ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

Learning to Walk Over Structured Terrains by Imitating MPC

Spring Term 2021

Supervised by:
Prof. Dr. Marco Hutter
Takahiro Miki
Alexander Reske

Author:
Chao Ni

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Learning to Walk Over Structured Terrains by Imitating MPC

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Chao	Ni

Student supervisor(s)

Takahiro	Miki
Alexander	Reske

Supervising lecturer

Marco	Hutter

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zürich, 14.10.2021

Place and date

Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Intellectual Property Agreement

The student acted under the supervision of Prof. Hutter and contributed to research of his group. Research results of students outside the scope of an employment contract with ETH Zurich belong to the students themselves. The results of the student within the present thesis shall be exploited by ETH Zurich, possibly together with results of other contributors in the same field. To facilitate and to enable a common exploitation of all combined research results, the student hereby assigns his rights to the research results to ETH Zurich. In exchange, the student shall be treated like an employee of ETH Zurich with respect to any income generated due to the research results.

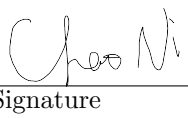
This agreement regulates the rights to the created research results.

1. Intellectual Property Rights

1. The student assigns his/her rights to the research results, including inventions and works protected by copyright, but not including his moral rights ("Urheberpersönlichkeitsrechte"), to ETH Zurich. Herewith, he cedes, in particular, all rights for commercial exploitations of research results to ETH Zurich. He is doing this voluntarily and with full awareness, in order to facilitate the commercial exploitation of the created Research Results. The student's moral rights ("Urheberpersönlichkeitsrechte") shall not be affected by this assignment.
2. In exchange, the student will be compensated by ETH Zurich in the case of income through the commercial exploitation of research results. Compensation will be made as if the student was an employee of ETH Zurich and according to the guidelines "Richtlinien für die wirtschaftliche Verwertung von Forschungsergebnissen der ETH Zürich".
3. The student agrees to keep all research results confidential. This obligation to confidentiality shall persist until he or she is informed by ETH Zurich that the intellectual property rights to the research results have been protected through patent applications or other adequate measures or that no protection is sought, but not longer than 12 months after the collaborator has signed this agreement.
4. If a patent application is filed for an invention based on the research results, the student will duly provide all necessary signatures. He/she also agrees to be available whenever his aid is necessary in the course of the patent application process, e.g. to respond to questions of patent examiners or the like.

2. Settlement of Disagreements

Should disagreements arise out between the parties, the parties will make an effort to settle them between them in good faith. In case of failure of these agreements, Swiss Law shall be applied and the Courts of Zurich shall have exclusive jurisdiction.

Zürich, 14.10.2021 

Place and date Signature

Contents

Abstract

Walking over structured terrain requires perception awareness of the environment. The robot needs to extract information from exteroceptive sensors and generate control signals by either model-based approaches, such as Model Predictive Control (MPC), or inferencing a neural network, trained with Reinforcement Learning (RL) or Imitation Learning (IL). While MPC can produce accurate solutions, it relies on an accurate perceptual sensor and is constrained by computational limits of onboard hardware. RL solves such problems by designing rewards for a successful traverse. But designing rewards can be tedious and training an RL policy often takes days. IL trains the policy by learning from the expert demonstration and requires less time if good demonstrations are available. MPC-Net is an IL approach that learns from an MPC expert. It minimizes the control Hamiltonian of the Optimal Control (OC) problem instead of imitating the observation-action mapping directly as Behavioral Cloning (BC). It explicitly encodes the constraints into the objective and achieves improved constraint satisfaction practically. In this work, we add perception to MPC-Net, using demonstrations from perceptive MPC, which can walk over structured obstacles. To train the policy, we use a teacher-student learning framework to incorporate noisy exteroceptive information. In the first stage, we train a privileged teacher with MPC-Net in simulation, which observes the ground truth exteroceptive information. In the second stage, a student policy is trained with BC to imitate and robustify the privileged teacher by experiencing more realistic sensor data.

We benchmark our MPC-Net teacher and validate our student policy on the hardware. We show that our student policy outperforms MPC expert under noisy perception inputs. We also show the necessity of using a two-stage "learning by cheating" framework. Moreover, we compare the performance of the robot when walking over a high step with Perceptive MPC-Net against blind MPC-Net policy.

Symbols

Symbols

ψ	yaw angle
x	base position x
y	base position y
z	base position z

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
ICRA	International Conference on Robotics and Automation
MEN	mixture-of-experts network
MPC	Model Predictive Control
HJB	Hamiltonian Jacobi Bellman
GRU	Gated Recurrent Network
IL	Imitation Learning
RL	Reinforcement Learning
WBC	whole-body controller
PD	proportional derivative
OC	Optiumal Control
DDP	Differential Dynamic Programming
MLP	Multilayer Perceptron
SLQ	Sequential Linear-Quadratic
BC	Behavioral Cloning
RNN	Recurrent Neural Network
LF	left front
RF	right front
MDP	markov decision process
CNN	convolutional neural network
GAIL	Generative Adversarial Imitation Learning

Chapter 1

Main Paper

This chapter presents the main paper that was considered for submission to IEEE International Conference on Robotics and Automation (ICRA) 2022 and reflects the author's work during the master thesis. The paper is supervised by Alexander Reske and Takahiro Miki, and benefits from the feedback from Jan Carius, Ruben Grandia, Farbod Farshidian and Marco Hutter. The structure of the paper is as follows. The introduction in Sec. I provides the motivation and main contribution of this work, and also introduces a subset of related work. Sec. II gives background in Model Predictive Control (MPC) and how it handles structured obstacles. It also provides the preliminary knowledge in MPC-Net, an Imitation Learning (IL) approach. In Sec. III, we elaborate the approach in this thesis, where we use an adaptive data generation method to ease domain mismatch and a learning-by-cheating training schedule to cope with noisy perception inputs. We introduce implementation details regarding ANYmal control, ANYmal perception and neural network design in Sec. IV. The main results are presented in Sec. V. In the end, we conclude the paper in Sec. VI.

We provide supplementary notes to the paper in Chapter 2 and give an outlook in Chapter 4.

Learning to Walk over Structured Terrains by Imitating MPC

Chao Ni, Alexander Reske, Takahiro Miki, Jan Carius, Ruben Grandia, Marco Hutter

***Abstract*— Walking over structured terrain requires perception awareness of the environment. The robot needs to extract information from exteroceptive sensors and generate control signals by either model-based approaches, such as Model Predictive Control (MPC), or inferencing a neural network, trained with Reinforcement Learning (RL) or Imitation Learning (IL). While MPC can produce accurate solutions, it relies on an accurate perceptual sensor and is constrained by computational costs. RL solves such problems by designing rewards for a successful traverse. But designing rewards can be tedious and training an RL often takes days. IL trains the policy by learning from the expert demonstration and requires less time if good demonstrations are available. MPC-Net is an IL approach that learns from MPC expert. It minimizes the control Hamiltonian of the Optimal Control (OC) problem instead of imitating the observation-action mapping directly as Behavioral Cloning (BC). In this work, we add perception to MPC-Net, using demonstrations from perceptive MPC, which can walk over structured obstacles. To train the policy, we use a teacher-student learning framework to incorporate noisy exteroceptive information.**

We benchmark our MPC-Net teacher and validate our student policy on the hardware. We show that our student policy outperforms MPC expert under noisy perception inputs. Moreover, we compare the performance of the robot when walking over a high step with Perceptive MPC-Net against blind MPC-Net policy.

I. Introduction

Legged robots, such as ANYmal [1], Laikago, Cassie, and MIT cheetah [2], have recently shown great ability to cope with challenging terrains, such as slopes [3], [4], stairs [5], [6], and stepping stones [7], [8]. The agility of the leg makes it suitable for conducting tasks in human-orientated environments, such as industrial inspection, environment exploration, and autonomous navigation. One common challenge of these tasks is to traverse structured obstacles, such as stairs, steps, and gaps. This requires the robot to be aware of the obstacle by onboard sensors, such as cameras or LiDAR. With visual information, the robot can traverse without compromising its speed, avoid collisions with obstacles, and achieve smooth motions. However, incorporating visual information into the robot locomotion control remains an active area of research.

Model-based methods have been used to cope with such scenarios. The control task can be formulated as an Optimal Control (OC) problem and walking over obstacles can be abstracted as selecting feasible foothold locations directly [9], [10] or encoded as constraints that the end-effectors satisfy [7]. Winkler et al. presented a framework for dynamic quadrupedal locomotion over challenging terrain and used a terrain cost map to select the foothold locations [11]. Kalakrishnan et al. proposed to decompose the control into sub-systems and used expert demonstrations to learn the footholds [12]. Instead of selecting footholds separately, Grandia et al. added the end-effector constraints into the optimization objective [7]. While Model Predictive Control (MPC) can produce accurate solutions if the obstacle can be perfectly detected and depicted, it is computationally intense, which enforces extra constraints to the perception module. Furthermore, the exteroceptive sensors are not perfect and have noises, which may be crucial to the modeling of the problem.

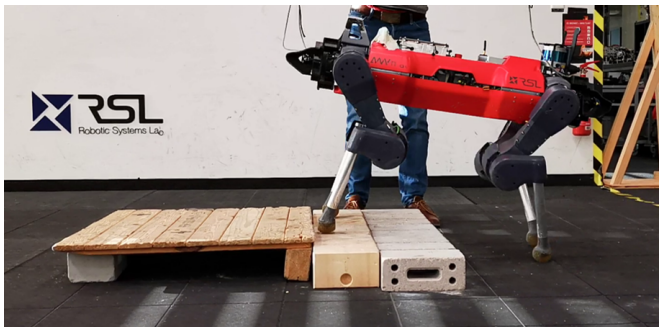


Fig. 1: ANYmal [1] walking over an obstacle.

On the other hand, learning-based approaches have seen a rapid development in the past years [13]–[19]. Among them Reinforcement Learning (RL) [20] and Imitation Learning (IL) [21] are the two main paradigms. RL encodes the problem as a markov decision process (MDP) and guides the agent with task-specific rewards. Lee et al. trained a policy that can walk on steps by gradually increasing the terrain difficulty during training [22]. Miki et al. incorporated the exteroceptive input explicitly and used a Recurrent Neural Network (RNN) encoder to combine multi-modal information to handle the noisy exteroceptive information [23]. While RL is capable of overcoming various structured terrains, it requires engineering efforts in reward design and usually takes days to train the policy. In contrast, IL learns the policy from expert demonstrations and can train the policy in a short time. It either seeks to replicate the behavior of the expert through learning the observation-action mapping directly, which is called Behavioral Cloning (BC) or learn the demonstrator's reward function for RL purpose, as known as Inverse Reinforcement Learning (IRL). When good experts are available, sample efficiency can be significantly improved compared to RL [24]. However, expert demonstrations for quadrupedal locomotion are often difficult to obtain. Peng et.al proposed to learn quadrupedal locomotion by imitating natural animals based on motion capture data [17]. Carius

All authors are with the Robotic Systems Lab, ETH Zürich, Switzerland. Email: chaoni@ethz.ch

et al. proposed to learn locomotion from the MPC expert by MPC-Net [18], which is a variant of IL approaches. Instead of imitating the action directly, it minimizes the control Hamiltonian, which also encodes the underlying constraints of the OC problem, and therefore improves constraint satisfaction practically [18]. The effectiveness of MPC-Net has also been shown in [19], where a robust multi-gait policy is learned. Nevertheless, most imitation approaches for locomotion are limited on flat terrain [17]–[19], where the terrain perception is not available and walking over structured obstacles often leads to failure.

In this work, we propose to traverse structured obstacles with IL via leveraging the knowledge from MPC. MPC can provide accurate solution given the perfect exterocpetive information in simulation, and we benefit from sample efficiency and improved constraint satisfaction by extending MPC-Net. We use a teacher-student learning framework [25] to incorporate the noisy exteroceptive information. To the best of our knowledge, this is the first approach that achieves perceptive IL for overcoming structured obstacles and transfers to the real robot. Our work is built upon the theoretical principle of a Hamiltonian loss for policy search [18] and the application in robust multi-gait locomotion [19] and contributes the following advances:

- We add exteroception to MPC-Net to traverse structured obstacles.
- We show improved convergence by introducing an adaptive data generation for IL.
- Benchmarking experiments confirm that teacher policy trained with MPC-Net leads to better performance compared to BC.
- Simulation results show that Perceptive MPC-Net achieves better performance than MPC expert under noisy environments.
- Sim-to-real transfer shows that Perceptive MPC-Net outperforms blind MPC-Net over a 10 cm step setup.

A. Related Work

This section covers a subset of model-based and learning-based approaches on robot locomotion over uneven obstacles and IL in robotics tasks.

1) Locomotion over Uneven Terrain

A large portion of model-based approaches focuses on the selection of foothold location, which is incorporated into the optimization problem. Jenelten et al. defined a grid map centered at the nominal foothold, where each cell is scored based on manually selected features [26]. Prediction of the foothold can also be done by a pretrained convolutional neural network (CNN) [9], [10]. Meduri et al. proposed to encode the foothold selection on uneven terrain as a MDP and trained a stepper via deep RL [27]. Besides predicting foothold locations, there are also works on learning to walk over uneven terrains directly through RL. An end-to-end planner and controller are proposed in [28] where the perception-aware robot learns to walk on difficult terrains through deep RL.

2) Imitation Learning

Learning from expert demonstration has also been widely used in multiple autonomous tasks. Johns proposed a coarse-to-fine process to imitate manipulation from a single demonstration [29]. Qin et al. built an IL platform for manipulation from human videos [30]. Cao et al. investigated the imperfect demonstrations and score the demonstrations by their feasibility and optimality [31]. An adaptive locomotion skill is learned from multiple motion clips via Generative Adversarial Imitation Learning (GAIL) in [32]. Despite the agility it achieves, it is still limited to the area of animation. Chen et al. proposed a "learning by cheating" schedule for IL, where a privileged agent is trained with ground truth observation in the first stage, and in the second stage, the sensorimotor agent, which has no access to ground truth, learns to imitate the privileged agent [25].

II. Preliminary

In this section, we give a brief background on MPC and the minimum-principle guided policy search approach: MPC-Net [18].

A. Model Predictive Control

We consider the following OC problem

$$\begin{aligned}
\underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \quad & \Phi(\boldsymbol{x}(t_f)) + \int_0^{t_f} l(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, \mathrm{d}t, && (1)\\
\text{subject to} \quad & \boldsymbol{x}(0) = \boldsymbol{x}_0, && (2a)\\
& \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), && (2b)\\
& \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{0}, && (2c)\\
& \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t) \geq \boldsymbol{0}, && (2d)
\end{aligned}$$

where $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ are the state and input at time t, t_f the time horizon, $\boldsymbol{x}_0$ the initial state, $\Phi(\cdot)$ the final cost and the $l(\cdot)$ the intermediate cost. The system dynamics is defined by $\boldsymbol{f}(\cdot)$ and has vectorized equality constraints $\boldsymbol{g}(\cdot)$ and inequality constraints $\boldsymbol{h}(\cdot)$.

To solve the optimization problem, we use a Sequential Linear-Quadratic (SLQ) algorithm [33], which is a variant of the Differential Dynamic Programming (DDP) algorithm. The equality constraints are incorporated via Lagrange multipliers $\boldsymbol{\nu}$ [33] and inequality constraints are handled by a barrier function $b(\cdot)$ [34]. The full Lagrangian of the OC problem (1, 2) is given by

$$\begin{aligned}
\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) = l(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\nu}(\boldsymbol{x}, t)^\top \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) \\
+ \sum_i b(h_i(\boldsymbol{x}, \boldsymbol{u}, t)). \qquad (3)
\end{aligned}$$

The solution to the OC problem (1,2) consists of a nominal state-input trajectory $\{\boldsymbol{x}_{\text{nom}(\cdot)}, \boldsymbol{u}_{\text{nom}}(\cdot)\}$ and a time-dependent feedback gain $\boldsymbol{K}(t)$, leading to the control policy

$$\boldsymbol{\pi}_{\text{mpc}}(\boldsymbol{x}, t) = \boldsymbol{u}_{\text{nom}}(t) + \boldsymbol{K}(t)(\boldsymbol{x} - \boldsymbol{x}_{\text{nom}}(t)). \qquad (4)$$

The OC problem (1, 2) has an associate optimal cost-to-go function

$$V(\boldsymbol{x}, t) = \min_{\substack{\boldsymbol{u}(\cdot) \\ \text{s.t. } (2)}} \left\{ \Phi(\boldsymbol{x}(t_f)) + \int_t^{t_f} l(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)\, \mathrm{d}\tau \right\}, \qquad (5)$$

and the control Hamiltonian is given by

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, t) = \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) + \partial_{\boldsymbol{x}} V(\boldsymbol{x}, t)^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \quad (6)$$

which for all t and $\boldsymbol{x}$ satisfies the Hamilton-Jacobi-Bellman (HJB) equation

$$0 = \min_{\boldsymbol{u}(\cdot)} \{\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, t) + \partial_t V(\boldsymbol{x}, t)\}. \quad (7)$$

B. Perceptive MPC

Perceptive MPC [7] handles obstacles by first extracting a convex segmentation from the elevation map either via LiDAR point cloud or perfect terrain in simulation, and forms state-only constraints

$$h_i^{\text{st}}(\boldsymbol{x}) = \boldsymbol{A}_i \cdot \boldsymbol{p}_{E_i}(\boldsymbol{x}) + \boldsymbol{c}_i \geq \boldsymbol{0}, \quad (8)$$

where $\boldsymbol{p}_{E_i} : \mathbb{R}^{24} \to \mathbb{R}^3$ maps the robot state to the foot position and the matrix $\boldsymbol{A}_i$ and $\boldsymbol{c}_i$ project the position of the foot i onto the target segmentation and form a set of half-space constraints. These state-only constraints are then incorporated into (2) and integrated into the objective by barrier functions [34].

C. MPC-Net

MPC-Net [18] imitates MPC and learns a policy by minimizing the Hamiltonian, which encodes the constraints of the OC problem including obstacle-related constraints (8). The policy $\boldsymbol{\pi}(\boldsymbol{x}, t; \boldsymbol{\theta})$ is parametrized by a mixture-of-experts network (MEN), each expert specializing for different modes, where each mode represents a contact configuration. Accordingly, the policy can be written as follows

$$\boldsymbol{\pi}(\boldsymbol{x}, t; \boldsymbol{\theta}) = \sum_{i=1}^{E} p_i(\boldsymbol{x}, t; \boldsymbol{\theta}) \boldsymbol{\pi}_i(\boldsymbol{x}, t; \boldsymbol{\theta}), \quad (9)$$

where $\boldsymbol{p} = (p_1, ..., p_E)$ is the weight for each expert and is computed by a gating network. E is the number of experts. To improve expert specialization, the cross-entropy is used that allows the incorporation of domain knowledge [19]:

$$CE(\tilde{p}, p) = -\sum_{i=1}^{E} \tilde{p}_i(t) \log\left(p_i(\boldsymbol{x}, t; \boldsymbol{\theta})\right), \quad (10)$$

where $\tilde{p}_i(t)$ is the probability to observe mode i at time t and is defined as $\tilde{p}_i(t) = \mathbb{1}_{\{m(t)=i\}}$, where $m(t)$ is the commanded mode schedule. This leads to the following loss function for MPC-Net

$$\text{loss} = \mathcal{H}\left(\boldsymbol{x}, \boldsymbol{\pi}, t\right) + CE(\tilde{p}, p), \quad (11)$$

where we define $\boldsymbol{\pi} := \boldsymbol{\pi}(\boldsymbol{x}, t; \boldsymbol{\theta})$.

One common problem in IL is the violation of i.i.d. assumption when the learner's actions effects its future observations and states [35]. MPC-Net uses a behavioral policy to address the mismatch:

$$\boldsymbol{\pi}_b(\boldsymbol{x}, t; \boldsymbol{\theta}) = \alpha \boldsymbol{\pi}_{\text{mpc}}(\boldsymbol{x}, t) + (1 - \alpha) \boldsymbol{\pi}(\boldsymbol{x}, t; \boldsymbol{\theta}), \quad (12)$$

where α linearly decreasing from one to zero in the training. The behavioral policy executes the rollout in the data generation and the collected data is later used for training.

Reske et al. proposed to replace the absolute state with the so-called relative state to achieve better tracking, and introduced a generalized time to guide the gait based on the commanded mode schedule [19].

III. Approach

Perceptive MPC-Net empowers the autonomous agent with visual ability to traverse structured terrain. Apart from the proprioceptive state estimation, the agent receives exteroceptive information and makes steps based on the observation.

We utilize a two-stage "learning by cheating" framework [25]. First, a privileged teacher is trained via imitating the MPC expert. Then, we train a student to imitate the behavior of the privileged teacher. The teacher is confined to simulation, while the student can be deployed on the hardware. Similar to [25], we provide the teacher with expert demonstration. We use Perceptive MPC [7] as the MPC expert. To avoid confusion of the usage of teacher in two stages in the rest of the paper, we refer to MPC as **expert**, privileged agent as **teacher** and the deployable agent as **student**.

In the following subsections, we explain each training component in detail.

A. Adaptive Data Generation

While the linear adaption for the mixture ratio has been proven practically effective in previous works [18], [19], the mismatch between the expert policy and learner's policy still exists given a more complex observed environment. To address this problem, we propose a two-step data generation approach, where in the first step, we linearly decrease the mixture ratio α and in the second step, we roll out the trajectory starting with $\alpha = 0$ and actively adapt α based on how well the learner's policy minimizes the Hamiltonian. As hinted by (7), we actively adapt the ratio at every contact configuration transition by

$$\alpha(t_i) = 1 - \exp\left(-\frac{\beta}{\Delta t} \int_{t_{i-1}}^{t_i} (\mathcal{H} + \partial_t V)\, \mathrm{d}t\right), \quad (13)$$

where β is a constant, t_i is the i-th contact configuration transition time extracted from the mode schedule. $\Delta t = t_i - t_{i-1}$, $\mathcal{H} = \mathcal{H}(\boldsymbol{x}, \boldsymbol{\pi}, t)$ and $\partial_t V = \partial_t V(\boldsymbol{x}, t)$. We update mixture ratio at each transition to avoid rapid change and have a better estimate of the quality of the learner's policy.

B. Privileged Teacher

The privileged teacher has access to the ground truth elevation map and privilege information regarding its contact status. The teacher observes a list of scan points around the foothold as its exteroceptive observation.

1) Architecture

The exteroceptive observation and privilege information are processed with a Multilayer Perceptron (MLP) and form the latent representation $\bar{\boldsymbol{z}}$. The downstream MEN is similar to [19] and we extend it such that it also receives the latent representation of the environment. The complete architecture is shown in Fig. (2).

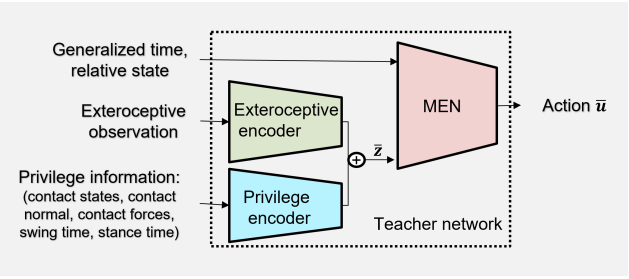


Fig. 2: Teacher network. Inputs include 1) generalized time and relative state; 2) exteroceptive observation; 3) privilege information (contact states, contact normals, contact forces, as well as swing and stance time). Privilege information are obtained from the RaiSim [36] simulator. The encoded exteroceptive observation and privilege information form the latent $\bar{z}$, which together with generalized time and relative state [19], is the input to the mixture-of-experts network (MEN) policy.

2) Objective

The teacher training is supervised in the MPC-Net fashion and the loss function is defined by (11), where the components are provided by the MPC solver.

3) Training

The data generation and policy search run asynchronously in a multi-thread scheme as in [19]. The data is generated by a behavioral policy (12), where the mixture ratio α is determined in a two-step fashion as described in Sec. III-A.

C. Student

After the teacher policy is trained, we copy the MEN and the exteroceptive encoder from the teacher's network and freeze them in the course of student training. A sequence of proprioceptive observation is received. The key assumption is that a sequence of proprioceptive observation helps to reconstruct the latent representation of the environment and the contact status of the agent.

1) Architecture

The architecture of the student network is shown in Fig. (3) leveraged from [23]. We use a Gated Recurrent Network (GRU) [37] to encode the exteroceptive observation as well as proprioceptive information. The noise of the exteroceptive observation can be eased through the averaging effect of history and GRU, and the contact status is related to the phase of locomotion. The GRU outputs a latent representation $\boldsymbol{z}$ supervised by the teacher's latent feature $\bar{\boldsymbol{z}}$. This representation $\boldsymbol{z}$ is further fed into the MEN with generalized time and relative state, and final action is generated.

2) Objective

The student is trained with BC and the loss is defined as

$$loss = ||\boldsymbol{u} - \bar{\boldsymbol{u}}||_{\boldsymbol{R}} + ||\boldsymbol{z} - \bar{\boldsymbol{z}}||_2 + \lambda\,||\boldsymbol{z} - \bar{\boldsymbol{z}}||_1, \qquad (14)$$

where $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{z}}$ are obtained by evaluating the teacher policy with privileged information given by the RaiSim [36] simulator and $\boldsymbol{R}$ is the cost matrix for balancing different input dimensions. We use a combination of 2-norm and 1-norm loss with a regularizer coefficient λ for the reconstruction of the latent representation.

3) Training

During the training, we use a similar two-step data generation approach as in the teacher training with a shortened stage one. However, in the second step, we only use student's policy and keep $\alpha = 0$. With frozen MEN and exteroceptive encoder, the student policy is able to roll out successfully within a few iterations.

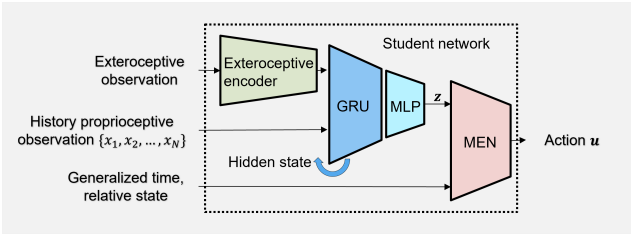


Fig. 3: Student network. The exteroceptive encoder and the multi-expert policy are copied from the teacher network and are frozen in the student training. The exterocpetive observation and a sequence of proprioceptive observation are used to reconstruct the robot's consensus about the environment with a Gated Recurrent Network (GRU) [37].

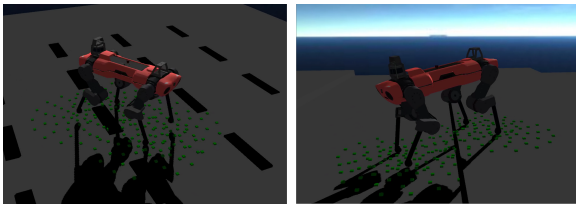


Fig. 4: Simulation terrains: gaps and steps. The width and depth of the gap as well as the shape and height of the step are randomized during the course of training.

IV. Implementation

This section introduces the setup of our quadrupedal robot and the structured terrain it traverses. We also explain some training details and how we deploy on a real robot.

A. ANYmal Control

We verified our approach on the quadrupedal robot ANYmal [1]. The kinodynamic model used by the MPC expert has a 24-dimensional state (base pose, base twist, joint angles) and 24-dimensional inputs (contact forces, joint velocities). The intermediate cost and final cost for the OC problem (1, 2) are formed as follows

$$\Phi(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_d(t_f))^\top \boldsymbol{Q}_f\,(\boldsymbol{x} - \boldsymbol{x}_d(t_f)), \qquad (15)$$

$$l(\boldsymbol{x}, \boldsymbol{u}, t) = (\boldsymbol{x} - \boldsymbol{x}_d(t))^\top \boldsymbol{Q}\,(\boldsymbol{x} - \boldsymbol{x}_d(t)) + \boldsymbol{u}^\top \boldsymbol{R}\,\boldsymbol{u}, \qquad (16)$$

where $\boldsymbol{x}_d(\cdot)$ is the desired state given by a user-defined reference trajectory. $\boldsymbol{Q}_F$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are cost matrices.

We use trotting gait in this work. The robot is controlled by torques, which are generated by inverse dynamics and PD control:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{id} + \boldsymbol{K}_d \cdot (\dot{\boldsymbol{q}}_{j,d} - \dot{\boldsymbol{q}}_{j,m}), \qquad (17)$$

where $\boldsymbol{\tau}_{id}$ is the inverse dynamics torque and $\dot{\boldsymbol{q}}_{j,d}$, $\dot{\boldsymbol{q}}_{j,m}$ are desired and measured joint velocities, respectively.

B. ANYmal Perception

We evaluate our approach mainly on two different terrains: gaps and steps, as is shown in Fig. (4). A curriculum factor is set for each terrain and represents the level of traversability. During training, we randomize the parameter of the terrains and the initial position of the robot to increase the variability of exteroceptive observations.

The robot is equipped with two RS-Bpearl LiDARs and an elevation map is constructed from the point clouds. A list

TABLE I: Scan configuration for each leg

Radius [m]	0.08	0.16	0.26	0.36	0.48
Number of Points	6	8	10	12	16

TABLE II: Hyperparameters

time step Δt	0.0025 s	rollout length T	8 s
number of threads n_t	5	adaption value β	0.1
number of jobs n_j	10	batch size B_t	256
horizon n_s	10	batch size B_s	256
data reuse n_b	2	learning rate η	1e-3
data decimation d_s	4	step one i_1	25k
maximum capacity N	5000	step two i_2	15k
number of experts E	3	regularizer λ	0.2

of scan points around the foothold is observed and the scan points are circled uniformly around the foothold projection onto the terrain in different radii. For each point, we take its vertical distance to the end-effector as scan input. The scan configuration is shown in Table. I [23].

C. Training Details

In this subsection, we give a detailed training schedule and summarize the parameters we use for training.

1) Hyper Parameters

Each trajectory lasts for a duration T and is forwarded every time step Δt. Note that we use a twist command to roll out the trajectory, instead of setting a random desired state as [18], [19]. This allows us a longer horizon and a complete cross over the obstacle. A rollout of trajectory is considered as failure, if the pitch or roll angle exceeds 30°, or if the height deviates more than 20 cm from the default value [18], [19]. When evaluating the policy, survival time defined as the earliest time of failure is used.

In the teacher training, data is generated on CPU by n_t threads that work on n_j jobs per run. For each job, a trajectory is generated with a random initialization at x_0 and twist command. Each trajectory is down-sampled by a factor d_s before being pushed into a circular buffer. We use batch size b_t for the teacher training. Moreover, the buffer has a maximum capacity N in the number of trajectories. As mentioned in Sec. III-B.3, we train the step one for i_1 gradient steps, and step two for i_2 steps.

In the student training, each data generation run gives a batch of B_s trajectories. To train the GRU, we forward the trajectory for n_s steps and sum up the loss, followed by one gradient step. Each batch is discarded after being forwarded n_b times.

We use Adam optimizer [38] for both teacher and student with a learning rate η. The mentioned hyperparameters are listed in Table II.

2) Dimensions

The exteroceptive encoder and the privilege encoder are both a MLP with two hidden layers, with each layer having 64 neurons. Each expert of MEN, as well as the gating network of MEN, is a two-layer MLP with 128 neurons. A softmax [39] function is applied to the top of the gating network. All activation functions as set to LeakyReLU [40].

D. Deployment

In the deployment, user controls the robot with a twist command. The robot receives the scan points from its LiDAR sensor and proprioceptive observation from the state estimator. The input $\boldsymbol{u}$ is inferred at 400 Hz from the policy, which is evaluated via ONNX Runtime [41]. In contrast to previous works [18], [19], where a whole-body controller (WBC) is used to track the input and computes the torques for the robot, we turn to inverse dynamics and PD control as used in the simulation. We find that such a change facilitates the sim-to-real transfer.

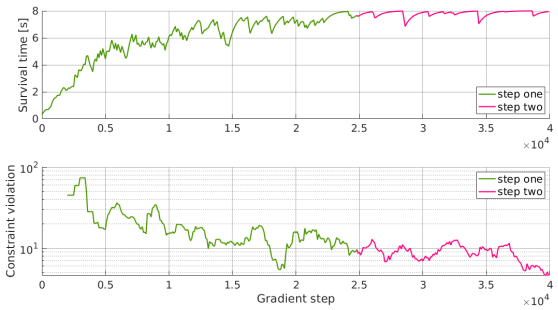


Fig. 5: The graph illustrates the improvement of survival time and constraint satisfaction with an adaptive data generation. The first 25k is trained with linearly decreasing mixture ratio, and the rest 15k is trained with adaptive ratio as defined in (13).

TABLE III: Comparison between our two step adaptive data generation and previous work [19]. We report the average survival time and the standard deviation by running each cases for 50 times. We evaluate this experiment on a 8 cm wide gap and 14 cm high step with a forward twist command 0.2 m/s.

	Our Approach	Linear Adaption
Gap	$\mathbf{25.59 \pm 6.59}$	21.18 ± 6.31
Step	$\mathbf{26.65 \pm 6.99}$	23.90 ± 7.59

V. Results

We evaluate our approach with thorough comparison results in simulation and hardware tests. For simulation results reported, our runs are executed on a single thread of the machine: Intel® Xeon(R) CPU E3-1280 v5 @ 3.70GHz x8.

A. Adaptive Data Generation

Using an adaptive data generation improves the convergence of the training and leads to a higher survival time and better constraint satisfaction as Fig. (5) shows. We also compared against policy that is trained with only step one [18], [19] with same network structure and gradient steps, and our policy survives longer with a similar standard deviation as shown in Table III.

B. Teacher Benchmarking

We benchmark our teacher policy with BC and MPC expert. In the case of BC, we replace the loss in (11) with

$$\text{loss} = ||\boldsymbol{\pi}_{\text{mpc}} - \boldsymbol{\pi}(t, \boldsymbol{x}; \boldsymbol{\theta})||_{\boldsymbol{R}} + CE(\tilde{p}, p), \qquad (18)$$

where we use the cost matrix $\boldsymbol{R}$ to normalize different dimensions. Perceptive MPC-Net achieves better performance than BC policy in both obstacles. The detailed results are shown in Table IV.

C. Comparison with MPC expert

We compared the performance of our student policy with MPC expert under noisy environments. We setup the step experiments on Gazebo and created a noisy elevation map for both MPC experts and our student policy. The performance is

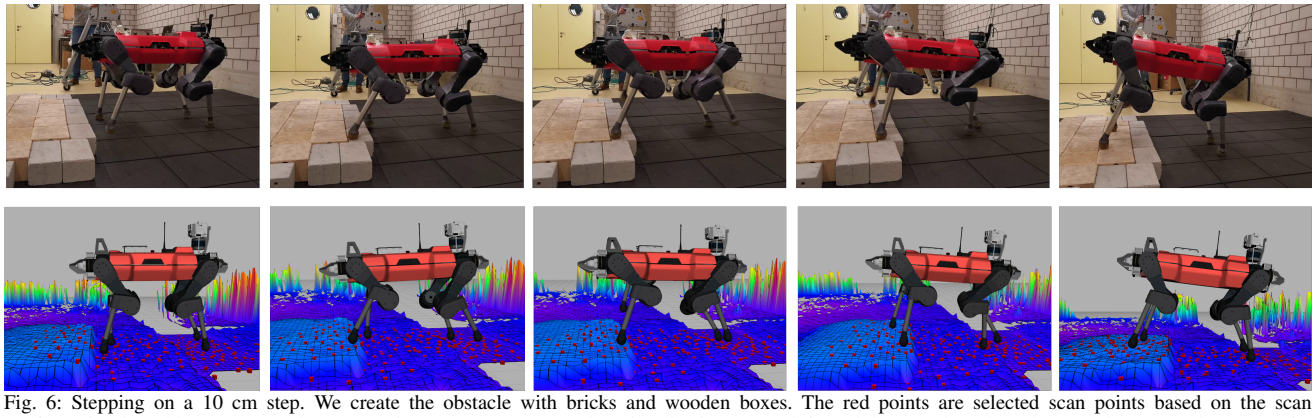


Fig. 6: Stepping on a 10 cm step. We create the obstacle with bricks and wooden boxes. The red points are selected scan points based on the scan configuration in Table I. Top: Hardware experiments, Bottom: A replay visualization. From left to right: 1). The robot observes the step and prepares for the lift of its left front (LF) leg; 2). Finishes LF step; 3). Prepares right front (RF) leg; 4) Finishes RF leg; 5). Stabilizes and prepares to walk forward. The full transverse of the obstacles is in the supplementary video.

TABLE IV: Survival time comparison against MPC expert and teacher policy trained with BC. The evaluation is on a 8 cm wide gap and $\{10, 12, 14\}$ cm high step terrain respectively. We collected 50 episodes, each with maximum 30 seconds.

		MPC	Perceptive MPC-Net	BC
Step	10 cm	**30.00±0.00**	**30.00±0.00**	29.06 ± 4.61
	12 cm	**30.00±0.00**	29.10 ± 4.39	25.05 ± 9.61
	14 cm	**30.00±0.00**	26.65 ± 6.99	23.25 ± 9.83
Gap	8 cm	**27.41±4.56**	26.21 ± 5.98	24.09 ± 6.67

TABLE V: Success rate comparison against MPC expert under noisy environment. For each case, we forward the robot with same twist command and tried 50 attempts. Note that our policy is only trained with a maximum height 14 cm.

	10 cm		12 cm		14 cm		16 cm	
σ	MPC	Ours	MPC	Ours	MPC	Ours	MPC	Ours
0.000	**1.00**	**1.00**	**1.00**	0.98	**0.98**	0.90	**1.00**	0.08
0.030	**1.00**	**1.00**	**1.00**	0.96	0.30	**0.86**	**0.32**	0.06
0.035	0.48	**1.00**	0.32	**0.96**	0.16	**0.82**	0.04	0.04
0.040	0.14	**1.00**	0.18	**0.90**	0.08	**0.80**	0.00	**0.04**
0.100	0.00	**1.00**	0.00	**0.80**	0.00	**0.76**	0.00	0.00

evaluated by the success rate, which is defined as the number of successful traversal over the step divided by total attempts. The noise level of the elevation map is controlled by the standard deviation σ, and the noisy elevation map is defined as:

$$\tilde{p}_{x,y} = p_{x,y} + \mathcal{N}(0, \sigma), \tag{19}$$

where $p_{x,y}$ is the true elevation value and $\tilde{p}_{x,y}$ is the noisy value at the coordinate (x, y) and $\mathcal{N}(0, \sigma)$ is a zero-mean normal distribution. When the noise level or step height increases, MPC fails to find the solution because of incorrect segmentation. On the other hand, our student policy outperforms under the noisy elevation map. The detailed results are in Table. V.

D. Sim-to-real Transfer

In this subsection, we validate the student policy on the quadruped ANYmal and show the benefit of using two-stage learning with practical evidence. Since gap terrain is not available in the lab environment and may cause danger to the robot, we limit the hardware experiment to the step obstacles. As shown in Fig. (6), the robot detects the obstacle through scan points and lifts the legs high enough to clear the obstacle. It is able to stabilize after stepping on the obstacle and sometimes even compensate for the missed step.

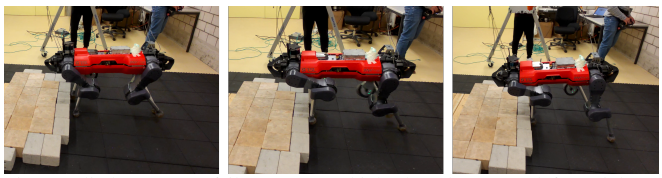


Fig. 7: Blind MPC-Net fails to step on high obstacles. From left to right, the robot is not aware of the existance of the obstacle, and therefore keeps the same trotting height. It hit the steps and fails to move forward.

1) Comparison with Blind MPC-Net

To show the visual ability of our policy, we compared to the previous work [19], where no perception input is given. We refer to it as blind MPC-Net. As shown in Fig. (7), since the robot has no perception inputs, it was unable to walk over high steps and failed the task.

2) Necessity of the Privileged Teacher

In the end, we investigated the necessity of using a two-stage learning approach. We trained a teacher without using privilege information and only used MEN to generate the input. However, it was unable to step over obstacles and much sensitive to noises.

VI. Conclusion

In this work, we added perception to MPC-Net to walk over structured obstacles by learning from MPC experts. A teacher-student framework was used to handle the noisy exteroceptive information and performed better than a single-stage method. We compared the performance under noisy environment against MPC expert and showed our policy is more robust to the noisy elevation map. Moreover, we proposed an adaptive data generation for MPC-Net and it leads to better domain transfer. The simulation results showed the benefit of using MPC-Net comparing against BC. Finally, we validated the viability of the approach on hardware by demonstrating a successful traverse over the structured obstacle.

This work proposed to learn to walk over obstacles by leveraging knowledge from MPC and opens the door to a wider range of traversable terrains. Future work includes training a single policy for different types of obstacles and in an uncertain dynamical environment.

REFERENCES

[1] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[2] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[3] J. Ding, Y. Wang, M. Yang, and X. Xiao, "Walking stabilization control for humanoid robots on unknown slope based on walking sequences adjustment," *Journal of Intelligent & Robotic Systems*, vol. 90, no. 3, pp. 323–338, 2018.

[4] M. Bando, M. Murooka, S. Nozawa, K. Okada, and M. Inaba, "Walking on a steep slope using a rope by a life-size humanoid robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 705–712.

[5] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *arXiv preprint arXiv:2105.08328*, 2021.

[6] E. Moore and M. Buehler, "Stable stair climbing in a simple hexapod robot," MCGILL RESEARCH CENTRE FOR INTELLIGENT MACHINES MONTREAL (QUEBEC), Tech. Rep., 2001.

[7] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *International Conference on Robotics and Automation (ICRA 2021)*, 2021, p. 3969.

[8] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 827–834.

[9] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2436–2442.

[10] O. A. V. Magana, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and continuous foothold adaptation for dynamic locomotion through cnns," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, 2019.

[11] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5148–5154.

[12] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2011.

[13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[14] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots." in *Robotics: Science and Systems*, 2018.

[15] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," in *Conference on Robot Learning*, 2020.

[16] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning." in *Robotics: Science and Systems*, 2019.

[17] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.

[18] J. Carius, F. Farshidian, and M. Hutter, "Mpc-net: A first principles guided policy search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, 2020.

[19] A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter, "Imitation learning from mpc for quadrupedal multi-gait control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[21] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[22] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[23] T. Miki, J. Lee, J. Hwanbo, L. Wellhausen, V. Koltun, and M. Hutter, "Wild anymal: Robust zero-shot perceptive locomotion," *Under review*, 2021.

[24] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply aggrevated: Differentiable imitation learning for sequential prediction," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3309–3318.

[25] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.

[26] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, 2020.

[27] A. Meduri, M. Khadiv, and L. Righetti, "Deepq stepper: A framework for reactive dynamic walking on uneven terrain," *arXiv preprint arXiv:2010.14834*, 2020.

[28] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[29] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," *arXiv preprint arXiv:2105.06411*, 2021.

[30] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang, "Dexmv: Imitation learning for dexterous manipulation from human videos," *arXiv preprint arXiv:2108.05877*, 2021.

[31] Z. Cao and D. Sadigh, "Learning from imperfect demonstrations from agents with varying dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5231–5238, 2021.

[32] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *arXiv preprint arXiv:2104.02180*, 2021.

[33] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 93–100.

[34] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[35] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15. PMLR, 11–13 Apr 2011, pp. 627–635.

[36] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com

[37] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[38] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[40] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Citeseer, 2013, p. 3.

[41] "ONNX Runtime: Cross-platform, High Performance ML Inferencing and Training Accelerator." [Online]. Available: https://github.com/microsoft/onnxruntime

Chapter 2

Supplementary Notes

This chapter presents supplementary notes on topics that are not covered in the paper. Sec. 2.1 provides a brief overview of the used software. Sec. 2.2 gives comments on the usage of the cost desired trajectory and proposes potential improvement for future development. Sec. 2.3 introduces the noise configuration used in this thesis. Sec. 2.4 presents thoughts on the two-step adaptive data generation. Sec. 2.5 covers the detail and the design choice of the student network. Sec. 2.6 comments on the training time for both teacher and student networks. Sec. 2.7 explains in brief the choice of tracking controller. Finally, Sec. 2.8 complements the analysis for some results of the main paper.

2.1 Software

The main training code developed in this thesis lie in the repository *ocs2_dev*, *ocs2_anymal* and *mpc_policy_learning* under the branch of *feature/perceptive_mpcnet*, *feature/perceptive_mpcnet* and *feature/perception*, respectively. There is also the repository named *anymal_control_ocs2* with the branch *feature/perceptive_policy* for the deployment of the perceptive policy. To deploy on the real robot, the branch of the repository *ocs2_dev* needs also to be switched to *feature/perceptive_mpcnet_deploy* correspondingly. In addition, to make use of the RaiSim generated terrain for the MPC expert, we also developed a separate branch under *convex_terrain_representation*. The detailed instructions on how to use the code for training and deployment, as well as evaluation of the policy, are written in the README of the *feature/perception* branch under the repository *mpc_policy_learning*.

In this thesis, we also developed the visualization package of the ANYmal simulation in RaiSim with the MPC expert or the learned policy. To visualize in RaiSim, one would need to install raisimOgre for extra tools.

Due to the fact that the perceptive policy requires more data, memory becomes an issue. We leverage Euler Cluster for large memory and the singularity software to run the ROS environment in the cluster. The detailed usage of cluster, as well as the Dockerfile used in this thesis can also be found at the *feature/perception* branch of *mpc_policy_learning* package.

2.2 Desired Trajectory

In contrast to previous works [1, 2], where a desired target state is tracked, this thesis tracks the desired trajectory and using twist command achieves a longer duration while training. This increases the potential variety of the observed scan points and covers the entire trajectory of the robot walking over the obstacle. The cost desired trajectory is a sequence of desired states $\{\boldsymbol{x}_{d,0}, \boldsymbol{x}_{d,1}, ..., \boldsymbol{x}_{d,N}\}$, where N is the trajectory horizon. In this work, we use a horizon of 8 seconds with 100Hz frequency, and this leads to a desired trajectory of length 800. Each state in the trajectory consists of orientation, position, angular velocity, translation velocity, and default joint positions. While setting the default joint position is trivial and doesn't depend on the exact robot state, base variables are dependent on the robot state, the twist command, and the terrain.

Fig. (2.1) illustrates the desired base trajectory in the deployment. The desired base trajectory assumes the knowledge of the obstacle and is planned accordingly. In the deployment, the angular velocity and translation velocity are filtered by a first-order filter to smooth the command. The orientation and position are integrated based on the filtered twist and the measurement of the current state. In the training, however, we have to set the cost desired trajectory before the start of the trajectory rollout, which means the obstacles are partly encoded to the problem before action is computed as we use so-called relative state instead of the absolute state as the input:

$$\boldsymbol{x}_r(t) = \boldsymbol{T}(\boldsymbol{\theta}_B)(\boldsymbol{x} - \boldsymbol{x}_d(t)), \tag{2.1}$$

where $\boldsymbol{\theta}_B$ is the current base orientation in world frame, $\boldsymbol{x}_d(t)$ is the desired state at time t, and the matrix $\boldsymbol{T}(\boldsymbol{\theta}_B)$ transforms the pose error from the world into the base frame.

To be specific, in order to set the cost desired trajectory, we need to estimate the so-called heading frame as used by OCS2. Since we don't have access to the actual contact configuration while determining the cost desired trajectory, we estimate the heading frame with a plane interpolated by four nominal footholds on the obstacle. In that sense, we use the information about the obstacle in the training, which may cause overfitting during training, resulting in degraded performance in the deployment. When we evaluate the policy in simulation, we also observe that the policy is very robust to the scan noise, but more studies need to be done to investigate how much the policy is simply tracking the desired trajectory and how much it is actually robust to the noise.

We also test the case where we remove the base position and reduce the state space for the policy. While we think having the angular and translation velocity would be enough to guide the robot in the correct direction, eliminating the position leads to a failure in the twist control. The robot's pose control fails, where the robot doesn't generate enough torques, which shows that the pose control relies heavily on the z variable of the robot state. The trotting control is better, but it is unable to control the lateral motion smoothly. Based on this observation, we add the robot base position back, but this is certainly a potential improvement direction that is worth exploring in the future.

One can try to address this issue by using the absolute value instead of the relative value for certain variables. Actually, in a stable version of the policy used in this work, we replace the relative joint position with the absolute joint position, and that works throughout time. However, there are other issues if one uses absolute state: the accumulating error due to using the absolute position may be too large and causes catastrophic effects; and the policy may ignore or overfit to the absolute

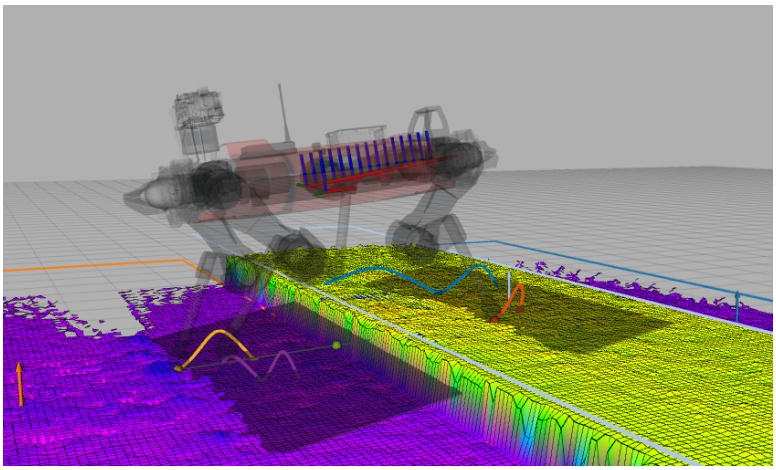


Figure 2.1: Illustration of desired base trajectory as part of the cost desired trajectory during training. The cost desired trajectory assumes the knowledge of the step height and plans accordingly.

state. One can also try to only use relative values for x, y and ψ, and use absolute values for other variables. This should solve the issue with pose control, while also avoiding the accumulating error.

2.3 Noise Configuration

In this section, we describe the noise configuration used in this thesis. The noise is applied to the robot state, robot input, and scan input with different scales.

2.3.1 State and Input Noise

One contribution in [1] is that it samples from the nominal trajectory generated by the behavioral controller and achieves better performance since a potentially larger state space is explored. In our work, due to the fact that we need a sequence of history proprioceptive information, we need to keep in order the states saved into the buffer. Therefore, we remove the samples around the nominal trajectory. Instead, we add a random offset to the robot state after the rollout of the behavioral policy:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_1) \tag{2.2}$$

where $\boldsymbol{\Sigma}_1$ is a diagonal matrix with each diagonal element representing the standard deviation of the normal distribution.

Such deviation achieves similar performance as sampling from nominal trajectories and speeds up the training. The detailed values of the diagonal matrix are shown in Table 2.1. In practice, the deviation value of the base state should be small and the rejection sampling is used to avoid abnormal values due to the random sampling.

On the other hand, we add noise to various kinds of objects, such as the state that policy receives and the output of the policy, both for the teacher network and student network. The policy is then a normal distribution with the neural network outputting its mean value: $\bar{\boldsymbol{u}} = \boldsymbol{\pi}(\bar{\boldsymbol{x}}, t, \bar{\boldsymbol{l}}) + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_3)$, where $\bar{\boldsymbol{x}} = \boldsymbol{x} + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_2)$.

Table 2.1: Elements of the diagonal matrix $\boldsymbol{\Sigma}_1$: deviation added to the rolled out state defined as: $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_1)$, where $\tilde{\boldsymbol{x}}$ is the deviated state and $\boldsymbol{x}$ is the nominal state.

index i	description	value
1	orientation x	0.01
2	orientation y	0.01
3	orientation z	0.01
4	position x	0.01
5	position y	0.01
6	position z	0.01
7	angular velocity x	0.01
8	angular velocity y	0.01
9	angular velocity z	0.01
10	translation velocity x	0.01
11	translation velocity y	0.01
12	translation velocity z	0.01
13	joint position LF HAA	0.05
14	joint position LF HFE	0.05
15	joint position LF KFE	0.05
16	joint position RF HAA	0.05
17	joint position RF HFE	0.05
18	joint position RF KFE	0.05
19	joint position LH HAA	0.05
20	joint position LH HFE	0.05
21	joint position LH KFE	0.05
22	joint position RH HAA	0.05
23	joint position RH HFE	0.05
24	joint position RH KFE	0.05

The summary of the diagonal elements of the matrix $\boldsymbol{\Sigma}_2$ and $\boldsymbol{\Sigma}_3$ is shown in Table 2.2 and Table 2.3. Based on the experiments, the final performance of the policy relies heavily on a clever choice of the noise level. When the noise level is low, the sim-to-real transfer is hard and when the noise level is too high, the robot finds it difficult to follow the twist command smoothly. In order that the policy works, we have to increase especially the velocity noise, where there exists a major gap between simulation and the real world. Notice that in the previous work, the velocity noise is incorporated in a way of setting an initial velocity offset. In the duration of tracking to the target state, the robot experiences enough states with a large offset to the default state. In this work, we recover this feature by adding noise to the state.

2.3.2 Perceptive Noise

To incorporate perceptive information is the key challenge in this thesis due to the complexity it adds to the entire network and its nature of being noisy in the real word environment. There are eight parameters that we use to set the scan points configuration. The parameters are summarized in Table 2.4. Notice that although we have options for all different noises, in this thesis, we only set the first two parameters. This can be potentially improved in future work. In the student training, the noise level is not fixed for each trajectory. We especially change the *heightnoiseFactor* and the noise level also follows a normal distribution

Table 2.2: Elements of the diagonal matrix $\boldsymbol{\Sigma}_2$: the noise added to the state before feeding into the policy.

index i	description	value
1	orientation x	0.05
2	orientation y	0.05
3	orientation z	0.05
4	position x	0.05
5	position y	0.05
6	position z	0.05
7	angular velocity x	0.05
8	angular velocity y	0.05
9	angular velocity z	0.05
10	translation velocity x	0.07
11	translation velocity y	0.07
12	translation velocity z	0.07
13	joint position LF HAA	0.05
14	joint position LF HFE	0.05
15	joint position LF KFE	0.05
16	joint position RF HAA	0.05
17	joint position RF HFE	0.05
18	joint position RF KFE	0.05
19	joint position LH HAA	0.05
20	joint position LH HFE	0.05
21	joint position LH KFE	0.05
22	joint position RH HAA	0.05
23	joint position RH HFE	0.05
24	joint position RH KFE	0.05

with $\mathcal{N}(0.03, 0.02)$ at the beginning of each trajectory. Besides, in the second half of each trajectory, we change the noise level by drawing a new value from the distribution of $\mathcal{N}(0.06, 0.02)$.

2.4 Adaptive Data Generation

This section covers more details about our proposed two-step adaptive data generation approach. We give an example of how the mixture ratio alpha evolves within a trajectory and how the mixture ratio is affected by the parameter β.

2.4.1 Evolvement of the mixture ratio

Unlike in the first step, where the mixture ratio α is fixed for a certain trajectory, the adaptive approach changes the mixture ratio at every contact configuration transition. Fig. (2.2) illustrates an example of how the mixture ratio evolves within one trajectory. The trajectory has a length of 8 seconds, and we update the ratio at each contact transition, that is every 0.35 seconds. We get overall 21 transitions starting from $\alpha = 0$. We choose $\beta = 0.1$ in this example. The maximum mixture ratio we get in this example is 0.157, while the mean ratio among 21 values is 0.063. The overall relatively low value indicates that the policy used here minimizes the control Hamiltonian well, and the actual performance gain is evaluated by compar-

Table 2.3: Elements of the diagonal matrix $\boldsymbol{\Sigma}_3$: the noise added to the output of the policy before feeding into the inverse dynamics.

index i	description	value
1	contact force LF x	1.5
2	contact force LF y	1.5
3	contact force LF z	15
4	contact force RF x	1.5
5	contact force RF y	1.5
6	contact force RF z	15
7	contact force LH x	1.5
8	contact force LH y	1.5
9	contact force LH z	15
10	contact force RH x	1.5
11	contact force RH y	1.5
12	contact force RH z	15
13	joint velocity LF HAA	0.5
14	joint velocity LF HFE	0.5
15	joint velocity LF KFE	0.5
16	joint velocity RF HAA	0.5
17	joint velocity RF HFE	0.5
18	joint velocity RF KFE	0.5
19	joint velocity LH HAA	0.5
20	joint velocity LH HFE	0.5
21	joint velocity LH KFE	0.5
22	joint velocity RH HAA	0.5
23	joint velocity RH HFE	0.5
24	joint velocity RH KFE	0.5

ing the survival time with the previous one-step approach as is done in the Sec. V of the main paper.

2.4.2 Impact of β

As explained in the main paper, we update the mixture ratio by:

$$\alpha(t_i) = 1 - \exp\left(-\frac{\beta}{\Delta t}\int_{t_{i-1}}^{t_i}(\mathcal{H} + \partial_t V)\mathrm{d}t\right), \tag{2.3}$$

where β is a constant, t_i is the i-th contact configuration transition time extracted from the mode schedule. $\Delta t = t_i - t_{i-1}$, $\mathcal{H} = \mathcal{H}(\boldsymbol{x}, \boldsymbol{\pi}, t)$ and $\partial_t V = \partial_t V(\boldsymbol{x}, t)$. The parameter β, however, remains a parameter to tune in the training.

We investigate the different relations between the mixture ratio and the average Hamilton-Jacobi-Bellman (HJB). As Fig. (2.3) shows, a small β can tolerate high HJB and leads to a small mixture ratio α, which means the behavioral controller uses more from the learner's policy. In practice, even the optimal solution of the MPC expert doesn't necessarily lead to a zero HJB, and sometimes it leads to a negative HJB, as is also observed in previous works. To address this issue, we choose a relatively low $\beta = 0.1$. In that case, we can keep rolling out the trajectories with mostly the learner's policy and ease the domain mismatch. One potential better way is to use a step function to determine the mixture ratio, as we would accept

Table 2.4: Parameters of the Scan Height Configuration. The noises are controlled by a zero-mean normal distribution, where the standard deviation is the value of interest when we set the parameters.

Name	Noise Description	Value
scanPointNoiseFactor	position of scan points in the terrain plane	0.004
heightNoiseFactor	height of the scan points to the foothold	0.03
perFootNoiseFactor	height of the base	0.00
perFootXYNoiseFactor	xy position of the foot	0.00
perFootZNoiseFactor	height of the foot	0.00
heightOutlierProb	probability of having outlier scan points	0.00
heightOutlierStd	noise of outlier scan points	0.00
blind	if value ≥ 0.5, the robot is blind	0.00

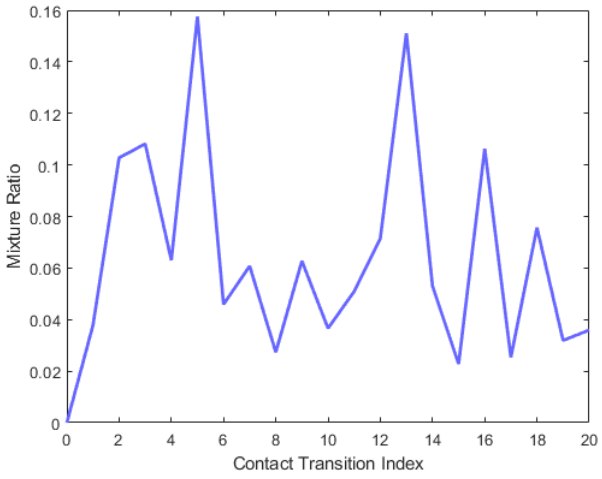


Figure 2.2: Evolvement of the mixture ratio in one example trajectory. Max: 0.157, min: 0.000, mean: 0.063.

solutions that make HJB be below a certain threshold as the "optimal" and only use the learner's policy in that situation.

2.5 Design of the Student Network

In this section, we explain details that are not covered in the main paper that why and how we choose the architecture of the student network.

2.5.1 Frozen MEN

The main purpose of the student network is to robustify the teacher policy while keeping the majority of the network unchanged. Motivated by this, we copy the Multi-Expert Network (MEN) and the exteroceptive encoder from the teacher policy. We tried using the teacher's MEN and exteroceptive encoder as a warm-up for the student policy, and train all parameters from end-to-end in the student training. However, this collapses the training. The main observation is that while the

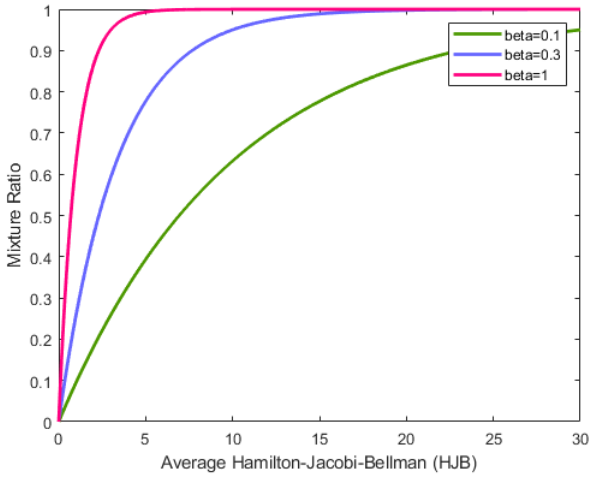


Figure 2.3: Impact of different β: A small β is more tolerant of the high HJB and uses more of the learner's policy. A big β is more conservative and tends to use the expert's policy.

second stage is aimed at coping with the noisy perception input, it has little to do with the MEN. The capacity of the MEN is mainly decided in the teacher training. Therefore, we freeze the MEN in the course of student training, and that turns out to be effective.

2.5.2 Representation Reconstruction

When we train the teacher, the privilege information such as contact forces and contact status is provided to the teacher policy to improve the performance. However, in the student training, we need to reconstruct that information by keeping track of the history. In this thesis, we achieve this in two aspects: by using the Gated Recurrent Network (GRU) and a sequence of proprioceptive observations. Recurrent modules can use their internal state to process variable-length sequences of inputs and is suitable to cope with time-dependent information and eliminate irrelevant noises. The horizon n_s over which one gradient step is optimized in the student training is a parameter to be tuned. We tried two cases: $n_s = 10$ and $n_s = 20$. While a large n_s leads to a longer training time, it doesn't bring significant improvement in terms of policy performance. Although a wider search can be explored, we think setting this parameter as $n_s = 10$ is a general election and won't affect the final performance significantly.

Another factor that can in theory play a role is the history length of the proprioceptive observation n_o. One can regard it as that computing the mean value of a sequence of close-in-time proprioceptive observation should give us a better estimation of the robot state, and the sudden change in the proprioceptive observation indicates the potential switch of contact configuration, which is related to the privilege information. We made a sweep search over the parameter n_o between 2 and 10 and found out that $n_0 = 4$ achieves the best performance among all. The reason that a longer length of history information doesn't bring too much improvement is multi-fold: the averaging effect of the proprioceptive information to incorporate robot estimation may not be necessary to reconstruct the latent representation, and GRU already plays such a role when keeping the internal hidden state. A too-long

history observation may also have counter effects in inferring the contact status of the robot as in theory, two consequent values should be enough to estimate the motion of the foot. Moreover, simply enlarging the state may also make the training less tractable.

2.6 Training Time

One key argument of the advantage of using IL is that IL is sample efficient when good demonstrations are available. In this thesis, the demonstrations are from the MPC expert, and getting them is not a bottleneck. However, the training time is heavily dependent on the operations that are required to evaluate the control Hamiltonian. For all the values reported in this section, the experiments are done in the Euler Cluster using 32 cores, each core having 3200 MB memory. On a fixed terrain, we can train the teacher in less than 12 hours with 40k gradient steps, and then the student training takes less than 3 hours with 10k gradient steps. The challenge is more coming from the randomization of the terrain. Due to the fact that in order to get the components to evaluate the Hamiltonian and the related gradients, we need to set the terrain model for each sampling point. And this is the most expensive part of the training. After adding the randomization feature of the terrain, the computing time reaches about 30 hours for the same number of gradient steps. While one can try to reduce the gradient steps to have a faster convergence, the performance under that situation is often degenerated.

2.7 Choice of Tracking Controller

In this thesis, we replace the whole-body controller (WBC) with inverse dynamics and PD control in the deployment. The main motivation is to mimic the RaiSim simulator, where we use inverse dynamics to compute the forward torque and PD control to track the joint velocity, which is denoted as control mode 1 in RaiSim. From the experiments, we also observed that with WBC, the robot tends to hit really hard on the ground and has stiff motions, which often leads to failures when walking over steps. There is also a reason from the implementation side: one of the tasks that WBC has to fulfill is the end-effector translation constraint, which requires a set of terrain-related constraints that are from the segmentation information in the original *feature_perception* branch of repository *anymal_ctrl_ocs2*. In the thesis, we do not have segmentation in the deployment, and simply removing the end-effector translation task doesn't generate promising solutions. Therefore, we chose the same tracking controller as in the RaiSim simulator.

2.8 Explanation of the Results

In this section, we explain in detail some of the results in the main paper. Table. 2.5 compares the survival time between MPC expert, teacher policy trained in MPC-Net fashion, and teacher policy trained with Behavioral Cloning (BC). We evaluate the experiments on a 8 cm wide gap and $\{10, 12, 14\}$ cm high step. For each experiment, we repeated 50 times with 30 seconds maximum duration. Comparing the Perceptive MPC-Net and BC column, we find that Perceptive MPC-Net always achieves better performance. This can be explained by that MPC-Net explicitly encodes the constraints into the objective and achieves improved constraint

Table 2.5: Survival time comparison against MPC expert and teacher policy trained with Behavioral Cloning (BC). The evaluation is on a 8 cm wide gap and $\{10, 12, 14\}$ cm high step terrain respectively. We collected 50 episodes, each with maximum 30 seconds.

		MPC	Perceptive MPC-Net	BC
Step	10 cm	**30.00±0.00**	**30.00±0.00**	29.06 ± 4.61
	12 cm	**30.00±0.00**	29.10 ± 4.39	25.05 ± 9.61
	14 cm	**30.00±0.00**	26.65 ± 6.99	23.25 ± 9.83
Gap	8 cm	**27.41±4.56**	26.21 ± 5.98	24.09 ± 6.67

Table 2.6: Success rate comparison against MPC expert under noisy environment. For each case, we forward the robot with same twist command and tried 50 attempts. Note that our policy is only trained with a maximum height 14 cm.

	10 cm		12 cm		14 cm		16 cm	
σ	MPC	Ours	MPC	Ours	MPC	Ours	MPC	Ours
0.000	**1.00**	**1.00**	**1.00**	0.98	**0.98**	0.90	**1.00**	0.08
0.030	**1.00**	**1.00**	**1.00**	0.96	0.30	**0.86**	**0.32**	0.06
0.035	0.48	**1.00**	0.32	**0.96**	0.16	**0.82**	0.04	0.04
0.040	0.14	**1.00**	0.18	**0.90**	0.08	**0.80**	0.00	**0.04**
0.100	0.00	**1.00**	0.00	**0.80**	0.00	**0.76**	0.00	0.00

satisfaction. MPC expert achieves the best performance in all experiments since in the simulation environment, the terrain is perfect and there is no noise in the resulting elevation map.

Table 2.6 shows different performances between our student policy and the MPC expert in the noisy environment. We define the success rate as the ratio of successful traverses over the step. When there is no noise as shown in the first row of the table, MPC expert almost traverses all obstacles successfully while our policy deteriorates in performance at the 14 cm step and fails at the 16 cm step. Note that our policy is only trained with a maximum 14 cm step during training, therefore there might be overfitting to the size of the step and it fails to generalize to a higher step. We also notice that when the noise level increases, MPC expert degenerates very fast. The reason is that with the noisy elevation map, the segmentation generator for the MPC expert cannot extract correct segmentation, which leads to unrealistic foothold optimization for the MPC solver. On the other hand, our policy can handle such scenarios very well, as is shown in the table, the performance of the student policy doesn't change too rapidly along with the increase of noise level, and it even achieves a 100% success rate on the 10 cm step. We point out that this noise level corresponds to the *heightNoiseFactor* parameter and therefore such noisy elevation maps are implicitly visited in the student training.

Chapter 3

Conclusion

In this thesis, we added perception to MPC-Net to walk over structured obstacles by learning from MPC experts. A teacher-student framework was used to handle the noisy exteroceptive information and performed better than a single-stage method. We compared the performance under noisy environment against MPC expert and showed our policy is more robust to the noisy elevation map. Moreover, we proposed an adaptive data generation for MPC-Net and it leads to better domain transfer. The simulation results showed the benefit of using MPC-Net comparing against BC. Finally, we validated the viability of the approach on hardware by demonstrating a successful traverse over the structured obstacle.

As a supplementary, we discussed some topics that are not covered in the main paper. First, we questioned the potential overfitting in the training since the relative state is used as the policy input. To avoid using obstacle information during training without compromising the policy performance, we provided some alternatives that can be considered in the future. For instance, one can use absolute value instead of relative value for all robot state except for the base position x, y, and ψ. One can also add twist command into the policy input and remove the relative base position x, y, and ψ completely. We also talked about the noise configuration used in this thesis. Specifically, we used different noise to the robot state during the rollout, policy input, and the policy output, as well as the perception input. Moreover, a more detailed discussion about the adaptive data generation is provided and we observed that using more of the learner's policy help with the better domain transfer. The reasoning about the design choice of the student network highlighted the importance and the methodology of using a teacher-student learning framework. The final discussion on the choice of tracking controller also gives insight on reducing the sim-to-real gap in the future.

To conclude, this thesis extends MPC-Net by adding perception, and the successful hardware validation opens up the door to a wider range of obstacles. With potential improvement on the policy as discussed before in the thesis, we are confident that the robot will one day walk in a human-orientated environment.

Chapter 4

Outlook

While Perceptive MPC-Net opens the door to more versatile kinds of obstacles and a wider range of perceptive applications, the current policy is limited in several ways, which leaves room for potential improvements. The policy is not robust, a successful traverse usually requires careful control over the speed and a properly placed obstacle setup. The experiments are limited to a single step and the size of the type is not given enough randomization during the training, which leads to overfitting over the deployment environment.

Compared to the blind MPC-Net, the robustness of the policy deteriorated. Although a perceptive policy is a more complex problem with a much larger state space, and the fact that we used the desired trajectory instead of a desired point further makes the policy deviate from being robust, we would expect a better policy to walk on steps with more confidence by a more careful design of the noise configuration and the improvement in the implementation.

For example, one can change the control mode in the RaiSim simulator such that no internal PD feedback of RaiSim is used, and the feedback should be explicitly computed and added to the direct torque. In that case, the way the robot receives the torque would be the same in the training and the deployment, thus reducing the sim-to-real gap. The main reason for this is that we found the internal PD gain of the simulator or the drives is usually scaled in different ways, and without the knowledge of how the simulator adds up the feedback part, the deployment would be problematic.

Secondly, one should rethink the usage of the cost desired trajectory, or more specifically, the choice of the network input. Using the so-called relative state assumes the knowledge of the terrain information, and the resulting consequence that the robot is tracking a reference trajectory makes it harder to incorporate the terrain information. Theoretically, the reference trajectory should also be inferred from the terrain information. We tried to remove the relative base state to address this issue. The preliminary results showed that removing the relative base state, however, makes it harder to control the robot by the twist command. One can try to use the absolute state and add twist command into the policy input to avoid using the terrain information during training But one potential problem as commonly addressed in RL is that the accumulating error in the absolute state would bring training difficulties.

Furthermore, the type of terrain can be increased. Currently, we have only trained the policy on a single step, and the randomization over the step size is not enough

to cover the lab environment. One can add slope, stepping stones into the training, and train a single policy that can handle different terrains.

Bibliography

[1] J. Carius, F. Farshidian, and M. Hutter, "Mpc-net: A first principles guided policy search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, 2020.

[2] A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter, "Imitation learning from mpc for quadrupedal multi-gait control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.