar (blue cones) offers a broad range, whereas the RGB camera (orange cone) is mounted on the chest and pitched downward to capture the immediate terrain in the robot's forward path. The top-down view depicts the specific ground footprint of each modality relative to the robot's center. The elevation map (green rectangle), synthesized temporally using a sequence of lidar readings, spans an area of 1.6 m × 1 m. In contrast, the RGB camera input (orange square) captures a subset of this area, covering a trapezoid with a nominal width of approx. 0.7 m directly in front of the robot. The distillation process replaces the original elevation map with a vision-centric input covering a more restricted area. 64

6.3 Sequential snapshots of the humanoid robot transitioning from flat ground to stairs using the diffusion transformer policy. The agent is commanded to steer toward a random heading using a constant forward velocity and variable yaw rate to approach the staircase. The first and third rows illustrate the third-person external view of the robot's motion, while the second and fourth rows display the corresponding first-person RGB camera inputs used by the policy to navigate the terrain. 65

6.4 Simulation sequence of the G1 robot transitioning from flat ground to stairs. The top row presents a third-person external view, while the bottom row shows the corresponding first-person view from the chest-mounted RGB camera. Red dots represent the sampled elevation map points used by the teacher policy. Notably, the camera's field of view captures only a subset of the elevation map (approximately the front 50 of total 187 points) according to Fig. 6.2; our results demonstrate that this forward-looking coverage is heavily partial yet sufficient for robust locomotion. 67

List of Tables

3.1 A comparison among popular geo-localization methods regarding a variety of attributes. Within each attribute, the methods which are regarded to produce results that are the most advantageous to aerial applications are highlighted in bold and colored red. The statistics are selected to demonstrate the nominal performance of each method by excluding the results from extreme cases presented in the corresponding papers. 16

3.2 A benchmark of the impact on RMSEs and percentage match from different numbers of initial candidates at the global descriptor matching stage. We preserve these candidates to enter the local descriptor matching stage. As the number of candidates decreases, the RMSEs in both 3D and 2D cases remain approximately unchanged but the percentage of successfully localized attempts among all query images (recall) declines depending on scenarios. The choice of the number of candidates is a trade-off between accuracy and the success rate of localization. 28

4.1 Detailed definitions of the components forming the observation space. The observation space is divided into 7 observation items whose corresponding symbols are listed side by side. The scale factors stand for the coefficients that are multiplied to the observation items as a preprocessing step. The upper bounds of the zero-mean uniform noise added to the observation are also listed below. 34

4.2 Detailed definitions of the proposed reward terms. Variables marked with an asterisk stand for target values. Δt is defined by the inverse of the control frequency, in this case, 0.005 s. Positive weights mean encouragement to such behaviors and negative ones penalty. In the hip (thigh) reward, H (T) is the set of 4 hip (thigh) motor indices. 36

5.1 Detailed descriptions of the observation terms, expressions, scale factors and noises added to the ground truth values. The observation terms are multiplied by corresponding scale factors as a normalization step before being input to the neural network. All intervals of the uniform noises are symmetric about zero. . 48

5.2 Initial joint position configurations (in rad) for training the baseline locomotion controller. The adducted hip joint configuration optimizes the trade-off between static stability and dynamic agility in quadrupedal locomotion. Values are specific to the Unitree Go1 robot platform. 52

5.3 Reward function terms, expressions and weights for training the baseline locomotion controller. The tracking hyperparameter $\sigma = 0.25$ modulates the exponential reward curvature to stabilize gradient magnitudes across tracking errors in a certain range. The simulation time step $\Delta t = 0.005$s corresponds to a 50Hz control frequency with a decimation of 4. For hip joint regularization terms, initial joint position configurations align with Table 5.2. 52
5.4 RMS errors from the velocity estimation and the command following tasks across selected combinations of the control system for quadrupedal locomotion are shown in the table. *RL-based* stands for the the proposed RL-based estimator concatenated with the baseline controller. *Concurrent* implies training the estimator using supervised learning and the controller using RL concurrently. *Teacher* and *student* represent the corresponding parts in the teacher-student architecture where the teacher is trained using RL with privileged observations and the student distills the teacher with practical observations. 54
5.5 Comparison between the RL-based controller-estimator pair and the concurrently trained pair along absolute steady-state error for velocity estimation and base linear velocity command tracking in x-direction. 55

6.1 Detailed descriptions of the observation terms, expressions and noises added to the ground truth values. All observation terms remain unscaled before being input to the neural network. All intervals of the uniform noises are symmetric about zero. The elevation map term has an original structure of $\boldsymbol{e}_{\mathrm{o}} \in \mathbb{R}^{17\times 11}$, covering a $1.6\,\mathrm{m} \times 1\,\mathrm{m}$ area around the robot. During the sampling process, the true projected position in the x-y plane of this area is offset by noise $\mathcal{U}^2(\pm 0.3)\,\mathrm{m}$ in both directions. In addition to the x-y plane projection offset, noise listed in the table is then added to the z-axis. Finally, the array $\boldsymbol{e}_{\mathrm{o}}$ is flattened to $\boldsymbol{e} \in \mathbb{R}^{187}$ and appended to the observation space. 62

Acknowledgments

This dissertation stands as a testament not just to my own efforts but to the mentorship, friendship and encouragement of the many wonderful people I have been fortunate enough to know.

First and foremost, I am deeply indebted to my advisor, Professor Koushil Sreenath. It has been a privilege to work under his guidance for the past seven years. Koushil, thank you for trusting me with such ambitious research problems alongside the absolutely coolest robots in the world. I will always appreciate the time you spent side-by-side with me troubleshooting experiments, as well as your constant support throughout my graduate studies. You have modeled the creativity and persistence that define a great researcher, and your mentorship has fundamentally shaped my approach to robotics.

I would also like to thank Professor Kameshwar Poolla and Professor Pieter Abbeel for serving on my committee. I have drawn much inspiration from your work and have benefited greatly from your insights over the years.

I extend my sincere gratitude to Professor Mark W. Mueller for his steadfast support on our drone projects during the pandemic. I learned a great deal from you during that challenging time. Thank you for generously providing access to your motion capture facilities and computational resources; without them, several of these projects would not have come to fruition. You widened my perspective on aerial systems and offered opportunities I would not otherwise have had.

I am also grateful to Professor Avideh Zakhor. Your excellent course introduced me to the field of computer vision and inspired me to further explore the vital intersection of perception and robotics.

To my collaborators, thank you for your immense efforts. To Bike, for the astronomical hours spent discussing various robotics problems and assisting with experiments. To Xiangyu, for your tireless help with coding and flight experiments at the Richmond Field Station. To Ting-Hao, for your assistance with coding, flight experiments and the motion capture facilities in your lab. To Jonathan, for the many days and nights we spent working on Cassie while you were in Berkeley.

I would like to thank my labmates: Ayush and Prasanth, for teaching me invaluable research skills; and Fernando, Jason, Jun, Qiayuan, and Yen-Jen for the fruitful discussions on a wide variety of topics. I also thank Akshay, Avinash, Emily, Katie, Mintae, Sangli, Xiaoyu and Zhongyu for your friendship and support throughout my graduate studies. Additionally, I am grateful to Chet, David, Gary, Ge, Hang, Hunter, Jamie, Jiaze, Johann, Johnathon, Julien, Lily, Long, Matthew, Mengyue, Ross, Scott, Sonny, Suiyi, Toby, Warren, Yuman, Zacks and Zhi for your assistance with my research.

Thank you to the cohort members and friends who supported my work: Carlo, Clark, Daisy, Dennis, Haoting, Jerry, Junseok, Karan, Kevin, Rui, Ruiqi, Teaya and Tony.

On a personal note, I must thank my roommates, Bike and Qilong, for tolerating me and being the best friends I could ever ask for. A special thanks to Bike for over four years of homemade dishes, fun concerts and discussions about life.

Most importantly, I dedicate this work to my parents. Thank you for your unwavering patience and unconditional support of my decisions. Your belief in me provided the strength I needed to persevere, and I could not have reached this milestone without you.

Finally, this work was supported by funding from the Army Research Laboratory under Cooperative Agreement Number W911NF-20-2-0105, the BAIR-Meta AI collaboration, InnoHK of the Government of the Hong Kong Special Administrative Region via the Hong Kong Centre for Logistics Robotics, the Graduate Division Block Grant, and the Chang-Lin Tien Chancellor Fellowship.

Chapter 1

Introduction

The unpredictability of unstructured environments poses a significant challenge for mobile robotics, demanding adaptable solutions for perception and control. Conventional paradigms often struggle in these scenarios, as their reliance on precise, high-cost hardware and stiff control structures makes them vulnerable to sensory noise and environmental constraints. To address these limitations, this dissertation explores a shift toward learning-based frameworks to enable robust, efficient, and self-reliant autonomy across aerial and legged platforms. By integrating learning-based perception and control, the proposed frameworks aim to support robot autonomy, addressing critical challenges ranging from aerial navigation in global positioning system-denied (GPS-denied) environments to dynamic legged locomotion on complex terrains.

1.1 Motivation

The deployment of autonomous robotic systems into unstructured, real-world environments presents a fundamental challenge to traditional engineering paradigms. While classical approaches have demonstrated efficacy in controlled settings, they frequently rely on precise, high-cost sensor suites and rigid control structures that degrade rapidly when subjected to the stochasticity and noise inherent in the wild. Consequently, the scalability of autonomous platforms is currently limited by frameworks that are brittle to environmental constraints and computationally inefficient. This dissertation is motivated by the necessity to advance robust autonomy through learning-based methodologies, addressing the interconnected challenges of self-reliant perception, compliant control, state estimation and efficient sensing across aerial and legged platforms.

A primary barrier to robust autonomy is the reliance on external infrastructure and restrictive sensing assumptions, particularly within aerial navigation. The dependence on GPS signals constitutes a critical vulnerability, as signal loss is often unpredictable in urban canyons or contested environments. Existing visual solutions often necessitate heavy off-board computation or enforce restrictive camera views, which precludes the simultaneous

execution of essential tasks such as obstacle avoidance. To achieve true self-reliance, there is a need for onboard, real-time frameworks capable of recovering a vehicle's full pose using only readily available data, thereby decoupling the vehicle from ground-based resources.

Complementing the need for robust localization is the requirement for compliant and dynamic interaction with the physical world. In the domain of legged locomotion, prevailing learning-based paradigms often utilize position-based control, where policies output target joint angles tracked by stiff and low-level controllers. This inherent stiffness limits a robot's ability to negotiate compliant terrains or absorb significant external disturbances. Advancing the status quo requires investigating end-to-end learning frameworks where policies directly modulate torque, thus combining the adaptability of data-driven methods with the physical compliance necessary for robust locomotion.

The efficacy of these control policies is fundamentally contingent upon accurate and robust state feedback. However, current learning-based architectures often create a tight coupling between state estimation and control, creating systems where modifications to one module necessitate the retraining of the other. This entanglement hinders system modularity and often obscures performance bottlenecks. To streamline the development of robust autonomous systems, a modular framework is required that decouples these components, allowing for the independent optimization of estimators that maintain high tracking performance without interfering with controller design.

Finally, the reliance on complex, high-fidelity perception hardware imposes significant constraints on the mechanical simplicity and accessibility of robotic platforms. Perceptive locomotion typically depends on dense elevation maps constructed from expensive sensors, which introduce substantial computational overhead and latency. To popularize capabilities such as obstacle traversal, it is essential to eliminate these hardware dependencies. By distilling geometric reasoning into image-based policies capable of mapping monocular visual inputs directly to control actions, robotic systems can achieve perceptive autonomy with significantly reduced hardware complexity. Addressing these challenges collectively moves the field toward systems that are not only theoretically capable but robustly deployable in the real world.

1.2 Contributions

This dissertation develops learning-based frameworks to enhance the robustness and autonomy of mobile robots. By addressing specific limitations in perception and control, we present four primary contributions that support robust real-world deployment across aerial and legged platforms.

- **Real-time Visual Geo-localization:** We introduce a real-time, image-based geo-localization framework designed for GPS-denied environments. Unlike traditional methods that require nadir camera views or heavy off-board computation, our two-stage pipeline utilizes a database of synthetic views rendered from satellite imagery

and topography to perform onboard inference. The system achieves a position root-mean-square error (RMSE) of approximately 2.8 m at an inference rate of 1.1 Hz. Crucially, it demonstrates robustness to significant variations in camera pitch, lens flare and motion blur, enabling flexible camera configurations for simultaneous tasks like obstacle avoidance.

- **Learning Torque Control for Quadrupedal Locomotion:** We propose a learning torque control framework that circumvents the limitations of stiffness inherent in conventional position-based controllers. By utilizing RL to output joint torques directly at high frequency, this approach bypasses the traditional low-level PD control loop. Validated on a quadrupedal robot, the framework demonstrates superior tracking accuracy and dynamic compliance compared to position-based baselines. This compliance enables the robot to recover from significant external disturbances and successfully traverse compliant terrains, such as air mattresses and loose pebbles.

- **Modular State Estimation:** We present a modular framework for legged state estimation that decouples estimator training from controller design. In contrast to concurrent training or teacher-student distillation which can hinder modularity and obscure performance bottlenecks, we optimize an RL-based estimator against an existing controller within a unified environment. This methodology ensures that the estimator inherits the sim-to-real transfer characteristics of the controller while achieving superior base linear velocity tracking compared to concurrent baselines. It allows for independent design and optimization of the estimator without necessitating the retraining of the control policy.

- **Cross-Modality Distillation for Perceptive Locomotion:** We propose a cross-modality distillation pipeline to achieve robust perceptive locomotion without relying on expensive depth sensors. We distill the geometric reasoning capabilities of an elevation map-based teacher policy which uses elevation maps into a student policy that relies solely on a monocular RGB camera. Utilizing a diffusion transformer architecture, the resulting image-based policy enables a humanoid robot to traverse complex terrain, such as stairs, matching the performance of the teacher policy in simulation. This framework eliminates the need for explicit depth perception hardware and map fusion, significantly reducing hardware complexity and cost.

1.3 Organization

The remainder of this dissertation is organized as follows:

- Chapter 2 provides a comprehensive literature survey of the research fields relevant to this work. It reviews existing methodologies in visual geo-localization within GPS-denied environments, control paradigms for legged locomotion, strategies for state estimation, and sensor modalities used in perceptive locomotion.

- Chapter 3 introduces a real-time, image-based geo-localization framework for aerial platforms. It details the methodology for offline database preparation using satellite imagery and topography, as well as the online inference pipeline for recovering the 6-DOF vehicle pose. The chapter concludes with experimental validation on multiple UAV platforms.

- Chapter 4 presents an end-to-end learning torque control framework for quadrupedal robots. This chapter describes the shift from position-based to torque-based reinforcement learning, outlining the policy architecture and reward formulation. It demonstrates the framework's effectiveness through real-world experiments on a quadrupedal robot traversing various terrains and resisting external disturbances.

- Chapter 5 proposes a modular framework for legged state estimation via RL. It explains the methodology for decoupling estimator training from controller design by treating the controller and simulator as a unified environment. The chapter presents simulation and real-world results that validate the system's tracking accuracy and sim-to-real robustness.

- Chapter 6 details a cross-modality distillation pipeline for achieving perceptive locomotion using a single monocular RGB camera. It describes the use of a diffusion transformer architecture to distill the geometric reasoning of an elevation map-based teacher into a image-based student policy. The chapter demonstrates the policy's ability to enable a humanoid robot to traverse complex terrain, such as stairs, without explicit depth sensors.

- Chapter 7 concludes the dissertation by summarizing the primary contributions and discussing potential directions for future work.

Chapter 2

Literature Survey

This chapter establishes the theoretical foundations and historical context necessary to situate the contributions of this dissertation within the broader field of robotics. As autonomous systems transition from controlled laboratory settings to unstructured real-world environments, the limitations of traditional, rigid pipelines for perception and control become increasingly apparent. To motivate the shift toward learning-based frameworks proposed in this work, we examine the state of the art across four distinct yet interconnected domains: aerial geo-localization, legged control paradigms, state estimation and perceptive locomotion. By critically analyzing existing methodologies, we identify key gaps, specifically regarding sensor dependency, computational constraints and adaptability that this dissertation aims to address.

2.1 Overview

The pursuit of robust autonomy for mobile robots involves a complex interplay between perception, state estimation and control. Historically, these sub-fields have been treated as distinct modules within a rigid pipeline: perception systems reconstruct the environment, state estimators filter sensor noise to provide pose and velocity, and controllers track trajectories based on these estimates. While this modular decomposition has enabled significant capabilities in structured environments, it often exposes fragility in the wild where global positioning system (GPS) signals are denied, terrain is compliant or slippery, and expensive sensor suites are impractical. Consequently, the literature has seen a paradigm shift toward learning-based frameworks that offer greater robustness and adaptability.

This chapter surveys the landscape of research relevant to bridging the gap between theoretical autonomy and real-world deployment across aerial and legged platforms. We structure the literature review into four primary domains, corresponding to the technical challenges addressed in this dissertation: visual geo-localization, legged control paradigms, modular state estimation and perceptive locomotion.

In Section 2.2, we examine the evolution of visual geo-localization for aerial platforms.

We review the progression from traditional template matching and horizon detection toward modern deep learning-based feature extraction. Particular attention is given to the trade-offs between onboard computational efficiency and global localization accuracy, as well as the limitations of existing methods that rely on strictly nadir camera views, which constrain multi-task capabilities such as obstacle avoidance.

Section 2.3 provides an emphasized survey of control strategies for quadrupedal locomotion. We contrast the historical dominance of model predictive control (MPC) which inherently leverages torque control for dynamic compliance with the rise of deep reinforcement learning (RL). We identify a critical divergence in the literature: while deep RL has achieved state-of-the-art results in traversing rough terrain, it has predominantly relied on position-based control interfaces. This section highlights the gap in learning-based torque control, setting the stage for frameworks that seek to combine the adaptability of RL with the compliant nature of direct torque modulation.

In Section 2.4, we analyze strategies mainly for legged state estimation, a prerequisite for stable locomotion. We discuss the limitations of kinematic-inertial fusion on slippery surfaces and the recent trend toward learning-based estimators. This section critically reviews the teacher-student architecture and concurrent training methods, identifying how tight coupling between estimation and control can hinder modularity and sim-to-real transfer. We explore the literature on decoupling these systems to allow for independent optimization and improved robustness.

Finally, Section 2.5 surveys sensor modalities and architectures for perceptive locomotion. We review the standard reliance on dense elevation maps constructed via lidar or depth cameras and the associated computational overhead. The survey covers recent advances in pixels-to-torques approaches and cross-modality distillation, framing the motivation for vision-centric policies that reduce hardware complexity while maintaining the ability to negotiate complex geometric features like stairs.

Collectively, this survey contextualizes the dissertation's contributions by highlighting the specific limitations of existing approaches, namely, reliance on external infrastructure, rigid control interfaces, coupled estimation architectures, and expensive sensing modalities that the proposed frameworks aim to overcome.

2.2 Visual Geo-localization in GPS-denied Environments

As aerial platforms are increasingly tasked with operating in urban canyons, jammed environments, and remote areas, reliance on GPS signals has become a critical vulnerability. To mitigate the risk of signal loss, the research community has turned to vision-based geo-localization using onboard cameras to match live aerial feeds against geo-referenced satellite imagery.

This domain has evolved significantly, transitioning from pixel-level template matching

to sophisticated deep learning pipelines. This section reviews this evolution, categorizing approaches into classical alignment methods, hand-crafted feature matching, and modern data-driven retrieval, while highlighting the persistent challenges of onboard computational constraints and camera view restrictions.

2.2.1 Direct Alignment and Feature Matching Methods

Early research in aerial global localization largely relied on direct alignment techniques. These approaches typically binarize query images and align them directly over a database of satellite maps [21]. While effective in high-contrast environments or at high altitudes where terrain features are distinct, these methods struggle with the complex visual discrepancies caused by differing lighting conditions, seasonal changes, and shadows. To address this, metrics such as mutual information (MI) were introduced to align images by minimizing statistical differences rather than relying on exact pixel matches [102]. While MI-based methods [69, 36] offer high accuracy and tolerance to visual disparities, they suffer from a restricted convergence basin, often necessitating a precise initial pose guess to prevent optimization failure.

Parallel to direct alignment, feature matching methods emerged with the advent of robust keypoint detectors. Algorithms utilizing scale-invariant feature transform (SIFT) [62] or speeded up robust features (SURF) [7] attempt to match specific points of interest between the aerial view and the satellite map [97, 95]. These handcrafted descriptors provide a more compact representation than whole-image alignment. However, they remain computationally expensive for real-time applications and are notoriously brittle when facing the extreme viewpoint changes inherent to aerial platforms, e.g., the difference between a slanted drone view and a nadir satellite image.

2.2.2 Deep Learning and Hierarchical Geo-localization

The limitations of handcrafted features have driven a shift toward deep learning-based approaches that semantically summarize images to tolerate large visual deformations. This is often formulated as a retrieval task: a neural network encodes the query image into a global descriptor, which is then used to retrieve the most similar map tiles from a database.

Architectures such as NetVLAD [6] have become standard for aggregating local features into robust global descriptors. In the context of aerial localization, cross-view matching networks like CVM-Net [42] and subsequent hierarchical approaches [81, 92] utilize these global descriptors to perform a coarse search, followed by fine-grained local matching. Recent advances have replaced handcrafted local features with learned descriptors such as SuperPoint [23], which are matched via graph neural networks like SuperGlue [83]. These learned pipelines demonstrate superior robustness to environmental changes compared to their classical counterparts.

2.2.3 Gap of Onboard Computing

Despite these algorithmic advances, a significant gap remains between accuracy and operational utility. A survey of the landscape (see Table 3.1 in Chapter 3) reveals that high-accuracy methods often rely on heavy, off-board computation stations, rendering them unsuitable for autonomous, long-range missions where communication with a ground station cannot be guaranteed.

Furthermore, many existing algorithms impose restrictive constraints on the vehicle's configuration. A common assumption is the flat-world model, which reduces the problem to 3-DOF (latitude, longitude and heading angle) by assuming a strictly nadir camera [36, 69]. This constraint is problematic for practical autonomy, as UAVs often require forward-facing cameras for simultaneous obstacle avoidance and path planning. Consequently, there is a pressing need for 6-DOF localization frameworks that can operate in real-time on embedded hardware, utilizing readily available satellite imagery without requiring specialized 3-dimensional (3D) city models.

While robust localization provides where to execute autonomy, the ability to execute dynamic maneuvers based on this state, presents its own set of challenges. This is particularly true when transitioning from stable aerial platforms to the hybrid dynamics of legged robots. The following section examines the control paradigms that govern these complex systems, contrasting traditional model-based approaches with emerging learning-based strategies.

2.3 Control Paradigms for Legged Locomotion

The control of legged robots presents a unique challenge due to their high dimensionality, underactuation, and hybrid dynamics characterized by intermittent ground contacts. Over the past few decades, strategies to address these challenges have broadly bifurcated into two dominant paradigms: model-based optimal control and learning-based control.

2.3.1 Model-Based Control and the Shift to Torque

Early legged locomotion relied heavily on heuristic control structures utilizing simplified dynamics models. Raibert's influential work [76] decomposed hopping into three separate controllers-hopping height, forward velocity, and posture, demonstrating dynamic balancing on hydraulic machines. As computing power increased, these heuristics evolved into optimization-based frameworks. MPC and whole-body control became the gold standards for dynamic locomotion, allowing robots to plan ground reaction forces and trajectories over a finite horizon while respecting physical constraints [24, 52, 31].

A critical evolution within model-based control was the transition from high-impedance position control to low-impedance torque control. While position control which tracks joint angles offers simplicity, it renders the robot stiff and unable to absorb impacts. In contrast, torque control allows for compliance, enabling the robot to act like a spring-mass system that can naturally adapt to terrain irregularities without explicit replanning. This shift

was instrumental in achieving highly dynamic maneuvers, such as the blind jumping and bounding demonstrated by the MIT Cheetah [24].

2.3.2 Deep Reinforcement Learning

More recently, deep RL has emerged as a powerful alternative, capable of synthesizing robust controllers without explicit dynamic modeling. By interacting with a physics simulator, RL agents learn policies that can generalize to unstructured environments that are difficult to model analytically. Landmark works in this domain have demonstrated policies capable of recovering from falls [44], traversing challenging terrains [57, 65] and transferring directly from simulation to the real world [91].

2.3.3 Gap of Action Space

Despite the success of deep RL, there remains a notable divergence in the choice of action space compared to model-based control. The vast majority of real-world RL implementations utilize position-based control, where the policy outputs target joint angles that are tracked by a high-frequency proportional-derivative (PD) controller [91, 57].

The preference for position-based RL stems from its training stability; the PD loop acts as a low-pass filter, smoothing out noisy policy actions and providing a reference posture that aids exploration. However, this approach inherits the limitations of high stiffness. The tuned PD gains often limit the robot's ability to interact dynamically with compliant surfaces (e.g., mattresses, loose pebbles) or absorb sudden external disturbances. While simulation studies have suggested that learning direct torque control can theoretically yield higher performance and agility [71], successful sim-to-real transfer of end-to-end torque policies has been scarce due to the reality gap in actuator dynamics and the lack of stabilizing PD feedback.

This dichotomy defines a significant gap in the literature: while model-based control has proven the superiority of torque modulation for dynamic feasibility, learning-based approaches have largely not made this shift in real-world deployments. Bridging this gap requires investigating frameworks that allow RL agents to output torques directly, thereby combining the adaptability of learning with the compliant dynamics of torque control.

Regardless of whether a controller is position-based or torque-based, its performance is fundamentally affected by the quality of its feedback. Precise knowledge of the robot's base velocity and orientation is a prerequisite for stability. The following section reviews the evolution of state estimation strategies, examining how the field is moving away from purely kinematic-inertial fusion toward modular, learning-based estimators.

2.4 State Estimation Strategies

For a floating-base system like a legged robot, accurate knowledge of the body's state, particularly its linear velocity is fundamental to feedback control. Unlike industrial manipulators

rigidly mounted to the floor, legged robots must infer their global motion through the integration of noisy sensors and intermittent ground contacts. This section reviews the progression of state estimation from kinematic-inertial fusion to modern data-driven approaches.

2.4.1 Kinematic-Inertial Fusion

The conventional approach to legged state estimation relies on fusing high-frequency inertial measurement unit (IMU) data with leg kinematics. Extended Kalman filters and unscented Kalman filters are the standard workhorses here [11, 45]. These filters typically predict the state using IMU acceleration integration and correct it using leg odometry, assuming that the feet in contact with the ground are stationary.

While robust on high-friction surfaces, these kinematic-based methods degrade significantly on slippery or deformable terrain. The core assumption that the foot is a stationary pivot point is violated during slippage, leading to drift in the estimated velocity [53]. Furthermore, accurately detecting contact events is itself a non-trivial challenge, often requiring threshold tuning for force sensors or impact detection algorithms that can be brittle in the wild [15].

2.4.2 Learning-based Estimation

To overcome the limitations of explicit modeling assumptions, the field has increasingly adopted learning-based estimators. These methods train neural networks to infer base velocity directly from proprioceptive history (joint encoders and IMU), implicitly learning to identify slip and soft contacts.

A dominant paradigm in this domain is the teacher-student architecture [57, 16]. In this framework, a teacher policy is trained with access to privileged information (e.g., ground truth velocity, friction coefficients and terrain height) available only in simulation. A student policy, which has access only to noisy proprioceptive sensors, is then trained to mimic the teacher's actions. Through this distillation process, the student implicitly learns a state estimator within its neural network layers to replicate the teacher's performance.

Another prevalent approach is concurrent training, where an explicit estimator network and a control policy are trained simultaneously [47]. In this setup, the estimator is supervised by ground truth simulation data, while the controller uses the estimator's output to maximize task reward.

2.4.3 The Modularity Bottleneck

While these coupled architectures yield high performance, they introduce significant engineering drawbacks regarding modularity. In both teacher-student and concurrent setups, the estimator and controller are tightly entangled. Modifying the control objective requires retraining the estimator, and updating the estimator requires retraining the controller. This

lack of separation complicates the design cycle and obscures the source of failure when sim-to-real transfer degrades.

This coupling creates a need for modular frameworks where estimation and control are decoupled. By treating an existing controller as part of the environment, an estimator can be optimized independently via Reinforcement Learning to provide the specific signals that maximize the controller's robustness, rather than merely minimizing a supervised reconstruction loss. Such an approach, which this dissertation explores, promises to streamline the development pipeline and improve the interpretability of learned behaviors.

While proprioceptive estimation allows a robot to stabilize itself, true autonomy requires the ability to anticipate and react to the environment before contact is made. The final section of this survey examines the transition from blind locomotion to perceptive systems that utilize exteroceptive sensing to traverse complex geometry.

2.5 Perceptive Locomotion and Sensor Modalities

While blind locomotion policies have achieved remarkable robustness on irregular terrain through proprioceptive feedback, they are fundamentally limited by their reactive nature. To traverse discrete terrains such as stairs, gaps or high ledges, a robot must perceive the geometry of the terrain and adapt its gait proactively. This capability, known as perceptive locomotion, relies heavily on the choice of exteroceptive sensors and the representation of the environment.

2.5.1 Elevation Maps and Dense Depth

The standard paradigm for perceptive locomotion involves constructing a dense 2.5D elevation map of the terrain. In this pipeline, data from lidars or depth cameras is fused over time into a grid-centric representation, where each cell contains a height value and an associated uncertainty variance [27].

Controllers utilizing these maps, whether model-based [32] or learning-based [65], sample height values around the robot's feet to make decisions about step height and location. While effective, this approach imposes a heavy burden on the system. High-resolution lidars are expensive, mechanically fragile, and power-hungry. Furthermore, the map fusion process requires precise state estimation to prevent smearing or drift; even small calibration errors can render the map inconsistent, leading to missed steps or collisions. The computational latency required to process point clouds and update the grid also limits the speed at which the robot can safely traverse dynamic environments.

2.5.2 Vision-based Policies and Cross-modality Distillation

To address the hardware and computational costs of dense mapping, recent research has explored learning policies directly from raw visual inputs, such as monocular RGB (red,

green and blue) images [2]. Unlike depth sensors, RGB cameras are ubiquitous, lightweight and mechanically simple. However, mapping high-dimensional pixel data directly to control actions, pixels-to-torques, is notoriously sample-inefficient and difficult to train end-to-end.

To bridge this gap, cross-modality distillation has emerged as a promising solution. In this framework, a teacher policy is first trained using high-quality geometric data, e.g., elevation maps. A student policy, constrained to use only the target sensor modality, e.g., a single RGB camera, is then trained to mimic the teacher's behavior [16, 61].

Recent advances in generative modeling, particularly diffusion transformers, offer new capabilities for this distillation process. By treating the policy as a conditional generation problem, these models can capture multi-modal action distributions and fuse heterogeneous inputs (vision and proprioception) more effectively than standard regression networks [19]. This dissertation leverages these advancements to demonstrate that complex behaviors, such as stair climbing, can be distilled from a map-based teacher into a streamlined, vision-only student, thereby democratizing perceptive autonomy.

2.6 Summary

This chapter has surveyed the critical technologies underpinning robot autonomy, highlighting a consistent trend: the limitations of rigid, sensor-heavy, and analytically modeled pipelines are driving a shift toward learning-based solutions.

- In **aerial geo-localization**, we identified the need for onboard, 6-DOF estimation that frees UAVs from the constraints of nadir-view cameras and external ground stations.
- In **legged control**, we contrasted the dynamic compliance of torque control with the stiffness of prevalent position-based RL, motivating the search for end-to-end learning torque control.
- In **state estimation**, we critiqued the lack of modularity in current teacher-student architectures, proposing a decoupled approach that preserves the independence of estimation and control.
- In **perceptive locomotion**, we highlighted the hardware and computational costs of elevation mapping, setting the stage for vision-based distillation that reduces system complexity.

The following chapters address these gaps individually, beginning with the challenge of robust estimation in the sky. Chapter 3 introduces a real-time framework for aerial geo-localization in GPS-denied environments.

Chapter 3

Real-time Geo-localization Using Satellite Imagery

The capabilities of autonomous flight with unmanned aerial vehicles (UAVs) have significantly increased in recent times. However, basic problems such as fast and robust geo-localization in GPS-denied environments still remain unsolved. Existing research has primarily concentrated on improving the accuracy of localization at the cost of long and varying computation time in various situations, which often necessitates the use of powerful ground station machines. In order to make image-based geo-localization online and pragmatic for lightweight embedded systems on UAVs, we propose a framework that is reliable in changing scenes, flexible about computing resource allocation and adaptable to common camera placements. The framework is comprised of two stages: offline database preparation and online inference. At the first stage, color images and depth maps are rendered as seen from potential vehicle poses quantized over the satellite and topography maps of anticipated flying areas. A database is then populated with the global and local descriptors of the rendered images. At the second stage, for each captured real-world query image, top global matches are retrieved from the database and the vehicle pose is further refined via local descriptor matching. We present field experiments of image-based localization on two different UAV platforms to validate our results.

3.1 Introduction

Geo-localization for UAVs operating in GPS-denied environments has been an active research area due to its critical need for practical applications. This is because GPS signal loss is inevitable due to various unpredictable factors in real-world flight missions. Image-based geo-localization methods are intuitive and also biologically inspired. For an agent that is

The material in this chapter is based on the following previously published work: Shuxiao Chen et al. "Real-time geo-localization using satellite imagery and topography for unmanned aerial vehicles". In: *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2021, pp. 2275–2281.

isolated from global localization sources, visual information is a reliable observation that the agent can actively collect for localization. As a growing number of satellite imagery sources are made public, they have unleashed the potential of image-based methods. Particularly, the more accessible topographic information of terrains and buildings offers a unique advantage.

An ideal set of capabilities for an image-based geo-localizer may contain the following aspects. Firstly, while multiple sensor inputs can be fused for localization, it should be able to localize purely with images. This requirement enhances the safety of the agent in the event of sensor failures. Next, it is desirable for the geo-localizer to use query images which are not captured strictly perpendicular or nadir to the ground, because the camera may not only be utilized for geo-localization but also have other purposes requiring capturing a forward view, e.g., obstacle avoidance at low elevations. This demand necessitates the localization outputs in more degrees-of-freedom (DOFs) as opposed to solely latitude and longitude. Furthermore, the geo-localizer should be ideally capable of being bootstrapped. In situations where no initial positions are available for the agent or long periods of GPS interruption is present, the geo-localizer needs to determine the position without prior knowledge. Ultimately, it is important to have the ability to infer location in real-time on low-power and low-compute UAV platforms.

This chapter is structured as follows. It begins with background knowledge covering image-based geo-localization, feature descriptors and camera pose estimation respectively. After that, we introduce our proposed method in Section 3.2 which is divided into the database preparation and onboard inference parts. The experimental setups and results are presented in Section 3.3 along with the detailed procedure and analyses. Finally, we conclude with future work in Section 3.4.

3.1.1 Image-based Geo-localization

In the early years of research into global localization of aerial vehicles using satellite maps, approaches largely concentrated on binarizing the query images and directly aligning them over a geo-referenced map database [21]. Key assumptions were made in early work to simplify the problem, such as the low complexity in the scenes and high elevations at which the experiments were operated. After the emergence of key point detectors and feature descriptors which offer more compact representations of the useful information contained in an image, these were then employed in the geo-localization field as a new genre, named the feature matching method, e.g., the robust scale-invariant feature transform (SIFT) detector and descriptor [62]. More detailed and advanced feature matching methods include matching the query images over an entire target map [97] and individual mosaics of the map [95] have been developed.

However, the handcrafted descriptors are computationally expensive and vulnerable to scene changes incurred by different seasons and lighting conditions. The first genre, the direct alignment method, again evolved to have the edge over the feature matching method in accuracy and scene tolerance. Mutual Information (MI) and optical flow are two typical modern direct alignment methods. They align two images in an incremental manner that

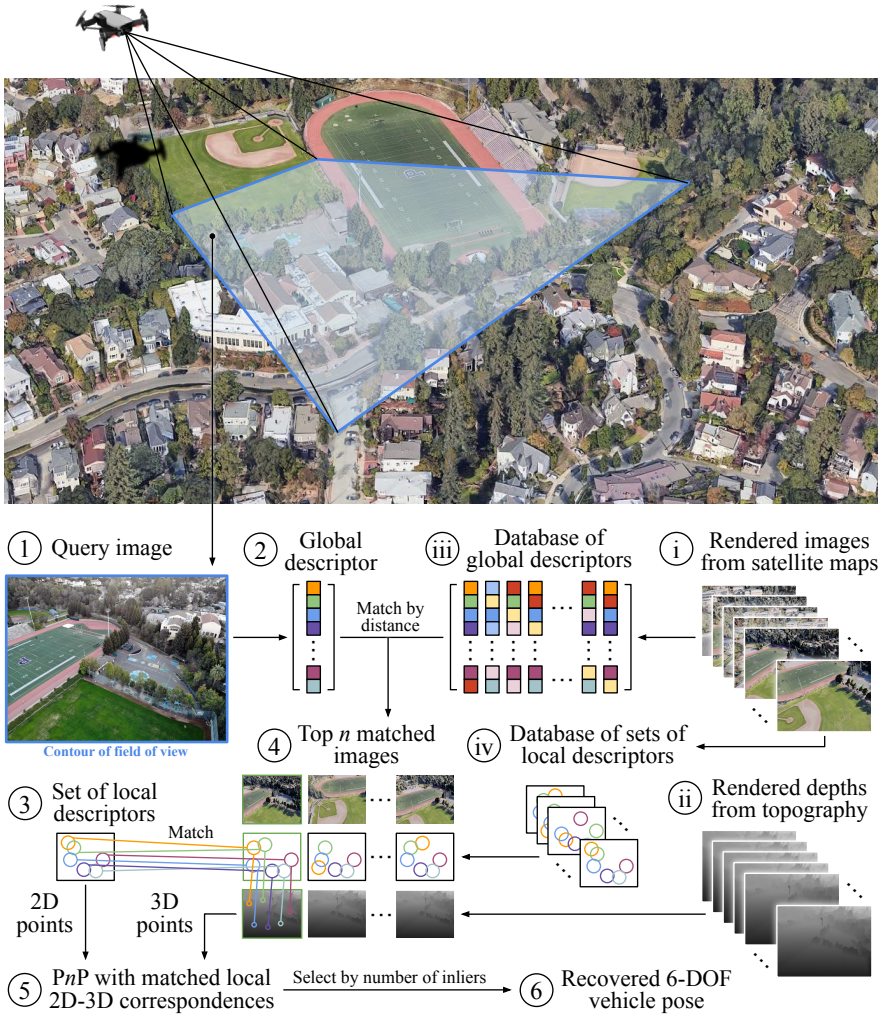


Figure 3.1: The proposed geo-localization pipeline. The steps numbered in Roman numerals (i-iv) introduce the procedure of the offline database preparation (Section 3.2.1) of the descriptor database and those in Arabic numerals (1-6) define the onboard inference (Section 3.2.2) based localization strategy. The pose and field of view of the quadrotor along with a captured query image from one of our experiments are shown.

Table 3.1: A comparison among popular geo-localization methods regarding a variety of attributes. Within each attribute, the methods which are regarded to produce results that are the most advantageous to aerial applications are highlighted in bold and colored red. The statistics are selected to demonstrate the nominal performance of each method by excluding the results from extreme cases presented in the corresponding papers.

Attribute	**Direct alignment**				**Feature matching**			
	NID [69]	MI [102]	ICLK with depth [36]	ICLK with homography [30]	Siamese NN [**shetty2019uav**]	SIFT [97]	Autoencoder [**bianchi2021uav**]	Ours
Stand-alone[a]	No, needs VO	No, needs IMUs	**Yes**	**Yes**	No, needs VO	**Yes**	**Yes**	**Yes**
Nadir camera	Yes	Yes	**No**	Yes	**No**	**No**	Yes	**No**
Min. working height	36-48 m	125-175 m	-	200-220 m	100-200 m	-	40 m	**18** m
Desired path	Yes	**No**	**No**	**No**	**No**	**No**	Yes	**No**
Initial position	Yes	Yes	Yes	Yes	**No**	**No**	Yes	**No**
Degrees of freedom[b]	4	4	**6**	**6**	**6**	**6**	3	**6**
onboard[c]	No	No	No	No	No	No	No	**Yes**
Real-time[d]	No	No	**Yes**	No	**Yes**	No	**Yes**	**Yes**
Computing platform	-	Xeon 2.8 GHz	GeForce RTX 2080 Ti	**Core i5 2.9 GHz**	GeForce GTX 1050	-	Core i7, Quadro P2000	**Jetson AGX Xavier**
Inference frequency	< 0.2 Hz	-	4.83 Hz	0.12 Hz	2.22 Hz	-	**9.09** Hz	1.12 Hz
Translational error	$<$ **3** m **(3D, RMSE)**	12.76 m (3D, RMSE)	**4.37** m **(3D, MAE)**	10.68 m (3D, MAE)	36.0 m (3D, RMSE)	-	**1.81** m **(2D, RMSE)**	**2.82** m **(3D, RMSE)**
Image dimensions[e]	560×315	**650×500**	**752×480**	200×200	-	-	320×160	**640×480**

[a]Requires no additional sensors or estimate sources to assist the localization. [b]The number of DOFs of the resulting poses directly from the geo-localizer. [e]The dimensions are defined as width $\times$ height.
[c]Both data collection and pose estimation are finished onboard on the vehicle. [d]The frequency of localization should not be too slow compared to the cruising speed and here 1 Hz is regarded as the boundary.

does not require explicit presence of specific and similar key points in both images. Relative shift and rotation between images are parameterized to be the decision variables and difference metrics such as MI are minimized. Recent research shows that the method is capable of achieving superb localization accuracy whilst allowing inevitable discrepancies between query and database images, although approximate pose guesses are essential to initializing the optimization [69, 36, 102, 30].

To eliminate the need for initial guesses, cross-view image-based hierarchical localization algorithms such as HF-Net [81, 82], or even those focusing on matching ground views to aerial imagery [41, 42, 92], are inspiring for aerial geo-localization applications. The general philosophy that these algorithms follow is to invoke the power of neural networks to semantically summarize the query and database images, either globally or locally, in order to tolerate the large visual difference caused by cross-view deformation and scene changes. Then, they formulate the localization problem as a retrieval task over a database consisting of mosaics of the collected maps. However, these databases are usually formed with real-world images which are not always available in aerial cases.

3.1.2 Image and Feature Descriptors

Feature descriptors play an important role in image retrieval problems. From the coverage perspective, descriptors are mainly categorized into global and local ones. Global descriptors, also known as whole-image descriptors, condense and encode the entire image to a lower dimensional space where similarity measures, or distance metrics, between images are easier to apply. Local descriptors aim at specific key points on the image and encode the content around the key points. Handcrafted descriptors, e.g., SIFT and Speeded Up Robust Features (SURF) [7], etc., have been used as both whole-image [95] and local descriptors [97] for localization by adjusting their effective radii.

Because of the limited complexity and handcrafted nature of these descriptors, they were then replaced by NN-based methods such as NetVLAD [6] that aggregates local descriptors, which can be SIFT, SURF or even NN-based ones such as SuperPoint [23], to encode the whole image.

3.1.3 Pose Estimation

Localizing an agent in three-dimensional (3D) space with 6 DOFs is widely adopted in the simultaneous localization and mapping (SLAM) community because of the short relative distance between the agent and environment. Nevertheless, by imposing a flat-ground assumption due to imaging from higher elevations, most of the aforementioned geo-localization algorithms can then target more concise outputs with fewer DOFs by excluding the altitude, pitch and roll of the vehicle. This flat-ground assumption also validates the use of 2D satellite maps which are more easily available. On the other hand, geo-localization algorithms which do not require a rigorously nadir camera usually output full 6-DOF poses by decomposing the calculated planar homography to a set of rotations and translations [30].

With more frequently updated modern 3D map sources such as Google Earth, it has become possible to reduce the minimum working height and to compute and output poses of more DOFs by replacing the flat-ground assumption with richer depth information, thus allowing the geo-localization to be formulated as a PnP problem.

3.1.4 Contributions

In order to meet the above requirements and address existing issues, we propose an image-based geo-localization pipeline shown in Figure 3.1. This pipeline consists of two phases. During the first offline database preparation phase, potential vehicle poses are sampled at a nominal flight attitude within a specified prospective flying area. Next, color images and depth maps are rendered as seen from the above sampled vehicle poses. Eventually, a database containing the global or whole-image descriptors and local descriptors of the rendered images, along with the depth maps, is constructed and loaded on the vehicle. The second phase is to perform online localization onboard the vehicle. This is achieved as follows. For each incoming query image collected by the UAV, the image is firstly encoded as a global descriptor and a set of local descriptors using neural networks (NNs). The encoded global descriptor is then matched over the database to find the most similar candidates. After that, the correspondences between the query and candidate sets of local descriptors are matched using NNs again. The absolute global coordinates of the matched correspondences are retrieved from the rendered depth maps. Ultimately, by combining the global coordinates and their projections on the query images, a Perspective-n-Point (PnP) problem is formulated and solved.

We next summarize our contributions below.

- We present a pragmatic image-based geo-localization package that executes on embedded systems at a high frequency.
- Our approach has the ability to adapt to downward-tilted camera configurations that are not rigorously nadir, therefore exploiting the existing setups.
- Our approach does not require a desired flight path for preparing the map database, thus enabling more flexible maneuvers.
- Our approach does not require prior knowledge of the initial positions for online localization, making it easier to choose the take-off location for the UAV.

3.2 Methodology

In this section, we present our methodology that is primarily divided into two parts: (1) the offline *database preparation* given a potential flight venue and (2) the *onboard inference* pipeline for real-time localization. Figure 3.1 gives a full visual overview of the two parts.

3.2.1 Database Preparation

The essence of image-based localization is the retrieval of recorded information at an acceptable speed and accuracy. Instead of storing the entire satellite map as a whole, cropped mosaics of the map are generated (see Section 3.2.1 on image rendering) and the mosaics are encoded into a lower dimensional space (see Section 3.2.1 on image encoding).

Image Rendering

Google Earth offers high-quality Earth maps in the sense of high-resolution satellite imagery, fine-grained 3D models of ground buildings and comprehensive topography information. We begin by retrieving these three parts from Google Earth via an open-source map builder [4]. Next, we use Blender to stitch the downloaded colored meshes into a single piece according to their coordinates and rotate to fit the built-in x-y plane. To achieve the coordinate switch, we override the latitude-longitude geographic coordinate system with the relative metric coordinate system. We call this the Blender coordinate system and define the origin arbitrarily to the stitched mesh. The geographic coordinate of the Blender origin is recorded. By assuming a negligible variation in the earth radius over the target terrain, the great-circle distance between a selected point and the Blender origin is regarded as the real-world distance. Conversely, to convert the distance back to the geographic coordinate, the Blender origin is used as the reference and the differences in distance are added or subtracted from it.

Next, we compute potential camera poses in the Blender coordinates in order to render images from the stitched map to generate mosaics. The ranges and spacings of the camera poses in x-direction, y-direction, elevation, heading and pitch angle are determined empirically in compliance with the area of the flight venue, the camera setup and the vehicle's onboard storage capacity. An effectual way of deciding the parameters for creating our database is detailed next.

(i) We begin with setting the simulated elevation to the nominal flight height of the mission. A camera with a vertical field of view (FOV) of around 68° flying at an elevation of about 70 m can give ±29% tolerance to the real vehicle height, i.e. with a single layer of potential camera poses simulated at an elevation of 70 m, the real-world localization can then be achieved between around 50-90 m. Additional layers of different elevations can be added to create hierarchical maps to increase the vertical working range of the vehicle.

(ii) The spacings of the camera poses in the x- and y-directions are identical. In practice, this spacing is determined using the following two steps. Firstly, we obtain the rectangle projected by the camera when it is pointed strictly downward to a level ground. Secondly, a quarter of the shorter side of the rectangle is adopted as the spacing, e.g., for the aforementioned camera and nominal elevation, the spacing is 10 m.

(iii) A spacing of 30° in the heading is used for a camera with a horizontal FOV of 84°. It is recommended to have a resulting overlap of 50% or above of the camera's horizontal FOV.

(iv) The simulated pitch angle of the camera is aligned to the installation angle of the real camera. Images can be rendered with additional pitch angles for extra robustness of localization.

Finally, the simulated images and corresponding depth maps are rendered at the spaced camera poses, as shown from Steps (i-ii) in Figure 3.1. To maximize the amount of useful information contained in the 3D map, diffuse reflection is desired because any artificial shadows may cause unwanted visual discrepancies in the images. Hence, in Blender, the specular attribute in the material settings is tuned to the minimum possible value so that bright highlights on glossy surfaces are suppressed and the resulting reflections are more viewpoint-independent.

Image Encoding

Instead of storing the generated mosaic images of the map, we store the descriptors of the images. There are a variety of choices of whole-image descriptors and compressed representation of images. NetVLAD [6] is employed in this case as the global descriptor because it is easily accessible in almost all deep learning frameworks. We use the version modified by [81] to combine the features extracted by MobileNet [80] with a NetVLAD layer. Their pre-trained weights are used out-of-the-box to generate the global descriptors for the rendered images. For a simple representation in our implementation, each global descriptor is a 1-by-4096 row vector and all global descriptors are vertically concatenated to form a N-by-4096 global descriptor array for N rendered images. This aggregated global descriptor array is then serialized as a Python pickle binary file that is loaded onto the UAV.

SuperPoint [23] is applied as the local descriptor choice. Unlike handcrafted descriptors such as SIFT whose number of detected key points can vary, NN-based descriptors trigger a fixed amount of computation from end to end while also being more robust to scene changes. For each pose, the calculated local descriptors are serialized as a single file. The database containing these encoded global and local descriptors is presented as Steps (iii-iv) in Figure 3.1.

Instead of matching descriptors via the brute-force matcher which iterates over all elements in the two sets without the awareness of semantic content, SuperGlue [83] (proposed as a supporting matcher for SuperPoint) is used to find the correspondences between the local descriptors from two images. SuperGlue's pre-trained weights are utilized for inference due to its good exposure to outdoor street photos in the training set.

3.2.2 Onboard Inference

After finishing the offline database preparation step, the 3D map and rendered images are discarded and only the serialized global descriptor array, the local descriptors and the depth maps are transferred to the aerial vehicle. NetVLAD, SuperPoint and SuperGlue network weights together with the calculated global descriptor array are de-serialized into the memory in advance for fast inference. When a query image is sent for inference, it is undistorted and down-sampled to match the dimensions of the prepared images, 640×480 in our case. We next detail the steps involved in online geo-localization.

Matching Strategy

We begin by calculating a global descriptor for the query image captured from the UAV camera. The ℓ^2-norm of the differences between the query image's global descriptor and every single global descriptor in the prepared array are computed and sorted in ascending order. After that, local descriptors are inferred for the query image and matched with the top n candidates in the sorted list using SuperGlue. Here, n is determined by the trade-off between performance and accuracy. Steps (1-4) in Figure 3.1 illustrate the procedure of the matching strategy.

Vehicle Pose Recovery

Using the matching strategy, within each pair of query and database global descriptors out of the n matched pairs, the correspondences of local key points are found from their individual sets of local descriptors. In terms of each retrieved set of local descriptors from the database, the simulated depths of the correlated key points are obtained from its rendered depth map. Therefore, both the set of matched 3D coordinates on the map and their corresponding 2D projections in the query image are collected. OpenCV's built-in implementation of the PnP solver with RANSAC outlier rejection is used to recover the vehicle pose. Empirically, the re-projection error during the RANSAC procedure is set to 1.0 pixel and the number of iterations to 1000. Among the converged trials, the result from the one with the most inliers is treated as the final pose. A refinement threshold is set for the distance between the refined pose and its initially matched quantized position. If the pose correction from the refinement exceeds the threshold, i.e. the pose after PnP is too far away from the initially matched position, the trial is rejected. Eventually, the pose in the relative metric coordinate is converted to the geographic coordinate in latitude, longitude and altitude. Steps (5-6) in Figure 3.1 illustrate this pose recovery procedure.

3.3 Experimental Setup and Results

Having shown our methodology for the geo-localization pipeline, we now introduce our experimental setups and present the field test results.



Figure 3.2: (Left) Front view of our customized quadrotor used at Location 1. Our camera is mounted and encapsulated by the blue cage. A GPS receiver is installed on the top to record ground truth trajectories. A Jetson AGX Xavier is attached to the back of the cage. (Right) The commercial DJI Mavic Air drone used to collect data at Location 2. The equipped gimballed camera can change the pitch angle resulting in a diversity of query images.

3.3.1 Apparatus

In order to test our pipeline's adaptability to images captured from various types of UAVs, we used two UAV platforms: a custom designed one as well as an inexpensive consumer UAV. As shown in Figure 3.2, our main vehicle was a customized quadrotor equipped with a passively cooled NVIDIA Jetson AGX Xavier as the primary computing resource. The Xavier possesses an 8-core Arch64 central processing unit (CPU) and a 512-core Volta graphics processing unit (GPU) to support the operating system and neural network inference. All motors and electronics were powered by a 16.8 V lithium polymer battery with a capacity of 5200 mA h. The vision part consists of a global-shutter camera from IDS Imaging along with a wide-angle lens from Lensation. The resulting horizontal FOV of this camera package is 84° after image undistortion. We used an image aspect ratio of 4:3 resulting in dimensions of 640×480 pixels. This camera package was installed at 45° tilted downward from the horizon level. The camera was then able to cover the bottom-front view.

Our secondary platform is a commercial Mavic Air quadrotor from DJI. The Mavic Air's camera offers a horizontal FOV of around 69° and is electronically gimballed. We pitched down this gimballed camera to mimic similar viewing angles and down-sampled its images to the same dimensions of those from the customized quadrotor.

3.3.2 Procedure

Two experiments at two different locations were conducted. We aimed to verify the versatility of the pipeline by adding this variation in localization scenarios.

At test Location 1, our custom UAV was flown to collect images. At this location, an area of 400 m × 400 m was selected for building the database so that the potential vehicle poses were confined in this space. An interval of 10 m was used to quantize the camera poses

along the latitude and longitude directions to generate the map mosaics. A fixed altitude of 70 m was used as the nominal flight elevation. The yaw angles were spaced by 30° within one revolution thus 12 directions in total. The pitch angle was fixed at 45° pointing to the bottom-front side. For these 19,200 poses, blank areas without map information may appear in the rendered images if only the selected area of the map is used. This is because the tilted viewing angle sees more than the area that the camera FOV projects perpendicularly on the ground. Thus, in practice, a larger map of around 1 km × 1.2 km was employed to cover the actual visual range.

Regarding the run-time settings, at Location 1, the top 3 matched candidates were used to enter the second stage for pose refinement via PnP. The refinement threshold was set to double the x- or y-direction spacing thus 20 m in this case. All global and local descriptors were stored on the vehicle in the format of Python pickle. The depth maps were saved using floating-point OpenEXR format.

At test Location 2, the DJI Mavic Air was flown to collect images. The parameters for generating the database were adjusted according to the camera settings of the Mavic Air. During the flight, we varied the pitch angle from 30° to 60° instead of keeping it constant as at Location 1 so that the database images were rendered at pitch angles of 30°, 45° and 60° respectively. The nominal flight elevation was also reduced from 70 m to 60 m. Particularly, the horizontal FOV was narrowed from 84° to 69°. Query images were captured at 0.5 Hz and of the same dimensions as at Location 1 after down-sampling. The potential vehicle poses were limited within an area of 250 m × 250 m. As this reduction in flying area releases more spare storage capacity for extra poses, the yaw angles were spaced by 24° thus 15 directions in total. An identical 1 km × 1.2 km area was used to cover the visual range. The serialization of descriptors and depth maps was completed in the same manner as was done in the first experiment. The same refinement threshold was set.

Since online inference was not possible on the Mavic Air due to the proprietary computation system, we ran the inference offline on a ground station. We still tested with different numbers of initial candidates that would enter the second stage for pose refinement via PnP.

3.3.3 Results

In the results from the first experiment conducted at Location 1, the database was generated using satellite imagery from 2018 while query images were collected from experiments in 2021. We reserved the top 3 matched global descriptors from the database and let them enter the pose refinement stage. The inferred and ground truth trajectories are overlaid on a satellite map and are demonstrated in Figure 3.3. The orientation of the satellite map in the background is aligned to the y-axis whose positive direction is strictly pointed to the North. In Figure 3.4, the inferred and ground truth altitudes are compared to show the performance of height estimation which is normally omitted in geo-localization benchmarks. The onboard localization pipeline was functioning during both the take-off and landing phases. The minimum localized altitude was 45 m even with database images that were rendered at a nominal altitude of 70 m. Figure 3.5 shows a comparison between the real query images and

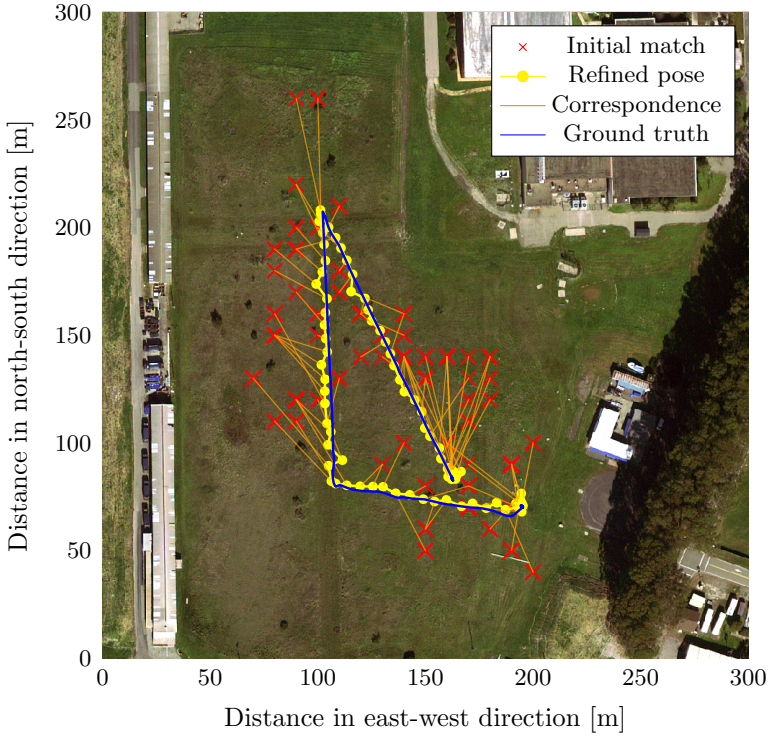


Figure 3.3: The top view of the trajectory generated by our geo-localization pipeline is shown overlaid on a satellite map of Location 1. The red crosses are the positions of the initial matched candidates at which database images were generated. The yellow markers stand for the vehicle positions after the PnP refinement. The orange line segments show the correspondences between the initial matched positions and those after the refinement. The blue trajectory represents the ground truth positions from GPS. The length of the flight path is 360 m.

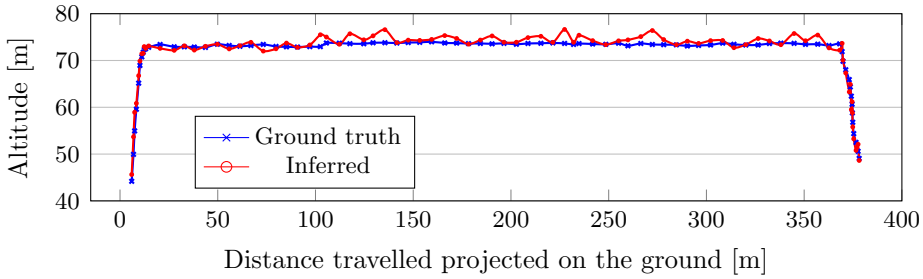


Figure 3.4: A comparison between the inferred and ground truth altitudes versus the distance traveled by the vehicle. Although the vehicle was able to localize itself from 45 m to 73 m in elevation, all database images were generated at a single altitude of 70 m



Figure 3.5: The top row shows the captured query images from the UAV and the bottom row presents the corresponding matched database images. In the two columns in the middle, there are significant discrepancies between the current query images and the dated satellite maps, e.g., the missing buildings in the scene.

database images. The localizer was able to overcome the significant discrepancies between the real scenes and outdated satellite maps with missing buildings.

In the results from the second experiment conducted at Location 2, the database was generated using satellite imagery from 2020 while query images were collected from experiments in 2021. A similar visual presentation of the results from this experiment can be found in Figure 3.6. This experiment was under a much more extreme condition, including variable pitch angles for the camera, image defects and lower altitudes. As shown in Figure 3.7, the minimum localized altitude was 18 m with database images rendered at a nominal altitude of 60 m. Due to the additional pitch angles added to the database, shown in Figure 3.8a, query images collected at pitch angles from 30° to 60° were able to be localized. When the camera was pointed too steep down, compared with the nominal pitch angles at which database images were rendered, the visual distortion and lack of features in the view led to failures in localizing the vehicle. This happens more frequently when an extremely low altitude, such as 18 m in this case, or a narrow FOV is present. From Figure 3.8b, it is noticed that the query images suffered from many environmental uncertainties. We encountered four types of interferences: lens flare, direct sunlight, high-contrast lighting and motion blur. Despite these visual interferences, all four of the listed images were localized successfully.

In terms of the real-time performance, this stand-alone package was able to infer the pose of the vehicle at 1.121 Hz on the customized quadrotor shown in Figure 3.2. Quantitatively, these inferred poses are compared with the ground truth positions recorded by a GPS receiver using the root-mean-square error (RMSE) metric. The 3D and 2D RMSEs are 2.816 m and 2.472 m respectively along a flight path of 360 m in the first experiment. The 3D RMSE includes the errors in altitude while the 2D RMSE only focuses on planar error distances. The localization was computed offline in the second experiment and the 3D and 2D RMSEs are 2.418 m and 2.108 m along a flight path of 220 m when three initial candidates were chosen.

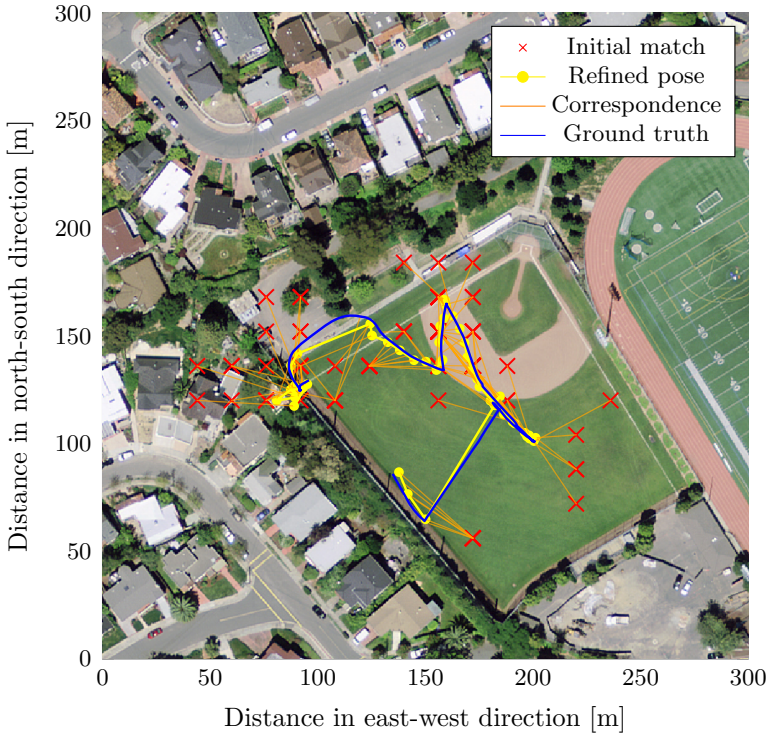


Figure 3.6: Top view of inferred and ground truth trajectories overlaid on a map of Location 2. The visualization format shares the same fashion as in Figure 3.3. The length of the flight path is 220 m.

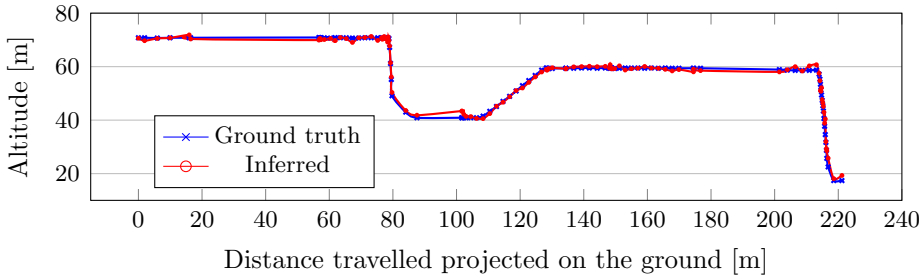
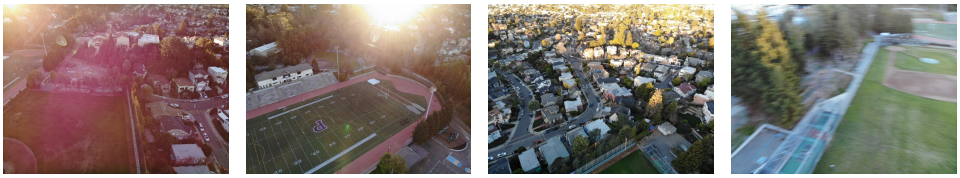


Figure 3.7: A comparison between the inferred and ground truth altitudes versus the distance traveled by the vehicle. Query images captured from 18 m to 70 m in elevation were successfully localized despite all database images being generated at a single altitude of 60 m.



(a) Query images captured at pitch angles from 30° to 60°.



(b) Sample query images collected under severe conditions, such as, from left to right, lens flare, direct sunlight, under/overexposure and motion blur, that were successfully matched and localized.

Figure 3.8: Images collected from the experiment conducted at Location 2.

A more thorough benchmark of the impact on the RMSEs from different numbers of initial candidates can be found in Table 3.2. In the first experiment at Location 1, a drastic reduction in the number of candidates from 50 to 1 brings negligible loss in the errors but around 30% decrease in the percentage of matched images among total query images, or alternatively, the recall. At Location 2, this decrease in the percentage grows further to 45%. Query images with features that are more distinct and unique in the flying area have a larger chance of convergence during the global descriptor matching stage. Therefore, just a few top candidates are sufficiently probable to capture the best match in a feature-rich area, and in contrast, more candidates are required to cover the ideal match for less distinguishing images such as those with plain colors, grass and repetitive patterns.

3.4 Summary

In this chapter, we proposed an image-based real-time geo-localization pipeline that features a reliable solution to modern localization needs on UAVs flying in GPS denied environments. The stand-alone pipeline executes on an embedded system at a high frequency and is able to accommodate to downward-tilted cameras. We show the practicality of our method by requiring no desired flight paths for preparing the map database and no prior knowledge of initial positions for online localization.

Some limitations and extensions of the pipeline are discussed next. Although the potential areas of search over the database can be confined according to the current position of the vehicle, the detection of outliers is still of vital importance when the very first global position is found. The current RANSAC approach applied during the PnP phase indiscriminately tests over the features extracted from the images. A content-aware inspector layer may be

Table 3.2: A benchmark of the impact on RMSEs and percentage match from different numbers of initial candidates at the global descriptor matching stage. We preserve these candidates to enter the local descriptor matching stage. As the number of candidates decreases, the RMSEs in both 3D and 2D cases remain approximately unchanged but the percentage of successfully localized attempts among all query images (recall) declines depending on scenarios. The choice of the number of candidates is a trade-off between accuracy and the success rate of localization.

(a) Experimental results at test Location 1.

Metric	**Number of initial candidates**					
	50	30	20	10	3	1
RMSE (3D) [m]	2.954	3.110	2.938	2.980	**2.816**	3.010
RMSE (2D) [m]	2.639	2.749	2.598	2.674	**2.472**	2.760
Recall [%]	**64.42**	61.13	55.05	50.39	46.90	42.79

(b) Experimental results at test Location 2.

Metric	**Number of initial candidates**					
	50	30	20	10	3	1
RMSE (3D) [m]	2.253	**2.198**	2.437	2.274	2.418	2.220
RMSE (2D) [m]	1.944	**1.894**	2.145	1.982	2.108	1.912
Recall [%]	**60.22**	56.91	52.49	45.30	38.67	33.15

added right after the global descriptor matching stage to further verify the validity of the matches because the descriptors can be similar even between completely different images. Additionally, the refinement threshold is assigned empirically and it may also reject correct estimates. Therefore, extensive research is still necessary to further enhance the robustness by improving the outlier rejection.

While this framework solves the problem of where a robot is in the world, autonomy also requires robustly solving how a robot interacts with it. Having established a method for state estimation in the air, we next turn our attention to the ground, addressing the challenge of dynamic control on legged platforms where physical compliance is paramount.

Chapter 4

Learning Torque Control for Quadrupedal Locomotion

Reinforcement learning (RL) has become a promising approach to developing controllers for quadrupedal robots. Conventionally, an RL design for locomotion follows a position-based paradigm, wherein an RL policy outputs target joint positions at a low frequency that are then tracked by a high-frequency proportional-derivative (PD) controller to produce joint torques. In contrast, for the model-based control of quadrupedal locomotion, there has been a paradigm shift from position-based control to torque-based control. In light of the recent advances in model-based control, we explore an alternative to the position-based RL paradigm, by introducing a torque-based RL framework, where an RL policy directly predicts joint torques at a high frequency, thus circumventing the use of a PD controller. The proposed learning torque control framework is validated with extensive experiments, in which a quadruped is capable of traversing various terrain and resisting external disturbances while following user-specified commands. Furthermore, compared to learning position control, learning torque control demonstrates the potential to achieve a higher reward and is more robust to significant external disturbances. To our knowledge, this is the first sim-to-real attempt for end-to-end learning torque control of quadrupedal locomotion.

4.1 Introduction

Legged locomotion is a longstanding challenge in the field of robotics. While the high degree of freedom, underactuation, and hybrid nonlinear dynamics of legged robots are particularly difficult, they have the advantage of being able to traverse rough terrain, making them highly useful in certain applications. Among the different types of legged robots, quadrupeds

The material in this chapter is based on the following previously published work: Shuxiao Chen et al. "Learning torque control for quadrupedal locomotion". In: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE. 2023, pp. 1–8.



Figure 4.1: A quadrupedal robot walks on a hill with a torque-based RL policy. The experimental video can be found at https://youtu.be/18v3Z9BD-W8.

have received considerable attention over the last few decades and have achieved significant advancements.

Since the 1980s, model-based control has been a primary method for achieving quadrupedal locomotion [76]. This approach uses a dynamics model to develop advanced controllers that enable a quadruped to perform a range of locomotion behaviors with rigorous stability and safety guarantees. Model-based control for quadrupeds has undergone a significant transformation, shifting from position control [75] to torque control [24]. This change has led to more dynamic and robust locomotion behaviors, indicating the superiority of torque control. On the other hand, learning-based approaches have recently emerged as a promising tool for quadrupedal locomotion, offering great potential for navigating rough terrain [44, 57, 65]. A notable difference between the learning-based approach and model-based control is that the former typically uses joint positions as the action space, whereas the latter favors torque control. In addition, it is worth noting that learning torque control has been utilized in various robotic applications beyond locomotion, e.g., manipulators [94] and quadrotors [104], with promising results. Inspired by the successful transition of model-based control, we explore an alternative approach for learning-based quadrupedal locomotion, where an RL policy directly predicts joint torques, instead of joint positions with a low-level PD controller.

4.1.1 Related Work

Learning quadrupedal locomotion has been widely studied in the past several decades. In the DARPA learning locomotion program, a hierarchical apprenticeship learning method was developed for quadrupedal walking [54]. More recently, deep reinforcement learning has emerged as a viable approach for quadrupedal locomotion. For instance, in [91], a walking policy was trained in simulation and successfully transferred to a real quadrupedal robot. Moreover, imitation from animals was explored to produce learned quadrupedal

motions in [72]. Several studies have also explored the use of model-based control in learned quadrupedal locomotion, as evidenced by works such as [22, 58, 28]. In addition to learning only in simulation, a substantial part of learning can also happen in the real world [89, 98]. Furthermore, a series of learning quadrupedal locomotion works have achieved state-of-the-art performance for traversing challenging terrain [44, 57, 65]. Additionally, other works have demonstrated different learned locomotion behaviors [86, 8, 9, 101, 20] and explored various learning approaches [90, 50, 55].

For the reinforcement learning-based legged locomotion, there are two main approaches in terms of the action space design: one is joint position [91, 57] and the other is joint torque [60, 84]. When joint positions are used as actions, the learned policy outputs target joint angles which are then sent to a PD controller. The PD controller is treated as a low-level tracking module, and the PD gains are usually tuned manually to track desired positions well. In addition, a prevalent variation is the residual joint position, where joint positions consist of heuristic or optimization-based joint positions and learned residual joint positions [57]. When the action space is joint torque, actions are directly applied to motors to actuate a robot. However, this action choice has only been demonstrated in simulation.

There has been some research to investigate the efficacy of both action choices. A study on different proportional gains for the PD controller was conducted in [99], which suggests that high proportional gains can cause instability on the robot while low proportional gains result in a policy behaving like a torque controller that leads to a more robust policy. Nevertheless, low proportional gains have larger tracking errors. In addition, the derivative gain for the PD controller was studied in [88], which validates that a small derivative gain leads to learning instabilities and a large derivative gain prevents the robot from tracking a target velocity. Thus, in order to have adaptive stiffness and damping at the joints, the PD gains can also be learned as part of the action space, as demonstrated in [87]. However, switching between different PD gains can potentially produce nonsmooth joint torques and cause instabilities of the robot.

In spite of the exploration in the design of PD controller, learning torque control was studied in [71] in a simulation setting, where a comparison among different action spaces was conducted, including joint positions, joint torques, joint velocities, and activations for musculotendon units. Note that in [71], joint torques are compared with joint positions at the same frequency for a quadruped. However, joint torques need to be executed at a higher frequency in order to achieve comparable performance to joint positions. Moreover, this figure shows that the performance of joint torques becomes better when the query rate increases, which implies the potential of learning torque control. Furthermore, in [77], a simulation study of joint positions and joint torques indicates that, while learning joint position control leads to a quicker early learning rate, learning joint position control may converge to lower-reward behaviors and may hinder the ability to adapt to rapid changes.

4.1.2 Contributions

Building upon the previous studies, in this work, we investigate the possibility of joint torques as the action space for a reinforcement learning policy with application to quadrupedal locomotion. To the best of our knowledge, torque control for real-world quadrupedal locomotion using end-to-end deep reinforcement learning has not been demonstrated yet, and the potential of learning torque control for quadrupedal locomotion is still underexplored.

The contributions of this work with respect to prior literature are summarized as follows.

- We present a sim-to-real learning torque control approach for quadrupedal locomotion, where an RL policy streams joint torques directly, without a PD controller.
- We validate the proposed approach through extensive experiments, in which a quadruped is capable of traversing different terrain and resisting external disturbances.
- We illustrate that learning torque control has the potential to achieve higher rewards, compared to the conventional learning position control.
- We demonstrate that a quadruped trained with torque-based RL is capable of resisting larger disturbances when compared with the position-based approach.
- We further discuss the key features of learning torque control and provide implementation suggestions.

4.2 Framework

In this section, we present our end-to-end learning torque control framework. Figure 4.2 illustrates the observations and policy architecture used to predict actions. Compared with the common position-based method, our proposed torque-based policy is queried at a high frequency, and it directly outputs joint torques, avoiding the use of a PD controller.

4.2.1 Simulation Environment

We choose Isaac Gym [63, 79] as our simulator. We train our policy end-to-end with 4096 parallel environments simulated on an NVIDIA GeForce RTX 3090 GPU. In addition, we use a curriculum of rough terrain in the simulation to improve the robustness of our policy.

4.2.2 Problem Formulation

We formulate quadrupedal locomotion as a reinforcement learning problem where the objective is to learn a policy π that allows an agent to maximize its expected return for a given task. At each time step t, a state $\boldsymbol{s}_t$ is observed by the agent from the environment, and an action $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is sampled from the policy π. The agent then applies this action

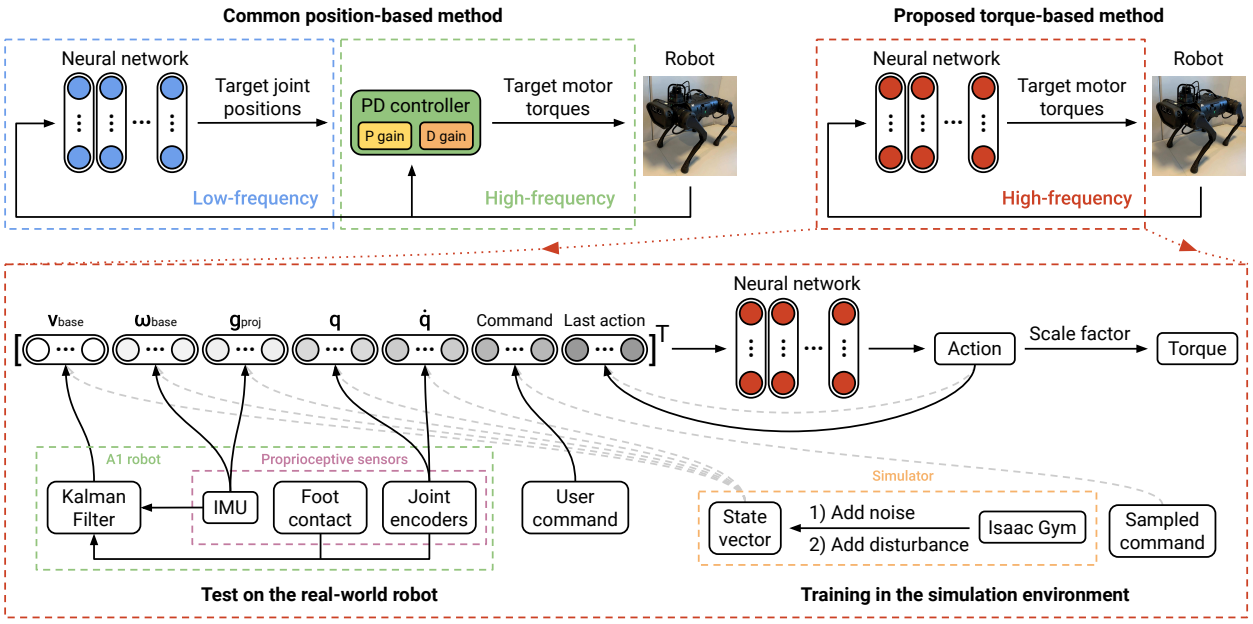


Figure 4.2: The proposed learning torque control framework. The common position-based method (top left) has two main modules where a neural network (blue) runs at a low frequency and a PD controller (green) executes at a high frequency. The proposed torque-based method (top right) merges the two modules in the position-based method into a high-frequency neural network module (red), as the RL policy is queried at a high frequency to directly output torques. A detailed illustration of the high-frequency neural network module is shown at the bottom, consisting of training in the simulation environment and test on the real quadruped. During the training, we add noise and disturbance to facilitate sim-to-real transfer. During the test, the state information is collected through proprioceptive sensors with appropriate preprocessing. The comprehensive explanation of the framework is presented in Section 4.2.

and drives its current state to a new state $\boldsymbol{s}_{t+1}$ and a scalar reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$ is received. This process is iterated to generate a trajectory $\tau = \{(\boldsymbol{s}_0, \boldsymbol{a}_0, r_0), (\boldsymbol{s}_1, \boldsymbol{a}_1, r_1), \cdots\}$. Therefore, the goal is to learn a policy that maximizes the agent's expected return $J(\pi)$ defined as

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right],$$

Table 4.1: Detailed definitions of the components forming the observation space. The observation space is divided into 7 observation items whose corresponding symbols are listed side by side. The scale factors stand for the coefficients that are multiplied to the observation items as a preprocessing step. The upper bounds of the zero-mean uniform noise added to the observation are also listed below.

Observation item	Symbol	Scale factor	Noise
Base linear velocity	$\boldsymbol{v}_{\text{base}} = [v_x\ v_y\ v_z]^\mathsf{T} \in \mathbb{R}^3$	2.0	0.2
Base angular velocity	$\boldsymbol{\omega}_{\text{base}} = [\omega_x\ \omega_y\ \omega_z]^\mathsf{T} \in \mathbb{R}^3$	0.25	0.2
Projected gravity	$\boldsymbol{g}_{\text{proj}} = [g_x\ g_y\ g_z]^\mathsf{T} \in \mathbb{R}^3$	1.0	0.05
Joint position	$\boldsymbol{q} = [q_1 \cdots q_{12}]^\mathsf{T} \in \mathbb{R}^{12}$	1.0	0.01
Joint velocity	$\dot{\boldsymbol{q}} = [\dot{q}_1 \cdots \dot{q}_{12}]^\mathsf{T} \in \mathbb{R}^{12}$	0.05	1.5
User command	$\boldsymbol{u} = [v_x\ v_y\ \omega_z]^\mathsf{T} \in \mathbb{R}^3$	$[2.0\ 2.0\ 0.25]^\mathsf{T}$	0
Last action	$\boldsymbol{a}_{\text{last}} = [\tau_1 \cdots \tau_{12}]^\mathsf{T} \in \mathbb{R}^{12}$	1.0	0

where T is the time horizon for each episode, and $\gamma \in [0, 1]$ is a discount factor. $p(\tau|\pi)$ represents the likelihood of a trajectory τ under a policy π,

$$p(\tau|\pi) = p(\boldsymbol{s}_0) \prod_{t=0}^{T-1} p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)\, \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\ ,$$

where $p(\boldsymbol{s}_0)$ is the initial state distribution, and $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ denotes the state transition.

4.2.3 Observation Space

We choose a concise representation of the robot that can prove the advantage of learning direct torque control, alleviate overfitting to simulated dynamics, and facilitate sim-to-real transfer [91], excluding any history of proprioceptive sensor measurements or privileged information. The observation space consists of 7 parts and they together form a 48×1 vector. The different components consist of robot base linear and angular velocities, orientation, joint positions and velocities, last action, and user commands. Comprehensive definitions of the observation space can be found in Table 4.1. Base linear and angular velocities and projected gravity are in the robot frame. The projected gravity is a unit vector capturing the robot's orientation with respect to gravity. Desired forward velocity, lateral velocity, and yaw rate are sent to the controller as the user commands to modify the policy behavior online. Joint position, joint velocity, and last action are each 12-dimensional corresponding to the hip, thigh, and calf joints of each of the four legs of the quadruped. The last action is the last neural network output without being multiplied by the action scale.

4.2.4 Action Space

The action space of our model is a 12×1 vector consisting of unscaled joint torques. This vector is then multiplied by a constant action scale to produce the actual desired torque values sent to the 12 motors.

The intuition behind the action scaling is as follows: typical position-based RL policies usually add or subtract position values from a reference or nominal position vector, like a standing pose for the robot. There are two reasons for this choice. First, learning a residual on a nominal configuration of joint positions makes it easier for the agent to survive during the early stages of training by giving it a prior standing pose. Secondly, it also helps establish a good starting point for policy exploration because desired gaits often occur around this nominal configuration. However, determining nominal torques is difficult for torque control, since these might vary with different environments and initial configurations. Hence, we remove the use of reference or nominal torques and instead use a scaling vector to give final torques to the motor controller.

Due to the fact that the numerical range of torque control (e.g., -33 to 33 Nm being the safe joint torque range on the A1 robot) is much larger than that of position control (e.g., -3.14 to 3.14 rad), we use a large value of 10 as the action scale. This action scale strikes a good balance between exploration and exploitation in the absence of nominal torques, and it also helps stabilize the robot in the early stage of training. Eventually, all final torques are clamped to the range $[-33, 33]$ Nm in both simulation and real-world tests to ensure the safety of the robot whilst retaining enough leeway for agile locomotion.

4.2.5 Rewards

Similar to having no nominal configurations for the action space, no reference trajectories or demonstrations are used in our reward functions either. The learned policy is able to evolve spontaneously from high-level task rewards, and generates gaits that are suitable for high-frequency torque control without any additional constraints to reference motions.

Overall, the total reward function encourages the agent to follow a given set of high-level user commands, whilst satisfying additional behavioral, heuristic, and cosmetic requirements. Detailed definitions of the individual reward terms can be seen in Table 4.2. The hip and thigh reward terms drive the positions of the joints close to nominal positions. This creates "natural looking" gaits on the robot, enhancing the sim-to-real transfer probability of learned policies.

4.2.6 Policy Architecture and Training

The policy is modeled with a multilayer perceptron (MLP) which has 3 layers of 512, 256, and 128 exponential linear units (ELUs). The standard deviation of the action distribution is set to a fixed value. The value function is modeled by a separate network with the same configuration as the policy network. The policy is optimized with proximal policy



(a) (b) (c) (d)

Figure 4.3: The proposed end-to-end learning torque control framework allows the A1 robot to traverse various terrains, such as (a) inclined soft lawn, (b) pebble path, (c) compliant terrain, and resist external disturbances such as (d) recovering from a kick. All the tests use the same RL policy queried at 200 Hz.

optimization (PPO) [85]. Note that we do not choose a complicated architecture, such as a teacher-student structure in [57], since we would like to use a simple architecture to validate our learning torque control framework.

Table 4.2: Detailed definitions of the proposed reward terms. Variables marked with an asterisk stand for target values. Δt is defined by the inverse of the control frequency, in this case, 0.005 s. Positive weights mean encouragement to such behaviors and negative ones penalty. In the hip (thigh) reward, H (T) is the set of 4 hip (thigh) motor indices.

Reward term	Expression	Weight
Base linear velocity x-y	$\exp\left(-\lVert \boldsymbol{v}^*_{\text{base},xy} - \boldsymbol{v}_{\text{base},xy} \rVert^2 / 0.25\right)$	$1.0\Delta t$
Base linear velocity z	$-v^2_{\text{base},z}$	$-2.0\Delta t$
Base angular velocity x-y	$-\lVert \boldsymbol{\omega}_{\text{base},xy} \rVert^2$	$-0.05\Delta t$
Base angular velocity z	$\exp\left(-(\omega^*_{\text{base},z} - \omega_{\text{base},z})^2 / 0.25\right)$	$0.5\Delta t$
Orientation	$\lVert \boldsymbol{g}_{\text{projected},xy} \rVert^2$	$-1.0\Delta t$
Joint torque	$\lVert \boldsymbol{\tau} \rVert^2$	$-0.0002\Delta t$
Joint acceleration	$\lVert \dot{\boldsymbol{q}}_{\text{last}} - \dot{\boldsymbol{q}} \rVert^2$	$-0.00000025\Delta t$
Feet air time	$\sum_{i=1}^{4}(t_{\text{swing},i} - 0.5)$	$2.0\Delta t$
Collision	$\sum_{i=1}^{4} c_{\text{knee},i}$ for $c_{\text{knee},i} \in \{0, 1\}$	$-1.0\Delta t$
Action rate	$\lVert \boldsymbol{a}_{\text{last}} - \boldsymbol{a} \rVert^2$	$-0.01\Delta t$
Hip	$\sum_{h \in H} \lvert q^*_h - q_h \rvert$	$-1.0\Delta t$
Thigh	$\sum_{h \in H} \lvert q^*_t - q_t \rvert$	$-1.0\Delta t$

4.2.7 Domain Adaptation

Randomization of ground frictions and external disturbances is included to facilitate the building of robustness. A friction coefficient is sampled uniformly in the range $[0.5, 1.25]$ and assigned to each simulated robot. External pushes are exerted on each robot every 15 s by sampling and setting the instantaneous x-y velocity of the robot base from $[-1.0, 1.0]$ m/s. Note that uncertainties of the physical parameters, e.g., dynamics randomization [5, 73], can be viewed as external forces applied to the robot [74, 91]. The linear velocity commands are also randomized in range $[-1.0, 1.0]$ m/s in both x- and y-directions, and the yaw rate is randomized in range $[-3.14, 3.14]$ rad/s, in order to produce a goal-conditioned RL policy.

4.3 Results and Analysis

Having presented the end-to-end learning torque control framework, we now evaluate it with various experiments. We first introduce the experimental apparatus and a baseline policy. Then, real-world experiments and comparisons are demonstrated. Finally, we provide an in-depth discussion on the learning torque control framework.

4.3.1 Apparatus

We use a Unitree A1 quadrupedal robot to evaluate our end-to-end learning torque control framework. The A1 robot has 18 degrees of freedom (DoFs) with 12 actuated joints and 6 generalized coordinates for the floating base. There is an NVIDIA Jetson TX2 inside the robot body to offer extra computing power. The TX2 has a 256-core GPU for neural network inference implemented with Mobile Neural Network (MNN) [49], an open-source inference engine that is lightweight and supports various hardware backends. In addition, our policy runs completely onboard.

4.3.2 Baseline

We develop a position-based RL policy for the same quadruped as a baseline. This policy uses joint positions as the action space and runs at 50 Hz, followed by a low-level PD controller tracking at 200 Hz, which is the same frequency as the torque-based RL policy. For the PD controller, the proportional gain is 20, and the derivative gain is 0.5. This position-based RL policy is trained with the same level of noise and disturbance as well as the terrain curriculum as the torque-based RL policy. In order to make a fair comparison, we make the reward function of the position-based RL close to the torque-based RL one. However, the same reward function does not work directly for position-based RL policy. We modify the reward function in a minimal way which excludes the orientation, hip, and thigh reward terms, and changes the weight of the feet air time reward from 2.0 to 1.0. The action scale for this position-based policy is 0.25 with a nominal position. This position-based policy also allows a quadruped robot to traverse different terrains.

4.3.3 Results

We conduct experiments both in simulation and on hardware to extensively study our proposed learning torque control framework. In the simulation, we compare the difference between learning torque control and learning position control in terms of their learning curves. On hardware, we validate the learned torque control policy on various terrains with user-specified commands. The snapshots of the learned locomotion behaviors are shown in Figure 4.1 and Figure 4.3.

In Figure 4.3a, a quadruped robot walks on an inclined soft lawn on campus, which shows that the learned torque policy allows the quadruped to locomote over non-flat terrain while following commands. In Figure 4.3b, the quadruped robot walks on a pebble path, where the pebbles are not embedded in the ground and can be slippery. While the robot walks forward, the pebbles collide with the feet, which causes additional disturbances to the robot. In this experiment, the robot gets stuck in pebbles for a moment. The operator increases the forward velocity command, and then the robot is able to get out of the pebbles and keeps walking forward. Figure 4.3c demonstrates that our torque policy enables the quadruped robot to walk over a compliant terrain, i.e., an air mattress. A similar test was conducted in [20] using a position-based RL policy with an adaptive control architecture. In addition to walking on various terrains, we also test the robot with external disturbances as shown in Fig. 4.3d. We kick the robot from different angles when it is standing or walking, and it is capable of resisting a certain amount of external forces. All of these locomotion behaviors are performed with the same torque-based RL locomotion policy queried at 200 Hz.

In order to study the difference between learning position control and learning torque control, we carry out real-world experiments to compare their action tracking performances. We perform real-world experiments on the same A1 robot given the same forward velocity command at 0.5 m/s. The action tracking result for the rear left calf joint is shown in Figure 4.4, where the top plot illustrates the result of learning position control and the bottom plot demonstrates the result of learning torque control. The green lines denote the target values produced from RL policies with post-processing, and the orange lines indicate the actual values measured from the hardware. Learning position control has a larger tracking error, whereas learning torque control has a much better tracking performance. It is worth noting that this comparison does not indicate that learning position control has worse performance, since the position-based RL can adjust the target position so that the actual position achieves the goal during training. Here, we only aim to demonstrate a major difference between the two methods.

We were concerned that joint torques need to be changed rapidly between the stance phase and the swing phase for a quadruped robot, compared to joint positions. Thus, it may pose challenges to RL to learn to control a quadruped robot using joint torques as the action space. Figure 4.4 indicates that RL is powerful enough to handle this rapid change in the action space. In addition, we were concerned that the action of RL might not be smooth enough to be fed to motors. For learning position control, there is a low-level PD controller to track the action of RL, and it can locally smooth out jerky values. Figure 4.4 demonstrates

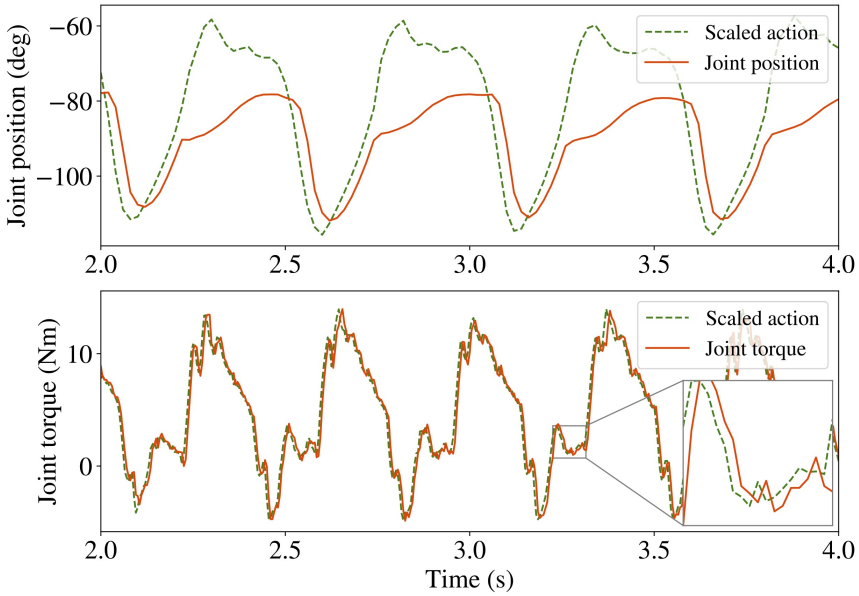


Figure 4.4: **Action tracking comparison** between learning position control (top) and learning torque control (bottom) on a real quadruped robot for the rear left calf joint. The green lines depict the target values queried from RL policies with post-processing, e.g., scales. The orange lines illustrate the actual values measured from the real quadruped robot.

that torque-based RL produces smooth values for motors. Although we occasionally notice some jerky values, the motors on the hardware are generally tolerant. There is no need to add another layer, e.g., a PD controller or a low pass filter, to smooth out the RL actions.

Furthermore, we compare the normalized learning curves for learning position control and learning torque control, as shown in Figure 4.5. We simplify the reward function design for this comparison, which only consists of the base velocity tracking reward. We choose three different proportional gains, i.e., 40, 60, and 80, which are illustrated as orange, green, and red lines, and the derivative gain is 0.5. The torque-based RL is depicted as the blue line. The learning curves for the position-based RL change with respect to different proportional gains, which indicates that proportional gains have an impact on the learning process. In contrast, while torque-based RL shows a slow learning rate at the early stage, it converges to a higher return, which shows that torque-based RL has the potential to achieve a higher expected return, compared to position-based RL. This result coincides with the findings presented in [77], which conducts a similar comparison but using a much simpler system.

In addition, we compare the robustness between the position-based RL policy and the torque-based RL policy. We perform a real-world experiment where the robot stands while getting consecutive kicks, and the result is presented in Figure 4.6. The green, blue, and red lines show the measured, nominal, and minimal base heights, respectively. The robot base heights are calculated by averaging the z-direction foot positions in the robot frame

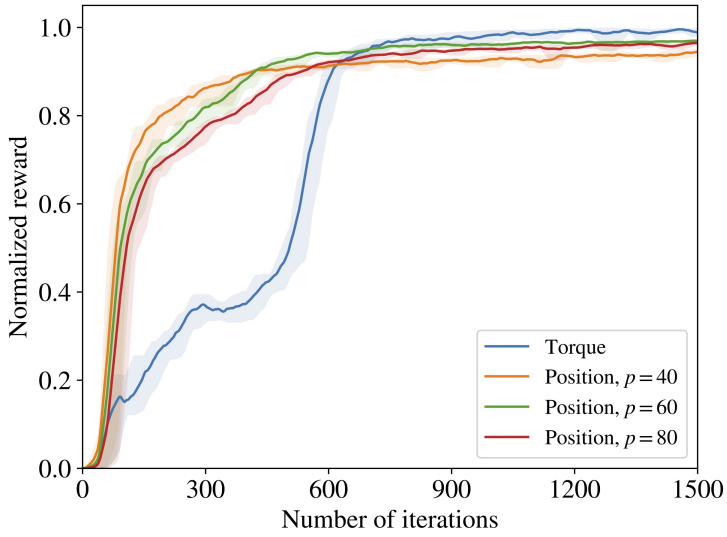


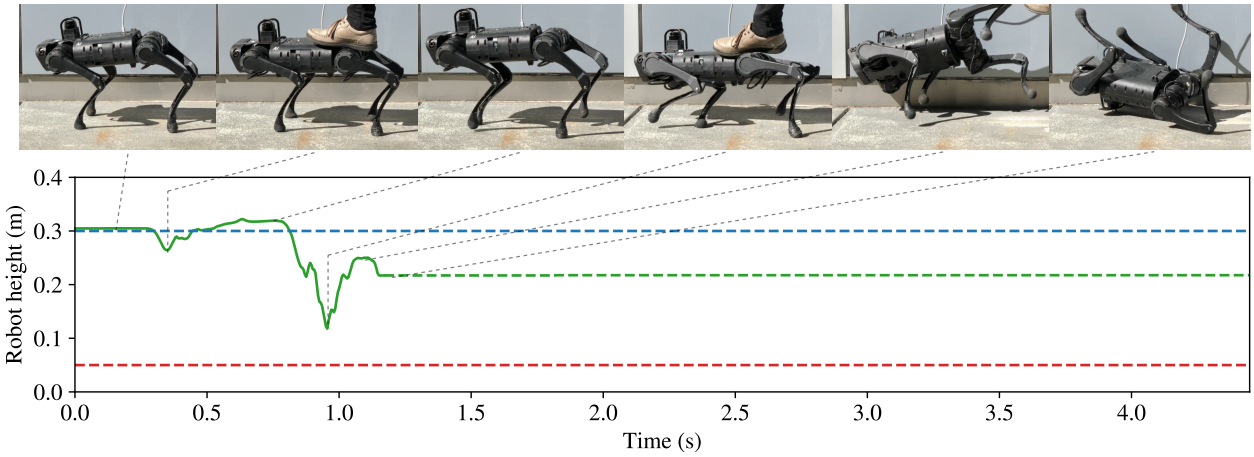
Figure 4.5: **Learning curve comparison** between learning position control and learning torque control. We choose three different proportional gains, i.e., 40, 60, and 80, for the position-based RL, which are denoted as orange, green, and red, respectively, and the torque-based RL is depicted as blue.

obtained by forward kinematics (FK). Since the robot body has non-negligible thickness, the minimally possible base height measured from FK is 0.05 m when the robot body touches the ground. In the top plot, a position-based RL policy allows the robot to resist the first kick, but the robot falls over after the second kick. In this case, the base height is no longer valid after 1.2 s due to the feet's loss of contact to the ground. In the bottom plot, with a torque-based RL policy, the robot is able to sustain itself during three successive kicks. Note that the third kick is the hardest one where the quadruped base even contacts the floor. This kicking experiment demonstrates that the torque-based RL policy shows more robustness than the position-based RL policy. We hypothesize that such robustness comes from torque control. Note that this robust behavior is different from the learned recovery behavior in [56], as we do not explicitly train the robot to resist vertical kicks. The robustness of the position-based RL policy can be further improved if we explicitly train with downward kicks.

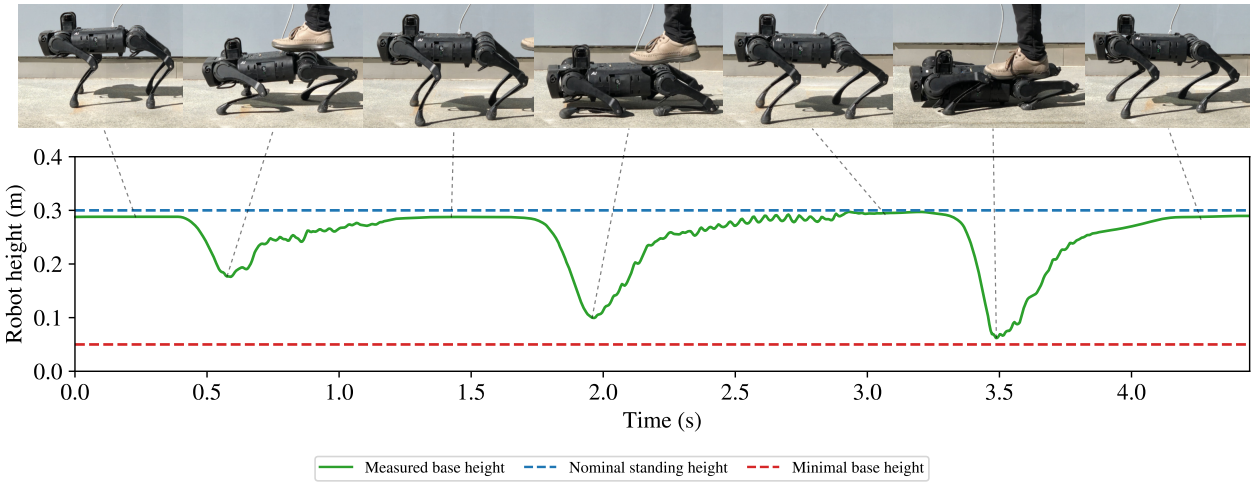
4.3.4 Discussion

Control Frequency

Our torque-based RL policy is queried at 200 Hz, whereas the query frequency of most position-based RL policies ranges from 20 to 100 Hz [57], followed by a low-level PD controller. During our preliminary test of the policy querying frequency, the GPU on the TX2 can stably infer the 3-layer MLP and maintain the inference at 20 kHz without executing its



(a) When using the position-based RL policy, the robot is able to sustain itself during the first gentle kick, but after the second larger kick, the robot falls over. The green dashed line stands for the last valid base height measurement before the flip.



(b) When using the torque-based RL policy, the robot is capable of resisting three consecutive kicks successfully, and the third kick is the strongest one during which the robot base even contacts the ground.

Figure 4.6: **Robustness comparison** between learning position control and learning torque control, where the robot base height is illustrated under successive downward kicks while it is standing. The green lines show the measured base heights, and the blue lines depict the nominal base heights. The red lines illustrate the minimal base heights where the base is on the ground. Note that these vertical kicks are not present during training.

outputs on the robot hardware, which indicates the potential for a higher-frequency policy if needed. Moreover, compared to the position-based RL policy, the minimum query frequency of the joint torque-based policy is supposed to be higher.

Sim-to-Real Transfer

One of our concerns in learning torque control was the difficulty of transferring a torque-based policy from simulation to hardware. Surprisingly, it turns out that there is no additional difficulty of sim-to-real transfer for learning torque control, compared to the conventional learning position control. The common domain randomization strategy leads us to successful sim-to-real transfer for both position-based and torque-based policies.

Circumvention of PD Controllers

Learning torque control allows us to circumvent the use of a low-level PD controller, as the torque-based RL policy directly outputs joint torques. For the position-based locomotion policies, PD gains are usually tuned manually and tightly related to the learning locomotion performance, as studied in [99, 88]. Thus, the torque-based RL method eliminates the number of parameters to tune. Especially, these PD gains need to be tuned on hardware involving extra engineering work.

Potential of Learning Torque Control

This is the first attempt, to our knowledge, of learning torque control for quadrupedal locomotion with real-world experiments. From a perspective of biomechanics, torque control is more inclined to the way that animals or humans locomote [1, 33]. Hence, we hypothesize that learning torque control can potentially lead to more dynamic and flexible behaviors in the future.

4.4 Summary

In this chapter, we studied an alternative approach to learning to control a quadruped robot, which uses joint torques as the action space in reinforcement learning, instead of joint positions as conventional learning quadrupedal locomotion work does. We validated the proposed method via real-world experiments, where a quadruped robot is capable of traversing different terrains and resisting external disturbances. Moreover, compared to the learning position control method, we showed that the learning torque control framework can potentially achieve a higher expected return and result in more robust locomotion behaviors. Finally, we provided discussions on the key features of learning torque control.

Even robust controllers are ultimately limited by the quality of the feedback. The dynamic capability demonstrated here relies on accurate knowledge of the robot's base velocity, a variable that is notoriously difficult to measure directly. To prevent state estimation errors

from becoming a bottleneck for this control authority, the following chapter introduces a framework to robustly infer these states.

Chapter 5

Modular Legged State Estimation via Reinforcement Learning

We present a modular legged state estimation framework via reinforcement learning (RL) that decouples estimator training from controller design. In contrast to prior approaches that tightly integrate estimation and control via concurrent training or the teacher-student architecture, our method bundles an existing controller and the simulator to form a unified environment, allowing the estimator to be optimized independently as the agent. Specifically, we train a base linear velocity estimator against a quadrupedal locomotion controller. From both simulation and real-world experiments, the controller-estimator system shows superior command tracking accuracy, maintains constant inference complexity, and inherits the sim-to-real characteristics of the controller.

5.1 Introduction

Recent years have seen rapid co-evolution of legged locomotion control algorithms and robotic platforms. The Mini Cheetah [51] has become a standard open-source benchmark, supporting a range of model-based control strategies. These include robust feedback controllers and perception-integrated planners for navigating complex terrain [31, 66, 32, 46]. Simultaneously, reinforcement learning (RL) has emerged as a powerful tool for synthesizing model-free controllers parameterized by neural networks. Foundational architectures [44, 57, 65] demonstrate strong generalization across diverse terrains, enabling deployment in slippery, uneven, and outdoor environments. RL-based motion imitation [72] further expands locomotion capabilities by replicating animal trajectories Recent works [38, 18] achieve dynamic skills such as parkour, made feasible by large-scale parallel simulation [79].

Despite the impressive performance of recent locomotion controllers [29, 52, 25, 40, 44, 57, 100], the vast majority remain heavily reliant on derived proprioceptive state estimates, most notably, the base linear velocity. This dependency stems from a shared design objective: tracking high-level base velocity commands as a control target. However, prevailing

state estimation techniques [45] exhibit significant limitations in real-world scenarios. Some degrade on low-friction surfaces [34, 53]; others require prior knowledge of the controller being used [51]; and many depend on dedicated contact sensing hardware, which is susceptible to failure due to repetitive ground impacts [72, 17, 3].

While it is possible to bypass state estimators for simple locomotion tasks [91], many methods integrate estimation directly into control. The teacher-student architecture [16, 57, 55] distills an implicit estimator into the control policy by training a student to mimic a teacher policy that uses privileged simulation data. This distillation requires policy retraining and real-world engineering effort, and often results in degraded performance due to divergence from the original design. To address this, concurrent training of an explicit estimator and controller with an intermediate supervision loss [47] has become popular in legged locomotion [48]. This tandem setup, combining a supervised estimator and an RL controller, enables mutual adaptation and encourages the controller to avoid states where estimation is unreliable, enhancing overall performance.

However, the aforementioned measures can be inadequate for fulfilling particular controller design requirements or alleviating burdens in engineering implementation. From the perspective of the design paradigm, they preclude researchers from independently tailoring a controller to meet specific and exact needs. Instead, one has to anticipate the performance degradation caused by the mutation of the student policy or the influence from the concatenated estimator when designing the overall system. Because the degradation is unpredictable and unquantifiable, almost the only pragmatic engineering of such a system is on a trial and error basis, i.e. any change made to any part in the system requires the whole student or tandem to be retrained. Consequently, the excess computational investment in design trials and hyperparameter tuning is more than fitting a stand-alone estimator for a controller generated separately.

To complement existing locomotion controllers, we propose a framework that trains an RL-based estimator against a fixed controller. The estimator is controller-agnostic, enabling independent design. By rewrapping the simulator and controller as a unified environment, we train the estimator similarly to an RL controller, with real-time feedback. If the controller exhibits a sim-to-real gap, the combined system naturally inherits it, enabling zero-shot deployment on hardware.

In the context of system engineering, we propose a structured pipeline for developing a robust controller-estimator tandem using RL. The pipeline begins with a high-level overview of the controller, from which a reduced version is generated by removing the estimated state fields from the observation space. While this reduction leads to significant performance degradation due to the loss of observability in the estimated state fields, this step is essential for sim-to-real validation. Once the reduced controller passes the sim-to-real test, the estimated state fields are reintroduced. After achieving satisfactory performance in simulation, the estimator counterpart is constructed using the proposed RL-based method. The resulting controller-estimator tandem is designed to operate seamlessly on real hardware, ensuring robust and reliable performance.

5.1.1 Contributions

Building upon prior studies, this work explores the potential of RL for state estimation in legged robotics, with a focus on estimating the base linear velocity, a critical state for legged locomotion. We demonstrate an RL-based state estimation framework that operates in a modular fashion, inheriting the controller's sim-to-real transfer characteristics in real-world scenarios. Our approach not only addresses the limitations of existing methods but also opens new avenues for efficient system design in legged robotics.

We summarize our contributions as follows.

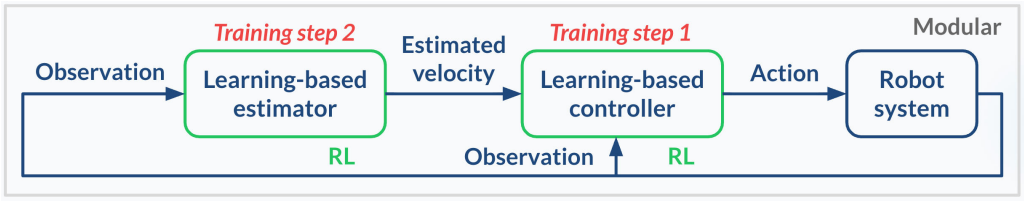
- We propose an RL-based approach for legged state estimation, with a focus on estimating the base linear velocity. The estimator training is conducted independently thus modular, ensuring no interference with the original controller design, thereby decoupling the development of the controller and the estimator in contrast to the concurrent solution [47].
- By leveraging locomotion controllers with known sim-to-real gaps, the estimator counterpart inherits the zero-shot deployment on real hardware. The performance feedback of the estimator during training is faithfully observable in simulation, avoiding costly real-world trial-and-error tests.
- The estimator operates with constant time complexity as an important advantage of neural networks, ensuring efficient real-world deployment as opposed to model-based estimators.

5.2 Methodology

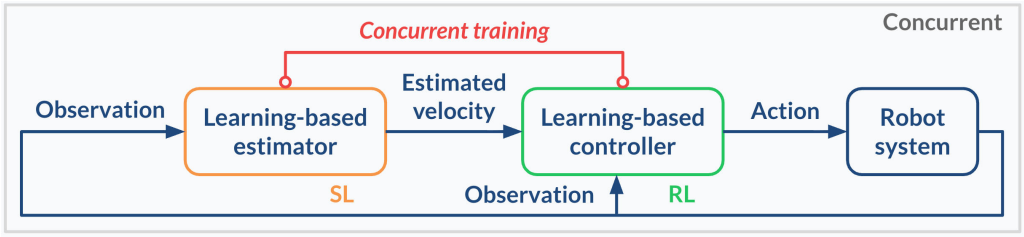
In this section, we present our approach to constructing legged state estimators based on RL in detail. Figure 5.1 visually contrasts the proposed approach with established legged locomotion and state estimation frameworks. Our proposed framework embeds an existing target controller as part of the environment agnostic to the estimator to be trained, allowing immediate feedback of the training performance and inheritance of the controller's sim-to-real characteristics. To better contextualize our methodology, specifically, we present the approach through the lens of base linear velocity estimators, a critical component that complements mainstream RL-based locomotion controllers.

5.2.1 Observation and Action Spaces

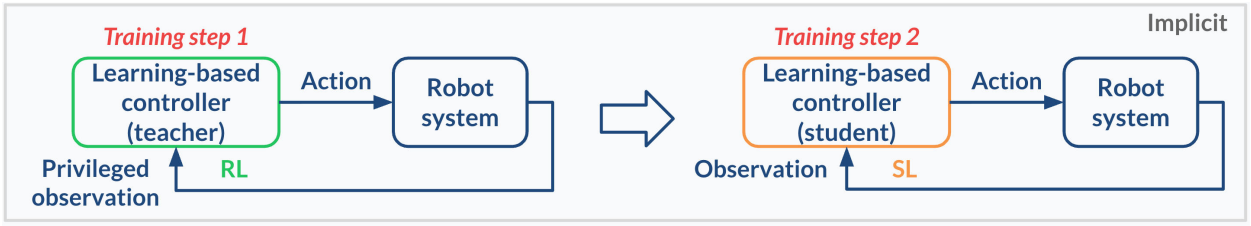
We adopt a common observation space composition for quadrupedal robots first introduced in [79] that is also widely utilized in many popular projects [64, 18]. After our simplification, the resulting observation terms are as follows: 3-axis base angular velocity $\boldsymbol{\omega}_{\mathrm{b}} \in \mathbb{R}^3$, unit projected gravity vector along the x- and y-axis $\boldsymbol{g}_{\mathrm{p}} \in \mathbb{R}^2$ representing the robot orientation in roll and pitch, user command $\boldsymbol{u} \in \mathbb{R}^3$ including target base linear velocity along the x- and



(a) Reinforcement learning for modular state estimation (proposed).



(b) Concurrent training of control policy and state estimator.



(c) Teacher-student architecture with implicit estimator.

Figure 5.1: Block diagrams of three architectures for legged locomotion and state estimation are illustrated here: (a) the proposed method, (b) a recently devised concurrent training approach for controller-estimator pairs that has gained popularity [47], and (c) the established, widely-used teacher-student architecture featuring an implicit and inseparable estimator [57]. (a) trains an estimator via RL in a unified simulator-controller environment, while keeping the existing controller fixed. (b) jointly trains a supervised-learned estimator and an RL-based controller tandem from scratch, promoting mutual adaptation. The controller inherently avoids states where the estimator exhibits uncertainty, enhancing system robustness. (c) trains a teacher policy using privileged states, then distills a student policy with practical states to replicate the teacher's performance. During distillation, the student not only adheres to the teacher's control objectives but also implicitly encodes an estimator component alongside.

Table 5.1: Detailed descriptions of the observation terms, expressions, scale factors and noises added to the ground truth values. The observation terms are multiplied by corresponding scale factors as a normalization step before being input to the neural network. All intervals of the uniform noises are symmetric about zero.

Obs. term	Expression	Scale	Noise
Base ang. vel.	$\boldsymbol{\omega}_\mathrm{b} = [\omega_x\, \omega_y\, \omega_z]^\intercal \in \mathbb{R}^3$	0.25	$\mathcal{U}^3(\pm 0.2)$ rad/s
Proj. gravity	$\boldsymbol{g}_\mathrm{p} = [g_x\, g_y]^\intercal \in \mathbb{R}^2$	1	$\mathcal{U}^2(\pm 0.05)$ rad/s^2
User cmd	$\boldsymbol{u} = [v_x\, v_y\, \omega_z]^\intercal \in \mathbb{R}^3$	$[1\,1\,0.25]^\intercal$	0
Joint position	$\boldsymbol{q} = [q_1 \cdots q_{12}]^\intercal \in \mathbb{R}^{12}$	1	$\mathcal{U}^{12}(\pm 0.01)$ rad
Joint velocity	$\dot{\boldsymbol{q}} = [\dot{q}_1 \cdots \dot{q}_{12}]^\intercal \in \mathbb{R}^{12}$	0.05	$\mathcal{U}^{12}(\pm 1.5)$ rad/s
Last con. ac.	$\boldsymbol{a}_\mathrm{c} = [\hat{q}_1 \cdots \hat{q}_{12}]^\intercal \in \mathbb{R}^{12}$	0.25	0
Last est. ac.	$\boldsymbol{a}_\mathrm{e} = [\hat{v}_x\, \hat{v}_y]^\intercal \in \mathbb{R}^2$	1	0

y-axis and target base angular velocity along the z-axis, joint position $\boldsymbol{q} \in \mathbb{R}^{12}$, joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^{12}$, last controller action $\boldsymbol{a}_\mathrm{c} \in \mathbb{R}^{12}$ and last estimator action $\boldsymbol{a}_\mathrm{e} \in \mathbb{R}^2$. All terms are in the robot base frame which usually coincides with the inertial measurement unit (IMU) frame. The controller action used in the latter experiments is the target joint position for a control subsystem to track. The estimator action is the base linear velocity in the x- and y-direction. Comprehensive definitions, dimensions, scale factors and noise descriptions of the observation terms are listed in Table 5.1. Thus, the observation tuple for the estimator is defined as

$$\boldsymbol{s}_t = \left(\boldsymbol{\omega}_\mathrm{b}, \boldsymbol{\Omega}^n_\mathrm{b}, \boldsymbol{g}_\mathrm{p}, \boldsymbol{G}^n_\mathrm{p}, \boldsymbol{q}, \boldsymbol{Q}^n, \dot{\boldsymbol{q}}, \dot{\boldsymbol{Q}}^n, \boldsymbol{a}_\mathrm{c}, \boldsymbol{A}^n_\mathrm{c}, \boldsymbol{a}_\mathrm{e}, \boldsymbol{A}^n_\mathrm{e}, \boldsymbol{u}\right),$$

where capital letters represent history of corresponding lowercase quantities and the superscript n stands for the length of the history, i.e. the number of steps of previous values to be appended.

Notice that the aforementioned terms are all commonly and natively accessible from the sensors installed on a quadrupedal robot. This representation is chosen to maximize the conciseness of information and the compatibility across different robot platforms.

For more convenient studies of the estimator performance, we choose a representative action space in significant demand by many tasks, merely the plain base linear velocity along the x- and y-axis, $\hat{\boldsymbol{v}}_\mathrm{b} = [\hat{v}_x\, \hat{v}_y]^\intercal \in \mathbb{R}^2$, as the upstream input to the controller. The z-axis component $\hat{v}_z$ is excluded for simplification because v_z is not often part of the user command for the controller to track. For an RL-based controller, v_z may appear as a penalization term to be incorporated in the reward to prevent excessive vertical body shake, but this behavior is usually implicitly learned by the controller without needing an explicit v_z in the observation space.

5.2.2 Rewards

A spontaneous design of the reward of such an estimator outputting the base linear velocity is tracking the ground truth base linear velocity accessible directly from the simulation environment

$$\boldsymbol{r}_{\text{estimator}} = \exp\left[-\left\|\hat{\boldsymbol{v}}_{\text{b},xy} - \boldsymbol{v}_{\text{b},xy}\right\|^2 / \sigma\right],$$

as the only term in the reward, where $\boldsymbol{v}_{\text{b}}$ and $\hat{\boldsymbol{v}}_{\text{b}}$ are the ground truth and estimated base linear velocities, σ is the tracking factor that helps shape the resulting exponential curve.

Prior work has explored estimators trained sequentially alongside existing controllers via supervised learning, as investigated in [47] and heavily adopted in [103]. However, this method exhibits performance degradation due to overfitting. Notably, significant sim-to-real gaps have been documented in [103], prompting mitigation strategies such as: (i) a smoothness loss combining first- and second-order penalization to enforce output continuity, and (ii) early stopping. While these measures partially address reality gaps, they introduce inherent trade-offs that antagonize estimation accuracy. For example, the smoothness loss imposes an implicit assumption that velocity estimates should exhibit temporal smoothness, a heuristic lacking causal justification under dynamic disturbances, e.g., sudden external perturbations or low-friction terrain. Similarly, early stopping introduces engineering uncertainties by necessitating a trade-off between accuracy, i.e. the number of iterations, and sim-to-real generalization, a balance that relies heavily on empirical human judgment rather than principled optimization.

5.3 Experimental Validation

Following the presentation of our RL-based state estimation methodology, we now systematically evaluate its design paradigms, experimental performance, and comparative advantages. The results section is structured as follows. First, we conduct a comprehensive analysis of the estimator's design space through simulation studies, including benchmarking against established methods referenced in Fig. 5.1. Second, we validate the framework's sim-to-real transfer capability through hardware experiments on a quadrupedal robot platform, comparing real-world performance metrics against simulation baselines.

5.3.1 Evaluation in Simulation

Before training the estimators, we first establish a baseline locomotion controller with the following attributes: stability, natural gait patterns, terrain traversability, disturbance rejection, a concise observation space, and a lightweight neural network architecture. The controller is trained using Isaac Gym [63, 79]. Building on the observation space defined in Section 5.2.1 for the estimator, we augment the controller's observation tuple with a base linear velocity field $\boldsymbol{v}_{\text{b}} = [v_x\, v_y]^\intercal \in \mathbb{R}^2$. This field remains unscaled and is perturbed with

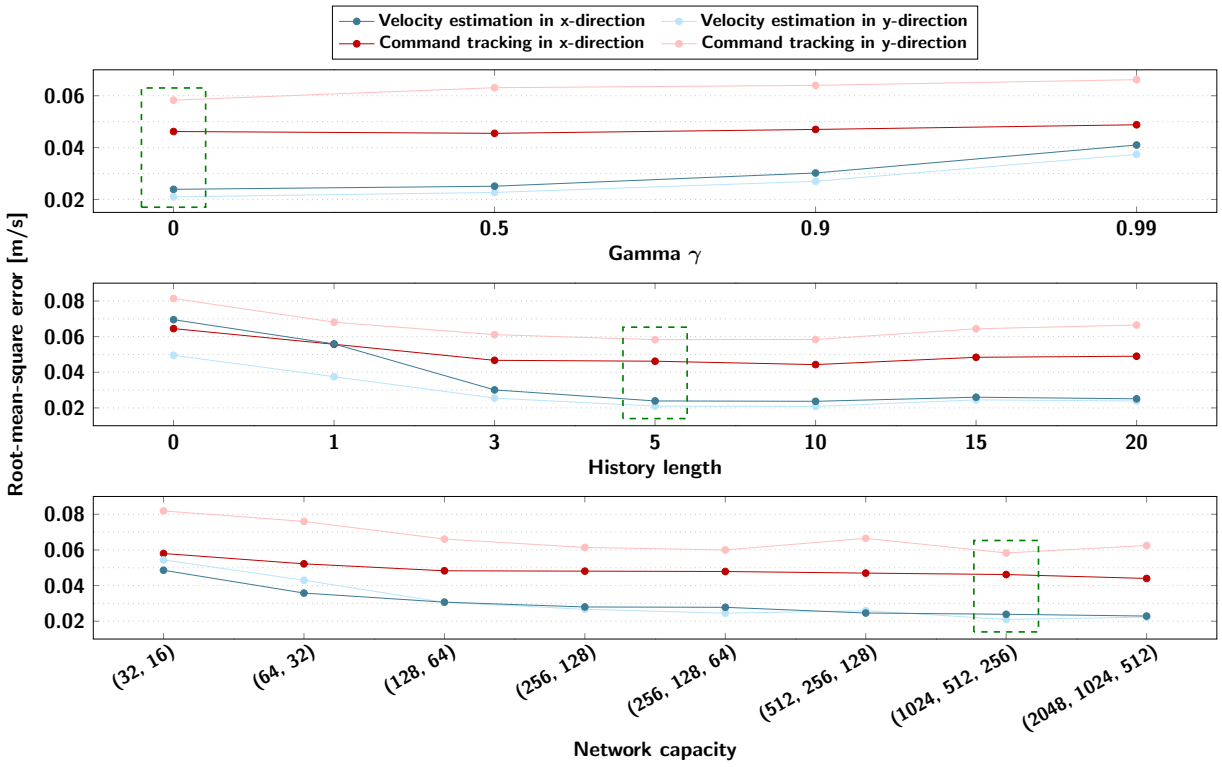


Figure 5.2: Root-mean-square (RMS) errors of both the base linear velocity output from the RL-based estimator and the command tracking performance achieved by the concatenated system against critical design aspects are presented in the above group of plots. The RMS errors are calculated using the metric mentioned in Section 5.2 from [47] for faithful comparison. Points surrounded by the dashed green box are the results from an identical baseline estimator with the following configuration: an MLP having a $(1024, 512, 256)$ structure activated by the leaky rectified linear unit (ReLU) with a negative slope of 0.01, a history length of 5 appended to the current observation and a discount factor γ of 0. All the estimators are produced against an identical baseline locomotion controller trained with the parameters mentioned in Section 5.2.

uniform noise $\mathcal{U}^2(\pm 0.05)\,\mathrm{m/s}$ to simulate sensor uncertainties. The final observation tuple for the baseline controller is

$$\boldsymbol{s}_t = (\boldsymbol{v}_\mathrm{b}, \boldsymbol{\omega}_\mathrm{b}, \boldsymbol{g}_\mathrm{p}, \boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{a})\,,$$

where $\boldsymbol{\omega}_\mathrm{b}$ is the base angular velocity, $\boldsymbol{g}_\mathrm{p}$ is the projected gravity vector in the x-y plane, $\boldsymbol{u}$ denotes user commands, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are joint positions and velocities, and $\boldsymbol{a}$ represents the previous action.

The controller employs a multi-layer perceptron (MLP) with a $(512, 256, 128)$ architecture and exponential linear unit (ELU) activations. The network processes the observations and outputs an action $\boldsymbol{a} \in \mathbb{R}^{12}$, interpreted as residual adjustments to the nominal joint positions, equivalent to the initial joint positions listed in Table 5.2. These resulting target joint positions are tracked by a proportional-derivative control subsystem with gains $K_\mathrm{p} = 20$ and $K_\mathrm{d} = 0.5$.

We adopt the adaptive curriculum learning scheme for terrain and velocity commands proposed in [79]. The terrain curriculum progressively increases difficulty: agents advance to steeper or rougher terrain tiles after achieving consistent success on simpler ones. Simultaneously, the forward velocity command curriculum initializes with a range of $[-1, 1]\,\mathrm{m/s}$ and increments by $0.5\,\mathrm{m/s}$ each time the agent's velocity tracking reward exceeds 80% of the maximum attainable reward per episode. This range is clamped at $2.5\,\mathrm{m/s}$ to limit the maximum command. For lateral motion, base linear velocity commands in the y-direction are sampled uniformly from $[-1, 1]\,\mathrm{m/s}$, while base angular velocity commands in the z-direction span $[-1.5, 1.5]\,\mathrm{rad/s}$.

The training terrain comprises a half-half mix of smooth slopes, generated via Isaac Gym's pyramid-shaped utility functions, and rough, uneven surfaces, constructed also using the built-in random uniform terrain perturbations. Although benchmark evaluations focus on flat ground, training on sloped and irregular terrain enhances robustness, agility, and gait quality, particularly improving foot clearance during swing phases which helps with the sim-to-real transfer. The reward functions and their weights for training the baseline controller are detailed in Table 5.3. The controller operates at 50 Hz, with the simulator stepping at 0.005 s intervals. All other hyperparameters align with the original implementation in [79]. A critical modification involves the initial joint positions in Table 5.2: the hip joints are adducted (bent inward) rather than the default abducted configuration. This adjustment optimizes the trade-off between static stability and dynamic agility, reducing lateral cumbersomeness during rapid maneuvers while maintaining aesthetic gaits.

With the baseline locomotion controller finalized, we construct an RL-based estimator atop this fixed controller. As outlined in Figure 5.1, the estimator's training framework follows a standard RL formulation. The reward function is exclusively the velocity tracking reward defined in Section 5.2.2, with a shaping hyperparameter $\sigma = 0.15$ to steepen the reward gradient near zero tracking error. This design choice prioritizes high-precision estimation by amplifying gradients for small deviations, which empirically improves the tracking accuracy during training.

Table 5.2: Initial joint position configurations (in rad) for training the baseline locomotion controller. The adducted hip joint configuration optimizes the trade-off between static stability and dynamic agility in quadrupedal locomotion. Values are specific to the Unitree Go1 robot platform.

Joint name	Pos.	Joint name	Pos.	Joint name	Pos.
Front-left hip	-0.1	Front-left thigh	0.8	Front-left calf	-1.5
Rear-left hip	-0.1	Rear-left thigh	1.0	Rear-left calf	-1.5
Front-right hip	0.1	Front-right thigh	0.8	Front-right calf	-1.5
Rear-right hip	0.1	Rear-right thigh	1.0	Rear-right calf	-1.5

Table 5.3: Reward function terms, expressions and weights for training the baseline locomotion controller. The tracking hyperparameter $\sigma = 0.25$ modulates the exponential reward curvature to stabilize gradient magnitudes across tracking errors in a certain range. The simulation time step $\Delta t = 0.005$s corresponds to a 50Hz control frequency with a decimation of 4. For hip joint regularization terms, initial joint position configurations align with Table 5.2.

Reward function term	Expression	Weight
Tracking base lin. vel. cmd x-y	$\exp\left[-\lVert\boldsymbol{v}_{\mathrm{cmd},xy} - \boldsymbol{v}_{\mathrm{b},xy}\rVert^2/\sigma\right]$	Δt
Tracking base ang. vel. cmd z	$\exp\left[-(\omega_{\mathrm{cmd},z} - \omega_{\mathrm{b},z})^2/\sigma\right]$	$0.5\Delta t$
Base linear velocity z	$-v_{\mathrm{b},z}^2$	$-2\Delta t$
Base angular velocity x-y	$-\lVert\boldsymbol{\omega}_{\mathrm{b},xy}\rVert^2$	$-0.05\Delta t$
Joint torque	$\lVert\boldsymbol{\tau}\rVert^2$	$-5 \cdot 10^{-5}\Delta t$
Joint velocity	$\lVert\dot{\boldsymbol{q}}\rVert^2$	$-2 \cdot 10^{-5}\Delta t$
Joint acceleration	$\lVert\ddot{\boldsymbol{q}}\rVert^2 = \lVert\dot{\boldsymbol{q}}_{\mathrm{last}} - \dot{\boldsymbol{q}}\rVert^2/\Delta t$	$-2.5 \cdot 10^{-7}\Delta t$
Feet air time	$\sum_{i=0}^{3}(t_{\mathrm{swing},i} - 0.5)$	Δt
Collision	$\sum_{i=0}^{3} c_{\mathrm{knee},i}$ for $c_{\mathrm{knee},i} \in \{0, 1\}$	$-\Delta t$
Action velocity	$\lVert\dot{\boldsymbol{a}}\rVert^2 = \lVert\boldsymbol{a}_{\mathrm{last}} - \boldsymbol{a}\rVert^2$	$-0.015\Delta t$
Action acceleration	$\lVert\ddot{\boldsymbol{a}}\rVert^2 = \lVert\dot{\boldsymbol{a}}_{\mathrm{last}} - \dot{\boldsymbol{a}}\rVert^2$	$-2 \cdot 10^{-3}\Delta t$
Hip cosmetic	$\sum_{i=0}^{3} \lvert q_{\mathrm{hip,init},i} - q_{\mathrm{hip},i}\rvert$	$-0.01\Delta t$

The estimator operates at 50 Hz, matching the inference frequency of the baseline controller to ensure temporal alignment. We retain the adaptive curriculum learning scheme for terrain and forward velocity command, with one modification: the velocity command curriculum now uses the velocity tracking reward instead of the original command tracking reward. All other parameters are inherited from [79]. By minimizing deviations from the baseline implementation, we ensure fair comparisons with prior methods while streamlining reproducibility.

Amongst the hyperparameters governing estimator training, the discount factor γ, which is often treated as a tunable parameter in proximal policy optimization yet inherently task-dependent, warrants particular attention. The training of legged locomotion controllers typically adopts $\gamma = 0.99$ as a convention reflecting the need to optimize long-horizon reward accumulation for locomotion stability. However, the estimator's objective differs fundamentally that it should prioritize instantaneous accuracy, avoiding assumptions about temporal smoothness, i.e. velocity continuity across steps. Also as shown in Fig. 5.2, we empirically evaluate γ and find that $\gamma = 0$ minimizes velocity estimation error. This result aligns with the estimator's role that unlike locomotion controllers, the estimator focuses solely on inferring current quantities. Consequently, future rewards are irrelevant, and a zero discount factor avoids conflating temporal dependencies, as justified in Section 5.2.2.

State estimation for legged robots demands richer historical context than locomotion control, as accurate inference relies on temporal patterns in sensor data. Through systematic ablation studies in Figure 5.2, we identify a history length of 5 steps as optimal, balancing sufficient temporal context with computational efficiency. Extending the history beyond this length yields diminishing returns: while estimation accuracy improves marginally, memory and processing costs grow substantially. This trade-off also reflects the capacity limits of the chosen neural architecture, an MLP with layers sized $(1024, 512, 256)$ and leaky ReLU activations (negative slope = 0.01). Though smaller networks such as $(256, 128)$ achieve comparable accuracy for the target task, the selected architecture ensures robust performance across diverse benchmarks. Larger networks such as $(2048, 1024, 512)$ showed no measurable gains, suggesting saturation in representational power.

To rigorously benchmark our framework against existing methods in Figure 5.1, we duplicate the network architectures and simulation settings, ensuring parity in model complexity and training conditions. For the concurrent controller-estimator baseline, we simplify the original design to align with our focus on base linear velocity estimation. The locomotion controller retains the baseline's core structure but replaces sensor noise in the velocity term with estimates from a supervised-learned estimator. Conversely, the estimator mirrors our proposed architecture but is trained via supervised learning alongside the controller. Both components share the adaptive terrain curriculum from our baseline, yet the concurrent framework proves incompatible with the original velocity command curriculum. Abrupt increases in velocity range induce catastrophic estimation errors, as the supervised-learned estimator fails to adapt to sudden command shifts. These errors destabilize the controller, which in turn generates anomalous observations, further degrading the estimator's performance, a feedback loop that collapses the coupled system. To mitigate this, we adopt a

Table 5.4: RMS errors from the velocity estimation and the command following tasks across selected combinations of the control system for quadrupedal locomotion are shown in the table. *RL-based* stands for the the proposed RL-based estimator concatenated with the baseline controller. *Concurrent* implies training the estimator using supervised learning and the controller using RL concurrently. *Teacher* and *student* represent the corresponding parts in the teacher-student architecture where the teacher is trained using RL with privileged observations and the student distills the teacher with practical observations.

Task	Combination	RMSE (v_x, v_y) [m/s]
Velocity estimation	*RL-based*	$(\mathbf{0.0239}, \mathbf{0.0210})$
	Concurrent	$(0.0398, 0.0549)$
Command tracking	*RL-based*	$(0.0462, 0.0583)$
	Concurrent	$(0.0439, 0.0625)$
	Teacher	$(\mathbf{0.0390}, 0.0541)$
	Student	$(0.1438, 0.1398)$
	No estimator	$(0.0788, 0.0814)$
	Asymmetric	$(0.0584, \mathbf{0.0481})$

smoothed velocity curriculum from [47], modified and defined as

$$v_{x,\max} = \min\left[0.7 + \frac{2.5}{1 + \exp\left[-0.002 \cdot (t - 1000)\right]}, 2.5\right],$$

where t denotes the training iteration. This schedule phases in higher velocities gradually, allowing the controller-estimator pair to adapt incrementally. However, the curriculum's dependence on absolute iteration counts, rather than agent performance, which introduces brittleness. Any change to hyperparameters or environmental dynamics necessitates laborious re-tuning of the schedule's coefficients, e.g., the 0.002 scaling factor, undermining reproducibility.

Learning rate selection further complicates concurrent training. A high learning rate accelerates initial estimator convergence but causes late-stage instability, as rapid updates overshoot optimal parameters, leading to inaccurate estimation. A low learning rate delays estimator adaptation, stalling controller progress, resulting in the early failure of the coupled system. Through empirical testing, we identify $1.2 \cdot 10^{-3}$ as optimal, balancing convergence speed and final accuracy. This value ensures the supervised-learned estimator evolves in lockstep with the RL controller, avoiding the divergent learning trajectories that plague extreme rates.

In the teacher-student architecture, the teacher policy (baseline controller) inherently encodes an implicit state estimator through privileged simulation observations. To ensure a fair comparison, we preserve the combined representational capacity of the original controller and estimator by horizontally merging along their network dimensions. This results in a student network with an aggregated architecture, specifically, an MLP of size $(1536, 768, 384)$,

Table 5.5: Comparison between the RL-based controller-estimator pair and the concurrently trained pair along absolute steady-state error for velocity estimation and base linear velocity command tracking in x-direction.

Task	Combination	Froward base linear velocity command [m/s]					
		1.0	1.25	1.5	1.75	2.0	2.25
Velocity estimation	*RL-based*	0.086	0.115	0.111	0.177	0.366	0.466
	Concurrent	0.045	0.012	0.031	0.009	0.177	0.210
Command tracking	*RL-based*	0.038	0.066	0.060	0.108	0.289	0.338
	Concurrent	0.183	0.164	0.193	0.173	0.035	0.053

designed to match the total parameter count of the stand-alone controller and estimator. The student employs leaky ReLU activations, chosen for their balance of gradient stability and nonlinear expressivity, to distill the teacher's policy while inferring states from practical sensor inputs. By aligning network capacity, we isolate the impact of architectural differences, ensuring observed performance variations stem from the distillation process itself rather than disparities in model complexity.

5.3.2 Real-world Verification

We validate the proposed RL-based estimator framework on a Unitree Go1 quadrupedal robot. The Go1 features 18 degrees of freedom: 12 actuated joints for leg control and 6 generalized coordinates defining the floating base's position and orientation. To establish ground truth reference data, we employ an OptiTrack Prime Series motion capture (mocap) system, which provides millimeter-level positional accuracy. The mocap system captures the robot's base position and orientation at 200 Hz, enabling precise benchmarking against the estimator's output. Detailed experimental results, including estimation errors and real-world performance metrics, are presented in Table 5.5.

We evaluated our framework's performance via forward velocity tracking, a key metric for legged locomotion. The robot executed step commands of $\{1.0, 1.25, 1.5, 1.75, 2.0, 2.25\}$ m/s, covering nominal to high-speed regimes within the training range $[-2.5, 2.5]$ m/s. Each trial ran until the robot exited the motion capture system's valid area. Tests were performed on foam puzzle flooring ($\mu \approx 0.52$) to simulate moderate terrain while limiting slippage. A key challenge arose from the interplay between velocity magnitude and testing room constraints. At higher velocities, the robot's acceleration phase consumed significant distance, leaving limited room length for steady-state measurement. To ensure fair comparison across velocities, we prioritized steady-state error analysis, discarding transient data during acceleration. This approach isolates the controller-estimator performance under stable operating conditions, mitigating biases introduced by spatial limitations.

For real-world comparisons, both frameworks share identical training parameters, differing only in training concurrency and curriculum design. The concurrent framework jointly

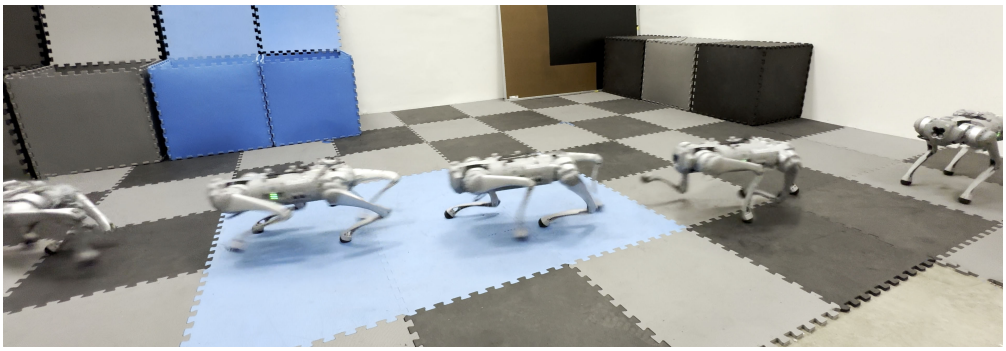


Figure 5.3: The quadruped is following a velocity command in the motion capture room.

trains the controller and estimator with a smoothed velocity curriculum, while the RL-based approach decouples them. This isolates performance disparities to architectural coupling: concurrent training may amplify the reality gap (e.g., overfitting in simulation worsening real-world instability), whereas the decoupled RL framework limits error propagation, prioritizing controller robustness despite estimation noise.

Table 5.5 shows two performance regimes based on velocity: nominal cruise ($< 2\,\mathrm{m/s}$) and extreme ($\geq 2\,\mathrm{m/s}$). The $2\,\mathrm{m/s}$ threshold, near the $2.5\,\mathrm{m/s}$ training limit, marks a point where both systems begin to degrade due to sim-to-real gaps. In the nominal range, the concurrent system yields better estimation, but the RL-based framework achieves more accurate tracking. This reflects a core trade-off: concurrent training favors estimation precision, while the RL approach prioritizes tracking fidelity despite noisier estimates. At extreme velocities, both systems struggle due to untrained commands. The RL-based framework shows sharp drops in estimation and tracking, revealing limits in policy extrapolation. In contrast, the concurrent system yields stabler tracking but worse estimation, as the controller aggressively compensates for poor estimates. This masks instability but compromises estimation integrity, leading to sporadic failures during high-speed operation.

These results underscore the interplay between estimation accuracy and control robustness. While concurrent training excels in narrow, simulation-aligned regimes, its fragility under extrapolation stresses the value of modularity in real-world systems. The RL-based framework's resilience, maintaining stable tracking despite estimation noise, validates its suitability for dynamic environments where operational conditions frequently exceed training boundaries.

The RL-based framework achieves accurate velocity tracking across the nominal range, with average steady-state errors of $0.068\,\mathrm{m/s}$. This stems from its decoupled training: the controller is optimized independently to track commands, unaffected by estimator errors. Operating in closed loop, it also leverages raw sensor inputs (e.g., encoders, IMU), allowing proprioceptive feedback to compensate for noisy estimates and maintains robust performance.

In contrast, the concurrent system shows persistent tracking errors averaging $0.178\,\mathrm{m/s}$,

despite better estimation accuracy. This stems from interdependent training: the controller over-relies on the estimator, which overfits to simulation dynamics. Using extended history (5 steps) further widens the sim-to-real gap, as real-world noise and latency disrupt learned correlations. As a result, the controller's reliance on idealized estimates leads to degraded tracking performance.

Notably, estimation errors do not uniformly degrade tracking performance. In the RL-based system, discrepancies between estimated and ground-truth velocities are mitigated by complementary sensor inputs. For example, joint encoder readings provide direct mechanical feedback, allowing the controller to adjust leg trajectories without relying solely on velocity estimates. This redundancy highlights the value of modular design: decoupling estimation and control allows each component to leverage domain-specific sensor modalities, enhancing overall system resilience.

5.4 Summary

We presented a modular legged robot state estimation framework using RL, decoupling estimator training from controller design by treating the controller-simulator combination as a unified environment. This approach enables independent estimator optimization while preserving the controller's sim-to-real robustness. Experiments in simulation and the real world demonstrated improved command tracking, constant inference complexity, and seamless integration with existing controllers. The framework simplifies estimator development and highlights the potential of modular, RL-driven solutions for advancing autonomous legged systems.

With robust proprioceptive estimation and torque control now in place, the robot is stable but remains fundamentally reactive; it can survive rough terrain but cannot anticipate it. To enable the traversal of discrete geometric obstacles like stairs, we must move beyond proprioception and integrate exteroceptive sensing. Chapter 6 addresses this final piece of the puzzle.

Chapter 6

Cross-modality Distillation for Perceptive Locomotion

Perceptive locomotion traditionally relies on explicit elevation maps constructed from fused depth data, a pipeline that imposes significant hardware complexity and computational overhead. This chapter presents a novel cross-modality distillation framework that replaces this dependency with a single monocular RGB image. We utilize a diffusion transformer architecture to distill the geometric reasoning of an elevation map-based teacher into an image-based student policy. This approach enables the robot to map raw visual observations directly to robust control actions, effectively eliminating the need for depth sensors and online map fusion. We demonstrate through simulation on a humanoid robot that our distilled policy successfully traverses complex terrain, such as stairs, matching the capabilities of its teacher policy. This work establishes a streamlined paradigm for perceptive locomotion, paving the way for universal, vision-centric robotic perception.

6.1 Introduction

The field of legged robotics has witnessed remarkable progress in recent years, with state-of-the-art systems demonstrating increasingly dynamic and agile locomotion. These platforms perform complex tasks such as mimicking human motions including even dancing, showcasing a significant leap in control paradigms [35, 59]. However, these impressive skills are predominantly validated in structured or controlled environments. For legged robots to fulfill their potential in real-world applications, they must transition from demonstrating in mild environments to dealing with challenging and random conditions. This requires a fundamental capability known as perceptive locomotion, i.e. the robot actively perceives using exteroceptive sensors, and adapts its gait and posture to complex terrains in real-time [65, 32].

A primary barrier to the widespread deployment of perceptive locomotion is its reliance on costly and fragile exteroceptive sensors. Conventional solutions, such as stereo vision and

lidar, present critical limitations that are exacerbated on dynamic legged platforms. Stereo vision, while cost-effective, suffers from performance degradation at longer ranges and in low-texture environments, leading to incomplete or noisy depth maps. To bridge the resulting sim-to-real gap, the simulated depth maps often require destructive post-processing that sacrifices fine geometric details [2, 18]. Lidar systems offer more precise depth information but are often prohibitively expensive, power-intensive and mechanically fragile. Furthermore, on a dynamically moving robot, the precise and continuous calibration of these sensors is critical; even a small absolute error in the estimated terrain height can translate into a significant relative error for the robot's leg-length-to-body ratio, potentially causing instability or failure. The computational burden of processing and fusing these sensor streams to build a consistent world map also adds latency and complexity [26, 27, 39].

In light of these challenges, monocular RGB cameras emerge as a compelling alternative. They are affordable, lightweight, ubiquitous and power-efficient. When strategically positioned, they capture a rich stream of information that often encapsulates not only the geometric properties of the environment but also vital semantic cues necessary for navigation that other sensors must infer through complex algorithms. The viability of a vision-centric approach is strongly supported by its success in other domains, most notably in the automotive industry, where systems like self-driving vehicles have demonstrated that camera-only perception can be sufficient for complex planning and navigation tasks [12]. Adopting a vision-based policy significantly simplifies the hardware stack, avoids the computational burden of dense depth estimation and map fusion, and generally simplifies system maintenance and deployment.

From a policy learning perspective, an image-based controller offers profound practical advantages [61]. It allows researchers to bypass the taxing procedure of setting up and calibrating specialized sensor suites, enabling faster and more robust deployment. Furthermore, visual input provides a natural and flexible interface for future additive integration of new skills. As new tasks, potentially guided by other modalities, are developed, the same distillation techniques can be employed to create an image-based version of the policy, avoiding redundant sensors and software stacks.

In parallel, the recent emergence of diffusion models [37, 78, 70] provides a powerful generative framework suited to this challenge. In robotic manipulation, diffusion policies have demonstrated a particular aptitude for fusing high-dimensional image inputs with low-dimensional proprioceptive states to generate coherent, multi-modal action sequences [19, 10]. This capacity for heterogeneous fusion is pivotal for tasks like grasping in cluttered environments [93]. We postulate that this paradigm is equally suited for the challenge of perceptive legged locomotion. Navigating complex, unstructured terrain requires precisely this multi-modal reasoning: exteroceptive perception to anticipate terrain properties and plan secure footholds, seamlessly combined with proprioceptive feedback for dynamic stability and gait execution. While previous approaches in locomotion have often relied on carefully engineered perception pipelines or privileged information [32], diffusion policies offer a promising end-to-end framework to directly map visual and proprioceptive observations to terrain-aware locomotion behaviors.

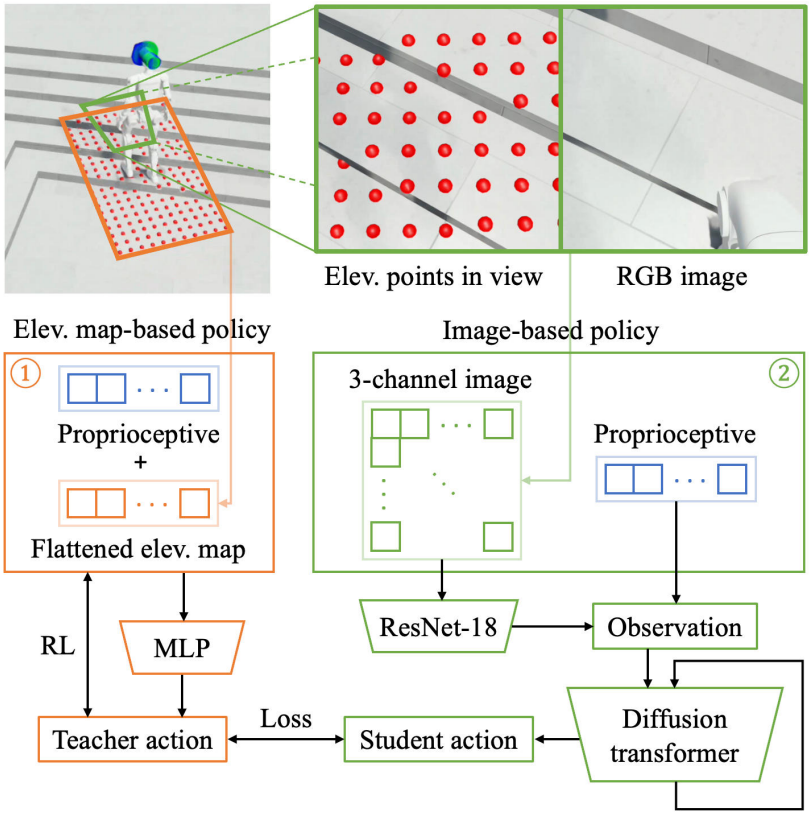


Figure 6.1: Architecture of the proposed cross-modality distillation method. The system consists of two stages: (1) an elevation map-based teacher perceptive locomotion policy trained via RL, which takes a local height map and a proprioceptive state as the input to an MLP, and (2) an image-based student that employs a diffusion transformer to encode the policy. At each time step, the student processes an RGB image through a ResNet-18 encoder, concatenates the latent space to a proprioceptive state, feed the combined observation to a diffusion transformer, and attempts to match the teacher's action through supervised learning. The distillation effectively eliminates the need for expensive depth hardware and computationally intensive fusion algorithms.

6.1.1 Contributions

In this work, we explore cross-modality distillation to achieve perceptive locomotion, specifically transferring skills from a teacher policy that uses temporally fused elevation maps to a student policy that relies only on monocular RGB images. To this end, we first train a reinforcement learning teacher policy that generates emerged and terrain-adaptive gaits for traversing complex environments like stairs based on simulated height points. We then collect demonstrations from this teacher policy and distill its capabilities into a student policy, implemented with a diffusion transformer architecture, that consumes only RGB images and proprioceptive data. The resulting image-based policy maintains the performance of the original elevation map-based controller while eliminating the dependency on costly and computationally expensive lidar sensor stacks, marking a significant step toward practical deployment.

We summarize our contributions as follows.

- We propose a novel cross-modality distillation pipeline that effectively transfers perceptive locomotion skills from an elevation map-based policy to an image-based policy, leveraging a diffusion transformer architecture as the core of the student model.
- We demonstrate that the resulting image-based policy enables successful stair traversal for a humanoid robot, matching the performance of the teacher policy while relying solely on a single RGB camera.
- Our method provides a scalable framework for modality consolidation. By converting policies dependent on specific sensors to a universal image-based input, our approach facilitates the additive integration of new skills and simplifies the hardware stack and computationally intensive fusion algorithms for deployment in complex and real-world environments.

6.2 Distillation Pipeline

This section details our approach for distilling perceptive locomotion behaviors from a policy that uses elevation maps as an input to one that relies solely on RGB images. Figure 6.1 illustrates the overall pipeline, which involves constructing an elevation map-based policy, transforming it into an image-based policy, and establishing a foundation for future integration with other modalities.

6.2.1 Elevation Map-based Locomotion Policy

We begin by constructing a high-quality demonstration policy in simulation. Adopting the elevation map configuration from [79] for the ANYmal C quadrupedal robot, we attach a local ground elevation map to a blind walking policy's observation space for the Unitree G1 humanoid robot. The map spans $1.6\,\mathrm{m} \times 1\,\mathrm{m}$ with the x-axis pointing forward in a

Table 6.1: Detailed descriptions of the observation terms, expressions and noises added to the ground truth values. All observation terms remain unscaled before being input to the neural network. All intervals of the uniform noises are symmetric about zero. The elevation map term has an original structure of $\boldsymbol{e}_{\mathrm{o}} \in \mathbb{R}^{17\times 11}$, covering a $1.6\,\mathrm{m} \times 1\,\mathrm{m}$ area around the robot. During the sampling process, the true projected position in the x-y plane of this area is offset by noise $\mathcal{U}^2(\pm 0.3)\,\mathrm{m}$ in both directions. In addition to the x-y plane projection offset, noise listed in the table is then added to the z-axis. Finally, the array $\boldsymbol{e}_{\mathrm{o}}$ is flattened to $\boldsymbol{e} \in \mathbb{R}^{187}$ and appended to the observation space.

Obs. term	Expression	Noise
Base ang. vel.	$\boldsymbol{\omega}_{\mathrm{b}} = [\omega_x\, \omega_y\, \omega_z]^\intercal \in \mathbb{R}^3$	$\mathcal{U}^3(\pm 0.2)\,\mathrm{rad/s}$
Proj. gravity	$\boldsymbol{g}_{\mathrm{p}} = [g_x\, g_y\, g_z]^\intercal \in \mathbb{R}^3$	$\mathcal{U}^3(\pm 0.05)\,\mathrm{rad/s^2}$
User cmd	$\boldsymbol{u} = [v_x\, v_y\, \omega_z]^\intercal \in \mathbb{R}^3$	0
Joint position	$\boldsymbol{q} = [q_1 \cdots q_{29}]^\intercal \in \mathbb{R}^{29}$	$\mathcal{U}^{29}(\pm 0.01)\,\mathrm{rad}$
Joint velocity	$\dot{\boldsymbol{q}} = [\dot{q}_1 \cdots \dot{q}_{29}]^\intercal \in \mathbb{R}^{29}$	$\mathcal{U}^{29}(\pm 1.5)\,\mathrm{rad/s}$
Last action	$\boldsymbol{a} = [\hat{q}_1 \cdots \hat{q}_{29}]^\intercal \in \mathbb{R}^{29}$	0
Elev. map	$\boldsymbol{e} = [e_1 \cdots e_{187}]^\intercal \in \mathbb{R}^{187}$	$\mathcal{U}^{187}(\pm 0.1)\,\mathrm{m}$

right-handed coordinate system, and is always expressed in the robot's base frame. With a resolution of 0.1 m, the map comprises 187 discrete ground elevation points.

In simulation, the elevation map is initially sampled in the global frame using Isaac Sim's height scanner [67] and then transformed into the robot's base frame, i.e. the lidar frame for the G1. A constant offset along the z-axis is then applied to normalize the map by shifting the points to have a mean elevation of nearly zero, under a normal walking gait.

We base our observation and action spaces on the composition for the G1 humanoid robot introduced in [96], chosen for its compatibility and representativeness in legged locomotion. The observation vector includes:

- Base angular velocity, $\boldsymbol{\omega}_{\mathrm{b}} \in \mathbb{R}^3$,
- Unit projected gravity, $\boldsymbol{g}_{\mathrm{p}} \in \mathbb{R}^3$,
- User command, $\boldsymbol{u} \in \mathbb{R}^3$ with the target base linear velocity along x- and y-axes, and the angular velocity along z-axis,
- Joint position, $\boldsymbol{q} \in \mathbb{R}^{29}$,
- Joint velocity, $\dot{\boldsymbol{q}} \in \mathbb{R}^{29}$,
- Previous action, $\boldsymbol{a} \in \mathbb{R}^{29}$,
- Flattened elevation map points, $\boldsymbol{e} \in \mathbb{R}^{187}$.

All terms are in the robot's base frame which coincides with the IMU frame. Comprehensive definitions, dimensions, scale factors and noise descriptions for these observations are provided in Table 6.1. We use only the current observation step without history, as the blind policy shows this configuration's sufficient robustness as well as the spatial coverage of the elevation map provides enough information for determining an action. This design also provides a clear target for the information that must be captured by the image-based policy during distillation.

The observation vector is processed by a single-layer long short-term memory network with a hidden size of 256, followed by an multi-layer perceptron with layers of size $[256, 256, 128]$. The output action $\boldsymbol{a} \in \mathbb{R}^{29}$ stands for residual values added to the default joint positions, with the resulting target positions tracked by a proportional-derivative controller. The default joint positions are determined empirically at a neutral standing pose for the G1 humanoid robot, and used by the base blind walking policy on flat ground.

6.2.2 Cross-Modality Distillation via Vision-Action Model

Given that the target modality for distillation is a single RGB camera, its placement is critical. The field of view must capture enough of the ground to provide evidence for inferring correct actions. We position the camera 10 cm in front of the chest along the x-axis, pitched 65° downward. This placement is depicted in Figure 6.2 in the side view.

In Isaac Sim, where focal length, aperture size and offset are set as one-tenth of the world unit, the camera is configured with an 18 cm focal length, a 400 m focus distance (effectively infinity), and a horizontal aperture of 20.955 cm (equivalent to a 35 mm spherical projector). The camera captures 96×96, 3-channel RGB images at 8-bit color depth and a frequency of 50 Hz, matching the policy control frequency.

During data collection, we isolate 50 agents across 50 parallel environments to prevent them from appearing in each other's camera views, as Isaac Sim renders all visible objects without exclusion. Simulated images are collected from the virtual RGB cameras rigidly attached to each agent as described above.

Vision-action models, which excel at combining visual and proprioceptive data, have shown superb performance in robot manipulation. We adopt the transformer-based diffusion network [19] to encode our image-based policy. To ensure a fair distillation, we include all proprioceptive observation terms from Table 6.1 except the elevation map. At time step t, these terms are concatenated into a vector $O_{t,\mathrm{p}} \in \mathbb{R}^{96}$.

A captured 8-bit RGB image $O_{t,\mathrm{i}} \in \mathbb{R}^{96\times96\times3}$ is first linearly rescaled to range $[0, 1]$ and then passed to a ResNet-18 visual encoder (without pre-training, and with specific modifications introduced in [19]). The resulting latent embedding $O_{t,\mathrm{l}}$ is merged with the proprioceptive vector $O_{t,\mathrm{p}}$ to form the total observation O_t. This is passed into a transformer architecture with 8 layers, 4 attention heads, and 256 embeddings, which outputs a horizon of 4 action steps. The diffusion policy uses 10 denoising steps during both training and inference, with only the first action step used for execution.

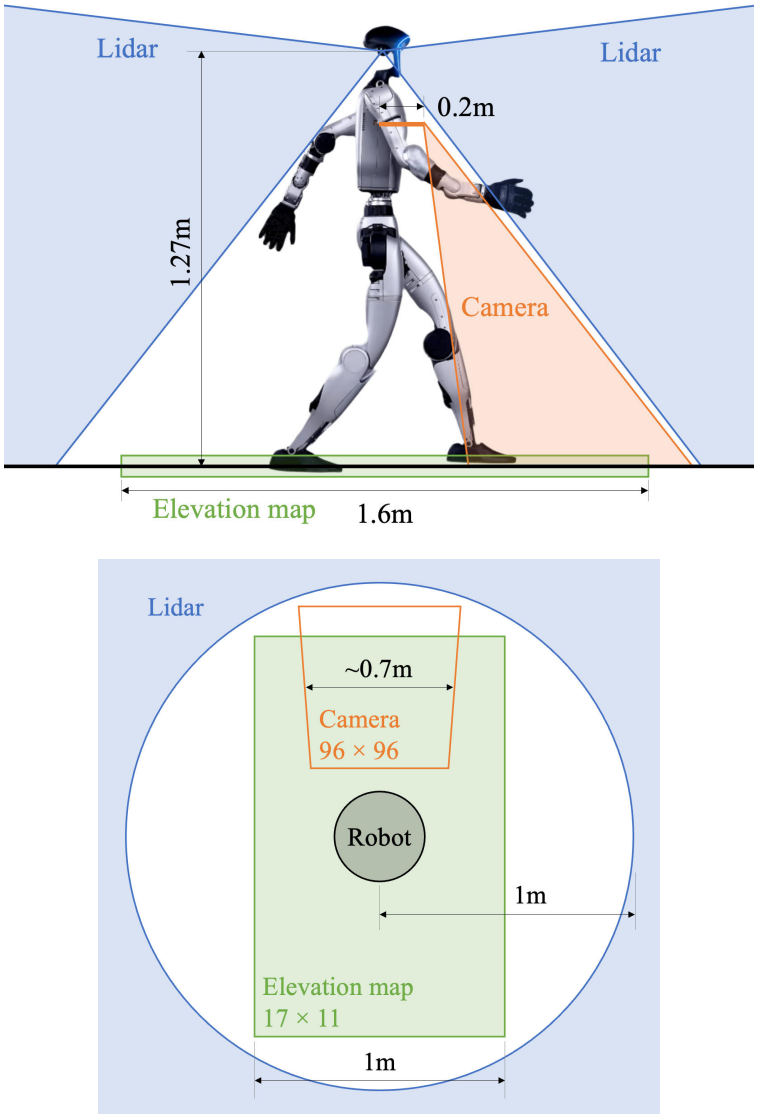


Figure 6.2: Spatial coverage comparison among sensing modalities. This diagram contrasts the field of view and ground coverage between the sensor inputs used for the elevation map-based and the image-based policies. The side view illustrates the vertical mounting and pitch of the sensors on the Unitree G1 robot; the lidar (blue cones) offers a broad range, whereas the RGB camera (orange cone) is mounted on the chest and pitched downward to capture the immediate terrain in the robot's forward path. The top-down view depicts the specific ground footprint of each modality relative to the robot's center. The elevation map (green rectangle), synthesized temporally using a sequence of lidar readings, spans an area of 1.6 m × 1 m. In contrast, the RGB camera input (orange square) captures a subset of this area, covering a trapezoid with a nominal width of approx. 0.7 m directly in front of the robot. The distillation process replaces the original elevation map with a vision-centric input covering a more restricted area.

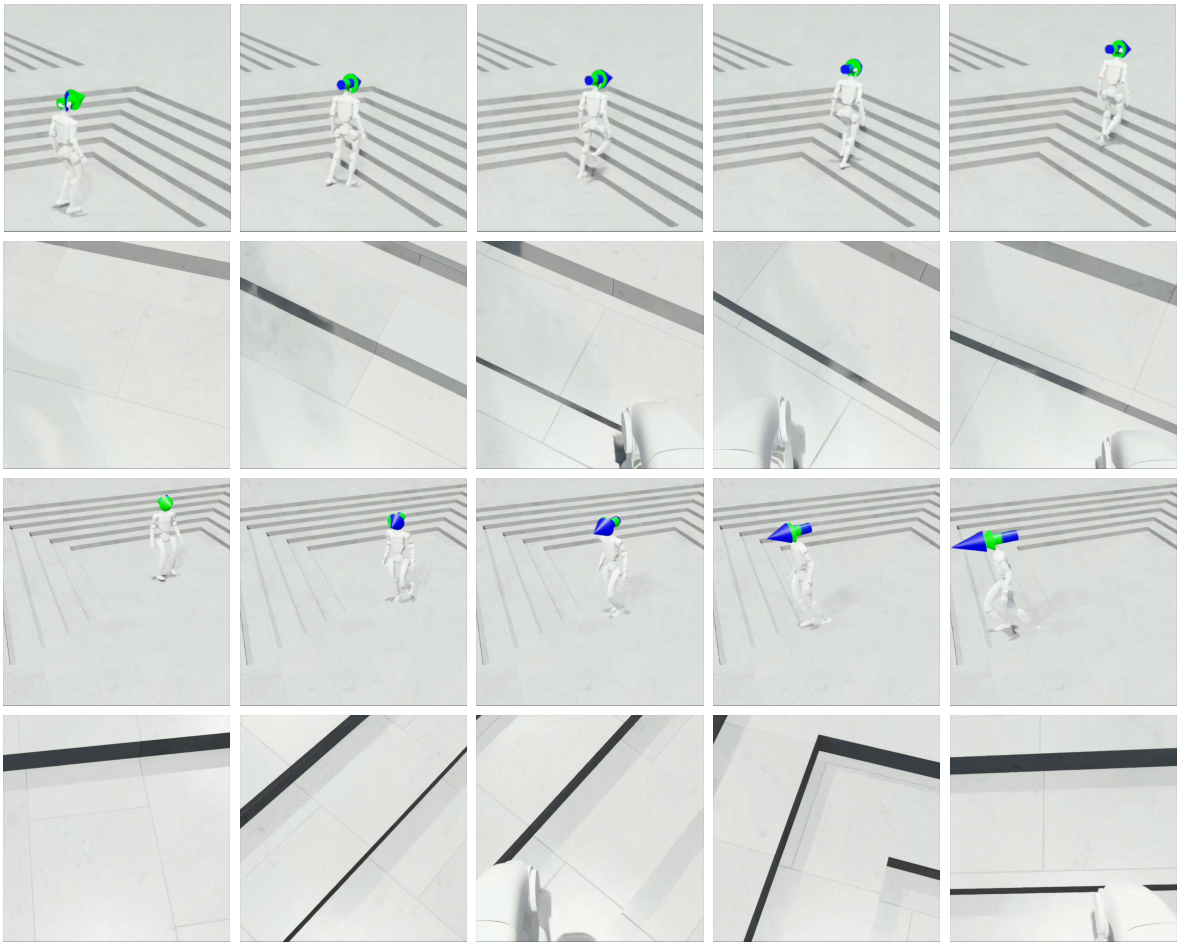


Figure 6.3: Sequential snapshots of the humanoid robot transitioning from flat ground to stairs using the diffusion transformer policy. The agent is commanded to steer toward a random heading using a constant forward velocity and variable yaw rate to approach the staircase. The first and third rows illustrate the third-person external view of the robot's motion, while the second and fourth rows display the corresponding first-person RGB camera inputs used by the policy to navigate the terrain.

We found that the camera's field of view, particularly its pitch angle, is crucial. If the camera does not see terrain close to the robot, the images lack the information needed to accommodate immediate terrain conditions. Unlike the lidar-based elevation map, which provides a 360° view, the camera only covers the front. Consequently, during data collection, we restrict the robot's motion: we provide only a forward velocity command along x-axis and a yaw rate command, setting the lateral velocity along y-axis to zero. This prevents the robot from walking backward or sideways, ensuring its motion aligns with the camera's frontal view. The robot is initialized with a random heading, and a PD controller generates yaw rate commands to steer it toward a random target heading.

6.3 Results

To validate the efficacy of the cross-modality distillation pipeline, we designed a sequential terrain traversal task in the simulation environment. The objective was to evaluate the robot's ability to transition seamlessly between distinct locomotion modes driven solely by visual information.

A G1 humanoid robot is initialized in a standing posture on flat ground surrounded by ascending stairs. The agent is commanded to follow a random heading angle with a constant forward velocity and a changing yaw rate to walk toward a staircase, requiring it to traverse a flat phase before facing a sudden geometric discontinuity. Upon reaching the stairs, the policy must autonomously detect the terrain change and adjust its foot clearance to ascend the steps without tripping.

We selected this plane-to-stair transition scenario because it serves as a representative test for perceptive locomotion. It forces the policy to conduct more than mastering single gaits, instead, it requires dynamic switching and interpolating between walking modes, particularly in an active fashion. This distinguishes a robust and terrain-aware policy from one trained using single or randomized terrains thus passive terrain adaptation.

As illustrated in Fig. 6.3, the image-based policy replicates the adaptive behavior of the teacher. The image sequence demonstrates the robot traversing the flat ground (columns 1-2) and seamlessly transitioning to stair climbing (columns 3-5). The image-based policy detects the visual features of the stairs before physical contact is made. This visual anticipation allows the robot to prepare its gait in advance.

It is also important to contrast the spatial coverage of the two modalities. While the teacher utilizes a broad, omnidirectional elevation grid depicted as red dots in Fig. 6.4, the student's image-based input captures only the immediate terrain ahead. This corresponds to approximately the front 27% of the teacher's map (50 out of 187 points). The successful transfer of skills despite this constrained view suggests that the critical features for stair traversal are concentrated entirely in the robot's forward path.

We identified the camera's pitch angle as a decisive factor in the distillation process. An excessively high pitch creates a blind spot directly in front of the robot, preventing the policy from assessing the immediate terrain. To achieve perceptive locomotion using only

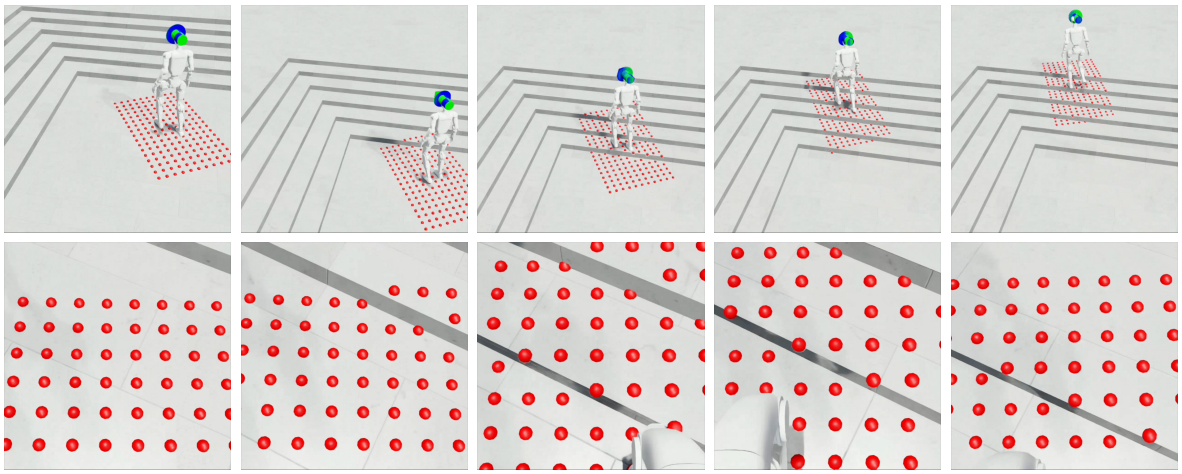


Figure 6.4: Simulation sequence of the G1 robot transitioning from flat ground to stairs. The top row presents a third-person external view, while the bottom row shows the corresponding first-person view from the chest-mounted RGB camera. Red dots represent the sampled elevation map points used by the teacher policy. Notably, the camera's field of view captures only a subset of the elevation map (approximately the front 50 of total 187 points) according to Fig. 6.2; our results demonstrate that this forward-looking coverage is heavily partial yet sufficient for robust locomotion.

single-step observations, the camera must be oriented to capture the impending foothold stair rather than focusing solely on the horizon. We deliberately excluded temporal history from the network architecture to validate the hypothesis that immediate visual perception is sufficient for terrain adaptation; while future work may incorporate memory for long-horizon tasks, our results prove it is not strictly necessary for this domain. Furthermore, to determine the minimal information threshold required for distillation, we restricted the camera's vertical field of view (FOV) to just 30°. This was found to be sufficient for robust performance and is well within the capabilities of standard hardware, such as the RealSense D435i, which offers a broader 42° vertical FOV.

Our results indicate that the image-based policy has learned to differentiate between terrain contexts. The robot maintains an energy-efficient and low-clearance gait on the flat ground and switches to a high-clearance and stability-focused gait only when the stairs are visually identified. The student policy adapts its control strategy in real-time, proving that the elevation map-based pipeline can be effectively compressed into an image-only modality with equivalent performance.

6.4 Summary

We have demonstrated that cross-modality distillation can successfully replace a complex elevation-mapping pipeline for perceptive locomotion with a single RGB camera. Our method trains a vision-based policy via a diffusion transformer to match the performance of a teacher policy that relies on privileged lidar data. In simulation and real-world experiments, the resulting policy enabled a humanoid robot to traverse complex terrain like stairs, proving it learned to adapt its gait from visual cues rather than just imitating a single skill.

This work confirms that robust, terrain-aware locomotion is achievable with a low-cost, vision-only sensing modality, eliminating the need for expensive and computationally intensive depth sensors. It establishes a more practical and scalable paradigm for deploying legged robots in real-world environments and provides a framework for consolidating future robotic skills under a universal visual representation. Future work will focus on extending the policy's perceptual field for omnidirectional navigation and generalizing to a broader range of unstructured terrains.

By successfully replacing heavy sensor stacks with a single RGB camera, this chapter completes our suite of learning-based solutions. Chapter 7 concludes the dissertation by synthesizing these contributions across aerial and legged domains and outlining the path forward for autonomous systems.

Chapter 7

Conclusion and Future Work

The deployment of autonomous robots into the wild represents one of the grand challenges of modern engineering. Unlike the structured, predictable confines of a factory floor or a laboratory, the real world is chaotic, noisy and unyielding. As demonstrated throughout this dissertation, traditional control and estimation pipelines, while mathematically elegant, often lack the plasticity required to survive in these environments. Rigidity in control leads to failure on compliant terrain; reliance on precise sensors leads to failure in GPS-denied zones; and tight coupling between modules leads to systems that are brittle to engineer and scale.

This dissertation has argued for a paradigm shift toward learning-based robustness. By integrating deep reinforcement learning (RL) and data-driven perception into the core control loops of aerial and legged robots, we have demonstrated that autonomy can be made efficient, modular, and resilient. We have moved from the stiff paradigms of position tracking and dense mapping to the compliant paradigms of torque control and visual distillation.

7.1 Summary of Contributions

This work has presented four distinct yet interconnected frameworks that support robot autonomy:

- **Real-Time Aerial Autonomy in GPS-Denied Environments:** In Chapter 3, we addressed the vulnerability of unmanned aerial vehicles (UAVs) to signal loss. We introduced a stand-alone, image-based geo-localization framework capable of recovering a vehicle's 6-degree-of-freedom pose using a database of synthetic views rendered from satellite imagery. By achieving onboard inference at 1.1 Hz with an accuracy of approximately 2.8 m, this system eliminates the reliance on heavy ground stations. Crucially, by accommodating non-nadir camera views, we unlocked the ability for UAVs to perform localization and obstacle avoidance simultaneously, paving the way for truly independent long-range flight.

- **End-to-End Learning Torque Control:** In Chapter 4, we challenged the prevailing reliance on position-based control in deep reinforcement learning. We demonstrated that by allowing an RL policy to output joint torques directly at high frequency, we could bypass the inherent stiffness of low-level proportional-derivative loops. The resulting policy endowed a quadrupedal robot with the physical compliance necessary to traverse deformable terrains such as air mattresses and loose pebbles, and resist significant external impacts, behaviors that remained elusive for rigid position-based baselines.
- **Modular State Estimation via RL:** In Chapter 5, we tackled the engineering bottleneck of system modularity. We proposed a framework that decouples state estimation from control by training the estimator as an independent RL agent against a fixed controller. This methodology breaks the cycle of mutual retraining inherent in concurrent or teacher-student architectures. We showed that this modular approach not only simplifies the design pipeline but also yields superior velocity tracking performance on real hardware, proving that estimators can be both performant and plug-and-play.
- **Perceptive Locomotion:** Finally, in Chapter 6, we addressed the hardware barriers to perceptive autonomy. We introduced a cross-modality distillation pipeline that transfers the geometric reasoning of an elevation map-based teacher to a image-only student. Utilizing a diffusion transformer architecture, we enabled a humanoid robot to navigate stairs using a single monocular RGB camera in simulation. This contribution significantly lowers the barrier to entry for perceptive robotics, replacing depth sensors which are often prone to mechanical fragility (e.g., lidar), measurement artifacts (e.g., stereo depth camera), and significant computational overhead for map fusion, with ubiquitous, low-cost imaging hardware.

7.2 Future Work

While this dissertation has addressed specific hurdles in perception and control, it opens new avenues of inquiry. The transition to more autonomous and general-purpose robots will require expanding these frameworks in several key directions.

7.2.1 Unified Multi-modal Models and Additive Skill Acquisition

In Chapter 6, we utilized a diffusion transformer to distill perception for a specific locomotion task, effectively collapsing a complex sensor stack into a single vision-based policy. The logical evolution of this work is the development of robotic foundation models that leverage this unified observation space for continuous and additive learning. Recent works such as RT-1 [14], RT-2 [105], and Octo [68] have demonstrated that policies can ingest visual, proprioceptive, and textual data to control manipulators. By standardizing the input modality to raw images, the distilled policy not only simplifies hardware implementation but also

facilitates robust deployment in diverse environments, allowing the robot to encounter and learn from a wider variety of terrains beyond the training simulation.

This vision-centric unification enables the seamless integration of downstream capabilities, such as visual localization and semantic navigation. Unlike traditional pipelines that segregate these tasks, a unified model can ingest demonstrations from heterogeneous sources, ranging from high-level algorithmic planners to human demonstrations provided via teleoperation. Crucially, because these diverse data streams share the same image-only format as the barebone perceptive locomotion policy, training becomes additive rather than repetitive. New skills can be layered onto the existing policy without the necessity of retraining from scratch, preventing catastrophic forgetting and allowing the robot's repertoire of skills to grow continuously.

Furthermore, this framework supports a reduction in computation. During the initial data collection or training phase, the robot may utilize a temporary, compute-heavy setup to run expert planners, process dense lidar maps, or fuse multi-modal sensor data. However, once these skills are successfully distilled into the efficient image-based student policy, the computational overhead required by the teacher is discarded. This process progressively frees up the onboard computational budget, reserving resources for the distillation of subsequent, more complex behaviors. In this additive manner, the system not only maximizes data efficiency but also evolves toward a more intelligent and computationally optimized robotic platform.

7.2.2 Semantic Navigation and Reasoning

The frameworks presented in this dissertation focus largely on the mechanics of movement, staying upright, estimating velocity, and finding footholds. The next step is to integrate and offer building blocks for semantic reasoning. A robot should not only perceive that the terrain ahead is stairs (geometry) but also understand that stairs lead to a door (semantics). Integrating the robust low-level torque controllers developed in Chapter 4 with high-level semantic planners, such as those leveraging visual-language maps [43] or affordance-based reasoning [13], would allow robots to execute complex, long-horizon missions, e.g., search the second floor for survivors, rather than just tracking velocity commands.

7.2.3 Heterogeneous Aerial-legged Collaboration

While this dissertation has advanced the autonomy of aerial and legged platforms individually, the synergistic potential of operating them in concert remains a frontier for robust exploration. Future research should investigate heterogeneous teams where the unique capabilities of each morphology are leveraged to overcome the limitations of the other. For instance, the aerial geo-localization framework developed in Chapter 3 could function as a high-altitude scout, generating global semantic maps and identifying traversable corridors in GPS-denied environments. This global context could then be consumed by the legged

agent, which utilizes the torque-based control and perceptive distillation frameworks from Chapters 4 and 6 to navigate the specific local terrain complexities.

By integrating the wide-area situational awareness of aerial platforms with the physical interaction capabilities of legged robots, we can develop systems capable of complex, multi-stage missions such as disaster response or remote exploration that neither platform could achieve in isolation. This collaborative autonomy represents the final step in moving from isolated proficient agents to coordinated, intelligent robotic ecosystems.

7.3 Summary

This dissertation has explored methods to enhance the robustness of autonomous systems operating in unstructured environments, advocating for a shift toward learning-based frameworks. By addressing specific challenges in aerial geo-localization, compliant torque control, modular state estimation and vision-based perception, this work aims to contribute to the development of more resilient and adaptable robotic platforms. The presented results suggest that reducing reliance on rigid control structures and expensive sensor suites can facilitate practical deployment.

Future research may build upon these foundations to investigate unified multi-modal learning and semantic reasoning, as well as the collaborative potential of heterogeneous robotic teams. Ultimately, this work aspires to support the continued advancement of autonomous agents capable of operating effectively in the real world.

Bibliography

[1] Karen E Adolph et al. "How do you learn to walk? Thousands of steps and dozens of falls per day". In: *Psychological science* 23.11 (2012), pp. 1387–1394.

[2] Ananye Agarwal et al. "Legged locomotion in challenging terrains using egocentric vision". In: *Conference on robot learning*. PMLR. 2023, pp. 403–415.

[3] Ayush Agrawal et al. "Vision-aided dynamic quadrupedal locomotion on discrete terrain using motion libraries". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 4708–4714.

[4] Andrian et al. *earth-reverse-engineering*. `https://github.com/retroplasma/earth-reverse-engineering`. 2021.

[5] Rika Antonova et al. "Reinforcement learning for pivoting task". In: *arXiv preprint arXiv:1703.00472* (2017).

[6] Relja Arandjelovic et al. "NetVLAD: CNN architecture for weakly supervised place recognition". In: *IEEE conference on computer vision and pattern recognition*. 2016.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006.

[8] Guillaume Bellegarda, Chuong Nguyen, and Quan Nguyen. "Robust quadruped jumping via deep reinforcement learning". In: *Robotics and Autonomous Systems* 182 (2024), p. 104799.

[9] Guillaume Bellegarda et al. "Robust high-speed running for quadruped robots via deep reinforcement learning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 10364–10370.

[10] Kevin Black et al. "Zero-shot robotic manipulation with pretrained image-editing diffusion models". In: *arXiv preprint arXiv:2310.10639* (2023).

[11] Michael Bloesch et al. "State estimation for legged robots on unstable and slippery terrain". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 6058–6064.

[12] Mariusz Bojarski et al. "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316* (2016).

[13] Anthony Brohan et al. "Do as i can, not as i say: Grounding language in robotic affordances". In: *Conference on robot learning*. PMLR. 2023, pp. 287–318.

[14] Anthony Brohan et al. "Rt-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[15] Marco Camurri et al. "Probabilistic contact estimation and impact detection for state estimation of quadruped robots". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1023–1030.

[16] Dian Chen et al. "Learning by cheating". In: *Conference on Robot Learning*. PMLR. 2020, pp. 66–75.

[17] Shuxiao Chen et al. "Learning torque control for quadrupedal locomotion". In: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE. 2023, pp. 1–8.

[18] Xuxin Cheng et al. "Extreme parkour with legged robots". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 11443–11450.

[19] Cheng Chi et al. "Diffusion policy: Visuomotor policy learning via action diffusion". In: *The International Journal of Robotics Research* 44.10-11 (2025), pp. 1684–1704.

[20] Suyoung Choi et al. "Learning quadrupedal locomotion on deformable terrain". In: *Science Robotics* 8.74 (2023).

[21] Gianpaolo Conte and Patrick Doherty. "An integrated UAV navigation system based on aerial image matching". In: *IEEE Aerospace Conference*. 2008.

[22] Xingye Da et al. "Learning a contact-adaptive controller for robust, efficient legged locomotion". In: *Conference on Robot Learning*. PMLR. 2021, pp. 883–894.

[23] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *IEEE conference on computer vision and pattern recognition workshops*. 2018.

[24] Jared Di Carlo et al. "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control". In: *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2018, pp. 1–9.

[25] Yanran Ding et al. "Representation-free model predictive control for dynamic motions in quadrupeds". In: *IEEE Transactions on Robotics* 37.4 (2021), pp. 1154–1171.

[26] Helei Duan et al. "Learning vision-based bipedal locomotion for challenging terrain". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 56–62.

[27] Péter Fankhauser, Michael Bloesch, and Marco Hutter. "Probabilistic terrain mapping for mobile robots with uncertain localization". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3019–3026.

[28] Siddhant Gangapurwala et al. "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control". In: *IEEE Transactions on Robotics* 38.5 (2022), pp. 2908–2927.

[29] Christian Gehring et al. "Dynamic trotting on slopes for quadrupedal robots". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5129–5135.

[30] Hunter Goforth and Simon Lucey. "GPS-denied UAV localization using pre-existing satellite imagery". In: *IEEE International Conference on Robotics and Automation*. 2019.

[31] Ruben Grandia et al. "Feedback mpc for torque-controlled legged robots". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4730–4737.

[32] Ruben Grandia et al. "Perceptive locomotion through nonlinear model-predictive control". In: *IEEE Transactions on Robotics* 39.5 (2023), pp. 3402–3421.

[33] Danyang Han and Karen E Adolph. "The impact of errors in infant development: Falling like a baby". In: *Developmental science* 24.5 (2021).

[34] Ross Hartley et al. "Contact-aided invariant extended Kalman filtering for robot state estimation". In: *The International Journal of Robotics Research* 39.4 (2020), pp. 402–430.

[35] Tairan He et al. "ASAP: Aligning Simulation and Real-World Physics for Learning Agile Humanoid Whole-Body Skills". In: *arXiv preprint arXiv:2502.01143* (2025).

[36] Timo Hinzmann and Roland Siegwart. "Deep UAV Localization with Reference View Rendering". In: *arXiv preprint arXiv:2008.04619* (2020).

[37] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *arXiv preprint arxiv:2006.11239* (2020).

[38] David Hoeller et al. "Anymal parkour: Learning agile navigation for quadrupedal robots". In: *Science Robotics* 9.88 (2024), eadi7566.

[39] David Hoeller et al. "Neural scene representation for locomotion on structured terrain". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8667–8674.

[40] Seungwoo Hong, Joon-Ha Kim, and Hae-Won Park. "Real-time constrained nonlinear model predictive control on so (3) for dynamic legged locomotion". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 3982–3989.

[41] Sixing Hu and Gim Hee Lee. "Image-based geo-localization using satellite imagery". In: *International Journal of Computer Vision* (2020).

[42] Sixing Hu et al. "Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[43] Chenguang Huang et al. "Visual language maps for robot navigation". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 10608–10615.

[44] Jemin Hwangbo et al. "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26 (2019).

[45] Jemin Hwangbo et al. "Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 3872–3878.

[46] Fabian Jenelten et al. "TAMOLS: Terrain-aware motion optimization for legged systems". In: *IEEE Transactions on Robotics* 38.6 (2022), pp. 3395–3413.

[47] Gwanghyeon Ji et al. "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4630–4637.

[48] Yandong Ji, Gabriel B Margolis, and Pulkit Agrawal. "Dribblebot: Dynamic legged manipulation in the wild". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5155–5162.

[49] Xiaotang Jiang et al. "MNN: A Universal and Efficient Inference Engine". In: *MLSys*. 2020.

[50] Yifeng Jiang et al. "Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2884–2890.

[51] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. "Mini cheetah: A platform for pushing the limits of dynamic quadruped control". In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 6295–6301.

[52] Donghyun Kim et al. "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control". In: *arXiv preprint arXiv:1909.06586* (2019).

[53] Joon-Ha Kim et al. "Legged robot state estimation with dynamic contact event information". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6733–6740.

[54] J Zico Kolter, Pieter Abbeel, and Andrew Y Ng. "Hierarchical apprenticeship learning with application to quadruped locomotion". In: *Advances in Neural Information Processing Systems*. Citeseer. 2008, pp. 769–776.

[55] Ashish Kumar et al. "Rma: Rapid motor adaptation for legged robots". In: *Robotics: Science and Systems*. 2021.

[56] Joonho Lee, Jemin Hwangbo, and Marco Hutter. "Robust recovery controller for a quadrupedal robot using deep reinforcement learning". In: *arXiv preprint arXiv: 1901.07517* (2019).

[57] Joonho Lee et al. "Learning quadrupedal locomotion over challenging terrain". In: *Science robotics* 5.47 (2020).

[58] Tianyu Li et al. "Learning generalizable locomotion skills with hierarchical reinforcement learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 413–419.

[59] Qiayuan Liao et al. "Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion". In: *arXiv preprint arXiv:2508.08241* (2025).

[60] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[61] Antonio Loquercio, Ashish Kumar, and Jitendra Malik. "Learning visual locomotion with cross-modal supervision". In: *arXiv preprint arXiv:2211.03785* (2022).

[62] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* (2004).

[63] Viktor Makoviychuk et al. "Isaac Gym: High performance GPU-based physics simulation for robot learning". In: *arXiv preprint arXiv:2108.10470* (2021).

[64] Gabriel B Margolis and Pulkit Agrawal. "Walk these ways: Tuning robot control for generalization with multiplicity of behavior". In: *Conference on Robot Learning*. PMLR. 2023, pp. 22–31.

[65] Takahiro Miki et al. "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science Robotics* 7.62 (2022).

[66] Maria Vittoria Minniti et al. "Adaptive CLF-MPC with application to quadrupedal robots". In: *IEEE Robotics and Automation Letters* 7.1 (2021), pp. 565–572.

[67] NVIDIA. *Isaac Sim*. Version 5.1.0. URL: `https://github.com/isaac-sim/IsaacSim`.

[68] Octo Model Team et al. "Octo: An Open-Source Generalist Robot Policy". In: *Proceedings of Robotics: Science and Systems*. Delft, Netherlands, 2024.

[69] Bhavit Patel, Timothy D Barfoot, and Angela P Schoellig. "Visual localization with google earth images for robust global pose estimation of UAVs". In: *IEEE International Conference on Robotics and Automation*. 2020.

[70] Tim Pearce et al. "Imitating human behaviour with diffusion models". In: *arXiv preprint arXiv:2301.10677* (2023).

[71] Xue Bin Peng and Michiel van de Panne. "Learning locomotion skills using deeprl: Does the choice of action space matter?" In: *Proceedings of the ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*. 2017, pp. 1–13.

[72] Xue Bin Peng et al. "Learning Agile Robotic Locomotion Skills by Imitating Animals". In: *Proceedings of Robotics: Science and Systems*. Corvalis, Oregon, USA, July 2020.

[73] Xue Bin Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[74] Lerrel Pinto et al. "Robust adversarial reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2817–2826.

[75] Marc Raibert et al. "Bigdog, the rough-terrain quadruped robot". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 10822–10825.

[76] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[77] Daniele Reda, Tianxin Tao, and Michiel van de Panne. "Learning to locomote: Understanding how environment design matters for deep reinforcement learning". In: *Proceedings of the 13th ACM SIGGRAPH conference on motion, interaction and games*. 2020, pp. 1–10.

[78] Moritz Reuss et al. "Goal-conditioned imitation learning using score-based diffusion policies". In: *arXiv preprint arXiv:2304.02532* (2023).

[79] Nikita Rudin et al. "Learning to walk in minutes using massively parallel deep reinforcement learning". In: *Conference on Robot Learning*. PMLR. 2022, pp. 91–100.

[80] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *IEEE conference on computer vision and pattern recognition*. 2018.

[81] Paul-Edouard Sarlin et al. "From coarse to fine: Robust hierarchical localization at large scale". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[82] Paul-Edouard Sarlin et al. "Leveraging deep visual descriptors for hierarchical efficient localization". In: *Conference on Robot Learning*. PMLR. 2018.

[83] Paul-Edouard Sarlin et al. "Superglue: Learning feature matching with graph neural networks". In: *IEEE conference on computer vision and pattern recognition*. 2020.

[84] John Schulman et al. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

[85] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[86] Haojie Shi et al. "Reinforcement learning with evolutionary trajectory generator: A general approach for quadrupedal locomotion". In: *IEEE Robotics and Automation Letters* (2022).

[87] Jonah Siekmann et al. "Sim-to-real learning of all common bipedal gaits via periodic reward composition". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 7309–7315.

[88] Laura Smith, Ilya Kostrikov, and Sergey Levine. "Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning". In: *Robotics: Science and Systems (RSS) Demo* 2.3 (2023), p. 4.

[89] Laura Smith et al. "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 1593–1599.

[90] Yu Sun et al. "Online learning of unknown dynamics for model-based controllers in legged locomotion". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 8442–8449.

[91] Jie Tan et al. "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots". In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018.

[92] Yicong Tian, Chen Chen, and Mubarak Shah. "Cross-view image matching for geo-localization in urban environments". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[93] Julen Urain et al. "Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion". In: *arXiv preprint arXiv:2209.03855* (2022).

[94] Patrick Varin, Lev Grossman, and Scott Kuindersma. "A comparison of action spaces for learning manipulation tasks". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6015–6021.

[95] Anirudh Viswanathan, Bernardo R Pires, and Daniel Huber. "Vision based robot localization by ground to satellite matching in GPS-denied situations". In: *IEEE International Conference on Intelligent Robots and Systems*. 2014.

[96] Sun Wandong. *Legged Lab: Direct IsaacLab Workflow for Legged Robots*. Version 1.0.0. 2025. URL: `https://github.com/Hellod035/LeggedLab`.

[97] Changchang Wu et al. "Image localization in satellite imagery with feature-based indexing". In: *ISPRS Congress: Technical Commission III*. 2008.

[98] Philipp Wu et al. "DayDreamer: World Models for Physical Robot Learning". In: *Conference on Robot Learning* (2022).

[99] Zhaoming Xie et al. "Dynamics randomization revisited: A case study for quadrupedal locomotion". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4955–4961.

[100] Chuanyu Yang et al. "Multi-expert learning of adaptive legged locomotion". In: *Science Robotics* 5.49 (2020), eabb2174.

[101] Yuxiang Yang et al. "Fast and efficient locomotion via learned gait transitions". In: *Conference on Robot Learning*. PMLR. 2022, pp. 773–783.

[102] Aurelien Yol et al. "Vision-based absolute localization for unmanned aerial vehicles". In: *IEEE International Conference on Intelligent Robots and Systems*. 2014.

[103] Donghoon Youm et al. "Legged robot state estimation with invariant extended kalman filter using neural measurement network". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2025, pp. 670–676.

[104] Dingqi Zhang et al. "Learning a Single Near-hover Position Controller for Vastly Different Quadcopters". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 1263–1269.

[105] Brianna Zitkovich et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *Conference on Robot Learning*. PMLR. 2023, pp. 2165–2183.