

Legged Locomotion and Collaborative Decision Making in Human-Robot Teams

David DeFazio

BS, State University of New York at Binghamton, 2018

MS, State University of New York at Binghamton, 2019

DISSERTATION

Submitted in partial fulfillment of the requirements for
the degree of Ph.D. in Computer Science
in the Graduate School of the
State University of New York at Binghamton
2024

© Copyright by David DeFazio 2024

All Rights Reserved

Accepted in partial fulfillment of the requirements for
the degree of Ph.D. in Computer Science
in the Graduate School of the
State University of New York at Binghamton
2024

Shiqi Zhang, Committee Chair
School of Computing, SUNY Binghamton

Yingxue Zhang, Committee Member
School of Computing, SUNY Binghamton

Weiyi Meng, Committee Member
School of Computing, SUNY Binghamton

Kaiyan Yu, Committee Member
Department of Mechanical Engineering, SUNY Binghamton

Yu Chen, Outside Examiner
Department of Electrical and Computer Engineering, SUNY Binghamton

Abstract

Legged robots are of great interest to the robotics community, due to their capacity for agile movements in diverse environments. Much of the recent research on legged robots focuses on developing locomotion capabilities in challenging terrain, and autonomous decision making. Although impressive locomotion and autonomous decision making has been demonstrated, existing approaches generally do not leverage human domain knowledge or support interactions with people for collaborative decision making. This highlights an important gap between human domain knowledge—encompassing locomotion gaits, commonsense reasoning, and spatial awareness—and the current legged robotics literature. In an effort to close this gap, our research adopts a human-in-the-loop approach for locomotion gait design, collaborative decision making applications, and floor plan navigation. To this end, our research innovates in three directions. First, we leverage human domain knowledge in quadruped locomotion to facilitate the learning of customized locomotion behaviors. Second, we develop a robotic guide dog system supporting bi-directional communication with a visually impaired human, who directly interacts with a robot. Third, we leverage existing floor plans designed by humans for planning and autonomous navigation tasks. These contributions help bridge the gap between existing work in legged robotics, and real-world applications involving human knowledge and decision making capabilities.

Acknowledgments

I'd like to acknowledge a few of the many people who have had a significant impact in producing the work in this dissertation, and training me to become a researcher. I first want to acknowledge my advisor Shiqi Zhang. When I joined the lab, I had no experience in robotics, and greatly underestimated some of the challenges in this field. Despite my early struggles in producing publishable results, Shiqi showed immense patience and provided important guidance, which was critical in producing the work in this dissertation, and training myself as a researcher. I acknowledge my committee members Yingxue Zhang, Weiyi Meng, and Kaiyan Yu, whose valuable feedback greatly improved this research. I acknowledge my current and former labmates in the AIR lab Saeid Amiri, Kishan Chandan, Yan Ding, Xiaohan Zhang, Zainab Altaweel, and Yohei Hayamizu. My labmates have been essential throughout my PhD, and have helped as co-authors, taking robot videos, discussing research, and providing feedback. I also acknowledge my past academic advisor Arti Ramesh, co-author and advisor Anand Seetharam, and corresponding labmates Gisella Bejarano, Yue Zhang, and Yunlong Xu, who introduced me to machine learning, and motivated me to begin a career in research. Finally, I'd like to acknowledge my friends and family who have been consistently supportive throughout my graduate studies.

Contents

List of Tables	ix
List of Figures	xii
1 Introduction	1
1.1 Dissertation Outline	5
2 Background	7
2.1 Legged Robots	7
2.2 Reinforcement Learning	8
2.3 Legged Locomotion Control	10
2.3.1 Model-based approaches for Legged Locomotion	10
2.3.2 Data-Driven approaches for Legged Locomotion	11
2.4 State Estimation for Learning-Based Legged Locomotion	14
2.5 Reward Machines	15
2.6 Robotic Guide Dog Systems	16
3 Learning Quadruped Locomotion Policies Using Logical Rules	19
3.1 Introduction	19
3.2 Related Work	22
3.2.1 Reward Machine	22
3.2.2 RL-based Locomotion Learning	23
3.2.3 Diverse Locomotion Gaits	23
3.3 RM-based Locomotion Learning	25
3.3.1 Reward Machines: Concepts and Terminologies	25
3.3.2 RM for Quadruped Locomotion	27
3.4 Reward Machines for Other Gaits	30
3.5 Experiments on Rule-Based Legged Locomotion Gaits	34
3.5.1 Training Details	34
3.5.2 Ablation Study	35
3.5.3 Gait Comparisons	40
3.5.4 Qualitative Results	41
3.6 Discussion	43

4	Robotic Guide Dog Navigation with Force Responsive Locomotion Control	45
4.1	Introduction	45
4.2	Related Work	48
4.3	Robotic Guide Dog System	49
4.3.1	Locomotion Controller	49
4.3.2	Robotic Guide Dog Navigation	52
4.4	Implementation Details	53
4.5	Experiments on Force Tolerance and Accuracy	54
4.5.1	Force Tolerance Evaluation	55
4.5.2	Force Estimation Evaluation	55
4.6	Hardware Demonstration	60
4.7	Discussion	61
5	Vision Language Models Can Parse Floor Plan Maps	63
5.1	Introduction	63
5.2	Related Work	66
5.2.1	Existing Map Representations and Navigation	66
5.2.2	VLM Prompting Strategies	67
5.2.3	Large Models in Robotics	67
5.3	Methodology in Floor Plan Parsing	68
5.3.1	Visual Prompting Strategy	68
5.3.2	VLM-Based Plan Generation	69
5.4	Experiments in VLM-Based Floor Plan Parsing	74
5.4.1	Experimental Setup	74
5.4.2	Experimental Results	78
5.5	Hardware Demonstration	81
5.6	VLM-Based Floor Plan Parsing Summary and Next Steps	81
6	Conclusions and Future Work	83
	Bibliography	87

List of Tables

3.1	All terms of R_{walk} . $\mathbf{v}$ refers to base velocity, $\mathbf{c}$ refers to commanded linear and angular base velocity, ω refers to base angular velocity, τ refers to joint torques, $\dot{\mathbf{q}}$ refers to joint velocities, $\mathbf{t}_{\text{air}}$ refers to each foot's air time, and $\mathbf{a}$ refers to an action.	28
3.2	Comparing average energy consumption per meter (rows labelled as E) and stability (rows labelled as S) across different gaits and terrains.	40
4.1	All terms of the reward function our locomotion policy is trained on. $\mathbf{v}$ refers to base velocity, $\mathbf{c}$ refers to commanded linear and angular base velocity, ω refers to base angular velocity, τ refers to joint torques, $\dot{\mathbf{q}}$ refers to joint velocities, $\mathbf{t}_{\text{air}}$ refers to each foot's air time, $\mathbf{a}$ refers to an action, and dt refers to the simulation time step.	50
4.2	Our learned controllers fell significantly less frequently, and better tracked velocity commands under external forces when compared to an MPC controller. Including the output of the force estimator in the state marginally improved robustness. The large variance in drift is caused by the large range of force strengths and directions we sample from.	56
4.3	Force estimation via accelerometer readings vs force estimator signal. . . .	60
5.1	An example navigation plan generated by GPT-4o in Map 1 shown on the left of Fig. 5.4. The initial location is "Terrasse Couverte" and the goal location is "Chambre 1". This navigation task requires a sequence of nine actions. GPT-4o achieved 0.96 success rate in this map on similar hard navigation tasks, which demonstrates great promises for VLM-based map parsing research.	79

List of Figures

1.1	Our research leverages different forms of human knowledge, and robotic sensors and algorithms in order to develop human-inspired legged robotic systems. We leverage knowledge in animal kinematics, commonsense knowledge, and spatial knowledge in this dissertation. Our human-inspired legged robot systems include rule-based locomotion gait design, a robotic guide dog tugging interface, and floor plan parsing for navigation tasks.	2
3.1	Snapshots of important poses of each of the six gaits learned with six different RMs. Specifying and learning the gaits require defining an automaton with no more than five automaton states (only two for half of the gaits). Red circles are around feet making contact with the ground.	20
3.2	Overview of RM-based Locomotion Learning (RMLL). We consider propositional statements specifying foot contacts. We then construct an automaton via propositional logic formulas for each locomotion gait (left side). To train gait-specific locomotion policies, we use observations which contain information from the RM, proprioception, velocity and gait frequency commands, and variables from a state estimator (right side).	26
3.3	Reward Machine for Trot gait, where we want to synchronize lifting the FL leg with the BR leg, and the FR leg with the BL leg. Trot is one of the six gaits considered in this chapter.	29
3.4	Bound gait synchronizes front feet and back feet	30
3.5	Pace gait synchronizes left feet and right feet	31
3.6	Walk gait lifts one foot at a time	31
3.7	Three-One gait alternates three feet with one of the front feet.	32
3.8	Half-Bound gait alternates the front feet with one of the back feet. State q_r discourages extraneous contacts with the wrong back foot, by setting all current and future reward to 0 when reached.	33
3.9	Isaac Gym simulation environment.	34
3.10	Reward curves for all gaits. RMLL more efficiently accumulates reward for each gait, particularly for the gaits with more complex foot contact sequences Walk, Three-One, and Half-Bound.	38
3.11	Visualization of each terrain we measured energy consumption and stability on.	39

3.12	Foot contact plots for each gait learned using RMLL. We report foot contacts from trials in simulation where we deploy each gait with a constant forward velocity command, zero angular velocity command, and constant gait frequency command. The colored horizontal bars indicate when the specified foot makes contact with the ground. The red highlighted region in each subplot denotes a full cycle of the gait.	42
4.1	Tugs on the robot guide dog are used as navigation cues during blindfolded navigation.	45
4.2	Overview of our approach. Our locomotion controller (circled in red) contains a velocity estimator, force estimator, and locomotion policy, all of which are trained in simulation. The base velocity estimator and force estimator are trained via supervised learning, using privileged information from the simulator as labels. The locomotion policy is trained via RL, and outputs target joint angles to a PD controller which converts them to joint torques which are directly applied to the robot. During deployment, our locomotion controller estimates external force at each time step. Force direction is derived from peaks in the estimated force signal. The direction of force determines the next local navigation goal for our navigation system to take, which returns velocity commands to our controller.	49
4.3	Map of our navigation environment. Yellow circles correspond to <i>decision points</i> , where the human needs to decide which direction to move in via tugging. The blue lines indicate the path the human took in our demonstration.	53
4.4	We report the accuracy and false positive rate of our force estimators, given forces of varied strength. The shaded region indicates the standard deviation between the five policies trained over five different random seeds. . . .	57
4.5	Measured acceleration (top) and estimated force (bottom) during a single trial. Tugs are denoted by red boxes.	58
5.1	Quadruped robot executing a VLM-generated plan (current action highlighted in red) to complete a navigation task while localizing directly on a floor plan image.	64
5.2	Overview of our method. A robot takes a raw image of a floor plan, which is then enhanced with labels and door indicators. The enhanced floor plan, along with a text prompt specifying the start and goal locations is given to a VLM. The VLM generates a navigation plan to reach the goal location, and the plan is executed on a mobile robot.	66
5.3	Text prompt input to VLM to generate navigation plans. We define the starting and ending locations, action types, and ask for explicit room and door connections to gain insights as to how the VLM understands the map.	70
5.4	Three maps used in our experiments for evaluating the performance of VLMs in map parsing and plan generation.	72
5.5	Example of a doubled map.	73

5.6	Comparison of GPT-4o results for map size, task difficulty and label density. We used original maps in the “label density” experiment. To accommodate hard tasks, we used doubled maps in the “task difficulty” experiment.	76
5.7	Comparison of Claude Sonnet 3.5 results for map size, task difficulty and label density. We used original maps in the “label density” experiment. To accommodate hard tasks, we used doubled maps in the “task difficulty” experiment.	77
5.8	The robot localizes directly on the floor plan image and performs a navigation task. The occupancy grid map that the robot used for localization and navigation was also derived from the floor plan map captured by the robot. We manually removed the labels that would otherwise be interpreted by the robot as obstacles. From right to left, the robot was performing 1) Going through Door N09, 2) Approaching Door N00, 3) Opening Door N00, and 4) Approaching Door T01.	80

1 Introduction

A commonly stated goal of robotics is to accomplish “dangerous, dirty, and dull” tasks, in order to improve the safety and productivity of society. The development of legged robots is necessary for making progress towards such a goal, due to the various types of challenging terrains in the world in which wheeled robots cannot traverse, such as steep slopes, uneven surfaces, or stairs. In order to maximize the potential of legged robots, they must be equipped with robust and energy-efficient locomotion controllers, alongside high-level decision-making capabilities. Unfortunately, the problem of developing such locomotion and decision-making capabilities sufficient for real-world applications is non-trivial.

Over the past few decades, significant research has been conducted in developing robust and energy efficient locomotion controllers for legged robots. Such controllers are commonly achieved either through model-based approaches [1, 2], or learning-based approaches [3, 4, 5, 6, 7, 8, 9, 10]. Model-based approaches assume knowledge of a dynamics model, which can predict the next world state given the current state and action. The dynamics model can be used in a constrained optimization problem which can be solved to generate control signals. Different from model-based approaches, learning-based approaches leverage Reinforcement Learning (RL) to learn from trial-and-error experiences

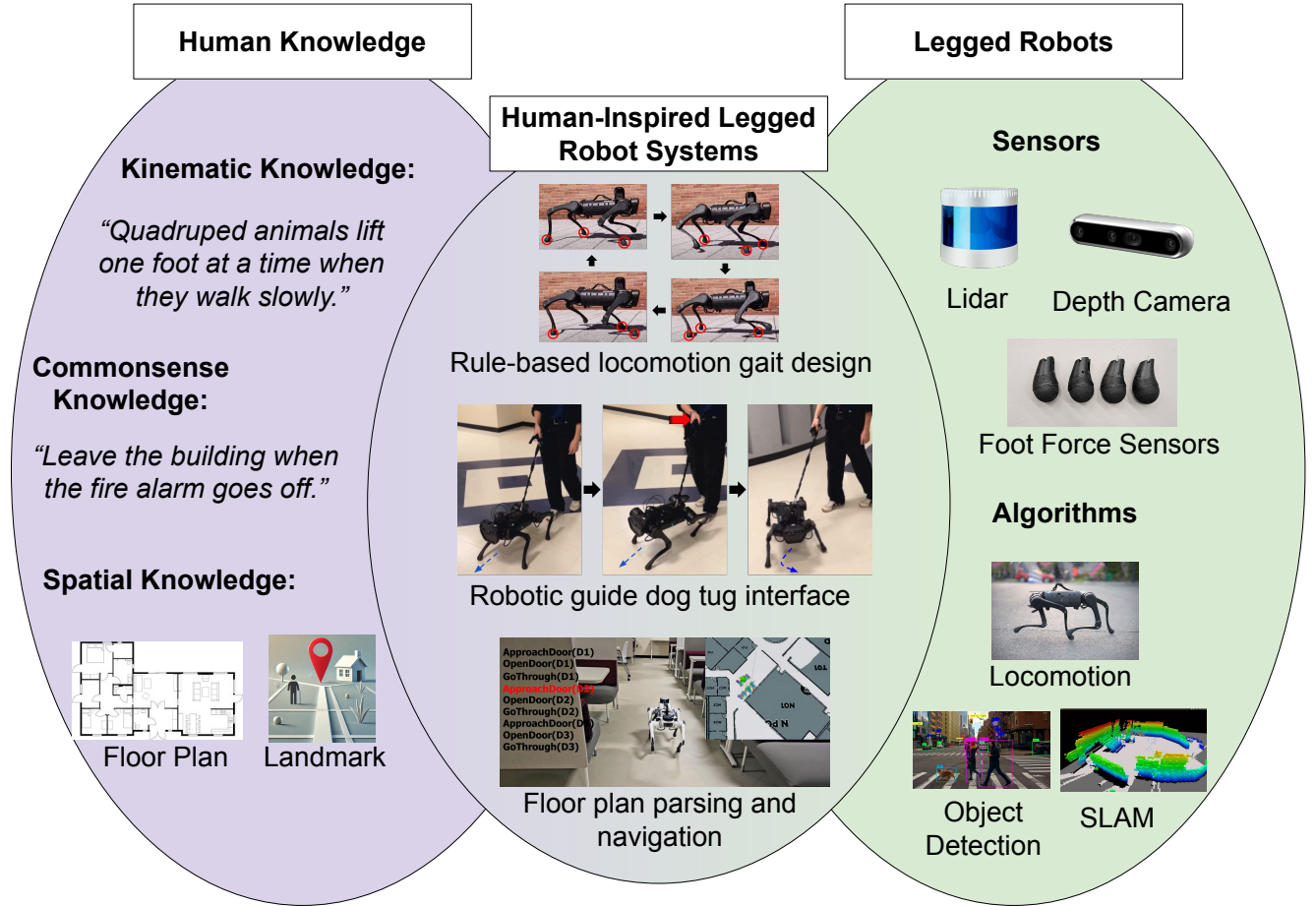


Figure 1.1: Our research leverages different forms of human knowledge, and robotic sensors and algorithms in order to develop human-inspired legged robotic systems. We leverage knowledge in animal kinematics, commonsense knowledge, and spatial knowledge in this dissertation. Our human-inspired legged robot systems include rule-based locomotion gait design, a robotic guide dog tugging interface, and floor plan parsing for navigation tasks.

without explicitly modeling the dynamics. Policies are then directly deployed on hardware which realize the learned locomotion gait. Impressive results have been achieved through both methods, each of which have demonstrated stable locomotion gaits on quadruped and biped robots.

Aside from locomotion control, legged robots must also be able to perform high-level decision making, including autonomous navigation capabilities. Such autonomous naviga-

tion is typically implemented through the use of Lidar and/or camera sensors for mapping and localization, and path planners to generate desired navigation trajectories. In order to address the unique demands of autonomous navigation for legged robots, existing work has focused on traversability estimation for path planning [11, 12], in order for the robot to generate efficient navigation plans based on the surrounding terrain. Aside from navigation tasks, other tasks requiring multi-agent coordination [13] or behavior trees for skill composition on complex quadruped manipulation tasks [14] have demonstrated impressive decision making capabilities.

Advancements in autonomously developing locomotion and high-level decision-making capabilities have pushed the field of legged robotics forward. While such advancements are important, the integration of human knowledge in existing legged robot systems is limited. However, humans possess expert domain knowledge, commonsense reasoning, and spatial awareness beyond the capabilities of current robotic systems. In order to expand upon the existing autonomous capabilities of legged robots, this dissertation focuses on the integration of human knowledge in such systems. More specifically, we leverage human knowledge in three directions.

First, we learn diverse legged locomotion gaits specified through simple logical rules over desired foot contacts. But why is it important for humans to specify the locomotion gait? Humans have studied legged animals for centuries, and have obtained vast amounts of expert domain knowledge on their locomotion strategies. Integrating human domain knowledge in legged locomotion training pipelines can expand the spectrum of legged robot locomotion behaviors. Our contributions in specifying arbitrary locomotion gaits through desired foot contact sequences enables human domain knowledge to be explicitly

encoded in the learned policies. These policies can then be used as priors in order to achieve more complex tasks, such as intelligent gait switching for efficient locomotion in varying environments.

Second, we develop a communication interface to facilitate bi-direction human-robot communication in the robotic guide dog setting. In this setting, the visually impaired human is generally aware of their global location, and makes high-level navigation decisions (i.e. turn left at the intersection). The robot dog is aware of its immediate environment, and must navigate around nearby obstacles. Thus, the human leverages common-sense knowledge about the environment to decide on the navigation route to take, while the robot leverages its perception capabilities for local path-planning which guides the human around nearby obstacles. Specifically, we develop a leash tugging interface which enables human-to-robot communication for global directional cues. Robot-to-human communication occurs through the forces felt through the leash during autonomous navigation. Such human-robot communication is crucial for leveraging the complementary strengths of humans (common sense knowledge) and robots (perception) in this setting.

Third, we leverage readily available floor plan maps of buildings to facilitate autonomous indoor robot navigation. This avoids the need for manual teleoperation of the robot to obtain an occupancy grid, and also provides semantic information relevant for navigation tasks. Vision-Language Models (VLMs) are leveraged to automatically parse the map for navigation plan generation, and the robot localizes itself directly on a copy of the floor plan map. We find that existing human knowledge in the form of floor plans serves as an effective map representation for planning and navigation in indoor environments.

The motivation of our research discussed in this dissertation is to advance legged robot

systems by adopting a human-in-the-loop approach in low-level locomotion gait design, collaborative human-robot decision making, and map parsing for navigation. Figure 1.1 provides an overview of our research.

1.1 Dissertation Outline

The remainder of this dissertation discusses the advancements we make in research related to the development of human-inspired legged robot systems. We organize as described below.

Chapter 2 discusses background information relevant to our research presented in the remaining chapters. In this chapter we discuss legged robots, RL and its application to locomotion learning, learned state estimators for legged robots, Reward Machines (RMs) [15], and robotic guide dog systems. This section is intended to concisely describe background information necessary for understanding the remaining chapters.

Chapter 3 presents our research in diverse quadruped locomotion gait learning. In order to leverage human domain knowledge in learning quadrupedal locomotion skills, we learn a diverse set of highly customized gaits via RMs. The gaits are specified via automata over simple logical rules which define target foot contact sequences. Aside from learning a diverse set of gaits, we also leverage RMs to specify gait frequency, which can be dynamically adjusted during policy execution. We learn six different gaits over a range of gait frequencies, evaluate the sample-efficiency of our policies, and demonstrate each gait on hardware.

Chapter 4 presents our research in force responsive robotic guide dog navigation. In order to develop a system for direct human-robot interactions which incorporates human

decision-making, we design a robotic guide dog system to help visually impaired people with navigation tasks. In this system, a human holds a leash directly connected to a robot which is autonomously navigating. We concurrently train a force tolerant locomotion controller, and a force detection model, to enable robust locomotion and human-robot communication. The locomotion controller tracks velocity commands, while the force detection model determines the direction the human is tugging. This tug direction is used to decide on the global navigation path, while a local path planner avoids immediate obstacles. We demonstrate this system on hardware for the task of blindfolded navigation.

Chapter 5 presents our research in floor plan parsing via VLMs. In order to leverage human knowledge for autonomous navigation and planning, we parse readily available floor plans with VLMs. Floor plans include valuable semantic room labels (i.e. “Bathroom”, “Elevator”) which we leverage to generate navigation plans. The floor plan also contains information about the geometry of the floor, which we use for robot localization and path planning. We explore the application of VLMs for floor plan parsing, and find generated plans to be surprisingly accurate. With simple modifications to the original floor plan image, we find significant improvements in the accuracy of the generated navigation plans. We additionally find the geometric information from the floor plan sufficient to serve as an occupancy grid, and demonstrate autonomous navigation via floor plans on a legged robot.

Chapter 6 concludes this dissertation with a summary, and directions for future work.

2 Background

In this chapter, we discuss some of the important building blocks relevant to our research presented in later chapters. This consists of a brief introduction to legged robotics, followed by Reinforcement Learning (RL), legged locomotion control, and state estimation for learning-based legged locomotion approaches. Next, we introduce Reward Machines. Finally, we discuss robotic guide dog systems.

2.1 Legged Robots

The advancement of legged robots is essential for achieving a comprehensive spectrum of autonomous capabilities. Compared to wheeled robots, legged robots have a unique advantage in traversing complex and uneven terrain. An early example of such a robot is BigDog [16], which was developed by Boston Dynamics to efficiently transport payloads over challenging terrains. Since then, there has been significant advancements in legged robot hardware and control, resulting in impressive behaviors [17] and more affordable legged robot platforms [18].

There exists many different areas of study in legged robotics. Significant efforts in producing stable locomotion gaits in challenging terrain has resulted in model-based and data-driven approaches, both of which are further discussed below. Some researchers leverage

perception capabilities via vision and lidar to more effectively interact with the environment. Machine learning methods are a crucial part of much of the legged robotics research, including sample-efficient reinforcement learning methods, lightweight vision models, and common-sense reasoning from large language models to name a few. Aside from this, advancements in physics simulation, sensor accuracy, and other hardware improvements are key areas of research in legged robot systems.

2.2 Reinforcement Learning

Reinforcement Learning (RL) refers to a machine learning paradigm in which a model is trained in an environment via trial and error experiences. This model, commonly referred to as a policy or agent, is trained for the purpose of completing tasks involving sequential decision making. Different from planning algorithms, RL agents do not have knowledge of the world dynamics, and must directly learn which actions to take based on observation and reward signal.

Formally, RL agents are trained over a Markov Decision Process (MDP), which is defined as a tuple (S, A, T, R, γ) . S refers to the state space, which represents all relevant information about the environment. A refers to the action space, which defines the set of actions the agent can take. $T : S \times A \times S \rightarrow [0, 1]$ is the transition function which outputs the probability of reaching state s' given state s and action a , $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward received by taking action a from state s and ending up in state s' , and γ is the discount factor which determines how valuable future reward should be considered in comparison to immediate reward. The goal in RL is to learn a policy $\pi : S \rightarrow A$ which selects actions that maximizes expected future discounted reward, given a state. Note that the Markov property

holds in an MDP, meaning the next state and reward depends only on the current state and action. Thus, optimal actions are determined only from the current state, and knowledge of state histories is not necessary.

Various RL algorithms have been developed to learn policies through trial-and-error experiences. RL algorithms fall under two categories: policy gradient methods and value-based methods. Both methods encourage selecting actions which maximize expected discounted future reward. Value-based approaches tries to learn the optimal value function that satisfies the Bellman Optimality Equation:

$$V^*(s) = \max_a \left(R(s, a) + \gamma \sum_{s'} P(s'|s, a) \cdot V^*(s') \right)$$

In order strike a balance between taking actions that we know lead to reward, and explore new actions, exploration strategies such as ϵ -greedy are used. ϵ -greedy will take a random action during training with probability ϵ . Common value-based approaches include Q-learning [19] or DQN [20].

Policy gradient approaches directly learn parameters of a policy which maximize the expected discounted cumulative reward:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \gamma^t R_t \right]$$

The gradient of $J(\theta)$ w.r.t to the parameters can then be derived:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) \cdot A(s_t, a_t) \right]$$

After a set of trajectories are collected from running the policy in the environment, the gradients are estimated and parameters are updated via gradient ascent. Common policy-gradient approaches include REINFORCE [21], or more recently TRPO [22] and PPO [23].

RL algorithms can be classified as either on-policy or off-policy. On-policy algorithms perform policy updates based on data generated from the current policy, while off-policy algorithms can update the policy from data not produced by the current iteration of the policy. Off-policy approaches typically use a replay buffer to store experiences from past policies, and perform updates based on random batches from the replay buffer.

2.3 Legged Locomotion Control

Approaches for developing legged locomotion controllers fall under two categories: model-based approaches or data-driven approaches. We introduce both methods in this section.

2.3.1 Model-based approaches for Legged Locomotion

Model Predictive Control (MPC) [24] is a common approach used to determine control signals while satisfying a set of constraints. Systems using MPC need the following components:

- Dynamics model which can predict the next state, given the current state and control signal.
- Cost function which assigns a cost to trajectories.
- Set of constraints for the state space and control signals.

MPC systems will use numerical optimization methods to predict trajectories of states and control signals which minimize cost while satisfying the given constraints. In order to alleviate compounding errors due to an inaccurate dynamics model, MPC systems predict trajectories over a limited horizon. After each control step is taken, a new trajectory is predicted based on the updated state. Note that there is a trade-off in selecting the desired trajectory length to predict. Short trajectories can be computed quickly, but can also lead to sub-optimal, or even infeasible trajectories which violate the given constraints. Longer trajectories lead to more optimal behaviors, but require more compute.

MPC is a common approach used for control of legged robot systems [1, 2, 25]. Existing systems have demonstrated stable locomotion in a variety of environments. Despite its success in legged robot control, significant efforts must be taken in order to achieve desired behaviors. Such efforts include accurate dynamics modeling, cost function design, constraint selection, and parameter selection such as prediction time horizon. MPC-based approaches are particularly challenging to implement for locomotion in extreme environments such as steep slopes or rocky terrains, where it is difficult to explicitly model the dynamics.

2.3.2 Data-Driven approaches for Legged Locomotion

The application of RL for legged locomotion has been a topic of interest since the early 2000s [3]. The idea is to learn locomotion controllers from many trial and error experiences in a physics simulator [26, 27] or directly on hardware [3, 28]. This has the advantage of learning desired locomotion behaviors without the need of dynamics models, which are necessary for the MPC-based approaches described above [1, 2, 25].

Imitation learning is another data-driven approach leveraged for locomotion learning [5, 28]. This involves capturing motion data from real dogs or artist generated motions, and training a policy which learns to imitate the given motions. The reward function is designed to reward motions similar to the given reference. While effective in providing a dense reward signal, the policy has less room for exploratory actions which deviate from the provided references.

In recent years, learning legged locomotion policies from simulation has seen much success [29, 4, 5, 30, 31, 6, 32, 7, 8, 9, 33, 34, 10], in part due to advancements in GPUs and sample-efficient RL algorithms [23, 35]. Training in simulation can be beneficial when compared to training directly on hardware, because it allows the policy to quickly explore millions of actions encompassing the full range of robot motion. Meanwhile, training on hardware takes longer to collect samples, requires an episode reset mechanism, and can lead to dangerous actions which can damage the robot.

Legged locomotion training environments generally share common features. In order for the policy to access relevant information for locomotion, the state space typically includes joint angles and joint velocities. The action space of nearly all learning-based legged locomotion controllers (with one notable exception [36]) consists of target joint angles. While reward functions across works vary based on the desired behavior, they typically follow a form similar to the following:

$$R = \exp(-\|\mathbf{v}_c - \mathbf{v}\|) - \tau \cdot \dot{q}$$

In this reward function, $\mathbf{v}$ and $\mathbf{v}_c$ corresponds to base velocity and commanded base

velocity, while τ and $\dot{q}$ correspond to motor torques and joint velocities. The first term encourages the robot to follow the velocity command, while the second term encourages minimizing energy consumption.

Achieving the target joint angles output by the policy is non-trivial. In order to get the joints to move, it's necessary to send control signals to the motors in the form of torque commands. Proportional-Derivative (PD) control is used to convert target joint angles to motor torques:

$$\tau = K_p(q^* - q) + K_d \cdot \dot{q}$$

where K_p and K_d are the proportional and derivative gains, q^* are the desired joint angles, and q are the current joint angles. The first term minimizes the absolute error between target and current joint angles, while the second term ensures the error changes smoothly to avoid overshooting the target. Torque commands are commonly sent at a higher frequency than the policy is queried. Thus, per each assignment of target joint angles, multiple control signals are sent to the motors.

Policies that are trained in simulation are challenging to deploy on hardware, due to the differences in the simulation environment and the real world, known as the sim-to-real gap. Closing the sim-to-real gap is commonly addressed through domain and dynamics randomization [37, 38, 8], adaptation modules [6, 39], or teacher-student training setups [31, 9, 33]. In our research, we use domain randomization to close the sim-to-real gap, which expands the range of environments we train on through varying environment variables such as friction or external forces. We additionally add noise to the state during training, to sim-

ulate sensor noise. We find this sufficient in aligning the simulation environment and real world for our research.

2.4 State Estimation for Learning-Based Legged Locomotion

In order to deploy locomotion policies on hardware, it is necessary to accurately estimate the state of the robot. As mentioned above, the state space commonly includes joint angles and joint velocities. These quantities can be directly estimated from sensors on the robot. Unfortunately, it's not the case that every component of the state space can be directly estimated via sensors on hardware. For example, base linear and angular velocity are commonly used components of the state space, as they are typically relevant components of the reward function. There is no sensor on the robot that can directly estimate such variables. Thus, some approaches have used motion capture systems or forward kinematics [5] to estimate these quantities.

Although such estimates have been successfully integrated with locomotion controllers, they oftentimes require extensive equipment and manual effort, or have large errors. If a new variable needs to be estimated (e.g. foot height), then a potentially new estimation technique must be implemented for this new variable. To mitigate the above mentioned issues, a learned state estimator has been used with much success [7, 40]. Under this paradigm, a state estimator is learned via supervised learning, concurrently with the policy. The simulation environment has ground-truth knowledge of the variables of interest, which are used as labels to train the state estimator, which uses the remainder of the state as input to predict such variables. When deployed on hardware, the robot inputs sensor readings into the state estimator, which predicts the other components of the state.

In our research, we use a learned state estimator to estimate base velocity, foot heights, and external forces. More details are provided in later chapters.

2.5 Reward Machines

A Reward Machine (RM) is typically used in settings where there is a set of “milestone” sub-goals to achieve in order to complete some larger task. Reward functions which do not encode these subgoals are oftentimes too sparse, while reward functions which explicitly reward sub-goal completion can be non-Markovian. An RM allows for specification of these sub-goals through an automaton, which can be leveraged to construct an MDP. Thus, through an RM, the reward function can give positive feedback for completing sub-goals, while also defining an MDP with a Markovian reward function.

Formally, an RM is defined as the tuple $(U, u_0, F, \delta_u, \delta_r)$ [41], where U is the set of automaton states, u_0 is the start state, F is the set of accepting states, $\delta_u : U \times 2^{\mathbf{P}} \rightarrow U \cup F$ is the automaton transition function, while $\delta_r : U \times 2^{\mathbf{P}} \rightarrow [S \times A \times S \rightarrow \mathbb{R}]$ is the reward function associated with each automaton transition. This RM definition assumes the existence of set $\mathbf{P}$, which contains propositional symbols that refer to high-level events from the environment that the agent can detect. A labelling function $L : S \times A \times S \rightarrow 2^{\mathbf{P}}$ determines the truth values of each symbol in $\mathbf{P}$ at each environment step. Then, the agent evaluates which automaton state transition to take via δ_u , and receives reward via δ_r .

An example referenced in the RM journal paper [15] presents a grid-world which contains a set of propositional symbols including “coffee”, “mail”, “office”, and “decorations”. The goal of the agent is to deliver the coffee and mail to the office, while avoiding the decorations. A sparse reward is received when the task is complete, otherwise there is no reward.

This reward function is non-Markovian when the agent only has knowledge of its location within the grid. A five state RM encourages the agent to select the coffee and mail in any order, followed by going to the office. The Cartesian product of the environment state and RM state (agents grid cell, and current automata state) makes the problem Markovian, and facilitates efficient learning of the task. Improved sample-efficiency is possible in some settings through CRM and HRM algorithms [15] which leverage counterfactual experience and the options framework respectively.

In Chapter 3, we leverage RMs in order to easily specify and learn diverse quadruped locomotion gaits.

2.6 Robotic Guide Dog Systems

Worldwide, it is estimated that there are 43.4 million people who are blind or visually impaired as of 2020 [42]. This inspired different forms of navigational aids which have been used to help visually impaired people perform different navigation tasks. Such aids include a smart cane with a camera and haptic feedback [43], mobile devices worn on a belt [44], and an augmented reality system [45]. Guide dogs are another type of navigational aid which has been effective at increasing the independence, confidence, companionship, and mobility of visually impaired people [46].

Although the above mentioned aids are useful in different aspects of guided navigation, they suffer from several limitations. Canes require the visually impaired person to actively sense the environment by physically moving the cane, which might not facilitate fluid navigation in crowded or unfamiliar environments. Mobile devices and augmented reality systems lack the ability to guide a person via physical forces from the environment,

and rely solely on audio and haptic feedback. Meanwhile, real guide dogs are expensive to train [47] with a low training success rate, and require substantial effort to take care of (such as daily walks). Guide dogs additionally have fundamental limitations in language and perception capabilities.

Robotic guide dogs have the potential to address the above mentioned issues with existing navigational aids. Such devices can physically traverse the environment alongside the human, even in the presence of steep slopes, stairs, or sudden drop-offs. The robot can be equipped with cameras, Lidar, microphones, and other sensors which can enable smooth obstacle avoidance, scene understanding, and bi-directional communication. Compared to real guide dogs, a robot guide dog is easier to maintain, and can be purchased without the need for a long and expensive training process.

To this end, research has been conducted in developing robotic guide dog systems, which perform human-robot co-navigation tasks where a human holds a taut leash or rigid connection to the quadruped robot. Some of these systems only support unidirectional robot-to-human communication via the forces the human feels from the leash [48, 49, 50]. The robot avoids nearby local obstacles, and navigates to a pre-determined goal location, while the human follows the robot based on leash forces. Other robotic guide dog systems allow for bidirectional human-robot communication, such that the human can indicate desired navigation directions via buttons [51] or predefined voice commands [52, 53]. Although these systems have demonstrated successful guided navigation with blindfolded participants, several advancements are needed to make this technology feasible for the public. In Chapter 4, we develop a robotic guide dog system which supports human-robot communication through a tugging interface.

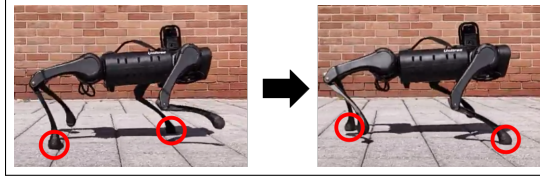
In this chapter, we present background information relevant to our research presented in later chapters. This consists of topics including legged robotics, RL, legged locomotion control, state estimation for learning-based legged locomotion, RMs, and robotic guide dog systems. In the next chapter, we discuss rule-based locomotion gait design with RMs.

3 Learning Quadruped Locomotion Policies Using Logical Rules

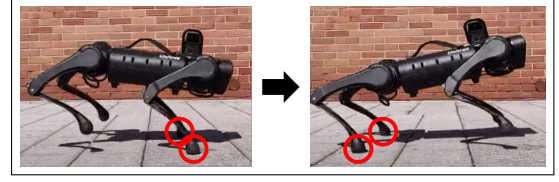
3.1 Introduction

Legged animals are capable of performing a variety of locomotion gaits, in order to move efficiently and robustly at different speeds and environments [54]. The same can be said of legged robots, where different locomotion gaits have been shown to minimize energy consumption at different speeds and environments [32, 55, 56]. Still, leveraging the full diversity of possible locomotion gaits has not been thoroughly explored. A larger variety of gaits can potentially expand a quadruped’s locomotion skills or even enable traversing terrains that are not possible before. Unfortunately, learning *specific* quadruped locomotion gaits is a challenging problem. To accomplish this, it is necessary to design a reward function which can express the desired behavior. Commonly used reward functions for quadruped locomotion encourage maximizing velocity command tracking, while minimizing energy consumption [4, 6]. While training over these types of reward functions oftentimes yields high quality locomotion policies, they do not specify any particular gait.

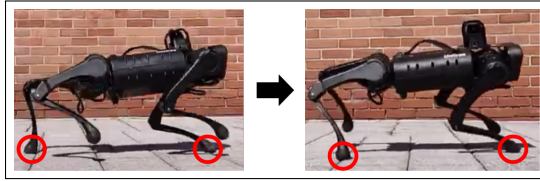
In order to incentivize the agent to learn a specific gait, the reward function must be encoded with such gait-specific knowledge. It is possible to design a naive reward function which explicitly encourages specific sequences of milestone foot contacts, which we refer



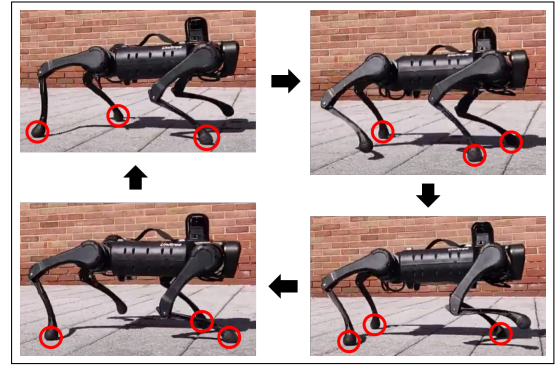
Trot



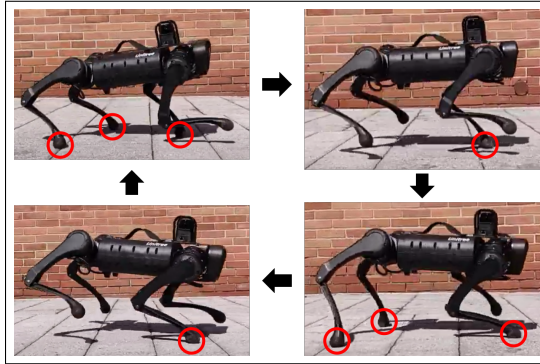
Bound



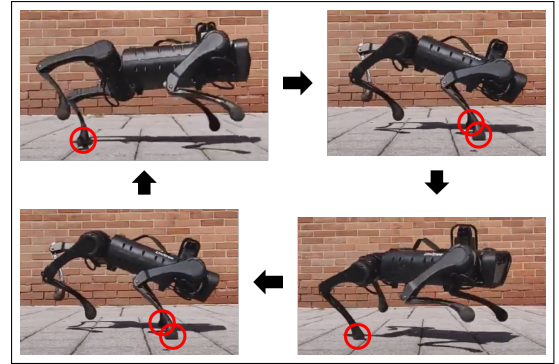
Pace



Walk



Three-One



Half-Bound

Figure 3.1: Snapshots of important poses of each of the six gaits learned with six different RMs. Specifying and learning the gaits require defining an automaton with no more than five automaton states (only two for half of the gaits). Red circles are around feet making contact with the ground.

to as poses. Unfortunately, this breaks the Markov property, because historical knowledge of previous poses within the gait is necessary to know which pose should be reached next in order to adhere to the specified gait. Quadruped locomotion controllers are commonly run at 50 Hz or more [6, 9], which generates a long history of states between each pose of a gait. Thus, naively satisfying the Markov property would require including all of these historical states in the state space, and would make the learning process more challenging as the policy would need to figure out which portion of this history is relevant.

Some researchers have taken advantage of motion priors in order to encode gait-specific knowledge in a reward function. One popular method for encoding such knowledge in a reward function is to maximize the similarity between the robot’s motion and a reference trajectory [5, 28]. While this approach has been successfully demonstrated on real robots, it requires significant manual effort to obtain reference trajectories, and constrains the robot’s motion to the given trajectory.

In this chapter, we alleviate the above mentioned problem of gait specification by leveraging Reward Machines (RMs) [15], which specify reward functions through deterministic finite automata. RMs have been applied to various domains for guiding RL agents [57, 58, 59, 60]. In this chapter, RM serves as high-level specifications of gaits for low-level locomotion policy learning. The RM transition function is defined through propositional logic formulas, which in our case specify foot contacts. Thus, changing the automaton state corresponds to reaching the next pose within the gait. The reward function is Markovian when considering the low-level state (robot sensor information), along with the current automaton state, because the automaton state encodes the relevant gait-level information needed to determine the next pose. This approach enables us to easily specify and learn

diverse gaits via logical rules, without the use of motion priors.

We refer to our approach as RM-based Locomotion Learning (RMLL), and train policies for six different gaits in simulation without the use of reference trajectories. Each policy is trained over a range of gait frequencies, which we can dynamically adjust during deployment. The reward function of each gait is defined through an automaton over desired foot contacts. We conduct an ablation study to evaluate the sample efficiency of RMLL in training the six different gaits, measure energy consumption and stability of each gait in different terrains, and deploy all gaits on a real Unitree A1 quadruped robot (see Figure 3.1). We compare RMLL to three baselines, each of which is designed to evaluate whether knowledge of the automaton state during training is actually beneficial in terms of sample efficiency. Results show that RMLL improves sample efficiency over its ablations for all gaits, more substantially for more complex gaits.

3.2 Related Work

In this section, we discuss prior work on RMs (beyond their introduction in Chapter 2), and legged locomotion via Reinforcement Learning (RL). We then focus on existing methods of gait specification and learning for legged locomotion, with and without motion priors.

3.2.1 Reward Machine

Since the introduction of Reward Machines (RMs) [41], there have been various new research directions such as learning the RM structure [57], RM for partially observable environments [61], multi-agent intention scheduling [62], and probabilistic RMs [60] to

name a few. While these works primarily focused on RM algorithmic improvements and theoretical analysis, their applications did not go beyond toy domains. RMs have also been used for simulated robotic arm pick-and-place tasks, which learn RM structures from demonstrations [59]. However, their approach was not implemented or evaluated in real-world robotic continuous control problems with high-dimensional action spaces. A recent journal article formally described the RM framework as well as a few algorithms for RM-based reinforcement learning [15]. We use RMs for robot locomotion learning in this chapter.

3.2.2 RL-based Locomotion Learning

There are numerous works on applications of RL for quadruped robot locomotion policy learning [4, 6, 28, 8, 9, 10]. Approaches of this type often lead to robust locomotion gaits, some of which can transfer to real robots. However, the above mentioned approaches generally focus on learning robust locomotion policies, and do not support the specification of particular gaits. Other works that support diverse locomotion gaits are described next.

3.2.3 Diverse Locomotion Gaits

With Motion Priors: Various works have learned diverse locomotion gaits for quadruped robots with the use of motion priors. For example, trajectory generators [29] and motion references [28, 5] have been leveraged for learning specified gaits. Obtaining these priors require extensive human (and sometimes even animal) effort, and restricts the robot to following the specified trajectory with little variation. While motion references can be generated, it requires highly tuned foot trajectory polynomials and phase generation func-

tions [63]. Our approach does not require such motion priors and can easily specify different gaits via a few logical rules. Our policies also have freedom to explore variations of the specified gait on its own and is not restricted by a predefined trajectory.

MPC-based: Various MPC-based approaches have successfully demonstrated diverse locomotion gaits without the use of motion priors [1, 2]. However, these methods require accurate dynamics models, and significant manual tuning for each gait.

Emergent Gaits: Different gaits can naturally emerge through minimizing energy [32], or selected from a high-level policy which selects foot contact configurations or contact schedules [55, 56]. More generally, diverse exploration strategies have been shown to improve policy performance and encourage learning different behaviors [64]. While these approaches lead to diverse locomotion gaits and behaviors, it does not provide the ability to learn any *arbitrary* gait or gait frequency specified beforehand.

Most Similar to ours: There are recent works that aim to learn locomotion gaits based on high-level gait descriptions – RMLL (ours) shares the same spirit. LLMs have been leveraged to specify and perform diverse locomotion behaviors [65, 66]. While the work of Tang et al. is useful in converting natural language to low-level control on hardware, they require extensive prompt engineering, and additional manual effort in defining a random pattern generator for each desired gait. Thus, this approach has only been demonstrated on two-beat gaits, and can be less effective to uncommon gaits, while our RMLL approach supports specifying and learning the two novel gaits of Three-One and Half-Bound. Other similar works involve specifying and learning diverse locomotion gaits through explicitly

defining swing and stance phases per leg [30, 34]. The former approach is designed for a bipedal robot, while the later only supports two-beat quadruped gaits. In contrast, RMLL only needs foot contact sequences (instead of leg-specific timings), and can specify and learn arbitrary quadrupedal gaits well beyond the set of two-beat gaits.

The main contribution of this research is a novel paradigm for gait specification. We focus on demonstrating the complete pipeline of this new paradigm using a real robot. This research paves the way for future research, e.g., on improving the efficiency of policy learning, developing novel gaits for quadrupedal robots, intelligently transitioning between gaits, and facilitating human-quadruped co-navigation e.g., for robotic guide dogs for the visually impaired [67].

3.3 RM-based Locomotion Learning

We present our RM-based reinforcement learning approach for learning quadruped locomotion policies below. Figure 3.2 presents an overview of how we use RMs to specify a diverse set of quadruped locomotion gaits and facilitate efficient policy learning.

3.3.1 Reward Machines: Concepts and Terminologies

As defined in chapter 2. RMs are formally defined as the tuple $(U, u_0, F, \delta_u, \delta_r)$ [41], where U is the set of automaton states, u_0 is the start state, F is the set of accepting states, $\delta_u : U \times 2^{\mathbf{P}} \rightarrow U \cup F$ is the automaton transition function, while $\delta_r : U \times 2^{\mathbf{P}} \rightarrow [S \times A \times S \rightarrow \mathbb{R}]$ is the reward function associated with each automaton transition. This RM definition assumes the existence of set $\mathbf{P}$, which contains propositional symbols that refer to high-level events from the environment that the agent can detect. A labelling function $L : S \times A \times S \rightarrow$

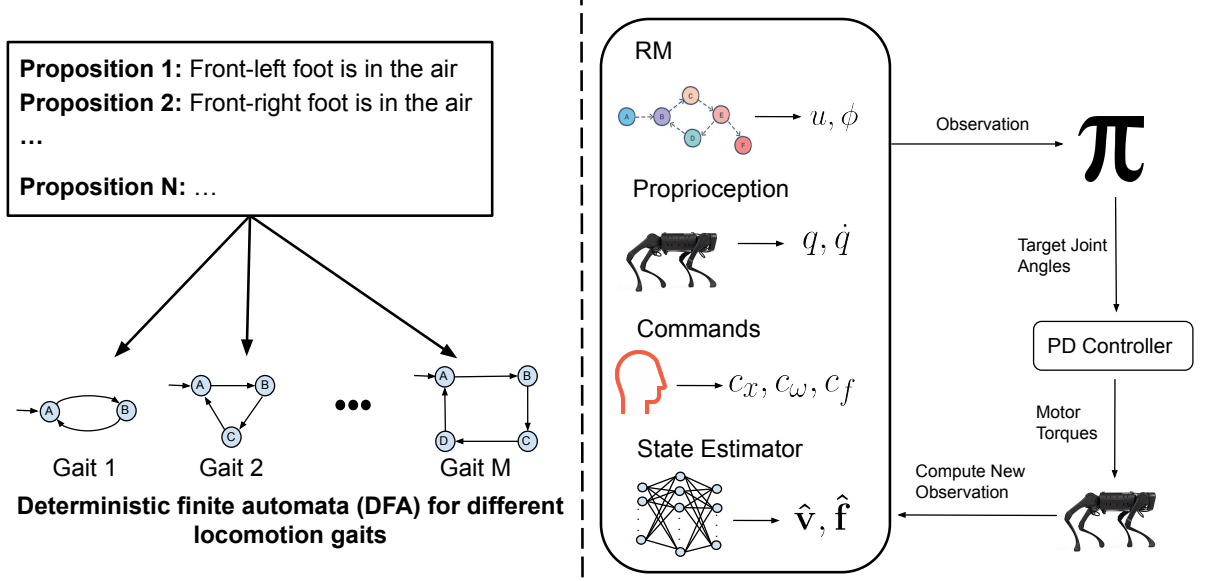


Figure 3.2: Overview of RM-based Locomotion Learning (RMLL). We consider propositional statements specifying foot contacts. We then construct an automaton via propositional logic formulas for each locomotion gait (left side). To train gait-specific locomotion policies, we use observations which contain information from the RM, proprioception, velocity and gait frequency commands, and variables from a state estimator (right side).

$2^{\mathbf{P}}$ determines the truth values of each symbol in $\mathbf{P}$ at each environment step. Then, the agent evaluates which automaton state transition to take via δ_u , and receives reward via δ_r .

Reward machines are defined alongside state space S , which describe the low-level observations the agent receives after each step in the environment. In order to construct an MDP from the non-Markovian reward defined by the RM, the agent considers its own observations from S , along with its current RM state from U . Training over state space $S \times U$ no longer violates the Markov property, because knowledge of the current RM state indicates which sub-goal was previously completed. The inclusion of this subsection is simply for completeness. More details are available in the RM article [15].

3.3.2 RM for Quadruped Locomotion

We use RMs to specify the sequence of foot contacts expected of the gait. In our domain, we consider $\mathbf{P} = \{P_{FL}, P_{FR}, P_{BL}, P_{BR}\}$, where $p \in \mathbf{P}$ is a Boolean variable. These indicate whether the front-left (FL), front-right (FR), back-left (BL), and back-right (BR) feet are making contact with the ground. Labelling function L evaluates whether a foot makes contact with the ground via the foot force sensors on the robot. Automaton states in U correspond to different poses in the gait, where u_0 corresponds to the last pose. Meanwhile, δ_u changes the automaton state when the next pose in the gait is reached. We define δ_r as:

$$\delta_r(u_t, a) = \begin{cases} R_{\text{walk}}(s) * b & \delta_u(u_t, a) \neq u_t \\ R_{\text{walk}}(s) & \text{otherwise} \end{cases}$$

where R_{walk} encourages maximizing velocity command tracking while minimizing energy consumption [8], and is fully defined in Table 4.1. Reward function δ_r encourages taking RM transitions which correspond to the specified gait, because R_{walk} is scaled by bonus b when such transitions occur. We leave F empty for all gaits, as quadruped locomotion is an infinite-horizon task.

We define our state space $S = (u, \phi, q, \dot{q}, a_{t-1}, c_x, c_\omega, c_f, \hat{\mathbf{v}}, \hat{\mathbf{f}})$, where u is the current RM state, ϕ is the number of time steps which occurred since the previous RM state changed, q and $\dot{q}$ are the 12 joint angles and joint velocities respectively, a_{t-1} is the previous action, c_x and c_ω are base linear and angular velocity commands respectively, c_f is the gait frequency command, and $\hat{\mathbf{v}}, \hat{\mathbf{f}}$ is estimated base velocity and foot heights. The RM state is encoded as

Term Description	Definition	Scale
Linear Velocity x	$\exp(-\ \mathbf{c}_x - \mathbf{v}_x\ ^2/0.25)$	1.0
Linear Velocity z	$\mathbf{v}_z^2$	-2.0
Angular Velocity x,y	$\ \omega_{x,y}\ ^2$	-0.05
Angular Velocity z	$\exp(-(\mathbf{c}_\omega - \omega_z)^2/0.25)$	0.5
Joint Torques	$\ \tau\ ^2$	-0.0002
Joint Accelerations	$\ (\dot{\mathbf{q}}_{\text{last}} - \dot{\mathbf{q}})/dt\ ^2$	-2.5e-7
Feet Air Time	$\sum_{f=1}^4 (\mathbf{t}_{\text{air},f} - 0.5)$	1.0
Action Rate	$\ \mathbf{a}_{\text{last}} - \mathbf{a}\ ^2$	-0.01

Table 3.1: All terms of R_{walk} . $\mathbf{v}$ refers to base velocity, $\mathbf{c}$ refers to commanded linear and angular base velocity, ω refers to base angular velocity, τ refers to joint torques, $\dot{\mathbf{q}}$ refers to joint velocities, $\mathbf{t}_{\text{air}}$ refers to each foots air time, and $\mathbf{a}$ refers to an action.

a one-hot vector, making the dimensions of $S \in [49, 52]$ based on the number of RM states defining the gait.

Gait Frequency: Aside from gait specification, we also leverage RMs to specify gait frequency. Our definition of δ_r naturally encourages high frequency gaits, because maximizing the number of pose transitions maximizes total accumulated reward. Thus, we introduce gait frequency command c_f , which denotes the minimum number of environment steps which must be taken until the agent is allowed to transition to a new RM state. When the agent maximizes the number of RM transitions it takes, while being restricted by c_f , then the commanded gait frequency is followed. Adding c_f on its own would cause the reward function to be non-Markovian, because the agent needs to remember how many environment steps have occurred since the RM state last changed. Thus, we also add timing variable ϕ to our observations, which keeps track of how many environment steps have occurred since the RM state has changed last. At every environment time step, we compare ϕ with c_f , and do not allow an RM transition to take place if $\phi < c_f$. Adding c_f and ϕ to

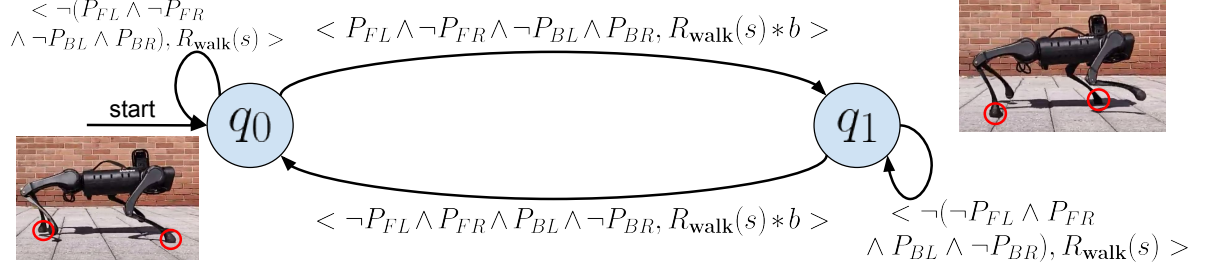


Figure 3.3: Reward Machine for Trot gait, where we want to synchronize lifting the FL leg with the BR leg, and the FR leg with the BL leg. Trot is one of the six gaits considered in this chapter.

the state enables gait frequency to be dynamically adjusted during policy deployment, and is demonstrated on hardware.

Illustrative Gait: We now discuss specifying a well known quadruped locomotion gait, **Trot**, via RM. Figure 3.3 shows the RM associated with this gait. In this **Trot** automaton, we want to synchronize lifting the FL leg with the BR leg, and the FR leg with the BL leg. Propositional logic formula $P_{FL} \wedge \neg P_{FR} \wedge \neg P_{BL} \wedge P_{BR}$ evaluates to true when only the FR and BL feet are in the air simultaneously, while $\neg P_{FL} \wedge P_{FR} \wedge P_{BL} \wedge \neg P_{BR}$ evaluates to true when only the FL and BR feet are in the air simultaneously. The two RM states correspond to which combination of feet were previously in the air. If the agent is in state q_1 , then $P_{FL} \wedge \neg P_{FR} \wedge \neg P_{BL} \wedge P_{BR}$ must have been evaluated as true at some point earlier. Note that when the agent does not achieve the desired pose, then the agent takes a self-loop to remain in the current RM state. ¹

Remark It is an intuitive idea of training a gait-specific locomotion policy via RM, because along with low-level sensor information, the policy also has access to the current

¹We provide the RMs for all other gaits we trained in Supplementary Materials, which can be downloaded on the project page <https://sites.google.com/view/rm-locomotion-learning/home>.

RM state, which is an abstract representation of the historical foot contacts relevant to the current pose in the gait. Rather than attempting to learn this from a long history of world states, the RM state explicitly encodes the previously reached gait pose. Thus, the policy can learn different gaits in a sample-efficient manner, because at each time step it can reference the RM state to indicate which pose within the gait to reach next.

3.4 Reward Machines for Other Gaits

In this section, we present the reward machines for the five gaits not already shown: **Bound**, **Pace**, **Walk**, **Three-One**, and **Half-Bound**.

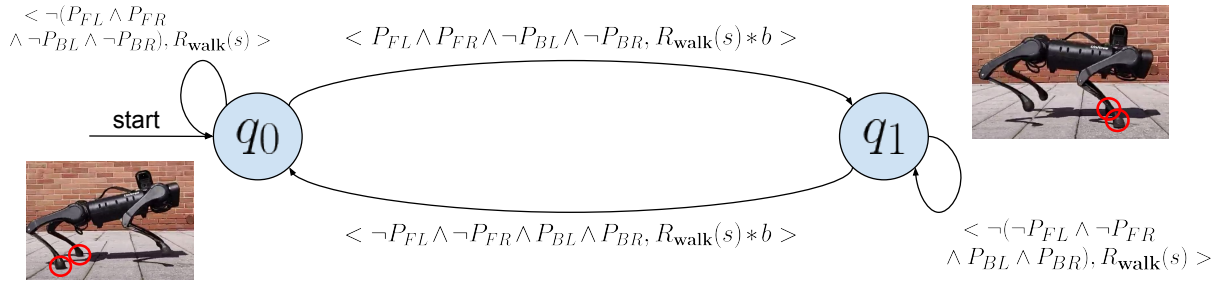


Figure 3.4: **Bound** gait synchronizes front feet and back feet

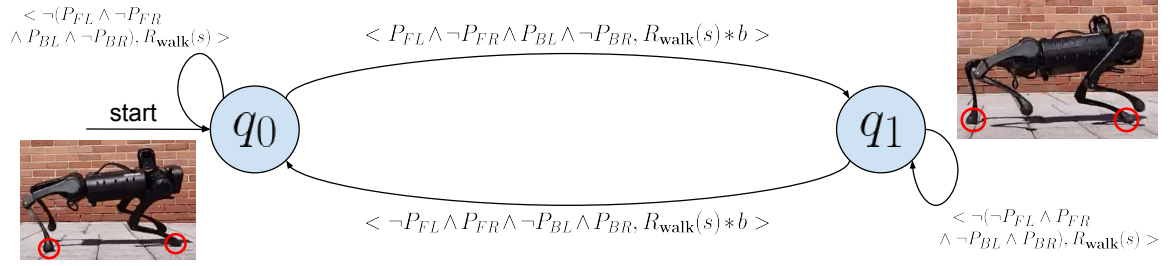


Figure 3.5: **Pace** gait synchronizes left feet and right feet

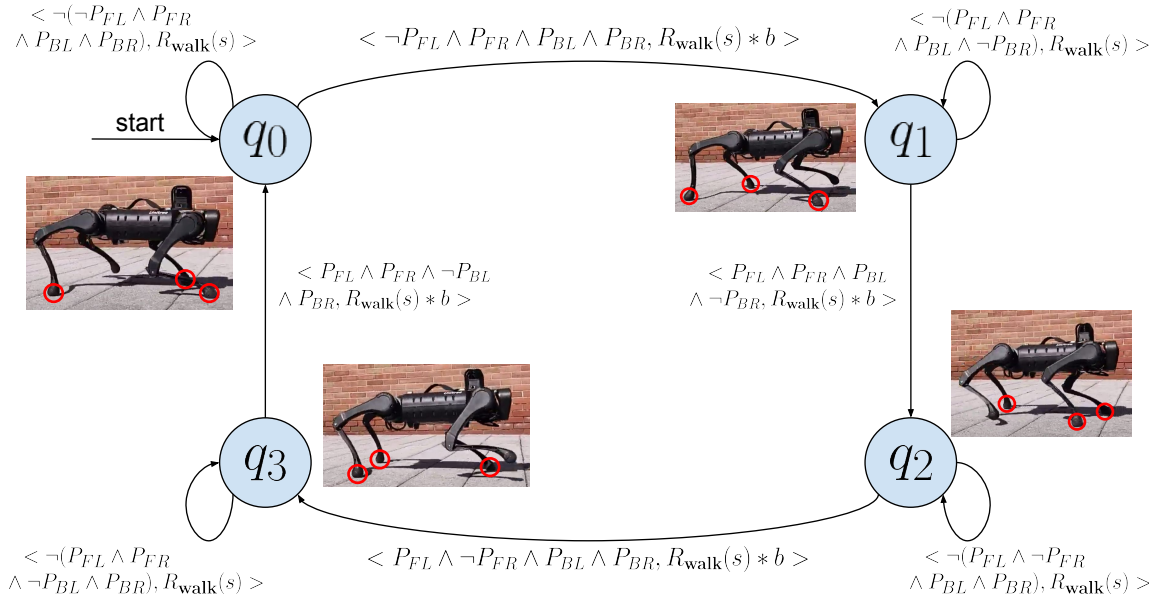


Figure 3.6: **Walk** gait lifts one foot at a time

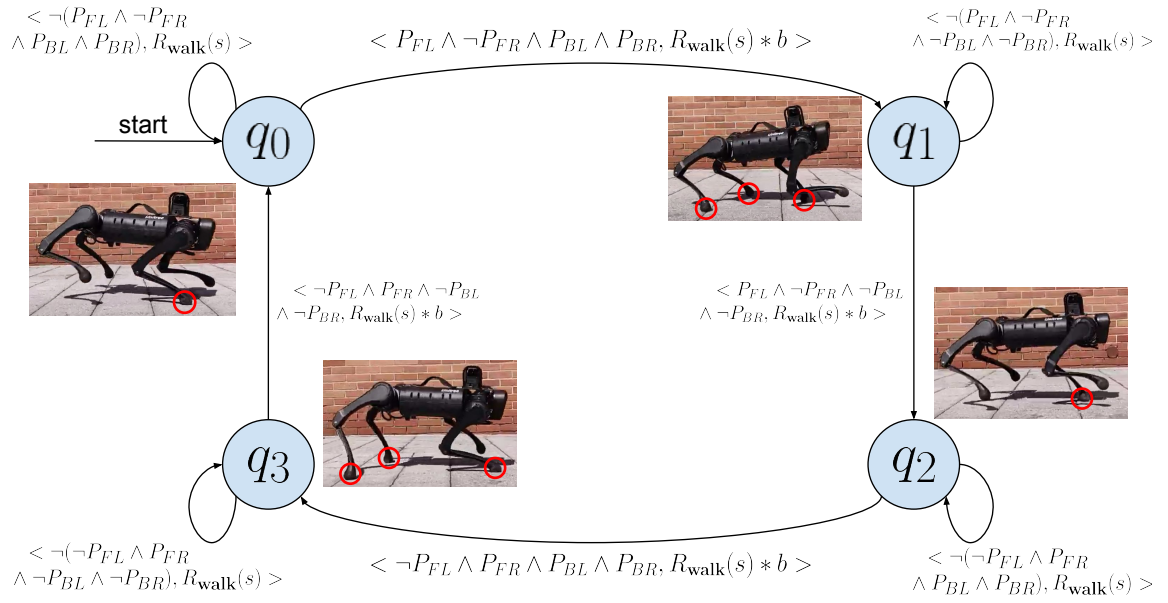


Figure 3.7: **Three-One** gait alternates three feet with one of the front feet.

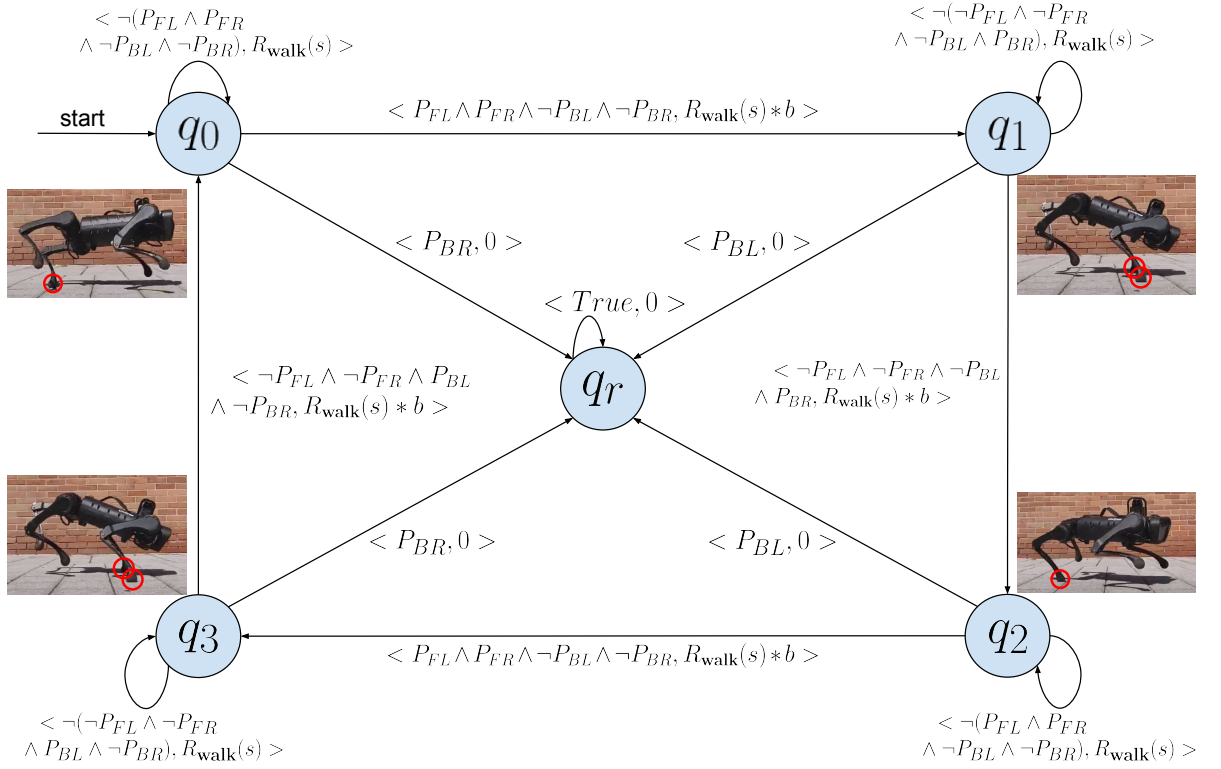


Figure 3.8: **Half-Bound** gait alternates the front feet with one of the back feet. State q_r discourages extraneous contacts with the wrong back foot, by setting all current and future reward to 0 when reached.

3.5 Experiments on Rule-Based Legged Locomotion Gaits

We train six different locomotion gaits via RMLL in simulation, and perform an ablation study to evaluate whether knowledge of the RM state improves sample efficiency when compared to ablations which do not access the RM state during training. After that, we compare energy consumption and stability of each gait across different terrains. Finally, we demonstrate all learned gaits on a Unitree A1 robot.

3.5.1 Training Details

State, Action, Reward We estimate base velocity $\hat{\mathbf{v}}$ and foot heights $\hat{\mathbf{f}}$ concurrently with the policy, via supervised learning [7]. Note that during training we only consider a foot in the air if it is higher than 0.03 meters. Actions include the target joint positions of each joint. These are input to a PD controller which computes the joint torques. The PD controller has a proportional gain $K_p = 20$ and derivative gain $K_d = 0.5$. The policy is queried at 50 Hz, and control signals are sent at 200 Hz. We set bonus $b = 1000$ in δ_r for all gaits.

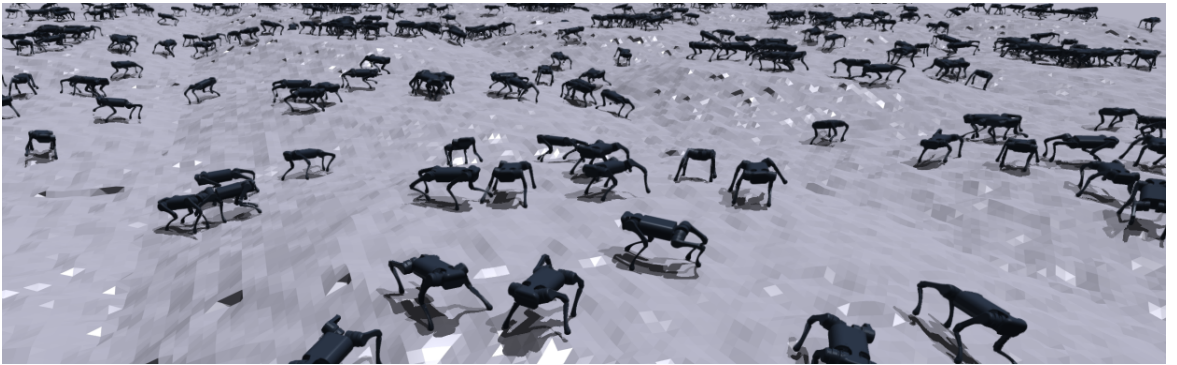


Figure 3.9: Isaac Gym simulation environment.

Environment Details We use the Isaac Gym [27] physics simulator and build upon a legged locomotion environment [8] to train our policies. We use a terrain called *random_uniform_terrain*, which includes hilly perturbations on the surface (see Figure 3.9). The robot traverses more challenging versions of this terrain based on a curriculum which increases terrain difficulty after the robot learns to traverse flatter versions of the terrain. Each episode lasts for 20 seconds, and ends early if the robot makes contact with the ground with anything other than a foot, if joint angle limits are exceeded, or if the base height goes below 0.25 meters. After each training episode, we sample a new velocity and gait frequency command for the robot to track. To facilitate sim-to-real transfer, we perform domain randomization over surface frictions, add external pushes, and add noise to observations [8].

Model Training We train our policy via PPO [23], with actor and critic architectures as 3-layer multi-layer perceptrons (MLPs) with hidden layers of size 256, and the same loss function. Each policy is trained for 100 million time steps (except for **Half-Bound**), where parameters are updated every 100,000 time steps. Data is collected from 4096 agents running simultaneously.

3.5.2 Ablation Study

We run an ablation study to determine whether knowledge of the RM state actually improves sample efficiency. We design the following baselines which we compare RMLL against:

1. **No-RM**: Remove the RM state from the state space, keeping everything else the

same.

2. **No-RM-Foot-Contacts**: Remove the RM state from the state space, and add a boolean vector of foot contacts.
3. **No-RM-History**: Remove the RM state from the state space, and add a boolean vector of foot contacts. Expand the state space to include states from the past 12 time steps.

Comparing against **No-RM** indicates whether the RM state is useful at all. Comparing against **No-RM-Foot-Contacts** indicates whether RM state is only useful because it contains information about foot contacts. Comparing against **No-RM-History** indicates whether the information provided by the RM state can be easily learned when given sufficient history.

It should be noted that there are no existing methods supporting the specification and learning of *arbitrary* gaits without using motion priors, dynamics models, or significant manual efforts such as prompt engineering and random pattern generators. Furthermore, highly customized gaits such as **Three-One** and **Half-Bound** are new to the literature, and to the best of our knowledge, there are no existing methods which support learning such gaits. Thus, we focus on evaluating how knowledge of the RM state contributes to the overall performance of RMLL.

We experiment over six different locomotion gaits: **Trot**, **Pace**, **Bound**, **Walk**, **Three-One**, and **Half-Bound**. Gaits **Trot**, **Bound**, **Pace**, **Three-One**, and **Half-Bound** sample linear and angular velocity commands from $[-1, 1]$ meters per second, and a gait frequency command from $[6, 12]$ time steps. Meanwhile, **Walk** samples from $[-0.5, 0.5]$ meters per

second and [5, 10] respectively. This is because quadruped animals naturally use **Walk** gait for slower locomotion speeds.

For each approach (ablation or not), we trained over five different random seeds per gait. For each training run, we save the policy after every 5 million steps. We then deploy each of those saved policies for 100 episodes, and average the accumulated reward over the five runs per approach. We report the resulting reward curves in Figure 3.10, where the shaded region indicates the standard deviation of the total accumulated reward across the five training runs.

The results indicate that knowledge of the RM state improves sample efficiency for all gaits when compared with the ablations. We believe this is the case, because the RM state can efficiently inform the policy of gait-relevant historical foot contacts, whereas the ablations either do not have access to historical foot contacts, or must learn the relevant contacts from a history of world states.

The results also show that **No-RM-History** does not perform better than the other ablations without history, indicating that it is challenging to learn gait-relevant information directly from 12 time steps of historical states. We also notice that **No-RM** performs similarly to **No-RM-Foot-Contacts**, which indicates **No-RM** learns to implicitly estimate foot contacts from the state. Finally, we notice a large performance gap between RMLL and all other ablations for **Half-Bound**. We believe this is the case due to the additional complexity in the RM structure of this gait.

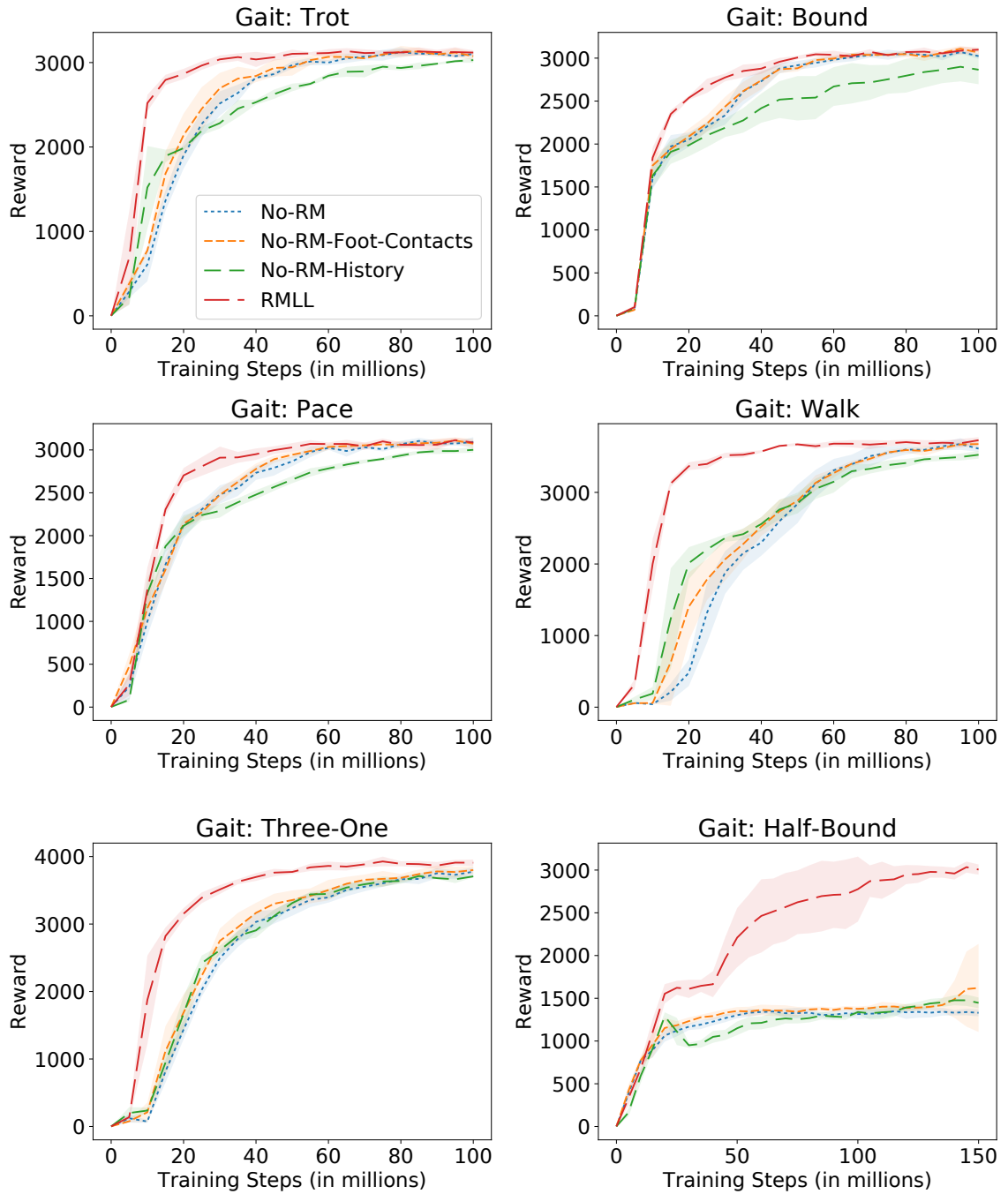
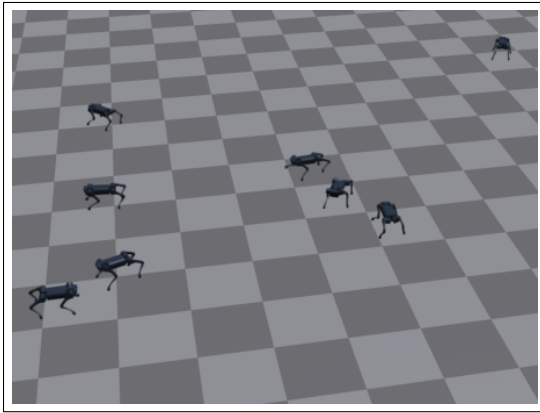
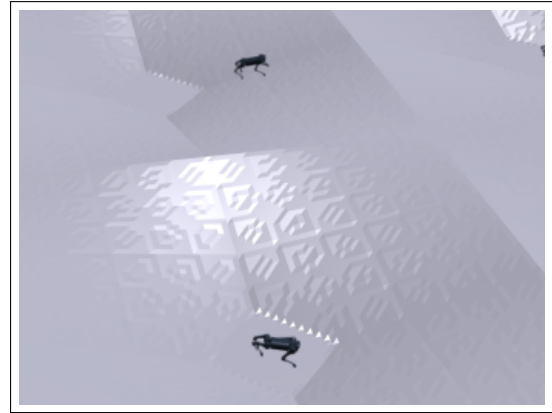


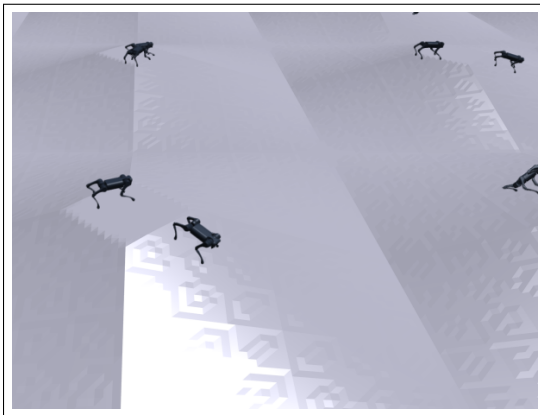
Figure 3.10: Reward curves for all gaits. RMLL more efficiently accumulates reward for each gait, particularly for the gaits with more complex foot contact sequences Walk, Three-One, and Half-Bound.



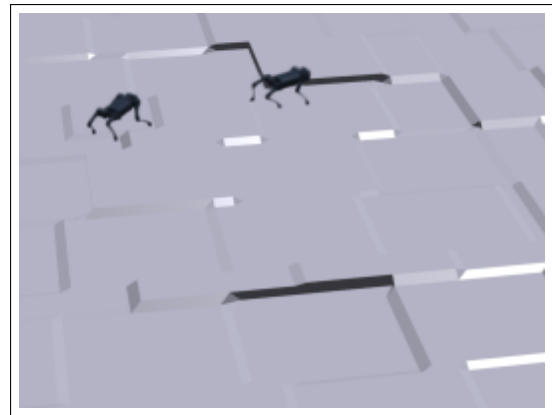
Flat



Uphill



Downhill



Stones

Figure 3.11: Visualization of each terrain we measured energy consumption and stability on.

Gait		Flat	Uphill	Uphill (S)	Downhill	Downhill (S)	Stones
Trot	E	2165.04	2628.88	5232.26	2137.38	2606.82	3149.81
	S	0.00	1.49	5.51	0.10	6.35	2.30
Bound	E	4657.94	4962.13	5118.81	4091.22	3187.78	5099.17
	S	0.00	3.09	5.70	2.77	6.79	2.16
Pace	E	2998.95	3023.70	3489.12	2811.13	2243.44	3493.30
	S	0.00	1.65	3.09	0.69	5.02	1.33
Walk	E	4271.82	4828.25	6048.98	3747.29	2828.21	4354.12
	S	0.00	0.70	2.73	0.56	4.54	1.28
Three-One	E	3145.81	3573.74	4118.70	3096.75	2509.71	3871.34
	S	0.00	1.45	3.22	1.79	5.27	1.89
Half-Bound	E	4873.29	5568.86	5935.81	4399.62	3594.13	5308.95
	S	0.00	3.56	7.87	3.18	6.76	3.25

Table 3.2: Comparing average energy consumption per meter (rows labelled as E) and stability (rows labelled as S) across different gaits and terrains.

3.5.3 Gait Comparisons

In order to further motivate learning different gaits, we compare energy consumption and stability for each gait across different terrains. We deploy each of our trained policies in simulation, over four different types of terrains: $\{Flat, Uphill, Downhill, Stones\}$ (See Figure 3.11). We also experiment with a version of *Uphill* and *Downhill* with steeper slopes, referred to as *Uphill (S)* and *Downhill (S)* respectively. Each policy (five policies over five random seeds per gait) is run for 20000 seconds per terrain type, and samples a base velocity command from [0.5, 1.0] meters/second ([0.25, 0.5] meters/second for **Walk**) and gait frequency command from the same range used during training. We measure energy consumption by multiplying motor torques by motor velocities, in the same manner as related work [32, 34]. Stability is reported as the average number of falls over all random seeds. We consider the robot to have fallen over when the base touches the ground, and reset the episode on a fall. Results are reported in Table 3.2. We find that **Trot** consumes the

least energy on most terrains, although **Pace** consumes the least energy on steeper slopes. Meanwhile, **Walk** is the most stable on all terrains except for *Downhill*, where **Trot** is the most stable.

3.5.4 Qualitative Results

Foot Contacts In order to better highlight the behavior of each of the six gaits we learned, we report foot contact plots for each gait. In simulation, we deploy each of our six gaits at a constant linear velocity, while setting $c_f = 6$ for all gaits. We record the foot contacts of each gait in Figure 3.12, which shows that each of our gaits follows the expected foot contact sequence and gait frequency. For example, green and orange bars in **Trot** are synchronized, indicating BR/FL feet are coordinated.

Hardware Demonstration We transfer each of our six learned policies from simulation to a real Unitree A1 robot, without any additional fine-tuning. Each trial is on an outdoor concrete walkway, where we increase and decrease gait frequency throughout the trial. The robot is sent velocity commands in real time via a joystick, operated by a human. We find that RMLL policies from all gaits successfully transfer to hardware, and the intended foot contact sequence is realized. We also find the robot is capable of dynamically adjusting gait frequency according to the gait frequency command.

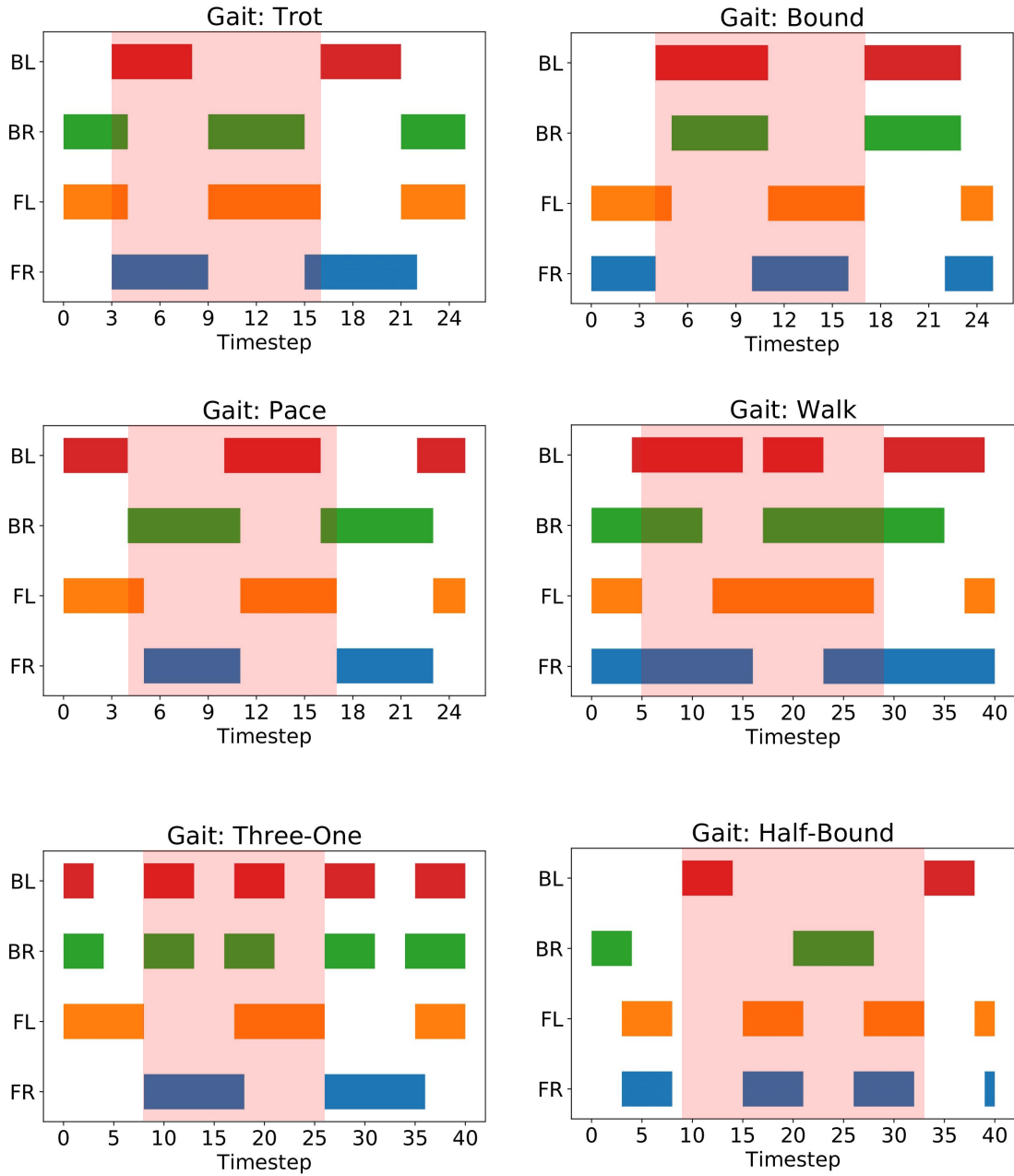


Figure 3.12: Foot contact plots for each gait learned using RMLL. We report foot contacts from trials in simulation where we deploy each gait with a constant forward velocity command, zero angular velocity command, and constant gait frequency command. The colored horizontal bars indicate when the specified foot makes contact with the ground. The red highlighted region in each subplot denotes a full cycle of the gait.

3.6 Discussion

Limitations and Next Steps in Rule-Based Gait Design While the approach discussed in this chapter can be used to easily specify and learn customized locomotion gaits and gait frequencies, we have not studied how to optimally leverage these different gaits to efficiently traverse various terrains, nor have we studied how to smoothly transition between gaits. In future work, researchers can train a wide variety of gaits, and learn how and when to transition between gaits and gait frequencies to most quickly or efficiently traverse different terrains. This can involve a two-stage procedure which trains a high-level policy to select pre-trained RM policies based on current terrain conditions, or an end-to-end approach which concurrently trains RM-based locomotion policies and a gait selector. Another interesting direction is to specify and dynamically adjust other gait parameters such as stride length, base height, or foot height. Finally, another avenue for future research is to investigate extracting descriptions of novel gaits from pre-trained large language models [66, 65], and to convert the descriptions to formal representations from which RL agents can learn locomotion policies.

RMLL Summary We leverage reward machines to easily learn and specify different quadruped locomotion gaits and gait frequencies. This is done via automata over simple logical rules which specify desired foot contact sequences. We efficiently train locomotion policies in simulation which learn these specified gaits over a range of gait frequencies, without the use of motion priors, dynamics models, or significant manual efforts when compared to existing work. We demonstrate policies of each of our six gaits on hardware,

and find that our robot can perform a variety of different gaits, while dynamically adjusting gait frequency.

In this chapter, we leverage human domain knowledge in the form of logical rules over foot contact sequences in order to easily specify and learn diverse locomotion behaviors. In the following chapters, we leverage different forms of human knowledge which we integrate in legged robot systems.

4 Robotic Guide Dog Navigation with Force

Responsive Locomotion Control

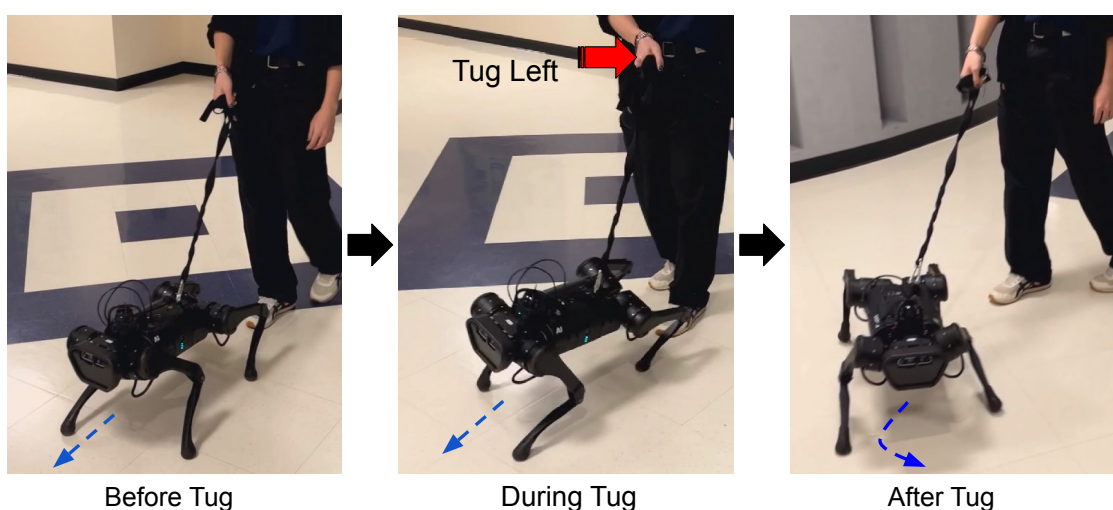


Figure 4.1: Tugs on the robot guide dog are used as navigation cues during blindfolded navigation.

4.1 Introduction

In the typical guide dog setting, a human holds a rigid handle attached to a harness on a dog. The dog will safely guide the human around nearby obstacles, while the human can tug on the dog to indicate which general direction to move in. Thus, the human has some idea of where they are, and is capable of making high-level navigation decisions, while the dog can better sense its immediate surroundings for obstacle avoidance and locomotion. Guide

dogs have been shown to improve the lives of visually impaired people, via increasing independence, confidence, companionship, and mobility [46]. Unfortunately, guide dogs need roughly two years of training, and cost over \$50,000 USD per dog [68]. Despite efforts to reduce the production cost of guide dogs [47], their supply is still significantly lower than demand.

To better meet this demand, and potentially improve performance, robotic guide dog systems are being developed [48, 49, 50, 51, 69, 52]. There is a growing interest in the development of systems of this type, which have undergone various human studies in order to measure the extent to which they are compatible with visually impaired humans, and their level of societal acceptance [70, 71, 72]. While these robotic guide dog systems have successfully demonstrated navigation tasks, none of them have considered settings involving human tugs, which is very common for guide dogs. It's important for robotic guide dogs to be robust to human forces, as the human is constantly holding a rigid handle directly attached to the robot, and large forces can cause the robot to stray from the optimal path, or even fall over. Furthermore, a typical method of communication between a human and a real guide dog is through tugs. This makes force estimation useful for a robotic guide dog, as knowledge of the direction and magnitude of the applied force can be used to better facilitate navigation according to the human's intentions (see Figure 4.1).

To address the above mentioned issues in human-robot communication, we develop a novel robotic guide dog system which is robust to external forces, and estimates the magnitude and direction of these forces to determine which navigation actions to take for human-robot co-navigation. We achieve this by simultaneously training a locomotion controller via RL, and a force estimator via supervised learning. Our locomotion policy is trained

over simulated tugs by sampling different base velocities which are suddenly applied to the robot [8]. These tugs serve as labels for training our force estimator, and are estimated during deployment.

While the controller is running on a real robot, the force estimator estimates the direction and magnitude of force the human applies. These force estimates are computed exclusively from sensors onboard the robot (joint encoders and IMU). From force estimates, the robot detects when and in what direction the human tugs occur. This informs the robot which direction to go at a global level, while a local planner using information from a LIDAR sensor is used to navigate the immediate environment. Different from existing robotic guide dog systems that require major hardware upgrades, e.g., customized traction device [50], or button interface [51], our robotic guide dog system needs is compatible with any attachable leash or handle.

Our main contributions include the following:

1. The first robotic guide dog system which takes directional cues via human tugs, while also safely navigating the immediate environment.
2. A force tolerant locomotion controller, jointly trained with a force estimator which can estimate the magnitude and direction of human forces.
3. Experimental results in simulation and on hardware to evaluate the robustness of our locomotion controller and accuracy of our force estimator.
4. Demonstration of our robotic guide dog system in an indoor environment with a blindfolded human.

4.2 Related Work

Various robotic guide dog systems have been demonstrated for the task of blindfolded navigation. Among these works, one of them considers an MPC-based motion planner for a wheeled robot [48]. Another considers an optimization based approach for path planning, which models a taut or slack leash [49]. A third system designs an adjustable leash, and optimizes for human comfort during navigation [50]. These works all stray from real-world guide dog settings, in part because they assume the robot has full knowledge of the destination beforehand, and that the human does not need to communicate with the robot during navigation. In real-world settings, the human must decide on which high-level navigation actions to take, and communicate these actions to the robot.

More similar to our work, other approaches consider settings where the human communicates high-level directions to the robot during navigation [51, 69, 52]. However, the medium of communication in these works are either a custom designed handle with buttons [51], predefined directional actions prior to navigation [69], or verbal commands [52]. In our work, the human communicates directional cues via tugging, and is compatible with any rigid connection to the robot. Additionally, none of these works consider human forces being applied to the robot, making their systems unresponsive or susceptible to failure upon navigation under human forces.

Numerous quadruped locomotion controllers have been developed, many of which are robust to external forces [3, 1, 4, 6, 36, 73, 34, 9, 8, 33, 10, 74]. These works do not explicitly estimate the applied forces, as the focus of these works is to develop controllers that are generally robust to varying environments. In our work, we explicitly estimate forces

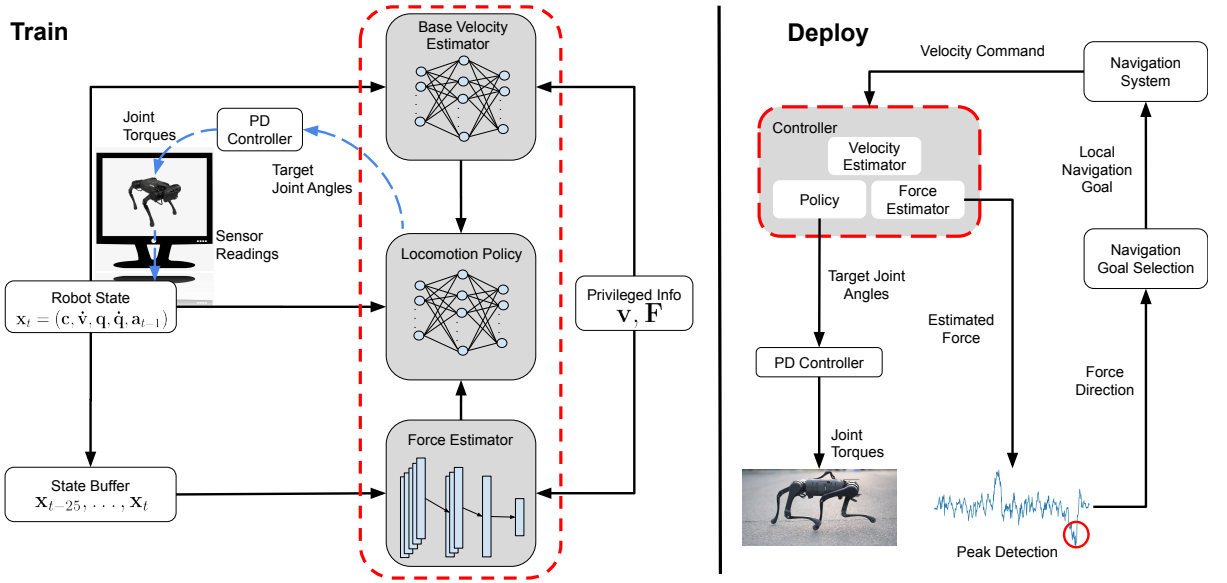


Figure 4.2: Overview of our approach. Our locomotion controller (circled in red) contains a velocity estimator, force estimator, and locomotion policy, all of which are trained in simulation. The base velocity estimator and force estimator are trained via supervised learning, using privileged information from the simulator as labels. The locomotion policy is trained via RL, and outputs target joint angles to a PD controller which converts them to joint torques which are directly applied to the robot. During deployment, our locomotion controller estimates external force at each time step. Force direction is derived from peaks in the estimated force signal. The direction of force determines the next local navigation goal for our navigation system to take, which returns velocity commands to our controller.

to the robot generated by human tugs, in order to communicate the human’s navigation intentions with the robot.

4.3 Robotic Guide Dog System

In this section, we present our robotic guide dog system that is robust and responsive to external forces from human tugs. Figure. 5.2 presents an overview of how we train our locomotion controller, and deploy for guided navigation.

4.3.1 Locomotion Controller

We train our locomotion controller via RL, which models environments as a Markov Decision Process (MDP). An MDP is defined as $M = (S, A, T, R, \gamma)$, where S is the set of

Table 4.1: All terms of the reward function our locomotion policy is trained on. $\mathbf{v}$ refers to base velocity, $\mathbf{c}$ refers to commanded linear and angular base velocity, $\boldsymbol{\omega}$ refers to base angular velocity, $\boldsymbol{\tau}$ refers to joint torques, $\dot{\mathbf{q}}$ refers to joint velocities, $\mathbf{t}_{\text{air}}$ refers to each foots air time, $\mathbf{a}$ refers to an action, and dt refers to the simulation time step.

Term Description	Definition	Scale
Linear Velocity x, y	$\exp(-\ \mathbf{c}_{x,y} - \mathbf{v}_{x,y}\ ^2/0.25)$	$1.0dt$
Linear Velocity z	$\mathbf{v}_z^2$	$-2.0dt$
Angular Velocity x, y	$\ \boldsymbol{\omega}_{x,y}\ ^2$	$-0.05dt$
Angular Velocity z	$\exp(-(\mathbf{c}_\omega - \boldsymbol{\omega}_z)^2/0.25)$	$0.5dt$
Joint Torques	$\ \boldsymbol{\tau}\ ^2$	$-0.0002dt$
Joint Accelerations	$\ (\dot{\mathbf{q}}_{\text{last}} - \dot{\mathbf{q}})/dt\ ^2$	$-2.5e-7dt$
Feet Air Time	$\sum_{f=1}^4 (\mathbf{t}_{\text{air},f} - 0.5)$	$1.0dt$
Action Rate	$\ \mathbf{a}_{\text{last}} - \mathbf{a}\ ^2$	$-0.01dt$

states, A is the set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is the transition function which outputs the probability of reaching state s' given state s and action a , $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function which returns feedback from taking action a from state s and ending up in state s' , and $\gamma \in [0, 1]$ is the discount factor which determines how valuable future reward should be considered in comparison to immediate reward.

We define a robot state at time t as $\mathbf{x}_t = (\mathbf{c}, \dot{\mathbf{v}}, \mathbf{q}, \dot{\mathbf{q}}, \mathbf{a}_{t-1})$, where $\mathbf{c} = (c_x, c_y, c_\omega)$ is the commanded base linear and angular velocity, $\dot{\mathbf{v}}$ is the base acceleration, $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the joint angles and velocities respectively, and $\mathbf{a}_{t-1}$ is the action taken at time $t - 1$. Actions are target joint angles, which are converted to torques via a PD controller. The reward function encourages tracking commands $\mathbf{c}$ while minimizing energy consumption and large action changes [8] – fully defined in Table 4.1.

Our locomotion policy also takes the base velocity and external force vector as input, to more easily learn to track commands $\mathbf{c}$ and respond to external forces. These variables are not easily estimated through robot sensors. Thus, we train state estimators via supervised learning over privileged information, which has been shown to be more effective than

classical methods, e.g., Kalman Filters [7]. It is a common setting to learn with privileged information in simulation, and insert a corresponding state estimator in real-world deployment. In line with those systems, we train a velocity estimator and force estimator for the real robot to bridge the sim-to-real gap in locomotion policy learning. These estimators are trained jointly with the locomotion policy, where $\mathbf{v} = (v_x, v_y, v_z)$ is ground truth base velocity, and $\mathbf{F} = (F_x, F_y, F_z)$ is ground truth external force, both obtained as privileged information from the simulator.

Two variations of forces are applied to the robot during training. One variation is small and frequent backward pushes, which are designed to simulate a human following the robot with a taut leash, where a human incidentally applies frequent small backward forces on the robot as they are being guided. The other variation is larger, less frequent pushes occurring in any direction. These pushes are designed to simulate human directional tugs, in which the human intentionally tugs the robot to communicate the direction they want to move in. The force estimator is only trained on data from the second variation of tugs, as we do not want to detect the small, incidental backward tugs which naturally occur during guided navigation.

The base velocity estimator is a multilayer perceptron (MLP), whose parameters are updated over the same data as the locomotion policy. To estimate forces, we find it helpful to access a history of states, to better capture the robot’s behavior over the duration of the applied force. Thus, similar to training adaptation modules [6], our force estimator uses 1-D convolutional layers to capture temporal relationships between states. Force estimator parameters are updated less frequently than the base velocity estimator and locomotion policy, because most time steps do not include external forces. Thus, most of the labels

for our force estimator includes zero vectors, indicating that no force was applied at those time steps. This causes imbalanced training data, which we resolve by only training the force estimator when the training data contains nonzero forces. We then further re-balance this to ensure at most 20% of the force estimator’s training samples include zero vectors as labels.

4.3.2 Robotic Guide Dog Navigation

To perform navigation tasks with our robotic guide dog, we need to estimate when and in what general direction a force is being applied. This is done by running peak detection [75] on the previous 200 time steps of the estimated force signal F_y , at a rate of 2Hz. In this work, we only analyze F_y , to determine whether a left or right tug has occurred. A peak in the signal indicates that the force estimator detected a significant external force applied to the robot. Thus, if a peak is detected within the past 50 time steps, then we consider it as a recent tug applied by the human. Positive peaks correspond to left tugs, while negative peaks correspond to right tugs.

Local navigation goals are then selected based on the robot’s location, orientation, and tug direction. A domain expert manually labels a map with *decision points*, where the human must decide which direction to go in next, based on the direction they came from, and their tug direction. A labelled map of the hallway domain we demonstrate our system on can be seen in Figure 4.3.

The local navigation goal is then sent to our navigation system, which uses a LIDAR sensor to localize itself on the map via AMCL [76], and compute local plans via DWA [77]. The local planner returns velocity commands $\mathbf{c}$ to our controller, which our controller

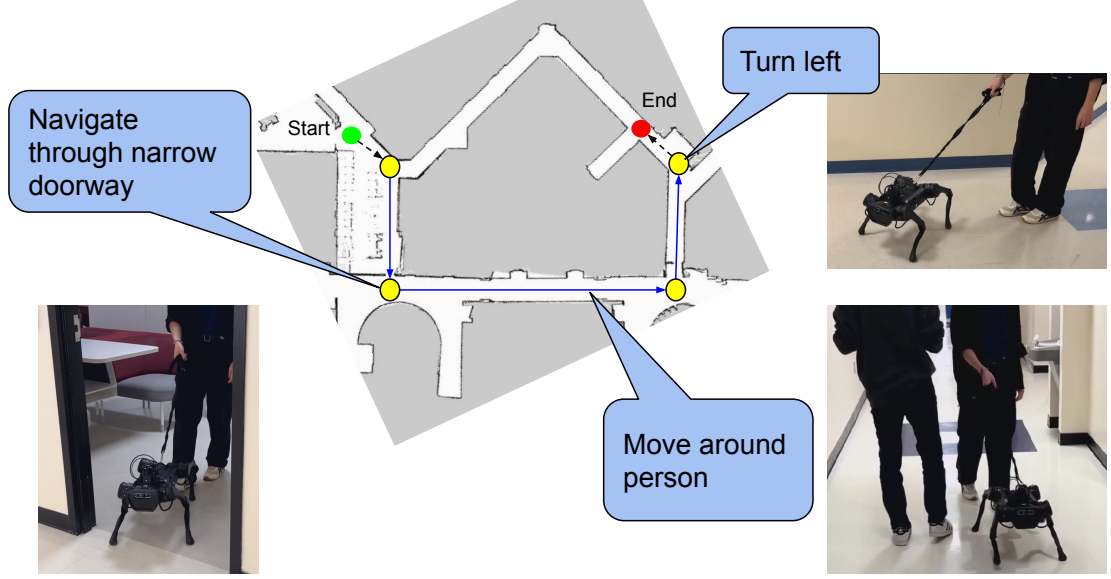


Figure 4.3: Map of our navigation environment. Yellow circles correspond to *decision points*, where the human needs to decide which direction to move in via tugging. The blue lines indicate the path the human took in our demonstration.

tracks.

4.4 Implementation Details

We use Isaac Gym [27] physics simulator, and train our controller with 2048 robots in parallel [8]. Our locomotion policy is trained via PPO [23], while our base velocity estimator and force estimator are trained via supervised learning, with Mean Squared Error loss. Our locomotion policy and base velocity estimator are both MLPs with three and two hidden layers respectively, while our force estimator is a convolutional neural network with three 1-D convolutional layers, which makes predictions over the past 25 time steps. The PD controller converts target joint angles to torques with proportional gain set to 20 and derivative gain set to 0.5. The policy is queried at 50Hz, and control signals are sent at 200Hz. New velocity commands are sampled after each episode, where c_x , c_y , and c_ω

are sampled uniformly from $[-1,1]$. An episode terminates when any link other than a foot touches the ground, the base height is below 0.25 meters, or the episode has lasted 20 seconds.

To better facilitate sim-to-real transfer, we train over `RANDOM_UNIFORM_TERRAIN` which increases in difficulty based on a curriculum [8]. We also include noise in observations, and domain randomization over different surface frictions.

We add random external forces in our environment, to improve robustness of our locomotion policy and collect data to train our force estimator. The small and frequent backward pushes occur every 0.6 seconds, have a duration of 0.1 seconds, and sets the base velocity to 0.25 m/s backward. Meanwhile, the large and infrequent pushes used to train the force estimator occur every 3 seconds, have a duration sampled from $[0.24, 0.48]$ seconds, and sets the base velocity to a vector sampled from $F_x \in [-0.75, 0.75]$, $F_y \in [-0.75, 0.75]$, and $F_z \in [0, 0.1]$. Note that forces from $\mathbf{F}$ are implemented as spontaneous updates in base velocity.

4.5 Experiments on Force Tolerance and Accuracy

We design experiments to evaluate the robustness of our locomotion controller, and accuracy of our force estimator. Although we are able to learn a force estimator in simulation, we could not directly evaluate it in the real world due to the missing ground-truth values. Instead, we chose to evaluate tug detection (LEFT, RIGHT, and NONE) on the real robot, where the participants followed our instructions, and hence ground truth was available. We then demonstrate our full robotic guide dog system via guiding a blindfolded human.

4.5.1 Force Tolerance Evaluation

To evaluate whether our learned force controller is actually robust to external forces, we run experiments in simulation where we apply random forces to the base of the robot. In each trial, a single force in a random direction is applied, whose duration is sampled from $[0.25, 0.5]$ seconds, and strength is sampled from $[25, 100]$ Newtons. Meanwhile, the robot is commanded to walk forward at 0.5 meters/second, for a duration of five seconds.

We run this experiment on three different types of locomotion controllers, for 1000 trials each. The controllers include a commonly used MPC controller [78], a variant of our learned controller which does not consider estimated force in its state (referred to as *Learned No Est*), and our controller described in Section 4.3.1 (referred to as *Learned Est*). All controllers are deployed in PyBullet [26].

We measure how frequently the robot fell across all trials (Proportion Fell), and how far the external force caused the robot to drift from its current trajectory on average (Drift from Trajectory). We consider a robot to have fell if a non-foot part of the robot touches the ground. Results are reported in Table 4.2, which indicate that our learned controllers fell significantly less frequently, and better maintained velocity tracking under external forces than the MPC controller. Including estimated forces in the state appears to marginally improve robustness. Note that for our two types of learning-based controllers, we train over five different random seeds each and average the results.

4.5.2 Force Estimation Evaluation

We evaluate the accuracy of our force estimator through experiments in simulation, and on hardware.

Table 4.2: Our learned controllers fell significantly less frequently, and better tracked velocity commands under external forces when compared to an MPC controller. Including the output of the force estimator in the state marginally improved robustness. The large variance in drift is caused by the large range of force strengths and directions we sample from.

Controller	Proportion Fell	Drift from Trajectory
MPC	0.5990	1.1256 ± 0.5908
Learned No Est	0.1904	0.6824 ± 0.5447
Learned Est	0.1762	0.6790 ± 0.5309

Simulation

We deploy our learned controllers in PyBullet with a velocity command of 0.5 meter/s/second, and a single external push per trial, for 1000 trials. Each push has a force whose x-component is sampled from $[-50, 50]$ Newtons, and a y-component which is at a fixed magnitude, and random direction (either left or right). There are three possible classes the force estimator can predict over: $\{\text{LEFT}, \text{RIGHT}, \text{NONE}\}$. Each trial includes 150 time steps and takes 3 seconds. In each trial, the force estimator is queried every 25 time steps, or six total queries per trial. Thus, in each trial, the force estimator makes a total of six predictions. A trial is deemed correct if one of the six queries matches the force direction being applied in the simulator.

In order to evaluate whether force direction can be estimated through only a history of base velocities, we train a baseline force estimator which only makes force estimates based on a history of ground-truth base velocity and base velocity commands. We refer to this baseline as *only vel*, while our force estimator trained over the full state and described in Section 4.3.1 is referred to as *full state*.

In Figure 4.4, we report the accuracy and false positive rates of our force estimators over varying force strengths. Accuracy is computed by dividing the number of trials in

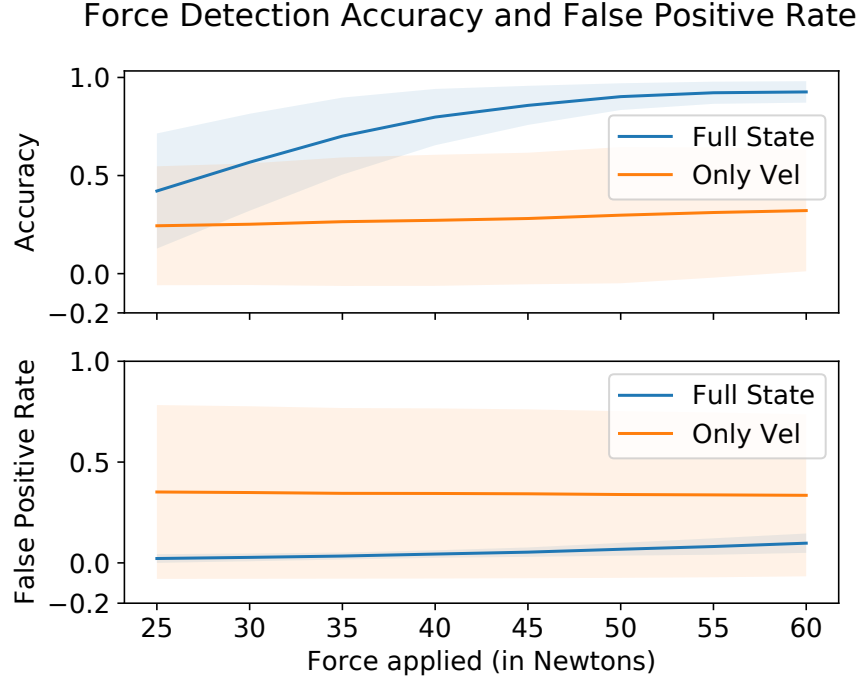


Figure 4.4: We report the accuracy and false positive rate of our force estimators, given forces of varied strength. The shaded region indicates the standard deviation between the five policies trained over five different random seeds.

which the force estimator predicted the correct force direction at any of the six queries in the trial, by the number of total trials. Computing accuracy alone in this manner is not informative enough, because it is possible to achieve high accuracy by predicting LEFT at one point in the trial, and RIGHT at a different point within the same trial.

Thus, we also consider the false positive rate, which we compute by dividing the number of extra forces (LEFT or RIGHT predicted when ground truth is NONE) predicted during the trial, by the number of times the force estimator is queried (every 25 time steps). A high false positive rate corresponds to the force estimator oftentimes predicting forces when they do not occur. As force strength increases, our estimators achieve a higher accuracy while maintaining a relatively low false positive rate. Results indicate that knowledge of the full state is significantly beneficial in estimating force direction, when compared to

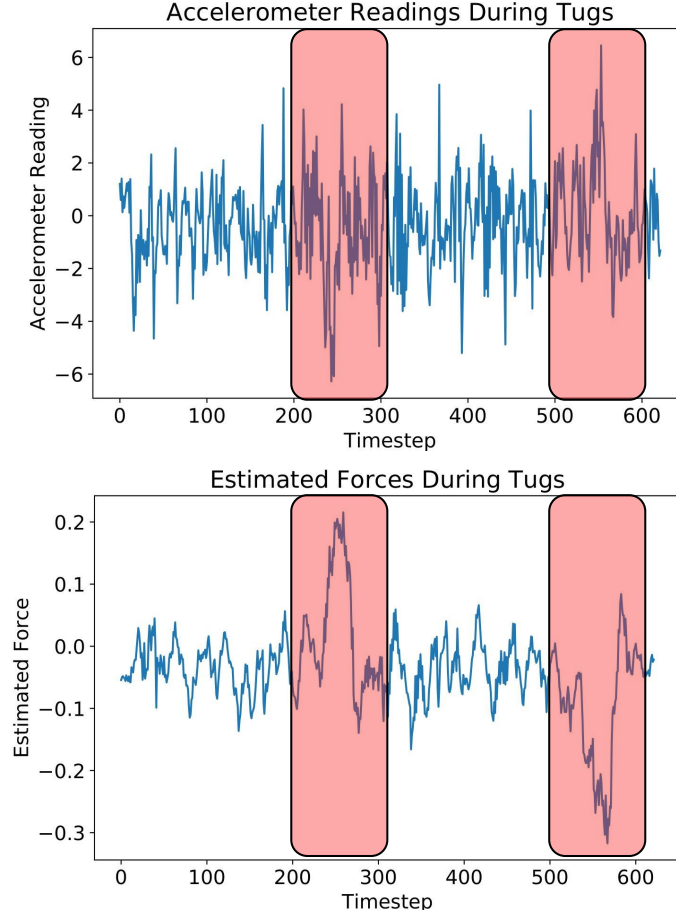


Figure 4.5: Measured acceleration (top) and estimated force (bottom) during a single trial. Tugs are denoted by red boxes.

a force estimator which is only trained over base velocity information.

Hardware

When a human tugs our robotic guide dog with sufficient force, the base of the robot will momentarily accelerate in the direction of the tug. Thus, one might wonder why we do not consider accelerometer signals to detect tug direction, rather than train a force estimator. In this sub-section, we validate the usefulness of our estimated force signals, which we compare to accelerometer readings.

In this experiment, we command a real Unitree A1 robot to move forward at 0.5m/s,

while a human participant tugs left after a few seconds of forward locomotion, followed by a right tug after another few seconds of locomotion. This trial is performed by four human participants, three of which have no prior experience in operating this tugging interface. Each participant is a robotics researcher in a university lab. The three participants with no prior experience with this system were given a demonstration of an example trial, before completing their own trials. In total, 42 trials were conducted, and data was collected from 40 trials (two trials were removed due to the robot falling over and data not being saved). Of these 40 trials, each participant performed ten of them.

Forces are detected every 25 timesteps, and can be predicted as one of three classes. We compute the accuracy and false positive rate for force detectors using the estimated force signal, compared to force detection using the signal directly from the accelerometer on the robot. Accuracy is computed as the percentage of trials which contain a LEFT force prediction before the halfway point of the trial, and a RIGHT force prediction after the halfway point of the trial. False positive rate is the average percentage of false positive forces being predicted across all trials. A force prediction is considered as a false positive if it is either LEFT or RIGHT, and does not correspond to the first expected LEFT force or later expected RIGHT force.

Results are reported in Table 4.3, where we find the force estimator more accurately predicted the correct forces, with fewer false positives compared to predictions from raw accelerometer signals. Both methods of detecting tug direction (accelerometer and force estimator) perform worse for beginner participants, indicating that adding diverse tugging styles is important for a more complete evaluation of force estimators.

In Figure 4.5, we plot the raw signals from the accelerometer and force estimator from

Table 4.3: Force estimation via accelerometer readings vs force estimator signal.

Participant Type	Method	Accuracy	False Positive Rate
Expert (10 trials)	Accelerometer	0.20	0.2149
	Force Estimator	1.00	0.0442
Beginner (30 trials)	Accelerometer	0.13	0.1906
	Force Estimator	0.70	0.0498

a single trial. We find the accelerometer signal is more noisy than the estimated force signal, which we hypothesize is because the force estimator has access to other sensor information along with accelerometer readings. Note that our force estimator is co-learned with the locomotion policy. The locomotion policy takes the estimated force as input during training, and the states generated from the policy are used to train the force estimator. We believe this co-learning mechanism leverages the additional sensor information to help the force estimator outperform the tug detection from raw accelerometer readings.

4.6 Hardware Demonstration

We demonstrate our full robotic guide dog system on a Unitree A1 robot, in an indoor hallway environment (see link in Abstract). Our locomotion policy and force estimator are deployed on hardware without any additional fine-tuning.

In this demonstration, a human is blindfolded, and holding a taut leash attached to the robot. The human desires to reach some particular goal location, and chooses the route to take by tugging the robot at decision points, while the robot autonomously avoids obstacles (including boxes, narrow doorways, and another human) along the way. Decision points occur at intersections in the hallway, where the human needs to decide in which direction they want to go. Similar to the setting in [51], the blindfolded human is not familiar with localizing himself without vision, so another sighted human verbally indicates

when a decision point is approaching.

This demonstration indicates that our locomotion policy and force detector are transferable to hardware, and can cooperate with a local planner to enable blindfolded navigation. Thus, successful human-robot communication occurred, such that the human avoided all obstacles while the robot navigated to the desired goal location through detecting human tugs at decision points.

4.7 Discussion

Limitations and Next Steps in Robotic Guide Dog Development While our robotic guide dog system can avoid obstacles and select routes at a course level in indoor hallway environments, real guide dog settings include outdoor environments, and situations with many possible directions to navigate in. In future work, researchers can leverage methods to determine which paths are traversable [12], and train force detectors which can estimate the direction of force at a finer level. Another future direction is to incorporate intelligent disobedience, which refers to rejecting a human’s navigation decision if unsafe [79]. Additionally, The evaluation can be further improved by replacing sighted people with those with visual impairments in the experiments. Finally, our robot’s locomotion speed is relatively low. It might be the human tugs, the learned locomotion policy, or both causing the low speed. Further investigation into those factors can potentially lead to very interesting future research and higher-speed robotic guide dog systems.

Force-Responsive Robotic Guide Dog Summary We train a locomotion controller which is tolerant to human tugs, and can estimate the direction of external forces. We evaluate

the robustness of our controller, and accuracy of our force estimator through experiments in simulation and on hardware. Finally, we demonstrate our controller on a real robot for the task of blindfolded navigation, where a blindfolded human is successfully guided to a destination, while giving directional cues through tugging on the robot.

In this chapter, we integrate human common-sense knowledge and global spatial awareness knowledge for collaborative decision making for navigation tasks in human-robot teams. In the next chapter, we further leverage human spatial knowledge in the form of floor plan maps for task planning and navigation.

5 Vision Language Models Can Parse Floor Plan Maps

5.1 Introduction

A key to mobile robotics is a deep understanding of the geometric configuration of the world that mobile robots live in. As a result, many mobile robots need some forms of a *map* for localization, obstacle avoidance and navigation. To build such maps, the robots use a predefined data structure, e.g., an occupancy grid [80] or visual features [81], and then perform simultaneous localization and mapping (SLAM). Human beings have a long history of building and using maps. These days one can easily read floorplan maps of airports and shopping centers, and figure out a plan for navigation. By comparison, robots can hardly reach comparable competency in map reading and task planning. It is a non-trivial task for the robots due to the many labels in the map, their ambiguous associations to different areas, and complex geometric configurations. As a result, there is no existing method for addressing the *map parsing* problem, i.e., computing a navigation plan given a map image and a goal text, which motivated this research.

For complex navigation tasks, a robot needs to compute a task plan, i.e., a sequence of navigation actions, and continuous trajectories for realizing those actions. Example actions can be *entering an area* and *going through a door*. Extensive engineering efforts

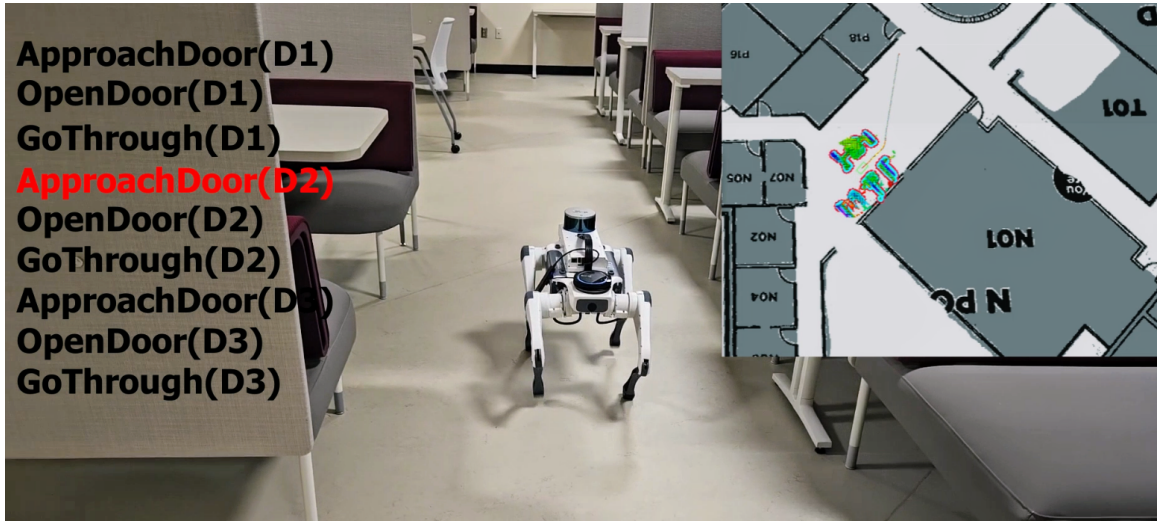


Figure 5.1: Quadruped robot executing a VLM-generated plan (current action highlighted in red) to complete a navigation task while localizing directly on a floor plan image.

are needed to realize such navigation systems, from building the map itself to labeling areas of the map. At the same time, professional architectural drafters have generated blueprints that accurately reflect the geometric configurations, which unfortunately cannot be used by current robots. From an application perspective, this research aims to leverage the readily accessible floorplan maps in human environments to fulfill the robots’ need of maps for navigation.

Vision language models (VLMs) are foundation models that learn from and reason about both images and texts, supporting a variety of downstream tasks from visual question answering to image captioning. VLMs have demonstrated impressive successes in a variety of applications [82, 83, 84] including those in robotics [85, 86]. We, for the first time, apply VLMs to the novel task of map parsing and evaluate its performance in navigation, a foundational problem in mobile robotics. Our approach is simple and intuitive. A floorplan map and a problem description, including the start and goal positions, are provided to a VLM, and the task is to compute a plan (i.e., an action sequence) to achieve the

goal. Despite the straightforward idea, the results are surprisingly good. Navigation plans generated by VLMs which can require a sequence of nine actions are generated correctly up to 90% of the time on some floorplans.

The main contribution of this research includes the introduction of the map parsing problem, evaluations of VLMs on this problem, practical suggestions that paves the way for future research on VLM-based map parsing, and a complete demonstration of real-robot system.

There are limitations in this research that can be addressed in future work. One is that we still need to slightly edit the map image, such as thickening walls and removing architectural annotations, to produce the best performance. Such steps can be automated in future work. Another is that the robot needs to stand close and forward-facing when capturing the map image. This can be a challenge to small robots because most floorplan maps are placed at human heights. In this chapter, we focus on highlighting the remarkable performance of VLMs on map parsing, and leave those topics to future work.

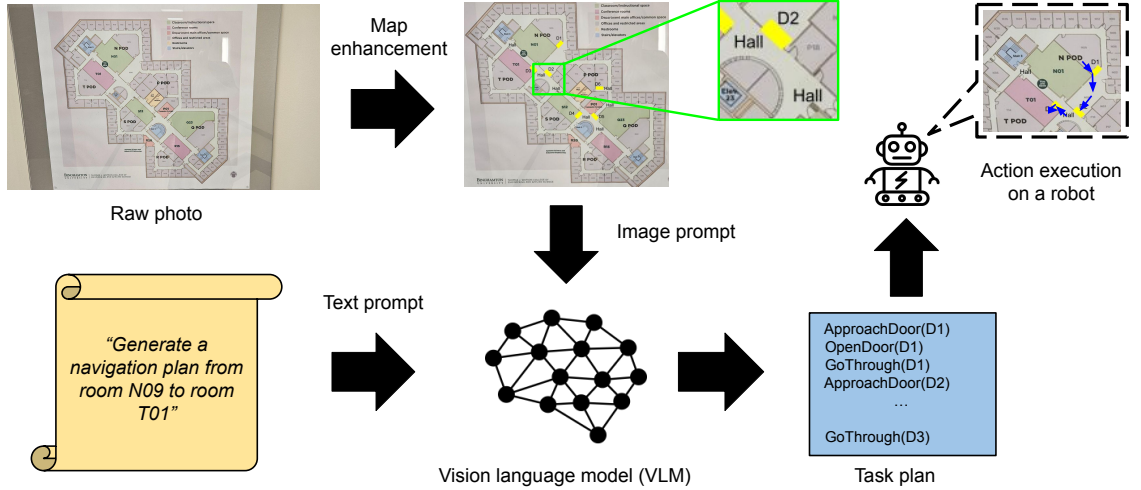


Figure 5.2: Overview of our method. A robot takes a raw image of a floor plan, which is then enhanced with labels and door indicators. The enhanced floor plan, along with a text prompt specifying the start and goal locations is given to a VLM. The VLM generates a navigation plan to reach the goal location, and the plan is executed on a mobile robot.

5.2 Related Work

In this section, we discuss existing work in autonomous navigation for mobile robots, VLM prompting strategies, and integrating large pre-trained models in robotics. We highlight how our work differs from existing works in each of these categories.

5.2.1 Existing Map Representations and Navigation

Existing works demonstrating autonomous navigation on mobile robots usually require generating an occupancy grid [87, 88, 89, 67], or leveraging vision-based methods for simultaneous localization and mapping (Visual SLAM) [81, 90, 91, 92, 51]. While such map representations have proven to be effective for autonomous navigation tasks, generating an accurate map is oftentimes labor-intensive. For instance, in vision-based settings, the robot has limited knowledge of the global environment, either leading to lengthy exploration, or navigational commands from a human. In this work, we greatly reduce the effort of gener-

ating accurate maps while still leveraging detailed information of the environment through the use of existing, and potentially in situ, floor plans.

5.2.2 VLM Prompting Strategies

The output of a large pre-trained model largely relies on the way it is prompted. Strategies like chain-of-thought [93] and in-context learning [94] are leveraged on LLMs to improve performance. Similar to language prompts, image prompting strategies can improve VLM outputs. Set-of Mark prompting, which segments and labels objects in an image has shown to improve VLM responses [95]. In our work, we design a new visual prompting strategy designed for obtaining a spatial understanding of floor plan images.

5.2.3 Large Models in Robotics

To improve the common-sense reasoning capabilities of robots, large pre-trained LLMs and VLMs have been integrated in robots for various tasks like housekeeping [96], object rearrangement [97], navigation [98, 99, 100, 101, 102], and quadruped locomotion [65, 66] to name a few. Alignment of the large model with the environment and robot’s skills is critical to perform tasks in the real world [103]. Various approaches have demonstrated planning capabilities [104, 105] and uncertainty estimation [106] of large models. Various visual prompting strategies designed for different robotic manipulation tasks have also been developed [85, 86]. In line with recent works that leverage large models to incorporate common-sense knowledge on robots, we generate feasible navigation plans directly from an image of a floor plan.

5.3 Methodology in Floor Plan Parsing

In this section, we present our approach for leveraging Vision-Language Models (VLMs) to interpret floor plan images and generate navigation instructions. We discuss the two key aspects of our approach: visual prompting strategy, and VLM-based plan generation.

5.3.1 Visual Prompting Strategy

Our study uses floor plan images that include detailed architectural layouts for various building types. To generate accurate plans from a VLM, it’s important to design a visual prompt which can facilitate learning the structure of the floor plan. Unfortunately, raw floor plan images tend to contain various markings (i.e. windows, furniture symbols, non-uniform wall thickness) which can potentially confuse the VLM in understanding the general layout of the floor plan. Thus, we remove such markings to produce a cleaner map which can be better leveraged by a VLM.

We find that removing extraneous details from the floor plan is insufficient for the VLM to understand the map layout. In particular, we find the VLM has limited spatial awareness for sparsely labelled rooms with lots of open space, and near key decision points like doors and intersections. To alleviate this, we add duplicate room labels in open spaces and near doors and intersections. This provides the VLM with further guidance in understanding the general structure of the floor plan. We later demonstrate the importance of such additional labels in Section 5.4. In this work, we manually remove extraneous markings and add additional room labels for visual prompt construction. We leave automated floor plan editing as future work.

Map enhancement with dense labeling: A key finding in this research is the importance of dense labeling in map enhancement. We apply a methodology in which rooms are labeled strategically at decision points (e.g., near doors or intersections). We later show that such label placement brings a considerable performance boost, which is particularly significant in complex environments where navigation paths can involve multiple rooms and transitions. This process can be automated (though not in this chapter) to make it scalable for broader applications without manual intervention.

5.3.2 VLM-Based Plan Generation

In our VLM-based framework, as shown in Fig. 5.2, VLMs formulate navigation plans based on a floor plan image and a text prompt. This method leverages an instruction-based text prompting strategy, which involves providing the VLM with explicit and detailed guidelines for the navigation task, along with the floor plan image. We now elaborate on the prompting strategy and the resulting output format.

The text prompt given to the VLM is shown in Fig. 5.3. The prompt explicitly defines the starting point and the destination. This allows the VLM to understand the required navigation path and objectives clearly. It provides detailed instructions on interpreting the floor plan and managing door interactions. These guidelines include specific protocols for door operations and decision-making processes.

A key aspect of this prompt is the request for all door and room connections. By generating this information at the start, we speculate that the VLM integrates it with the floor plan image to produce an accurate navigation sequence. We believe this step helps the VLM better understand the spatial relationships in the map, leading to accurate navigation

I am a robot that cannot go through walls and must use doors to navigate. This is the floor plan of the building I am in right now (provided as an image). You are a navigation agent, and your task is to give me a detailed, efficient navigation plan that strictly follows a sequence of actions to achieve the navigation task: Begin in **<location A>** and arrive at **<location B>**. The only doors which exist are represented as yellow rectangles and labeled with 'D(N)' distinct positive integers(1,2,3...N). A plan consists of a sequence of the following actions:

`ApproachDoor(x)`: Move in front of door x.

`OpenDoor(x)`: Open door x.

`GoThrough(x)`: Move through open door x to the location on the other side.

Include only the necessary doors that are part of the path being used, and do not mention doors that won't be traversed even if they are in the path.

Explicit Room and Door Descriptions: Alongside the image, make a clear list of all rooms and doors with their connections - which is to be used for the navigation task.

Remember that the door symbol can overlap with the boundaries or common spaces. Remember to only use the generated door room connections for making the action plan. Double-check if each action is necessary and correct for traversal to the end goal. Common spaces (eg Hall) and larger rooms may have multiple instances of the same labels to help you understand their boundaries.

Important: The doors close after every `GoThrough(x)` action. Carefully inspect the floor plan image to ensure the correct correspondence between doors and rooms. Prioritize providing a correct path over the shortest path. Make sure the path avoids any unnecessary doors or rooms. If any unnecessary doors or rooms are included, silently correct the plan before providing the final sequence. Give the final path in a json format.

Remember to make explicit connections for each door, then make a step by step solution for each navigation step and ONLY use the door-room connections to generate the final navigation plan.

Figure 5.3: Text prompt input to VLM to generate navigation plans. We define the starting and ending locations, action types, and ask for explicit room and door connections to gain insights as to how the VLM understands the map.

path planning.

Based on the text prompt and floor plan image, the VLM generates a sequence of actions required to navigate from the initial to goal location. This sequence includes specific steps, e.g., approaching, opening, and passing through doors.

The output from the VLM is a detailed sequence of actions formatted as follows:

- `ApproachDoor (x)` : Move in front of door x .
- `OpenDoor (x)` : Open door x .
- `GoThrough (x)` : Move through open door x to the location on the other side.

The final navigation plan is output in JSON format, specifying each action. This structured format facilitates easy interpretation and execution of the navigation instructions. This plan is then parsed and executed by the robot.

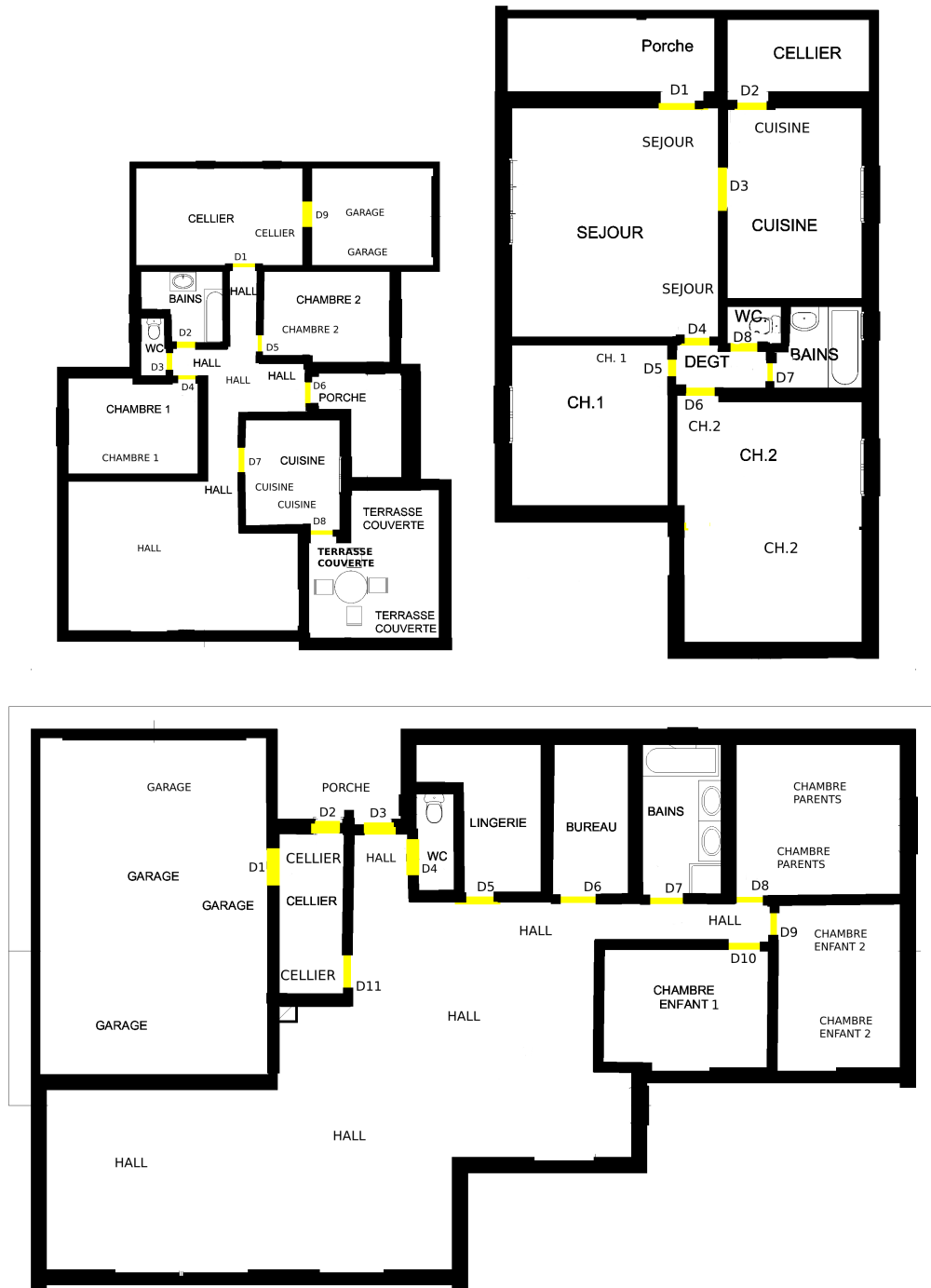


Figure 5.4: Three maps used in our experiments for evaluating the performance of VLMs in map parsing and plan generation.

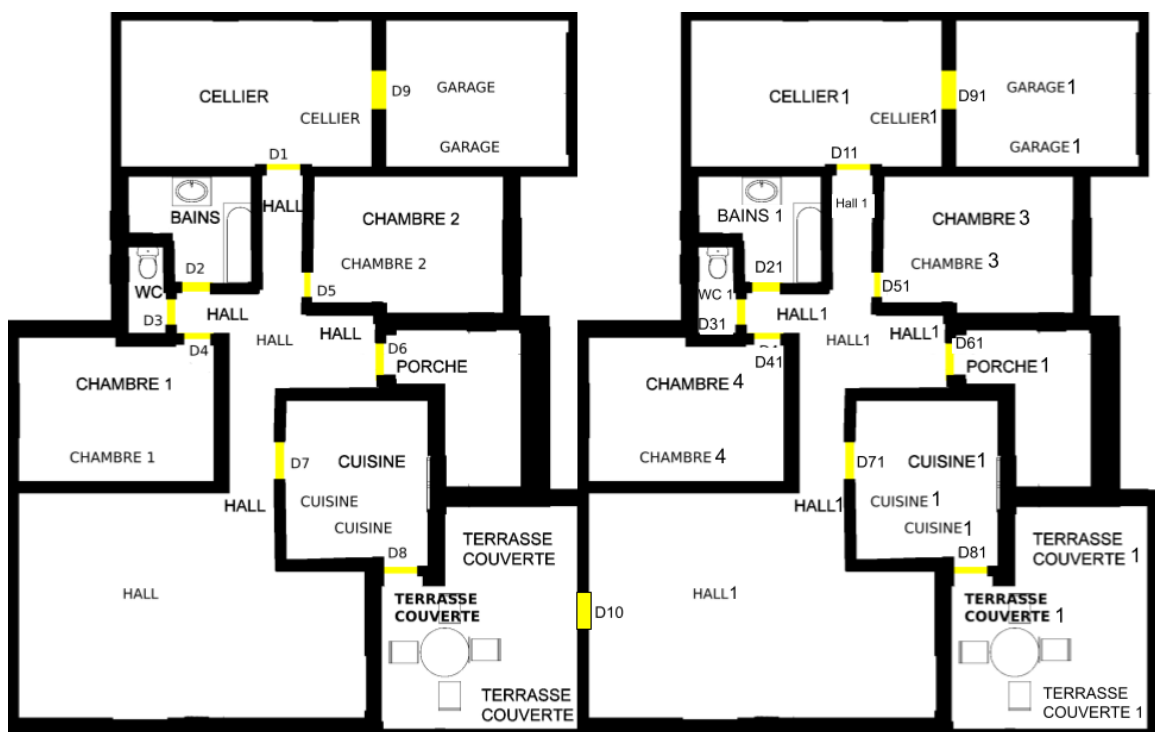


Figure 5.5: Example of a doubled map.

5.4 Experiments in VLM-Based Floor Plan Parsing

In order to evaluate whether the VLM produces accurate navigation plans, we design and run experiments over a dataset of floor plans. The experiments are designed to measure the effect of floor plan size, task difficulty, and label density on the VLM’s plan accuracy.

5.4.1 Experimental Setup

Our study uses floor plan images from a publicly available dataset CVC-FP [107], which includes detailed architectural floor plans for various building types. Three floor plans were randomly selected from those that contain 9-11 rooms and clear labels. Those maps are referred to as “original maps” and are shown in Fig. 5.4. Our experiments use two state-of-the-art VLMs: GPT-4o [108] and Claude-3.5 Sonnet [109].

For each floor plan, we generate five pairs of start and goal locations. To account for the stochastic nature of VLM responses, we run each VLM ten times per navigation task, resulting in a total of 50 navigation trials per floor plan. We evaluate performance based on the correctness of the generated plans. A plan is considered correct if it uses only those actions defined in the text prompt, includes feasible actions, and leads a sequence of transitions from the start location to the goal. Example infeasible actions include navigating to a room that is not connected to the current room and opening a door that belongs to a distant room.

Our experimental design focuses on three key dimensions:

1. **Map Size:** We hypothesize that increasing the map size will result in a decrease in VLM’s map parsing performance. This hypothesis is based on the assumption that

larger maps introduce greater complexity.

2. **Task Difficulty:** We hypothesize that when the start and end locations are far away (based on number of rooms required to traverse), the VLMs will have a low accuracy in map parsing and plan generation.
3. **Label Density:** We hypothesize that a densely labelled floor plan map will facilitate accurate navigation plan generation from VLMs.

Next, we describe our experiment setup for evaluating each of the three hypotheses.

Map Size

To examine the impact of map size on navigation performance, we developed two types of maps: **Original Maps** that are enhanced versions of the maps selected from the CVC-FP dataset, and **Doubled Maps** that were created by connecting two copies of an original map through an additional door. Fig. 5.5 presents an example of a doubled map. A door denoted as D10 is added to establish connectivity, thereby forming the final doubled map.

Task Difficulty

We design two types of navigation tasks to evaluate model performance across two levels of difficulty. **Easy Tasks** are the navigation tasks that can be completed by navigating from Room A to Room B *without* traversing any intermediate rooms. **Hard Tasks** are those that require a robot to navigate through at least two intermediate rooms before the goal can be achieved. As a result, an optimal solution of hard tasks will involve four rooms in total.

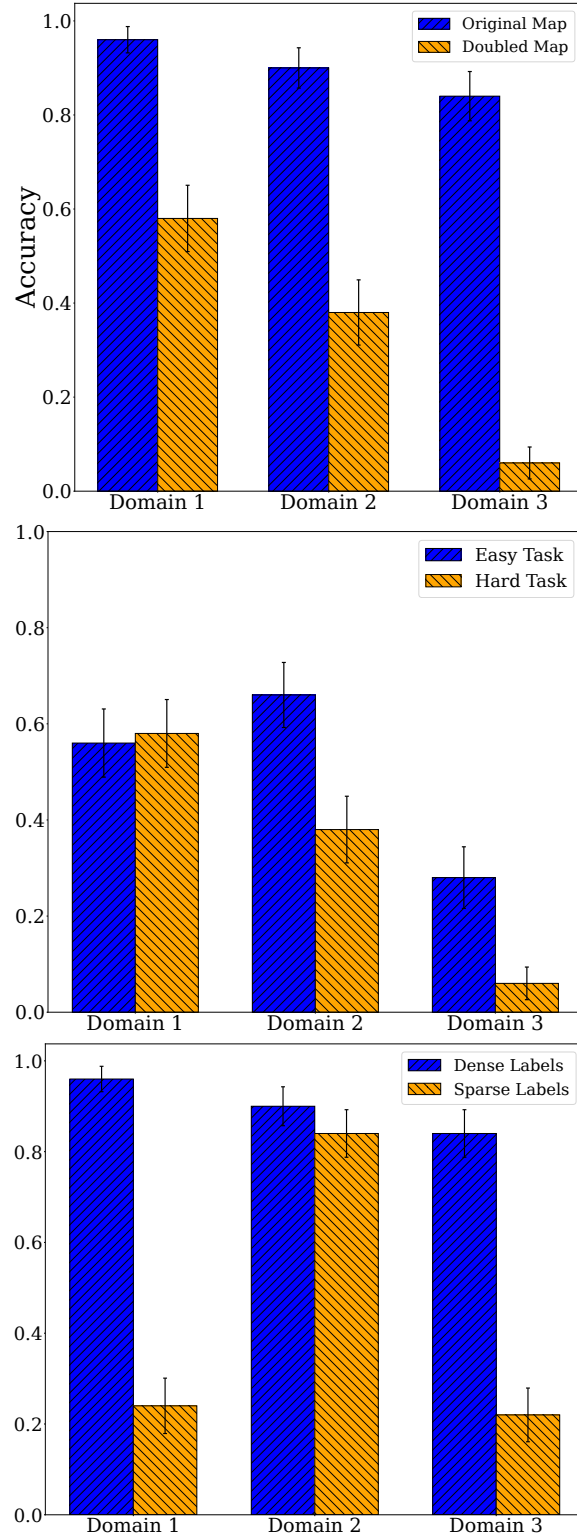


Figure 5.6: Comparison of GPT-4o results for map size, task difficulty and label density. We used original maps in the “label density” experiment. To accommodate hard tasks, we used doubled maps in the “task difficulty” experiment.

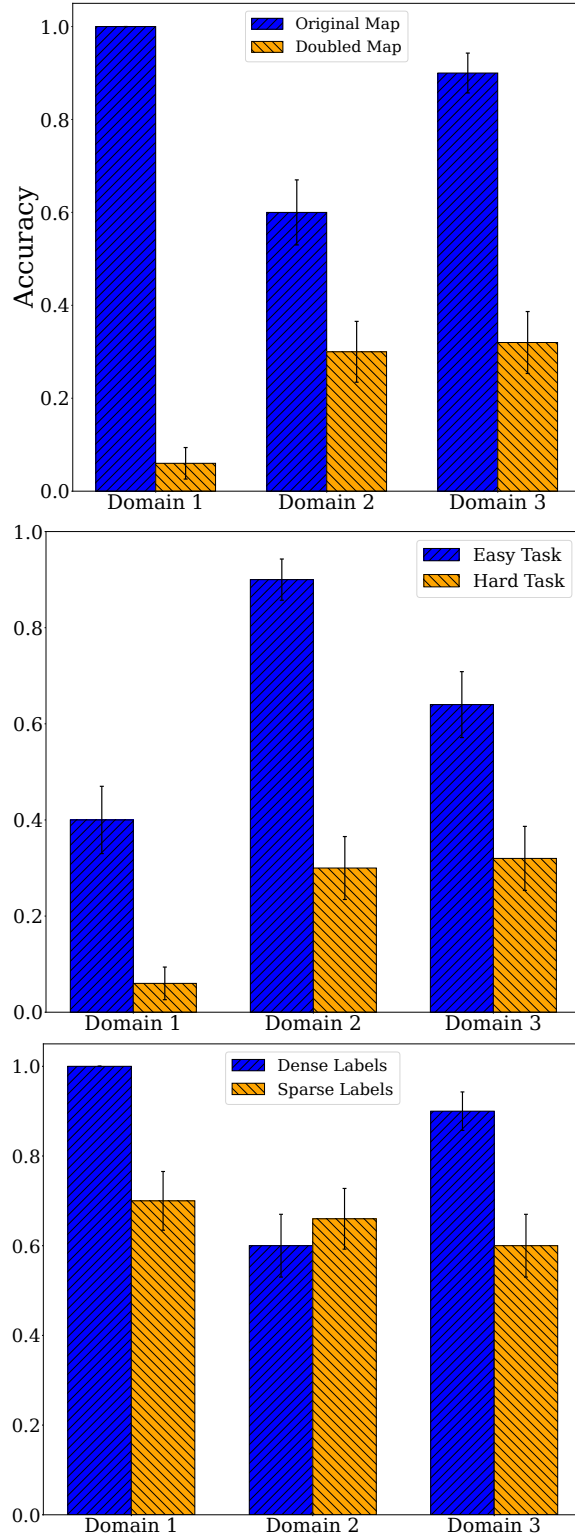


Figure 5.7: Comparison of Claude Sonnet 3.5 results for map size, task difficulty and label density. We used original maps in the “label density” experiment. To accommodate hard tasks, we used doubled maps in the “task difficulty” experiment.

The increased complexity introduces more decision points and possible paths, producing a more challenging task.

Label Density

We evaluated the impact of label density on model performance by implementing two labeling schemes. **Sparse-Labeled** maps are those where each room was labeled with a single label, usually placed at the center. This minimalistic approach offered fewer cues for the model to base its navigation decisions on. **Dense-Labeled** maps include multiple labels for each room, where the placement is described in Section 5.3.1.

5.4.2 Experimental Results

Hypothesis 1 (Map Size)

The first hypothesis explores how the size of the map influences the model’s accuracy. We compared the performance between original maps and doubled maps over hard tasks to assess the same.

The results indicate that accuracy decreases as map size increases in the overall domain analysis, with the VLMs performing better in smaller maps. The difference is significant, as a T-test on trials with GPT-4o revealed a drop in accuracy ($t = 6.13$, $p < 0.0001$). This supports our hypothesis that accuracy decreases as map size increases. The results are reported in Fig. 5.6 and 5.7. Both GPT and Claude exhibit a similar pattern of performance decline with larger maps.

Hypothesis 2 (Task Difficulty)

The second hypothesis investigates how task difficulty impacts accuracy. Easy tasks involve straightforward navigation between rooms, while hard tasks, require traversing multiple intermediate rooms, making the tasks more complex. We compared the performance between doubled maps over easy tasks and hard tasks.

The accuracy of the GPT-4o models generally decreased with increased task difficulty, as more complex tasks led to lower accuracy in two out of the three maps. For example, a T-test on `doubled_map_2` indicated a drop in accuracy with $t = 2.88$ and $p = 0.0047$. The results from both VLMs generally support our hypothesis that more complex tasks lead to lower accuracy, as navigating through multiple rooms introduces additional challenges. One example task is detailed in Table 5.1.

Table 5.1: An example navigation plan generated by GPT-4o in Map 1 shown on the left of Fig. 5.4. The initial location is “Terrasse Couverte” and the goal location is “Chambre 1”. This navigation task requires a sequence of nine actions. GPT-4o achieved 0.96 success rate in this map on similar hard navigation tasks, which demonstrates great promises for VLM-based map parsing research.

Number	Action
1	ApproachDoor(D8)
2	OpenDoor(D8)
3	GoThrough(D8)
4	ApproachDoor(D7)
5	OpenDoor(D7)
6	GoThrough(D7)
7	ApproachDoor(D4)
8	OpenDoor(D4)
9	GoThrough(D4)

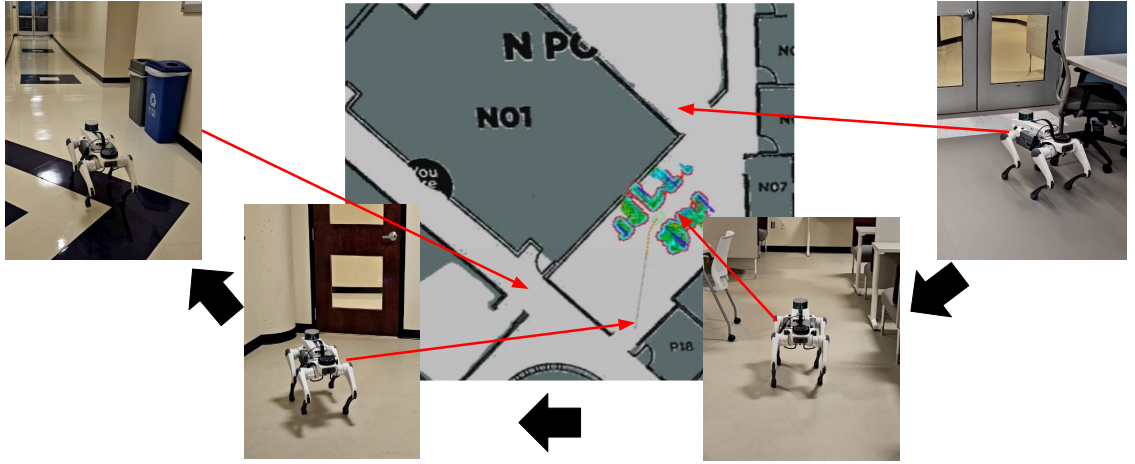


Figure 5.8: The robot localizes directly on the floor plan image and performs a navigation task. The occupancy grid map that the robot used for localization and navigation was also derived from the floor plan map captured by the robot. We manually removed the labels that would otherwise be interpreted by the robot as obstacles. From right to left, the robot was performing 1) Going through Door N09, 2) Approaching Door N00, 3) Opening Door N00, and 4) Approaching Door T01.

Hypothesis 3 (Label Density)

The third hypothesis examines the impact of label density on accuracy. This is evaluated by comparing the performance of original maps from sparse-label and dense-label datasets over hard tasks. Dense labels provide more contextual information, while sparse labels offer minimal context, making the navigation task more challenging.

The accuracy of the GPT-4o models significantly improved with dense labels compared to sparse labels across all maps. For instance, a T-test on `original_map_1` showed a significant improvement with $t = 10.72$ and $p < 0.0001$. This result indicates that dense labels substantially enhance accuracy, supporting our hypothesis that dense labels provide crucial contextual information, enabling the models to navigate more effectively. The improvement in accuracy with dense labels was consistent across all maps for GPT-4o, highlighting

the importance of label density in successful navigation.

These findings suggest that our approach can effectively handle a range of navigation challenges while demonstrating the critical importance of label density, task difficulty, and map size in determining overall performance.

5.5 Hardware Demonstration

Our VLM-based planning and navigation system is demonstrated on a DEEPRobotics Lite3 quadruped robot. An image of the floor plan of a building on a college campus is captured from the robot’s camera. This image is then edited as described in Section 5.3 to make it suitable for the VLM query. Another version of the raw image with space whited out is directly used for robot localization, as seen in Fig. 5.8. The VLM is then queried with the edited photo, and a navigation plan consisting of a sequence of navigation and door opening actions is generated. The robot executes this navigation plan, to move from the robotics lab (room N09), to a classroom (room T01). The robot localizes itself directly on a grayscale version of the floor plan, without the need for generating an accurate occupancy grid via SLAM. The robot successfully avoids the obstacles not present on the floor plan, asks for doors to be opened as needed, and achieves the desired navigation goal.

5.6 VLM-Based Floor Plan Parsing Summary and Next Steps

In this chapter, we introduce a novel task, named map parsing, that is unexplored in the VLM literature while pointing to the foundation of mobile robotics. We demonstrate remarkable performance of two VLMs on map parsing tasks, as applied to robot navigation. We develop a VLM-based planning system that generates navigation plans directly from a

floor plan image and validated our approach through experiments on a floor plan dataset and on hardware, demonstrating the feasibility of VLM-driven navigation across different environments.

While our results show that this approach is viable, several challenges remain, opening up avenues for further research. The process of modifying floor plans to optimize VLM performance is still a manual task. Future research could investigate leveraging segmentation models [110, 111] or other techniques to automate map refinement. Also, we can investigate strategies to improve the VLM’s ability to handle larger and more complex maps, e.g., outdoor environments [112]. Another direction is that VLMs (and all transformer based autoregressive models) have a host of known issues, such as hallucinations and biases [113, 114, 115, 116] that can be addressed in robot planning. We anticipate that this work will inspire further studies that expand upon the ideas on VLM-based map parsing in this chapter.

6 Conclusions and Future Work

This dissertation studies the integration of human knowledge for legged robot systems and human-robot collaboration [117, 67, 118]. Logical rules and floor plans are leveraged to learn low-level locomotion controllers, and high-level decision making in navigation, respectively. Robotic guide dogs serve as an important application domain that showcases the integrated decision-making and human-robot collaboration skills.

Additional work has focused more broadly on machine learning, including interpretability of relational models, and various applications. Explanations for the predictions of Graph Neural Networks and other relational models are obtained through learning a local approximation model and optimizing a mask over a local computation graph [119]. Energy consumption signals from homes are disaggregated into appliance energy signals through the use of deep learning models [120]. Response times for non-emergency service calls are predicted with probabilistic graphical models [121].

One way human knowledge is integrated in legged robot systems is through the incorporation of logical rules defining foot contact sequences for policy learning. The rules capture human domain knowledge about quadruped animal locomotion, which are encoded into a reward function via the Reward Machine framework. The policy then learns to satisfy the human-defined logical rules in order to learn the desired locomotion gait. Gait

frequency can be dynamically adjusted based on the gait frequency command. Following this framework, human knowledge in the form of simple logical rules is encoded into low-level quadruped locomotion policies, which leads to the learning of diverse locomotion behaviors.

We also leverage human spatial knowledge and enable human-robot collaboration through the development of a robotic guide dog system. A locomotion controller and force estimation module is concurrently trained in simulation. A tug interface which leverages the force estimation module is developed to facilitate communication through high-level directional tug commands from a visually impaired human, based on their spatial knowledge of the building, and commonsense navigational knowledge. The robot leverages its perception capabilities and path planning algorithms to help the human-robot team to avoid nearby obstacles.

Human spatial knowledge is also represented in floor plans, which are commonly available in many buildings. Such floor plans include geometric information on the structure of the rooms and halls, and also semantic labels for each room. Floor plan information is leveraged in this dissertation in order to generate plans and perform autonomous navigation tasks. We find that Vision Language Models can directly parse floor plan images, and can accurately generate navigation plans with limited image edits for visual prompting. The floor plan can be directly used for robot localization, avoiding the need to manually generate a cost map.

An important direction of future work is to leverage other sensing modalities of legged robots such as vision to further expand the scope of rule-based knowledge, beyond foot contact sequences. This can allow for new propositional symbols, such as terrain type,

which can help further encode human domain knowledge for intelligent locomotion and decision making capabilities across diverse environments. A limiting assumption in this dissertation is that propositional symbols defining logical rules are perfectly detected. Real world environments are partially observable, and we can hardly expect to always obtain correct truth values of propositional symbols - especially considering modalities beyond foot contacts. In order to relax this assumption, new learning algorithms must be developed which can deal with uncertainty in rule satisfaction. One possible formulation can be to learn over a *belief state* over reward machine states.

Another direction of future work is to better leverage robot perception and human commonsense reasoning to further develop robotic guided navigation systems. This can include the introduction of dialog capabilities, scene verbalization, intelligent disobedience, and social navigation to name a few. Additional skill learning is also required, such as guided stair climbing and door opening. Each of these improvements require an understanding of the human's intentions, along with the environment, to perform safe and accurate navigation as a human-robot team.

In this dissertation, we focus on four-legged quadruped robots. However, other legged robot forms such as humanoid robots, or wheeled-legged robots have been recently gaining popularity due to their potential for manipulation capabilities, or more efficient locomotion. In order to further develop locomotion and decision making capabilities for these new robotic platforms, it is necessary to incorporate human knowledge. For example, humans might have domain knowledge on which surfaces might be more suitable for the use of wheels, and which surfaces require lifting the legs. On the other hand, humanoids might be a suitable alternative robot form for guided navigation tasks, due to their potential to

open doors and perform other manipulation tasks during navigation. In future work, we can further develop the capabilities of different forms of legged robots.

Bibliography

- [1] 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)*. 2018.
- [2] Donghyun Kim et al. “Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control”. In: *arXiv preprint arXiv:1909.06586* (2019).
- [3] Nate Kohl and Peter Stone. “Policy gradient reinforcement learning for fast quadrupedal locomotion”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*. Vol. 3. IEEE. 2004, pp. 2619–2624.
- [4] Jie Tan et al. “Sim-to-real: Learning agile locomotion for quadruped robots”. In: *arXiv preprint arXiv:1804.10332* (2018).
- [5] Xue Bin Peng et al. “Learning agile robotic locomotion skills by imitating animals”. In: *arXiv preprint arXiv:2004.00784* (2020).
- [6] Ashish Kumar et al. “Rma: Rapid motor adaptation for legged robots”. In: *arXiv preprint arXiv:2107.04034* (2021).
- [7] 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.
- [8] Nikita Rudin et al. “Learning to walk in minutes using massively parallel deep reinforcement learning”. In: *Conference on Robot Learning*. 2022.
- [9] Takahiro Miki et al. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science Robotics* (2022).
- [10] Ziwen Zhuang et al. “Robot Parkour Learning”. In: *arXiv preprint arXiv:2309.05665* (2023).
- [11] Martin Wermelinger et al. “Navigation planning for legged robots in challenging terrain”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 1184–1189.
- [12] Jonas Frey et al. “Locomotion policy guided traversability learning using volumetric representations of complex environments”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 5722–5729.
- [13] Siqi Liu et al. “From motor control to team play in simulated humanoid football”. In: *Science Robotics* 7.69 (2022), eabo0235.

- [14] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. “Legs as manipulator: Pushing quadrupedal agility beyond locomotion”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5106–5112.
- [15] Rodrigo Toro Icarte et al. “Reward machines: Exploiting reward function structure in reinforcement learning”. In: *Journal of Artificial Intelligence Research* 73 (2022), pp. 173–208.
- [16] Robert Playter, Martin Buehler, and Marc Raibert. “BigDog”. In: *Unmanned Systems Technology VIII*. Vol. 6230. SPIE. 2006, pp. 896–901.
- [17] Boston Dynamics. *Hey Buddy, Can You Give Me a Hand?* YouTube video. Sept. 2019. URL: <https://www.youtube.com/watch?v=wlkCQXHEgjA>.
- [18] Unitree. *Unitree Robotics*. 2024. URL: <https://www.unitree.com/>.
- [19] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. In: (1989).
- [20] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).
- [21] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8 (1992), pp. 229–256.
- [22] John Schulman et al. “Trust region policy optimization”. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.
- [23] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).
- [24] Carlos E Garcia, David M Prett, and Manfred Morari. “Model predictive control: Theory and practice—A survey”. In: *Automatica* 25.3 (1989), pp. 335–348.
- [25] Marko Bjelonic et al. “Whole-body mpc and online gait sequence generation for wheeled-legged robots”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 8388–8395.
- [26] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. <http://pybullet.org>. 2016–2021.
- [27] Viktor Makoviychuk et al. “Isaac gym: High performance gpu-based physics simulation for robot learning”. In: *arXiv preprint arXiv:2108.10470* (2021).
- [28] Laura Smith et al. “Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World”. In: *arXiv preprint arXiv:2110.05457* (2021).
- [29] Atil Iscen et al. “Policies modulating trajectory generators”. In: *Conference on Robot Learning*. 2018.
- [30] Jonah Siekmann et al. “Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition”. In: *arXiv preprint arXiv:2011.01387* (2020).
- [31] Joonho Lee et al. “Learning quadrupedal locomotion over challenging terrain”. In: *Science robotics* 5.47 (2020), eabc5986.

- [32] Zipeng Fu et al. “Minimizing energy consumption leads to the emergence of gaits in legged robots”. In: *arXiv preprint arXiv:2111.01674* (2021).
- [33] Ananye Agarwal et al. “Legged locomotion in challenging terrains using egocentric vision”. In: *Conference on robot learning*. PMLR. 2023, pp. 403–415.
- [34] 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.
- [35] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.
- [36] 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.
- [37] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.
- [38] 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.
- [39] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. “Deep whole-body control: Learning a unified policy for manipulation and locomotion”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 138–149.
- [40] Gabriel B Margolis et al. “Learning to See Physical Properties with Active Sensing Motor Policies”. In: *arXiv preprint arXiv:2311.01405* (2023).
- [41] Rodrigo Toro Icarte et al. “Using reward machines for high-level task specification and decomposition in reinforcement learning”. In: *International Conference on Machine Learning*. 2018, pp. 2107–2116.
- [42] Rupert Bourne et al. “Trends in prevalence of blindness and distance and near vision impairment over 30 years: an analysis for the Global Burden of Disease Study”. In: *The Lancet global health* 9.2 (2021), e130–e143.
- [43] Shivendra Agrawal, Mary Etta West, and Bradley Hayes. “A novel perceptive robotic cane with haptic navigation for enabling vision-independent participation in the social dynamics of seat choice”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022.
- [44] Xuan Yang. *Enabling Those with Low Vision to Run Independently*. <https://blog.research.google/2021/05/project-guideline-enabling-those-with.html>.
- [45] Brian FG Katz et al. “NAVIG: Augmented reality guidance system for the visually impaired: Combining object localization, GNSS, and spatial audio”. In: *Virtual Reality* 16 (2012), pp. 253–269.

- [46] Lorraine Whitmarsh. “The benefits of guide dog ownership”. In: *Visual impairment research* 7.1 (2005), pp. 27–42.
- [47] Lisa M Tomkins, Peter C Thomson, and Paul D McGreevy. “Behavioral and physiological predictors of guide dog success”. In: *Journal of Veterinary Behavior* 6.3 (2011), pp. 178–187.
- [48] Liyang Wang, Jinxin Zhao, and Liangjun Zhang. “Navdog: robotic navigation guide dog via model predictive control and human-robot modeling”. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, pp. 815–818.
- [49] Anxing Xiao et al. “Robotic guide dog: Leading a human with leash-guided hybrid physical interaction”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021.
- [50] Yanbo Chen et al. “Quadruped guidance robot for the visually impaired: A comfort-based approach”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023.
- [51] Hochul Hwang et al. “System configuration and navigation of a guide dog robot: Toward animal guide dog-level guiding work”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023.
- [52] J Taery Kim et al. “Transforming a Quadruped into a Guide Robot for the Visually Impaired: Formalizing Wayfinding, Interaction Modeling, and Safety Mechanism”. In: *Conference on Robot Learning*. PMLR. 2023.
- [53] Rayna Hata et al. “See Spot Guide: Accessible Interfaces for an Assistive Quadruped Robot”. In: *arXiv preprint arXiv:2402.11125* (2024).
- [54] Donald F Hoyt and C Richard Taylor. “Gait and the energetics of locomotion in horses”. In: *Nature* 292.5820 (1981), pp. 239–240.
- [55] Xingye Da et al. “Learning a contact-adaptive controller for robust, efficient legged locomotion”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 883–894.
- [56] Yuxiang Yang et al. “Fast and efficient locomotion via learned gait transitions”. In: *Conference on Robot Learning*. 2022, pp. 773–783.
- [57] Zhe Xu et al. “Joint inference of reward machines and policies for reinforcement learning”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 30. 2020, pp. 590–598.
- [58] Cyrus Neary et al. “Reward machines for cooperative multi-agent reinforcement learning”. In: *arXiv preprint arXiv:2007.01962* (2020).
- [59] Alberto Camacho et al. “Reward machines for vision-based robotic manipulation”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.
- [60] Taylor Dohmen et al. “Inferring Probabilistic Reward Machines from Non-Markovian Reward Signals for Reinforcement Learning”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. 2022.

- [61] Rodrigo Toro Icarte et al. “Learning reward machines for partially observable reinforcement learning”. In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 15523–15534.
- [62] Michael Dann et al. “Multi-Agent Intention Progression with Reward Machines”. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*. 2022, pp. 215–222.
- [63] Yecheng Shao et al. “Learning free gait transition for quadruped robots via phase-guided controller”. In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 1230–1237.
- [64] Andrew Cohen, Lei Yu, and Robert Wright. “Diverse exploration for fast and safe policy improvement”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.
- [65] Wenhao Yu et al. “Language to Rewards for Robotic Skill Synthesis”. In: *arXiv preprint arXiv:2306.08647* (2023).
- [66] Yujin Tang et al. “SayTap: Language to Quadrupedal Locomotion”. In: *arXiv preprint arXiv:2306.07580* (2023).
- [67] David DeFazio, Eisuke Hirota, and Shiqi Zhang. “Seeing-Eye Quadruped Navigation with Force Responsive Locomotion Control”. In: *Conference on Robot Learning (CoRL)*. PMLR. 2023.
- [68] Colby Morita. *How Much Does A Guide Dog Cost?* <https://puppyintraining.com/how-much-does-a-guide-dog-cost/#:~:text=Initial%20cost%20for%20Guide%20Dog,for%20a%20guide%20dog%20%3D%20%2459%2C600>.
- [69] J Taery Kim et al. “How to Train Your Guide Dog: Wayfinding and Safe Navigation with Human-Robot Modeling”. In: *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. 2023, pp. 221–225.
- [70] Qihe Chen et al. “Can Quadruped Navigation Robots be Used as Guide Dogs?” In: *arXiv preprint arXiv:2210.08727* (2022).
- [71] Yan Zhang et al. ““I am the follower, also the boss”: Exploring Different Levels of Autonomy and Machine Forms of Guiding Robots for the Visually Impaired”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–22.
- [72] Parth Chonkar et al. “Look to my Lead: How Does a Leash Affect Perceptions of a Quadruped Robot?” In: ().
- [73] Luigi Campanaro et al. “Learning and Deploying Robust Locomotion Policies with Minimal Dynamics Randomization”. In: *arXiv preprint arXiv:2209.12878* (2022).
- [74] Xuxin Cheng et al. “Extreme Parkour with Legged Robots”. In: *arXiv preprint arXiv:2309.14341* (2023).
- [75] Mert Dede. *peakdetect*. <https://github.com/avhn/peakdetect>. 2022.
- [76] Brian P. Gerkey. *AMCL*. 2013. URL: <http://wiki.ros.org/amcl>.

- [77] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.
- [78] Xue Bin Peng et al. “Learning Agile Robotic Locomotion Skills by Imitating Animals”. In: *Robotics: Science and Systems*. July 2020. DOI: 10.15607/RSS.2020.XVI.064.
- [79] Reuth Mirsky and Peter Stone. “The Seeing-Eye Robot Grand Challenge: Rethinking Automated Care”. In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*. 2021.
- [80] Alberto Elfes. “Using occupancy grids for mobile robot perception and navigation”. In: *Computer* 22.6 (1989), pp. 46–57.
- [81] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: A survey from 2010 to 2016”. In: *IPSJ transactions on computer vision and applications* 9 (2017), pp. 1–11.
- [82] Stanislaw Antol et al. “Vqa: Visual question answering”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.
- [83] Yen-Chun Chen et al. “Uniter: Universal image-text representation learning”. In: *European conference on computer vision*. Springer. 2020, pp. 104–120.
- [84] Tim Brooks, Aleksander Holynski, and Alexei A Efros. “Instructpix2pix: Learning to follow image editing instructions”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 18392–18402.
- [85] Soroush Nasiriany et al. “PIVOT: Iterative Visual Prompting Elicits Actionable Knowledge for VLMs”. In: *arXiv preprint arXiv:2402.07872* (2024).
- [86] Fangchen Liu et al. “MOKA: Open-Vocabulary Robotic Manipulation through Mark-Based Visual Prompting”. In: *arXiv preprint arXiv:2403.03174* (2024).
- [87] Shiqi Zhang et al. “Mobile Robot Planning Using Action Language with an Abstraction Hierarchy”. In: *International Conference on Logic Programming and Non-monotonic Reasoning*. Springer. 2015, pp. 502–516.
- [88] Yohei Hayamizu et al. “Guiding robot exploration in reinforcement learning via automated planning”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 31. 2021, pp. 625–633.
- [89] Zipeng Fu et al. “Coupling vision and proprioception for navigation of legged robots”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17273–17283.
- [90] Chi Zhang et al. “FRL-SLAM: A fast, robust and lightweight SLAM system for quadruped robot navigation”. In: *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2021, pp. 1165–1170.
- [91] Andréa Macario Barros et al. “A comprehensive survey of visual slam algorithms”. In: *Robotics* 11.1 (2022), p. 24.

- [92] Maks Sorokin et al. “Learning to navigate sidewalks in outdoor environments”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3906–3913.
- [93] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.
- [94] Qingxiu Dong et al. “A survey on in-context learning”. In: *arXiv preprint arXiv:2301.00234* (2022).
- [95] Jianwei Yang et al. “Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v”. In: *arXiv preprint arXiv:2310.11441* (2023).
- [96] Yash Kant et al. “Housekeep: Tidying virtual households using commonsense reasoning”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 355–373.
- [97] Yan Ding et al. “Task and motion planning with large language models for object rearrangement”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 2086–2092.
- [98] Naoki Yokoyama et al. “Vlfn: Vision-language frontier maps for zero-shot semantic navigation”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 42–48.
- [99] Abhinav Rajvanshi et al. “Saynav: Grounding large language models for dynamic planning to navigation in new environments”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 34. 2024, pp. 464–474.
- [100] Daeun Song et al. “Socially aware robot navigation through scoring using vision-language models”. In: *arXiv preprint arXiv:2404.00210* (2024).
- [101] Adarsh Jagan Sathyamoorthy et al. “CoNVOI: Context-aware Navigation using Vision Language Models in Outdoor and Indoor Environments”. In: *arXiv preprint arXiv:2403.15637* (2024).
- [102] Annie S Chen et al. “Commonsense Reasoning for Legged Robot Adaptation with Vision-Language Models”. In: *arXiv preprint arXiv:2407.02666* (2024).
- [103] Michael Ahn et al. “Do as i can, not as i say: Grounding language in robotic affordances”. In: *arXiv preprint arXiv:2204.01691* (2022).
- [104] Wenlong Huang et al. “Inner monologue: Embodied reasoning through planning with language models”. In: *arXiv preprint arXiv:2207.05608* (2022).
- [105] Bo Liu et al. “Llm+ p: Empowering large language models with optimal planning proficiency”. In: *arXiv preprint arXiv:2304.11477* (2023).
- [106] Allen Z Ren et al. “Robots that ask for help: Uncertainty alignment for large language model planners”. In: *arXiv preprint arXiv:2307.01928* (2023).
- [107] Lluís-Pere de las Heras et al. “CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool”. In: *International Journal on Document Analysis and Recognition* (2015).

- [108] OpenAI. *GPT-4o*. <https://openai.com/index/hello-gpt-4o/>.
- [109] Anthropic. *Claude-3.5 sonnet*. <https://www.anthropic.com/news/claude-3-5-sonnet>.
- [110] Chaoning Zhang et al. “Faster segment anything: Towards lightweight sam for mobile applications”. In: *arXiv preprint arXiv:2306.14289* (2023).
- [111] Nikhila Ravi et al. “Sam 2: Segment anything in images and videos”. In: *arXiv preprint arXiv:2408.00714* (2024).
- [112] Dhruv Shah et al. “LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action”. In: *Proceedings of The 6th Conference on Robot Learning*. Ed. by Karen Liu, Dana Kulic, and Jeff Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, 14–18 Dec 2023, pp. 492–504. URL: <https://proceedings.mlr.press/v205/shah23b.html>.
- [113] Yue Zhang et al. *Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models*. 2023. arXiv: 2309.01219 [cs.CL]. URL: <https://arxiv.org/abs/2309.01219>.
- [114] Emily M. Bender et al. “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?” In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623. ISBN: 9781450383097. DOI: 10.1145/3442188.3445922. URL: <https://doi.org/10.1145/3442188.3445922>.
- [115] Wai Man Si et al. “Why So Toxic? Measuring and Triggering Toxic Behavior in Open-Domain Chatbots”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’22. 2022, pp. 2659–2673. ISBN: 978-1-4503-9450-5. (Visited on 04/05/2023).
- [116] Siddhant Garg and Goutham Ramakrishnan. “BAE: BERT-based Adversarial Examples for Text Classification”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 6174–6181. DOI: 10.18653/v1/2020.emnlp-main.498. URL: <https://aclanthology.org/2020.emnlp-main.498>.
- [117] David DeFazio, Yohei Hayamizu, and Shiqi Zhang. “Learning quadruped locomotion policies using logical rules”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 34. 2024, pp. 142–150.
- [118] David DeFazio et al. “Vision Language Models Can Parse Floor Plan Maps”. In: *arXiv preprint arXiv:2409.12842* (2024).
- [119] Yue Zhang, David Defazio, and Arti Ramesh. “Relex: A model-agnostic relational model explainer”. In: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 2021, pp. 1042–1049.

- [120] Gissella Bejarano, David DeFazio, and Arti Ramesh. “Deep latent generative models for energy disaggregation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 850–857.
- [121] David DeFazio, Arti Ramesh, and Anand Seetharam. “Nycer: A non-emergency response predictor for nyc using sparse gaussian conditional random fields”. In: *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 2018, pp. 187–196.