Advancing Legged Robot Agility: from Video Imitation to GPU Acceleration

John Z. Zhang

CMU-RI-TR-24-46

August 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Zachary Manchester, Chair
Deva Ramanan
Guanya Shi
Kevin Tracy

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To mom and dad.

Abstract

Achieving human and animal-level agility has been a long-standing goal in robotics research. Recent advancements in numerical optimization and machine learning have pushed legged systems to greater capabilities than ever before, enabling black flips, parkour, and manipulation of heavy objects. Despite these exciting developments, this thesis identifies two key limitations of current legged robot technology and aims to improve upon existing art.

First, legged robots today require manual specifications of desired behaviors and fail to learn from their human and animal counterparts. We introduce SLoMo, a first-of-its-kind framework for transferring skilled motions from casually captured videos of humans and animals to legged robots. From a monocular RGB video, SLoMo synthesizes physically plausible trajectories for downstream offline trajectory optimization and online predictive control of quadruped or humanoid robots. We demonstrate SLoMo by transferring cat and dog motions to quadruped robot hardware and human motions to a simulated humanoid robot.

Second, current model-predictive control (MPC) for legged systems often resort to simplified models due to computational limitations in real-time settings. This is due to the high dimensionality of these robots and the reliance of existing numerical optimization algorithms on fundamentally serial, CPU-friendly linear algebra routines. We leverage advancements in GPU parallelization by developing a quadratic programming (QP) solver that uses only GPU-friendly operations. We refer to our solver as ReLU-QP, thanks to its computational similarities to inferencing a deep neural network with rectified linear unit (ReLU) activation functions. Across benchmarks on solving random QPs and high-dimensional MPC tasks in simulation, including balancing a full-order Atlas humanoid robot on one foot under control limits, ReLU-QP shows an order-of-magnitude speed improvement over state-of-the-art CPU-based QP solvers and solves MPC for modern legged robots at kilohertz rates.

Acknowledgments

The work in this thesis is only made possible thanks to the support of many incredible people over the past two years.

I owe much gratitude to my advisor, Zachary Manchester. Zac's deep theoretical understanding, profound practical insights, and countless crazy ideas have been a fountain of inspiration during my Master's studies. Perhaps more importantly, I was fortunate to learn from Zac's generous attitude toward people and optimistic perspectives for research and life.

To my thesis committee members: Deva Ramanan, Guanya Shi, and Kevin Tracy, thank you for your valuable feedback on my work. I was incredibly fortunate to collaborate with Deva as a first-year graduate student. Every meeting with Deva left me more optimistic and energetic about the next steps in my project. I am grateful to have met Guanya at the beginning of my second year at CMU. Guanya has never failed to engage me in interesting technical and philosophical discussions during our encounters. Kevin has been an amazing friend and constant beacon of reason since I joined the RExLab. I appreciate your patience while answering all my random questions.

I'm grateful for all my collaborators during my master's research: Shuo Yang, Arun Bishop, Swaminathan Gurumurthy, Gengshan Yang, Zixin Zhang, Kevin Tracy, Simon Le Cleac'h, Taylor Howell, and Chi-yen Lee. The opportunity to learn from each of you every day has been my favorite part of grad school and research. Putting together this thesis is another reminder of how much we've accomplished together and how lucky I have been to be a small part of your success.

Grad school would not be the same without all the amazing people in the (extended) RExLab and RI family, especially Sofia Kwok, Fausto Vega, Jeong-Hun (JJ) Lee, Samuel Schoedel, Bowen Jiang, Jonathan Francis, Peter Schaldenbrand, Uksang Yoo, Matthew Sivaprakasam, Conner Pulling, Brian Jackson, Jacob Willis, Mitchell Fogelson, Brian Plancher, Zhanyi Sun, Yilin Wu, and Yaru Niu. Thank you for celebrating the incremental achievements with me and supporting me through all the challenges that came along the way.

Outside of CMU, I must thank Ruoyang (Alex) Xu and Shiyu (Kelly) Chen for your years of friendship and company in my often-crazy life. I always look forward to and cherish our get-togethers. Thank you for

allowing me to be part of your lives. I additionally want to thank my landlady, Judie Compher, for welcoming me to your house with open arms and providing a peaceful place I have the pleasure of calling home.

Finally, Mom and Dad have been my strongest support in my journey thus far. The writing of this thesis also concludes my 10th year studying in the U.S. It's hard to imagine making all the difficult decisions to get here without having both of you as my backstop. I cannot thank you enough your belief in me as I searched for my interests over the years and eventually pursued research and robotics. This thesis is dedicated to you.

Funding

This work was partly supported by a Google Faculty Research Award.

Contents

1 Introduction **1**

2 Legged Robot Motion Imitation from Monocular Videos **3**
2.1 Introduction 4
2.2 Background and Related Work 6
2.2.1 Human and Animal Motion Capture 6
2.2.2 Trajectory Optimization through Contact 7
2.2.3 Online Control for Legged Robots 8
2.2.4 Motion Imitation 9
2.3 Video-to-Robot Motion Transfer 10
2.3.1 Physics-Informed Reconstruction from Casual Videos 10
2.3.2 Contact-Implicit Trajectory Optimization 13
2.3.3 Contact-Implicit Model-Predictive Control 14
2.4 Experiments and Results 15
2.4.1 Experimental Setup 15
2.4.2 Quadruped 16
2.4.3 Humanoid 17
2.4.4 Comparisons to an RL Policy 18
2.5 Discussion and Conclusions 19
2.5.1 Limitations 19
2.5.2 Future Work 20

3 GPU Accelerated Quadratic Programming Solver **21**
3.1 Introduction 21
3.2 Background 25
3.2.1 Convex Quadratic Programs 25
3.2.2 Alternating Direction Method of Multipliers 25
3.2.3 Model-Predictive Control 28
3.3 The ReLU-QP Algorithm 29
3.3.1 Modified ADMM Iterations 29
3.3.2 Algorithm Summary 30
3.3.3 Implementation details 30
3.4 Dense Model-Predictive Control 32
3.5 Simulation Experiments 33

3.5.1 Random Dense QPs 33
3.5.2 Random Linear MPC Problems 34
3.5.3 Atlas Balance 35
3.5.4 Whole-Body Pick-up Motion on a Quadruped with an Arm . . 36
3.6 Discussion and Conclusions 38
3.6.1 Limitations 38
3.6.2 Future Work 38

4 Conclusion **39**

Bibliography **41**

List of Figures

2.1 A collection of video-to-robot motion-transfer demonstrations on the Unitree Go1 quadrupedal robot on hardware (left three) and the Atlas humanoid robot in simulation (right two). From left to right: a dog reaching for a water feeder with one of its front feet, a house cat pacing, a trained dog performing a Cardiopulmonary Resuscitation (CPR) exercise on its human partner, a human stretching his body and limbs, and a human demonstrating a jumping-jack exercise. 3

2.2 The SLoMo algorithm: (a) Stage 1: Process visual RGB inputs and generate body and foot trajectories in $\mathbb{R}^3$. This reconstruction considers physics but does not produce a trajectory that is dynamically feasible on the target robot. (b) Stage 2: Solve for open-loop robot state, control, and contact-force reference trajectories that imitate the 3D reconstruction while obeying robot dynamics and contact constraints. (c) Stage 3: Track the reference trajectory on robot hardware while managing model and contact-timing mismatch and disturbances. . . . 5

2.3 The physics-informed 3D reconstruction pipeline: Differentiable rendering from a monocular video (left) and differentiable physics simulation (right) update key-point motion estimates (middle). Solid red and green arrows represent rendering and physics-roll-out costs; dashed arrows are corresponding gradients. 12

2.4 *Dog CPR* (left), *Human Jumping Jack* (right) motion imitation demonstrations. The top row shows frames from the original video, the second row shows the key-point trajectory, and the third row shows the dynamically feasible trajectory being executed by the Unitree Go1 and Atlas robots. The bottom row on the left shows *dog CPR* on hardware. 16

2.5 Foot height trajectories comparison for the dog-CPR experiment. Each plot corresponds to one foot of the robot. The blue lines are trajectories generated by the 3D reconstruction stage (S1). The red lines are the outputs of the trajectory optimization stage (S2). The yellow lines are the state feedback on robot hardware showing the result foot trajectories of the MPC policy (S3). 18

2.6 A comparison between the reconstructed key-point reference (yellow), optimized reference (teal, ours), and learned imitation policy [8] (purple). 19

3.1 MPC solve times for random linear systems with control limits (horizon = 40, 3:1 state-to-control dimension ratio). ReLU-QP is an order of magnitude faster on high-dimensional control problems. The preconditioned condensed formulation is used except for *OSQP-Direct*, which solves the sparse direct formulation. 23
3.2 Solve times for random dense QPs with both equality and inequality constraints demonstrating that ReLU-QP is up to 30 times faster than OSQP and ProxQP on large problems. 34
3.3 An Atlas humanoid robot balancing on one foot subject to control limits and disturbances. ReLU-QP successfully solves the task at up to 2600 Hz. 35
3.4 A Go1 robot equipped with a 6 DoF arm performing a pick-up motion using linear MPC. ReLU-QP solves the problem at 834 Hz. 37

List of Tables

2.1 A comparison between recent motion imitation methods for legged systems 9

3.1 Average MPC solve frequencies for a whole-body Atlas stabilizing task. ReLU-QP demonstrates 10-20 times speed ups. 36

3.2 Average MPC solve frequencies for a whole-body quadruped model with an arm picking up an object from the ground. ReLU-QP demonstrates a 2× speed up over OSQP. 38

Chapter 1

Introduction

Achieving human and animal-level agility has been a long-standing goal in robotics research. Recent advancements in numerical optimization and machine learning have pushed legged systems to greater capabilities than ever before, enabling black flips, parkour, and manipulation of heavy objects. As these robots walk into people's homes, inspection sites, and manufacturing plants, key challenges have to be overcome so they can successfully operate in these challenging real-world scenarios.

In chapter 2, we introduce the first framework for transferring skilled motions from casually captured videos of humans and animals to legged robots. We refer to this framework as SLoMo. Traditional motion imitation for legged motor skills often requires expert animators, collaborative demonstrations, and/or expensive motion-capture equipment, all of which limit scalability. Monocular videos, on the other hand, are widely available on the open web and easily obtained on personal smartphone cameras. From a monocular RGB video, SLoMo synthesizes physically plausible trajectories for downstream offline trajectory optimization and online predictive control of quadruped or humanoid robots. We demonstrate SLoMo by transferring cat and dog motions to quadruped robot hardware and human motions to a simulated humanoid robot. This work was performed in collaboration with Gengshan Yang, Shuo Yang, Arun Bishop, Swaminathan Gurumurthy, Deva Ramanan, and Zachary Manchester and was published in Robotics and Automation Letters in 2023 [92].

In chapter 3, we leverage advancements in GPU parallelization by developing a quadratic programming (QP) solver that uses only GPU-friendly operations. We refer

to our solver as ReLU-QP, thanks to its computational similarities to inferencing a deep neural network with rectified linear unit (ReLU) activation functions. Across benchmarks on solving random QPs and high-dimensional model predictive control tasks in simulation, including balancing a full-order Atlas humanoid robot on one foot under control limits, ReLU-QP shows an order-of-magnitude speed improvement over state-of-the-art CPU-based QP solvers and achieves MPC at kilohertz rates. This work was co-led with Arun Bishop and in collaboration with Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester and was presented at the 2024 International Conference for Robotics and Automation in Yokohama, Japan [9].

Chapter 2

Legged Robot Motion Imitation from Monocular Videos

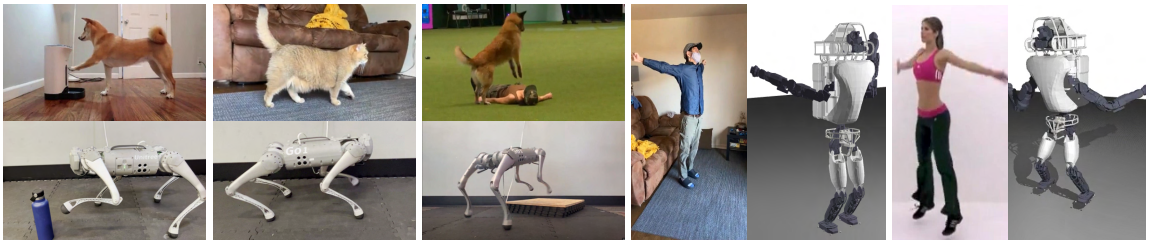

Figure 2.1: A collection of video-to-robot motion-transfer demonstrations on the Unitree Go1 quadrupedal robot on hardware (left three) and the Atlas humanoid robot in simulation (right two). From left to right: a dog reaching for a water feeder with one of its front feet, a house cat pacing, a trained dog performing a Cardiopulmonary Resuscitation (CPR) exercise on its human partner, a human stretching his body and limbs, and a human demonstrating a jumping-jack exercise.

We present SLoMo: a first-of-its-kind framework for transferring skilled motions from casually captured "in-the-wild" video footage of humans and animals to legged robots. SLoMo works in three stages: 1) synthesize a physically plausible reconstructed key-point trajectory from monocular videos; 2) optimize a dynamically feasible reference trajectory for the robot offline that includes body and foot motion, as well as a contact sequence that closely tracks the key points; 3) track the reference trajectory online using a general-purpose model-predictive controller on

robot hardware. Traditional motion imitation for legged motor skills often requires expert animators, collaborative demonstrations, and/or expensive motion-capture equipment, all of which limit scalability. Instead, SLoMo only relies on easy-to-obtain videos, readily available in online repositories like YouTube. It converts videos into motion primitives that can be executed reliably by real-world robots. We demonstrate our approach by transferring the motions of cats, dogs, and humans to example robots including a quadruped (on hardware) and a humanoid (in simulation). Videos are available at https://slomo-www.github.io/website.

2.1 Introduction

One of the grand challenges in robotics is to enable human- and animal-level agility for legged robots by directly imitating their natural counterparts. This motion-imitation procedure typically involves three steps: extracting motion primitives from videos or image sequences, processing motion primitives to ensure they are within the physical limits of the robot and, finally, executing those movements on robot hardware. Prior work on motion imitation relies on marker-based, multi-camera motion-capture (MoCap) systems, limiting the diversity of captured movements. An end-to-end motion-transfer solution that takes in raw video data and executes novel behaviors on real-world robots has, so far, remained elusive.

In this paper, we leverage recent advancements in neural rendering and reconstruction, trajectory optimization, and model-predictive control to build, to the best of the authors' knowledge, the first successful general framework for transferring human and animal motion skills captured by a single, moving camera to robot hardware. The framework, illustrated in Fig. 2.2, contains three modules: 1) a reconstruction pipeline that produces physically plausible 3D key-point trajectories from casual video footage; 2) a trajectory optimizer that solves for dynamically feasible robot state, control, and contact-force reference trajectories that closely mimic the key-point trajectories while respecting the physical limitations of the robot; and 3) a model-predictive controller (MPC) that runs at real-time rates on robot hardware to track reference trajectories. An important feature of our approach is explicit reasoning about contact interactions between the robot and environment at every stage of the framework, ensuring that offline reference trajectories can be safely executed on hardware and that the online

MPC is robust to contact-timing and model mismatch. Additionally, model-based trajectory generation and control methods allow us to maintain explainability in each stage of the pipeline — something difficult to achieve with current black-box reinforcement learning (RL) approaches. Finally, our work also differs from prior works in its generality across robot morphology; our 3D reconstruction pipeline, trajectory optimizer, and MPC are all robot agnostic — enabling motion transfer from humans to humanoid robots and from animals to quadrupedal robots within a single, unified framework.

Our specific contributions are:

- A general-purpose motion-transfer framework, SLoMo, for enabling legged robots to mimic human and animal motions from casual videos
- A novel offline reference-trajectory and contact-sequence generation technique that ensures physical feasibility
- End-to-end experimental demonstrations transferring animal behaviors to a quadruped robot on hardware and human motions to a humanoid robot in simulation

This paper is organized as follows: We review related literature in Section 2.2. Our methodology is introduced in Section 2.3. Results of simulation and hardware experiments are reported in Section 2.4. Finally, we summarize our conclusions and discuss directions for future research in Section 2.5.

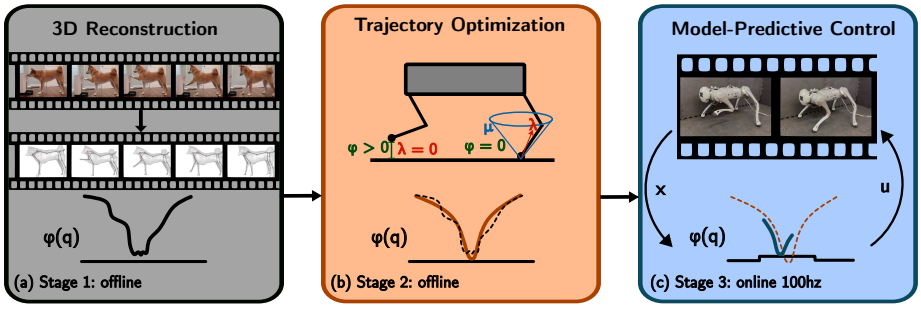

Figure 2.2: The SLoMo algorithm: (a) Stage 1: Process visual RGB inputs and generate body and foot trajectories in $\mathbb{R}^3$. This reconstruction considers physics but does not produce a trajectory that is dynamically feasible on the target robot. (b) Stage 2: Solve for open-loop robot state, control, and contact-force reference trajectories that imitate the 3D reconstruction while obeying robot dynamics and contact constraints. (c) Stage 3: Track the reference trajectory on robot hardware while managing model and contact-timing mismatch and disturbances.

2.2 Background and Related Work

In this section, we review related literature on human and animal motion capture, trajectory optimization through contact, and online control for legged robots.

2.2.1 Human and Animal Motion Capture

To study human and animal body movements, MoCap systems have been widely adopted in the movie industry and research labs [34, 49, 82]. An optical MoCap system typically consists of multiple high-resolution cameras whose positions and orientations are precisely measured through a sophisticated calibration process. Multiple cameras can observe and triangulate the positions of markers, which can be mounted on human or animal subjects. Although some MoCap datasets have been made publicly available [82], obtaining novel motion sequences remains a challenge. Despite these shortcomings, most existing works in motion imitation [8, 33] rely on MoCap data. Marker-based MoCap systems have inherently limited capture volume and are very sensitive to camera configuration changes, making outdoor usage extremely difficult and unreliable. MoCap systems also typically cost tens to hundreds of thousands of dollars. All of these factors limit the diversity of environments and targets that can be studied with a MoCap system. We aim to develop a low-cost solution for imitating diverse, in-the-wild motion skills from easily-accessible videos.

Alternatively, markerless motion capture has become increasingly popular in recent years [6, 31, 90]. For example, [31] built a multi-view video-capture system to capture and reconstruct human motion. Although those systems capture whole-body movement without using markers, they still require an indoor studio with hundreds of synchronized cameras, making it challenging to generalize to in-the-wild targets and behaviors. Some recent works [32, 40] learn data-driven models to predict full body movements from a single monocular camera. However, they heavily rely on carefully-constructed template models (e.g., SMPL [48]) and do not generalize well to in-the-wild videos or non-human subjects.

Recent advances in differentiable rendering [54, 81] and robust dense point tracking (e.g. optical flow [80, 85] and DensePose [24]) have enabled test-time optimization of dense surface structure and motion given real-life videos [86, 87]. Building on

these prior algorithms, our work performs key-point trajectory tracking of human and animal motions in 3D from a single, moving camera video without assuming a predefined shape template.

2.2.2 Trajectory Optimization through Contact

Trajectory optimization is a powerful tool for designing dynamic behaviors for robotic systems. Given an initial guess, trajectory optimization formulates a nonlinear program (NLP) to solve for an optimal control sequence under robot dynamics and environmental constraints. This technique has been a major component of important breakthroughs in legged autonomy in recent years [10, 11, 35, 45].

One of the hardest problems in planning and control for legged robots is reasoning about contact forces and timing as feet make and break contact with the environment, producing discontinuous impact events. A common approach for modeling rigid-body contact interactions is to formulate the dynamics as a linear complementarity problem (LCP) [27, 78], which solves for the next system state under impact and friction constraints. These LCP dynamics can be enforced as constraints in trajectory optimization methods [51, 65].

If the contact schedule is predefined [10], the dynamics can be written as a hybrid system with known transition times. This method, commonly referred to as hybrid trajectory optimization [52, 61], can be solved quickly and is effective for periodic gaits on flat terrain. This approach has also enabled diverse locomotion behaviors on robot hardware through motion-template libraries that can then be combined and tracked online to form long-horizon locomotion behaviors over challenging terrains [11].

In contrast, if the contact schedule cannot be determined a priori, the contact interactions must be solved *implicitly*, resulting in a much larger and more challenging nonlinear optimization problem [1, 28]. This method is referred to as contact-implicit trajectory optimization [51, 65]. Reliably solving contact-implicit trajectory optimization problems to high accuracy is challenging even in offline settings, and the solution can be sensitive to model mismatch. Several previous studies have aimed to improve numerical accuracy [51], convergence properties [28], or robustness to contact model uncertainty [16, 17].

In this work, we reason about contact interactions in each stage: During 3D reconstruction, we roll out reconstructed motions in a differentiable simulator [50], where contact is approximated with a spring-damper model. In trajectory optimization, we enforce rigid-body contact dynamics using LCP constraints [51]. Finally, during online tracking, we solve a simplified contact-implicit problem that can fully reason about contact mode timing.

2.2.3 Online Control for Legged Robots

Online feedback control has been widely studied for both quadrupedal [10] and bipedal [66] robots in recent years. MPC [10, 14] and RL [20, 75] have emerged as popular approaches for executing dynamic locomotion behaviors on legged systems.

Different from offline trajectory optimization, MPC typically uses a simplified model and only solves a finite-horizon variant of the optimal control problem to achieve real-time performance. Notably, [10] uses a heuristic foothold location scheduler and solves a convex quadratic program (QP) for desired stance-foot forces. This method achieves robust locomotion without any offline computation. A lower-level whole-body controller (WBC) [36] or inverse kinematics (IK) is then used to map the MPC solution into the robot joint space. At the motor level, a hand-tuned proportional-derivative (PD) controller is used to track this joint-level trajectory on the robot. The offline trajectory optimization and online MPC pipeline have enabled several impressive real-world robot behaviors, such as quadruped jumping [57, 93], humanoid parkour [11], and even coordinated heterogeneous muti-robot dance routines [12].

Model-free RL aims to learn a feedback control policy, typically represented by a deep neural network, by collecting experience in a simulated environment or the real world. The learned policy is optimized by maximizing a reward function using gradient descent algorithms. Similar to model-based pipelines, recent RL methods also rely on hand-tuned low-level PD controllers to execute behaviors in the real world [20, 75]. RL has also been successfully applied to animal imitation from motion capture reference data on a quadrupedal robot[8].

In this work, we adopt a general-purpose contact-implicit MPC (CI-MPC) algorithm [14], capable of controlling both quadruped and humanoid robots, and pair it with a low-level joint-space controller based on IK and PD feedback for hardware

Table 2.1: A comparison between recent motion imitation methods for legged systems

	[63]	[8]	[47]	[89]	[33]	SLoMo
Single RGB Camera	✗	✗	✗	✓	✗	✓
No Manual Label	✓	✓	✗	✓	✓	✓
Robot Hardware	✗	✓	✓	✓	✓	✓
System Agnostic	✓	✗	✗	✗	✗	✓

execution. Our contact-implicit MPC formulation can reason about contact timing and forces in real time, enabling robust tracking of complex behaviors.

2.2.4 Motion Imitation

Imitating motion primitives from nature can be an effective strategy for producing natural-looking robot behaviors while avoiding tedious, manual trajectory design. Researchers have studied various approaches for mimicking human motions [41, 64]. For example, learning-from-observation (LFO) converts MoCap motion sequences into reference motion primitives that can be tracked by a zero-moment point (ZMP) controller, which enables a humanoid robot to graciously perform traditional Japanese dance routines [55].

Recently, imitation learning has been used to produce animal-like movements for animation [63]. [8, 37, 89] demonstrated imitated motions where the sim-to-real gap was bridged through online adaptation on a quadrupedal robot. Similarly, model-based imitation [33, 47] has also produced animal-like locomotion on a real-world quadrupedal robot using hybrid trajectory optimization, where foothold locations and timing were predetermined by thresholding MoCap data. These prior works rely on marker-based motion capture to acquire motion priors. [89] uses RGB videos but still relies on manual labeling to acquire trajectories. As discussed in Section 2.2.1, these methods have some major limitations. Recent videos from Boston Dynamics [12] demonstrate impressive robot dancing through complicated choreography design, character animation, and robot trajectory optimization procedures; an expensive, time-consuming process.

In robot manipulation, recent work [5, 74] takes essential steps towards acquiring in-the-wild robotic skills from real-world RGB videos through reinforcement learning. The key insight from these two studies is that computer-vision techniques can now

process widely-available human and animal footage into representations that can be very effective priors for acquiring robotic skills. In this work, we tackle the problem of imitating locomotion skills by extracting target motions from videos using 3D reconstruction. Different from previous RL-based imitation methods that only consider robot kinematics, we use optimal control methods that allow us to explicitly reason about dynamics.

We propose a simple and cost-effective method: synthesize legged robot motion primitives directly from casual RGB videos without manual labels and leverage a model-based controller for robust execution on robot hardware. Additionally, we demonstrate that our control strategy generalizes across quadruped and humanoid robots. Moreover, the model-based approach allows us to interpret the output trajectories and explicitly reason about hardware limitations like torque limits.

2.3 Video-to-Robot Motion Transfer

In this section, we present the SLoMo algorithm for robot motion imitation from in-the-wild videos: Section 2.3.1 (Fig. 2.2 (a)) describes the 3D reconstruction pipeline for generating physically-plausible key-point trajectories from casual footage. Section 2.3.2 (Fig. 2.2 (b)) explains the offline trajectory-optimization problem used for constructing dynamically feasible robot reference trajectories and foot-contact sequences that mimic key-point movements. Section 2.3.3 (Fig. 2.2 (c)) details the contact-implicit MPC algorithm[14] for tracking the optimal reference online.

2.3.1 Physics-Informed Reconstruction from Casual Videos

Given videos of a target animal or human, our goal is to estimate its kinematic key-point trajectory in the world coordinates. Similar to prior work [86, 87, 88], we simultaneously reconstruct articulated shapes and kinematic skeleton trajectories, connected by a blend skinning model. To reconstruct physically plausible trajectories, we set up a physics-informed optimization by coupling differentiable rendering costs with an additional physics-roll-out cost.

We define the $\mathbf{x}(t) \in \mathbb{R}^{3N}$ to be the estimated key-point trajectory, where N is the total number of key-points, $t \in \{1 \dots T\}$ to be discrete time steps, and T to be

the number of frames in a given video.

Shape Model: We use a Multi-Layer Perceptron (MLP) parameterized by σ to represent the visual properties of the default state of the object:

$$(d, \mathbf{c}) = \mathbf{MLP}_{\sigma}(\mathbf{X}), \tag{2.1}$$

where $d \in \mathbb{R}$ is the assigned signed distance and $\mathbf{c} \in \mathbb{R}^3$ is the color at each point $X \in \mathbb{R}^3$. The points that have distance $d = 0$ are on the surface of the object. This representation is similar to a neural radiance field (NeRF) [54] except we remove the view dependence of color.

Kinematic Skeleton Model: To model the motion of the subject animal or human, we use a predefined target kinematic skeleton model consisting of B rigid links and N spherical joints, among which a root link is selected to define the model's location and orientation in space. For simplicity, we utilize joint locations as key points (Fig. 2.3). The key-point trajectory $\mathbf{x}(\mathbf{t})$ can be calculated using forward kinematics. When the skeleton changes configuration, the object's shape should deform and move along with the skeleton. This motion is described by a neural blend-skinning model $\mathcal{W}_t(\mathbf{X})$ [86] as an MLP parameterized by λ:

$$\mathcal{W}_t(\mathbf{X}) = \mathbf{MLP}_{\lambda}(\mathbf{X}; Q, G, t). \tag{2.2}$$

The neural blend-skinning model warps 3D points of the new configuration to the default configuration, which can then be used to query the visual properties of the object. We further parameterize the joint angles $Q = \mathbf{MLP}_{\kappa}(t)$ and SE(3) root-link transformation $G = \mathbf{MLP}_{\eta}(t)$. Then, after training, the zero-level set of d given by

$$(d, \mathbf{c}) = \mathbf{MLP}_{\sigma}(\mathcal{W}_t(\mathbf{X})) \tag{2.3}$$

represents the surface of the object at time t.

Differentiable Volume Rendering: In addition to the object shape model, we train another parameterized background scene model similar to Eq. (2.1). The shape and scene models can then be used together to differentiably render images given a camera's view transformation and intrinsic parameters [58]. We achieve this by performing ray casting for each image pixel p. For all 3D points along the ray

Figure 2.3: The physics-informed 3D reconstruction pipeline: Differentiable rendering from a monocular video (left) and differentiable physics simulation (right) update key-point motion estimates (middle). Solid red and green arrows represent rendering and physics-roll-out costs; dashed arrows are corresponding gradients.

within a given distance range, we use Eq. (2.3) to query their 3D color and density and perform a weighted average to compute $\hat{\mathbf{c}}_t(p)$, the pixel values on the rendered image, including 2D color and object silhouette. We further render optical flow from t to $t+1$ by warping and projecting the queried points from the current configuration to the next frame configuration.

Rendering Cost: We compare the expected color, object silhouette, and optical flow of pixels $\hat{\mathbf{c}}_t(p)$ on the rendered image with the input video observations at time t. The observation $\bar{\mathbf{c}}_t(p)$ comes from basic image-processing and segmentation methods [39, 85]. A rendering cost function is defined as:

$$\mathcal{L}_{render} = \sum_t \sum_p \|\hat{\mathbf{c}}_t(p) - \bar{\mathbf{c}}_t(p)\|^2 \tag{2.4}$$

We use its gradients to update the model parameters σ, λ, κ, and η with the Adam optimizer [38].

Physics Roll-Out Cost: Using only the rendering cost, the trained shape model can generate motions from the same view point [86]. However, these motions are often physically unrealistic due to piece-wise scale ambiguity at each individual patch [46] — a common issue for reconstructing monocular videos. For example, a dog can be up close and floating in the air or far away and on the ground. Both are equally valid from the same monocular visual evidence, but only the latter obeys physics. To resolve this ambiguity, we introduce a physics-roll-out cost in the optimization problem to encourage a physically plausible rendering solution.

Treating the key-point skeleton as a floating-base multi-rigid-body system, we denote its generalized coordinates as $q(t)$. During differentiable rendering, using the latest parameters κ and η, we can generate a reference trajectory $q_d(t)$ with

$t \in \{1 \dots T\}$. Then, in a differentiable simulator [50], we simulate the multi-body key-point skeleton model with a simple PD controller attempting to track $q_d(t)$ to compute the following cost,

$$\mathcal{L}_{physics} = \sum_t \|q(t) - q_d(t)\|^2. \tag{2.5}$$

This cost function is differentiable with respect to κ, η, and ξ. Note that, in this stage, the physics cost only encourages physical realism in a "soft" way and does not guarantee dynamic feasibility like later stages in SLoMo. Thus, we deem the rendered key-point trajectory physically *plausible*.

Coordinate-Descent Optimization: In theory, the cost $\mathcal{L}_{render} + \mathcal{L}_{physics}$ can be optimized together. However, in practice, the volume rendering and the physics simulator run at different frequencies, making joint optimization inefficient. Instead, we use coordinate descent to alternately minimize each cost function. In each iteration, we first minimize the rendering cost, during which we regularize Q and G's output toward the previous simulator-generated trajectory. We then perform a physics roll-out optimization step, where only physics parameters ξ are updated. We also use other strategies to improve convergence such as over-parameterization . At convergence, we take the rendered key-point trajectories $x^*(t)$ as output to the next stage of the motion-imitation pipeline. More details of the reconstruction optimization can be found in [86, 88].

2.3.2 Contact-Implicit Trajectory Optimization

After generating key-point trajectories from monocular videos, we solve a trajectory-optimization problem subject to robot dynamics and contact constraints. In particular, we use contact-implicit trajectory optimization [51, 65], which jointly solves robot states, controls, and contact forces with a direct collocation formulation. Compared to hybrid trajectory optimization [52, 61], the contact-implicit method does not require a predefined contact sequence, which is an important advantage in our motion-imitation workflow since the reconstructed key-point trajectories can suffer from dynamically infeasible artifacts (e.g. foot sliding), even with the physics roll-out cost, which makes applying a heuristic contact schedule impractical. By using the contact-implicit

formulation, we allow the optimizer to automatically generate a feasible robot gait sequence and corresponding reference trajectory that is similar to the key-point trajectory without separately predefining a contact schedule.

The offline contact-implicit trajectory-optimization problem has the following form,

$$\begin{aligned} \underset{\mathcal{H},\mathcal{X},\mathcal{U},\lambda}{\text{minimize}} \quad & \sum_{t=1}^{T-1} \frac{h_t}{2}\Big[(x_t - x_t^*)^\top Q(x_t - x_t^*) + u_t^\top R u_t\Big] \\ & + (x_N - x_N^*)^\top Q_N(x_N - x_N^*) \\ \text{subject to} \quad & x_{t+1} = \mathbf{NCP}_t(h_t, x_t, u_t, \lambda_t), \\ & u_t \leq u_{max}, \\ & u_t \geq u_{min}, \end{aligned} \tag{2.6}$$

where h_t is the time step, $x_t = (q_t, v_t)$ is the state, and u_t are the controls, and λ_t are contact forces at time t, respectively. $\mathbf{NCP}(x, u)$ represents the robot's nonlinear dynamics, including contact constraints, as a nonlinear complementarity problem [27, 51]. We optimize a quadratic tracking cost where x_t^* is the reconstructed key-point state at time t and Q and R are diagonal weighting matrices. This formulation allows the solver to infer a contact sequence and corresponding robot trajectories by imitating noisy and dynamically infeasible key-point data. We refer to a solution of (2.6) as a *reference trajectory*.

2.3.3 Contact-Implicit Model-Predictive Control

To stabilize and track reference trajectories in real-time, we use contact-implicit model-predictive control [14]. CI-MPC is a general method for controlling robots that make and break contact with their environment. Different from standard convex MPC approaches, CI-MPC models the robot's dynamics with a time-varying linear complementarity problem. This LCP can be thought of as a local approximation of the nonlinear complementarity problem in (2.6) about the reference trajectory, which makes it computationally easier to solve. Importantly, however, it maintains the ability to reason about discontinuous contact-switching events.

The CI-MPC tracking problem is:

$$\begin{aligned} \underset{\mathcal{X},\mathcal{U}}{\text{minimize}} \quad & \sum_{t=1}^{H} \frac{1}{2}\Big[(x_t - \bar{x}_t)^\top Q(x_t - \bar{x}_t) \\ & + (u_t - \bar{u}_t)^\top R(u_t - \bar{u}_t)\Big] \\ \text{subject to} \quad & x_{t+1} = \mathbf{LCP}_t(x_t, u_t), \\ & u_t \leq u_{max}, \\ & u_t \geq u_{min}, \end{aligned} \tag{2.7}$$

where x_t, u_t are state and control decision variables and $\bar{x}_t$, $\bar{u}_t$ are the reference state and control trajectories from (2.6) and H is the MPC horizon, which is generally shorter than the full length T of the reference trajectory to enable faster online solution times. The interested reader is referred to [14] for more details on CI-MPC.

2.4 Experiments and Results

In this section, we present the results of experiments demonstrating the capabilities of SLoMo on a variety of robotic systems in simulation and on hardware. In particular, we demonstrate two important features of our method: 1) our entire framework can successfully transfer motions from casual videos to robot hardware and 2) our pipeline is model-agnostic and can support both quadruped and humanoid robots. We carefully identify canonical-legged robot movements that require no contact switching, periodic contact switching, and dynamic non-periodic contact switching. Experiment videos are available on our website. An implementation of the framework can be found on GitHub.

2.4.1 Experimental Setup

Our 3D reconstruction stage is implemented with PyTorch. Processing a one-minute video takes eight hours on a computer with 8 NVIDIA GeForce RTX 3080 GPUs. We run offline trajectory optimization and online MPC on a workstation computer equipped with an Intel i9-12900KS CPU and 64GB of memory. This workstation computer is connected to the robot via Ethernet. All hardware experiments are run

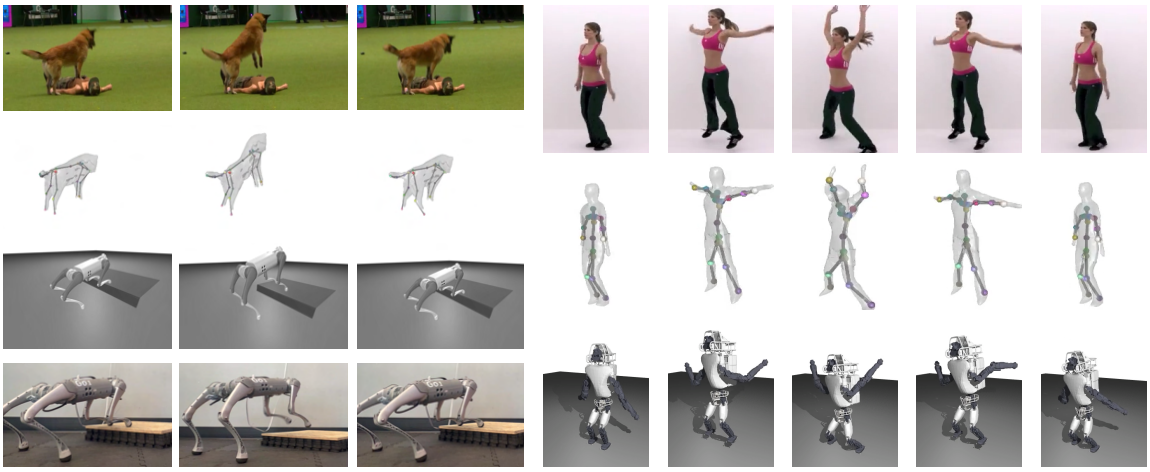

Figure 2.4: *Dog CPR* (left), *Human Jumping Jack* (right) motion imitation demonstrations. The top row shows frames from the original video, the second row shows the key-point trajectory, and the third row shows the dynamically feasible trajectory being executed by the Unitree Go1 and Atlas robots. The bottom row on the left shows *dog CPR* on hardware.

on a Unitree Go1 quadruped robot.

2.4.2 Quadruped

We verify our video-to-robot motion transfer approach on three video input examples: a dog reaching for water (*dog-reach*) — shown in Fig. 2.1 (left); a cat pacing across a living room (*cat pace*) — shown in Fig. 2.1 (second to left); and a dog performing CPR on a human (*dog CPR*) — shown in Fig. 2.1 (second to right).

Point-foot quadruped model: We use a simplified robot dynamics model for offline trajectory optimization and online control. This point-foot representation of the robot dynamics neglects the leg dynamics and instead models the feet as point masses. The model has 18 degrees of freedom and 12 control inputs. The controls are modeled as three-dimensional internal forces between each foot and the body. Note that this model is different from the skeleton model used during reconstruction, which contains additional links and joints (e.g. legs, tails, torso, etc.).

Hardware setup: For the hardware experiments, a hand-tuned PD controller running at 1000 Hz is used to track the forces and foot positions computed by the MPC policy on the Unitree Go1 robot. State estimation is also provided at 1000 Hz with a Kalman Filter that utilizes robot joint encoders and an onboard IMU. We

take time discretization of 0.05 s offline and track the reference with online MPC at 100 Hz with a prediction horizon of 0.15 s on the Unitree Go1 robot.

Dog reach: We take a video clip of a dog reaching for an automatic water feeder and compute an offline reference trajectory that is 5.0 s long. (Fig. 2.1 left).

Cat pace: We take a video clip of a cat pacing across a living room (Fig. 2.1 second to left) and compute an offline reference trajectory that is 3.0 s long, then repeated to form a continuous forward walking gait.

Dog CPR: We take a video clip of a dog performing CPR on a human (Fig. 2.4 left) and compute an offline reference trajectory that is 0.65 s long We model the terrain as a step where the front feet plan for a contact distance 0.1 m higher than the back feet. This motion primitive is fairly dynamic and would be difficult to design manually. Fig. 2.5 compares the output trajectories of each stage of SLoMo for each foot, where the key-point trajectory (blue) contains infeasible ground penetration, while the optimized reference (red) eliminates this unphysical artifact.

2.4.3 Humanoid

We show that our framework can also be applied to imitating human movements on humanoid robots in simulation on the Atlas robot.

Point-foot humanoid model: We use a simplified model to represent the dynamics of the humanoid robot. We model each foot as a rectangular prism with two contact points: one at the toe and the other at the heel. We similarly model each hand as a point mass. The model has 24 degrees of freedom and 18 control inputs. We use time discretization of 0.1 s for the experiments below.

Human Stretching: We take a video of a human raising both arms in a stretching motion (Fig. 2.1 second to left) and compute an offline reference trajectory that is 3.8 s long.

Human Jumping Jacks We take a video of a human performing a jumping jack exercise (Fig. 2.4 right) and compute an offline reference trajectory that is 1.5 s long.

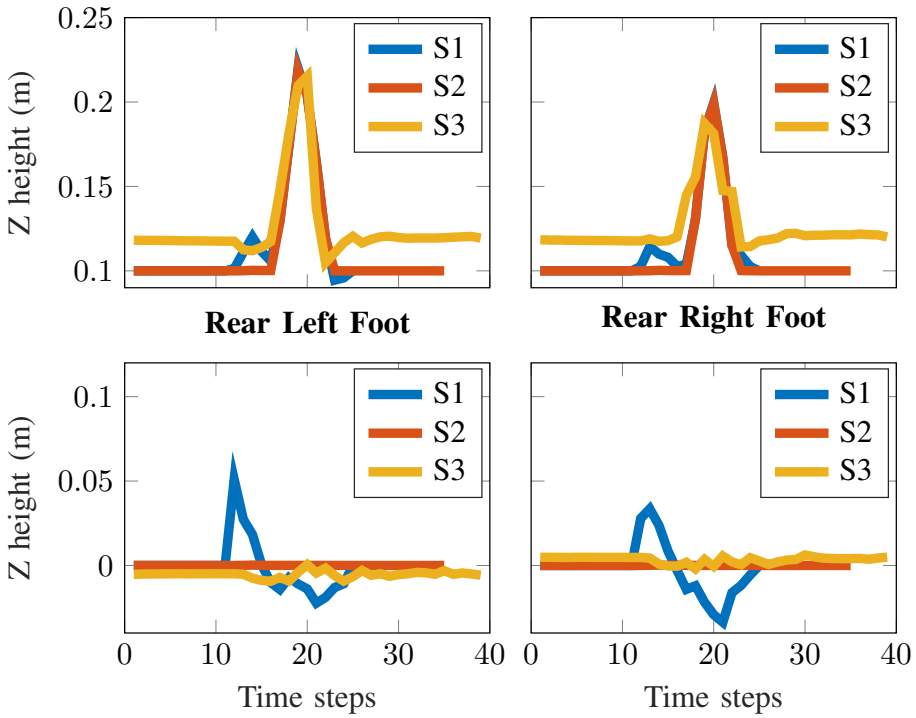

Figure 2.5: Foot height trajectories comparison for the dog-CPR experiment. Each plot corresponds to one foot of the robot. The blue lines are trajectories generated by the 3D reconstruction stage (S1). The red lines are the outputs of the trajectory optimization stage (S2). The yellow lines are the state feedback on robot hardware showing the result foot trajectories of the MPC policy (S3).

2.4.4 Comparisons to an RL Policy

We show that it is also possible to replace the trajectory-optimization and MPC steps of our method with an RL method [8] in simulation. We train RL policies for the Dog Reach and Cat Pace examples with 4 random seeds each. In Fig. 2.6, we showcase the highest-performing policy on the robot. Notably, while the best policy shows reasonable performance, we find that the RL policies exhibit a high degree of variance across different seeds in the Dog Reach example. This high variance makes fair, rigorous comparisons between model-based optimization and model-free RL difficult. However, we still believe imitating animal movements using RL is a promising approach and deserves further investigation.

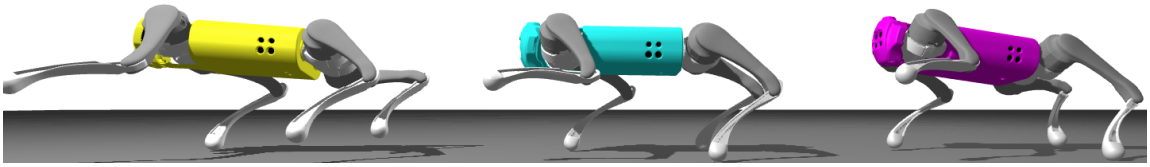

Figure 2.6: A comparison between the reconstructed key-point reference (yellow), optimized reference (teal, ours), and learned imitation policy [8] (purple).

2.5 Discussion and Conclusions

We present SLoMo, a first-of-its-kind framework for enabling legged robots to imitate animal and human motions captured from real-world casual monocular videos. Our research highlights that recent advancements in 3D reconstruction [86, 88] are effective at extracting physically plausible motion trajectories solely from monocular RGB videos. These trajectories are quite noisy and dynamically infeasible, even with the physics-based roll-out cost, making them unsafe to execute directly on real robots. To overcome this, an off-the-shelf trajectory optimizer [51] or RL method [8] is necessary to plan for dynamically feasible trajectories that track the retrieved motions from videos.

In this paper, we use a contact-implicit trajectory optimization method to reason about contact events and timing, and an MPC that is robot-agnostic and handles non-periodic contacts. This specific offline trajectory optimization and MPC combination facilitates motion transfer regardless of behavior periodicity, allowing us to generalize to arbitrary human and animal behaviors.

2.5.1 Limitations

SLoMo is a promising first step towards imitating human and animal behaviors on real-world robots from in-the-wild video footage. However, several limitations remain that should be addressed in future research: First, we make key model simplifications and assumptions in Sections 2.3.2 and 2.3.3 by using a point-foot model to represent the quadruped and humanoid dynamics. It should be possible to extend this work to use full-body dynamics in both offline and online optimization steps to fully leverage a robot's capabilities. For example, humans and animals are capable of making contact with the world in very rich ways (e.g. a dog can use its head to move an object), but

executing such behaviors for legged robots remains an open research problem. Second, the reconstruction step is computationally expensive, and we manually scale the reconstructed character to better match the kinematics of the target robot in Section 2.3.1. It should be possible to automate this scaling in Problem 2.6. Addressing morphological differences between video characters and corresponding robots in a principled, automated manner and accelerating reconstruction will be crucial for scaling our framework to large video datasets.

2.5.2 Future Work

Many exciting directions for future research remain: First, several trade-offs should be investigated in which components of our framework are swapped out. For example, leveraging RGB-D video data can likely improve reconstruction quality at the expense of data availability. Secondly, it should be possible to deploy the SLoMo pipeline on humanoid hardware, imitate more challenging humanoid behaviors, and execute behaviors on more challenging terrains where humans and animals have demonstrated highly athletic and robust behaviors but manually designing trajectories for robots can be challenging. Finally, this work can benefit from a proven, high-performance online control stack (e.g. whole-body control [36]). We believe that real-world visual data can be a rich source of robot behaviors, and taking advantage of recent advancements in reasoning about such visual data can be extremely powerful for robotics.

Chapter 3

GPU Accelerated Quadratic Programming Solver

This chapter introduces ReLU-QP, a GPU-accelerated solver for quadratic programs (QPs) capable of solving high-dimensional control problems at real-time rates. ReLU-QP is derived by exactly reformulating the Alternating Direction Method of Multipliers (ADMM) algorithm for solving QPs as a deep, weight-tied neural network with rectified linear unit (ReLU) activations. This reformulation enables the deployment of ReLU-QP on GPUs using standard machine-learning toolboxes. We evaluate the performance of ReLU-QP across three model-predictive control (MPC) benchmarks: stabilizing random linear dynamical systems with control limits, balancing an Atlas humanoid robot on a single foot, and performing a whole-body pick-up motion on a quadruped equipped with a six-degree-of-freedom arm. These benchmarks indicate that ReLU-QP is competitive with state-of-the-art CPU-based solvers for small-to-medium-scale problems and offers order-of-magnitude speed improvements for larger-scale problems.

3.1 Introduction

Convex quadratic programming is a core technology in many areas of robotics, including inverse kinematics [73], contact dynamics [68], actuator design [76], and model-predictive control (MPC) [43]. In many cases, problem instances with thousands

or tens of thousands of variables need to be solved in milliseconds to achieve real-time performance.

In this paper, we focus on MPC, which places particularly challenging demands on solvers by requiring large problems to be solved robustly and quickly. Model-predictive control (MPC) is a widely used control strategy that solves a receding-horizon optimization problem while reasoning about linear (or linearized) system dynamics and additional linear equality and inequality constraints. Successful applications of MPC include autonomous driving [18], rocket landing [4], and legged locomotion [15]. MPC problems frequently include hundreds to tens of thousands of variables, with solver time complexity scaling cubically with state and control dimensions and linearly with the time horizon.

To reduce solve times, it is common practice to use simplified point-mass [2, 4] or centroidal [10, 44] models that limit the controller's ability to reason about system dynamics and state and control constraints, such as joint and torque limits. When whole-body models are used in MPC controllers, the increased computational burden results in longer solve times and lower sample rates, requiring a faster low-level PD controller to compensate [56], and often producing more conservative and less robust performance.

Current methods for solving quadratic programs generally fall into three categories: interior-point methods, active-set methods, and the closely related augmented-Lagrangian method and alternating-direction method of multipliers (ADMM). Interior-point solvers like GUROBI [25], MOSEK [3], and IPOPT [84], offer robust convergence with a few iterations of Newton's method, but each iteration requires the expensive solution of a linear system. Interior-point methods are also difficult to warm start, making them less well suited to MPC applications where similar problem instances are encountered at each time step.

Active-set methods like QuadProg [23] and qpOASES [19] treat active inequality constraints as equalities and remove inactive ones from the problem. They are easy to warm start and can be very fast if the active set is guessed correctly, but their worst-case time complexity is combinatorial in the number of constraints [59].

Augmented-Lagrangian and ADMM methods, including ProxQP [7], OSQP [77], and ALTRO [29], have recently gained popularity in the robotics community for MPC. These solvers are easy to warm start and converge quickly to coarse tolerances,

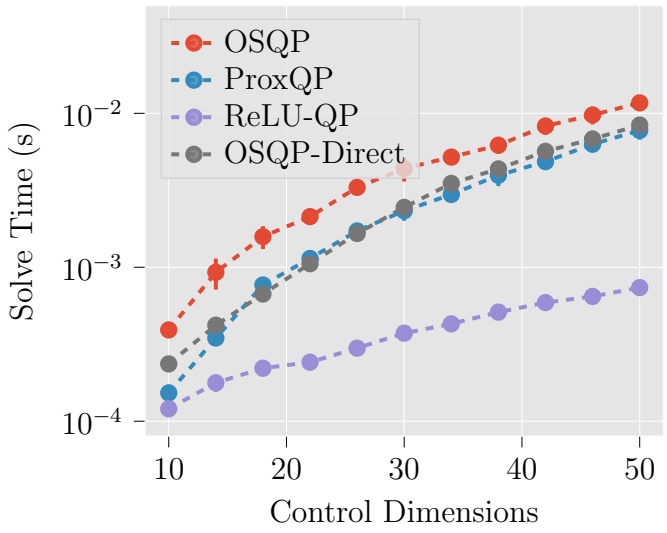

Figure 3.1: MPC solve times for random linear systems with control limits (horizon = 40, 3:1 state-to-control dimension ratio). ReLU-QP is an order of magnitude faster on high-dimensional control problems. The preconditioned condensed formulation is used except for *OSQP-Direct*, which solves the sparse direct formulation.

but generally have slow "tail convergence" when trying to achieve tight constraint tolerances.

While most existing MPC algorithms run on CPUs, there has been significant recent interest in exploiting the parallel computing capabilities of GPUs to speed up MPC. One such approach is model predictive path integral control (MPPI) [67, 83], which randomly samples control sequences and performs thousands of simulation rollouts in parallel to determine the next control action. While fast, MPPI can struggle to stabilize open-loop unstable systems and suffers from poor sample efficiency, making it difficult to scale to high-dimensional systems.

There have also been recent efforts to parallelize ADMM algorithms. A GPU version of OSQP uses an indirect conjugate-gradient method in place of a direct matrix factorization to exploit fast parallel matrix-vector products [71]. Another parallelized ADMM implementation pre-computes matrix inverses and relies on custom CUDA kernels for computing matrix-vector products on the GPU, but otherwise maintains the conventional sequential structure of ADMM [91].

In this work, we introduce ReLU-QP, a GPU-accelerated quadratic programming solver that achieves speed-ups of an order of magnitude over existing solvers on

large-scale problems. ReLU-QP achieves this by analytically mapping the sequential updates of the ADMM algorithm into the structure of a deep, weight-tied neural network with rectified linear unit (ReLU) activations. With this mapping, each iteration of the ADMM algorithm corresponds to a single layer of the network, comprising a dense matrix-vector product followed by a projection onto the positive orthant (clamping negative values to zero).

ReLU-QP can handle large MPC problem instances, enabling the use of detailed whole-body models subject to joint and torque limits, friction cones, and other linear constraints.

Despite a large number of constraints and high-dimensional state and input spaces in these problems, ReLU-QP can achieve kilohertz solution rates, allowing us to replace the standard hierarchical control stack — in which an MPC controller is cascaded with low-level PD controllers — with a single, unified, MPC controller that can more fully reason about a robot's dynamics for better performance, safety, and robustness. The simplicity of ReLU-QP also makes it easy to implement and deploy with standard machine-learning toolboxes such as PyTorch, TensorFlow, and JAX.

Our contributions are:

- An analytically exact transformation of the ADMM algorithm into a deep, weight-tied, neural network
- Open-source implementations of the ReLU-QP solver in PyTorch and Julia
- Experimental validation of ReLU-QP on several representative model-predictive control benchmarks with comparisons to state-of-the-art CPU-based solvers

Our paper is organized as follows: In Section 3.2, we review quadratic programs, the ADMM algorithm, and MPC. In Section 3.3, we derive the mapping between ADMM and a deep, weight-tied neural network and present the ReLU-QP solver. In Section 3.4, we discuss the details of our MPC implementation for use with ReLU-QP. In Section 3.5 we benchmark ReLU-QP against OSQP and ProxQP on dense QPs and three MPC applications: random linear problems, the Atlas humanoid balancing on one foot, and a quadruped equipped with a 6-DoF arm performing a whole-body pick-up motion. Finally, we summarize our results, discuss the limitations of our work, and outline future research directions in Section 3.6.

3.2 Background

3.2.1 Convex Quadratic Programs

Convex quadratic programs are defined as follows [60]:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \frac{1}{2} y^T H y + g^T y \\ \text{subject to} \quad & c \leq Gy \leq d, \end{aligned} \tag{3.1}$$

where $g \in \mathbb{R}^n$ and the symmetric positive-definite matrix $H \in \mathbb{R}^{n \times n}$ definite the cost function, $G \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^m$ and $d \in \mathbb{R}^m$ define the constraints, and n and m are the number of decision variables and constraints, respectively.

3.2.2 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) is an efficient approach for solving (3.1) that has been widely deployed in robotics and machine learning in recent years [77]. The ADMM algorithm introduces "splitting variables" $\bar{y}$ and z to transform (3.1) into

$$\begin{aligned} \underset{y, \bar{y}, z}{\text{minimize}} \quad & \frac{1}{2} \bar{y}^T H \bar{y} + g^T \bar{y} \\ \text{subject to} \quad & G\bar{y} = z, \\ & y = \bar{y}, \\ & c \leq z \leq d. \end{aligned} \tag{3.2}$$

The augmented Lagrangian for (3.2) is,

$$\begin{aligned} L_\rho(y, z, \lambda) = \frac{1}{2} y^T H y + g^T y + \frac{\sigma}{2} ||\bar{y} - y||^2 \\ + \mu^T (\bar{y} - y) + \frac{1}{2} (G\bar{y} - z)^T \rho (G\bar{y} - z) \\ + \lambda^T (G\bar{y} - z) + \mathbb{I}_{(c,d)}(z), \end{aligned} \tag{3.3}$$

where $\sigma \in \mathbb{R}$ is a penalty weight, ρ is a diagonal penalty matrix, μ and λ are Lagrange

multipliers, and $\mathbb{I}$ is the indicator function:

$$\mathbb{I}_{(c,d)}(z) = \begin{cases} 0 & c \leq z \leq d \\ \infty & otherwise \end{cases} \tag{3.4}$$

The diagonal elements in ρ are used to weight progress towards satisfying each constraint.

ADMM alternates minimizing over y and z by performing the following steps [77]:

$$\begin{aligned} \bar{y}^+ &= (H + \sigma I + G^T \rho G)^{-1}(-g + \sigma y - \mu \\ &\quad + G^T(\rho z - \lambda)), \end{aligned} \tag{3.5}$$

$$y^+ = \bar{y}^+ + \sigma^{-1}\mu, \tag{3.6}$$

$$z^+ = \Pi_{(c,d)}(G\bar{y}^+ + \rho^{-1}\lambda), \tag{3.7}$$

$$\mu^+ = \mu + \sigma(\bar{y}^+ - y^+), \tag{3.8}$$

$$\lambda^+ = \lambda + \rho(Gy^+ - z^+), \tag{3.9}$$

where $\Pi_{(c,d)}$ is a projection onto the box constraints $c \leq z \leq d$, (3.5) is usually computed through a direct matrix factorization or using an indirect method like conjugate gradient, and steps (3.6)-(3.9) are composed of simple addition, multiplication, and clamping operations. We note that, from the second iteration onwards, both equations (3.8) and (3.6) ensure that $\mu = 0$. Consequently, $\bar{y}$ and y are equivalent, making it possible to eliminate $\bar{y}$ and μ and simplify the overall algorithm, leaving us with y, z, and λ as the essential variables.

Convergence Criteria

A natural choice of stopping criterion of the ADMM algorithm is the ∞-norm of the primal and dual residuals of (3.2) [13]:

$$||Gy - z||_\infty \leq \epsilon_{prim}, \tag{3.10}$$

$$||Hy + g + G^T\lambda||_\infty \leq \epsilon_{dual}, \tag{3.11}$$

where $\epsilon_{prim} \geq 0$ and $\epsilon_{dual} \geq 0$ are user-specified accuracy tolerances.

Preconditioning

Numerical conditioning plays a key role in the convergence behavior of ADMM [77]. Heuristic preconditioners [21, 22] are often used to reduce condition numbers and increase convergence rate. Variants of Ruiz equilibration [70] are common in modern solvers [7, 77].

Adaptive Penalty

The most important parameter in the ADMM algorithm is the parameter ρ [77]. Empirically, convergence rates can be slow when a fixed penalty parameter is used [13]. A common heuristic that works well in practice is to adapt the penalty based on the ratio between the primal and dual residuals [26]. OSQP [77] implements a similar strategy that adjusts ρ based on the relative magnitudes of the residuals:

$$\rho^{+} = \rho \sqrt{\frac{||r_p||_\infty \max(||Hy||_\infty, ||G^T\lambda||_\infty, ||g||_\infty, 10^{-4})}{||r_d||_\infty \max(||Gy||_\infty, ||z||_\infty, 10^{-4})}}, \tag{3.12}$$

where r_p and r_d are the primal and dual residuals from (3.10) and (3.11). While adjusting the penalty term online is critical for fast convergence, each update requires a new matrix factorization, which is computationally expensive. Therefore, minimizing overall solution time requires a carefully balanced strategy.

Warm Starting

When solving many similar problem instances sequentially, a standard warm-starting strategy for ADMM is to initialize the solver with y, λ, and ρ from the previous solution and set $z = Gy$.

3.2.3 Model-Predictive Control

Linear MPC solves the receding-horizon optimal control problem:

$$\begin{aligned} \underset{x_{0:N},\, u_{0:N-1}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + \frac{1}{2} x_N^T Q_N x_N \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \quad \forall k \in 0, \ldots, N-1 \end{aligned} \tag{3.13}$$

where x_0 is the current state of the system, Q and R are stage cost weights on the states and controls, respectively, Q_N is the terminal state cost weight, A is the discrete state transition matrix, B is the discrete control Jacobian, and N is the MPC horizon length. This optimization problem can be put into the standard form of (3.1) with the following cost and constraint matrices:

$$y = \begin{bmatrix} u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_N \end{bmatrix}, \qquad H = \begin{bmatrix} R & 0 & 0 & \ldots & 0 \\ 0 & Q & 0 & \ldots & 0 \\ 0 & 0 & R & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & Q_N \end{bmatrix},$$

$$G = \begin{bmatrix} B & -I & 0 & 0 & \ldots & 0 \\ 0 & A & B & -I & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & -I \end{bmatrix}, \quad c = d = \begin{bmatrix} -Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where G, c, and d encode the dynamics constraints, current state, and potentially additional state and control constraints. In this form, it is also straightforward to add additional constraints on the state and control variables. In the absence of additional constraints, (3.13) is a Linear-Quadratic Regulator (LQR) problem and can be efficiently solved with a Riccati recursion [72].

We refer to (3.13) as the "direct" formulation of the MPC problem, as the decision variables include both states and controls. In practice, an MPC controller repeatedly re-solves this problem using the latest state information to produce a feedback policy.

3.3 The ReLU-QP Algorithm

In this section, we derive a reformulation of the ADMM algorithm as a deep, weight-tied neural network with ReLU activations. This reformulation involves only GPU-friendly operations such as matrix-vector products and projection onto the positive orthant (clamping negative values to zero).

3.3.1 Modified ADMM Iterations

We make two key modifications to the ADMM iteration in Section 3.2.2. First, we move the λ update before the y and z updates.

$$\lambda^+ = \lambda + \rho(Gy - z), \tag{3.14}$$

$$\begin{aligned} y^+ = (H + \sigma I + G^T \rho G)^{-1}(-g + \sigma y \\ + G^T(\rho z - \lambda^+)), \end{aligned} \tag{3.15}$$

$$z^+ = \Pi_{(c,d)}(Gy^+ + \rho^{-1}\lambda^+). \tag{3.16}$$

We then combine (3.14) and (3.15) into a single matrix equation, followed by the projection (3.16), where $\tilde{c}$ and $\tilde{d}$ are $(-\infty, +\infty)$ for y and λ:

$$\begin{bmatrix} y^+ \\ z^+ \\ \lambda^+ \end{bmatrix} = \Pi_{(\tilde{c},\tilde{d})} \left(W \begin{bmatrix} y \\ z \\ \lambda \end{bmatrix} + b \right), \tag{3.17}$$

where

$$W = \begin{bmatrix} D(\sigma I - G^T \rho G) & 2DG^T\rho & -DG^T \\ GD(\sigma I - G^T\rho G) + G & 2GDG^T\rho - I & -GDG^T + \rho^{-1} \\ \rho G & -\rho & I \end{bmatrix},$$

$$D = (H + \sigma I + G^T \rho G)^{-1}, \qquad b = \begin{bmatrix} -D \\ -GD \\ 0 \end{bmatrix} g.$$

Note that, despite some additional computation introduced by reordering the ADMM update steps, the matrix-vector product and projection operations in (3.17) can be easily parallelized row-wise, and can therefore be efficiently computed on a GPU. In addition, the reordering does not affect the convergence properties of the algorithm. The only effect is on the residuals since λ is now lagged by one iteration, which has negligible practical performance impact.

3.3.2 Algorithm Summary

The ReLU-QP algorithm includes both offline and online stages: The offline stage involves pre-computing a set of matrices W in (3.17) for a predefined set of penalty parameters. In the online stage, we initialize the primal and dual variables and then perform the modified ADMM iterations (3.17) until convergence criteria (3.10) and (3.11) are satisfied, as summarized in Algorithm 1.

3.3.3 Implementation details

Penalty Scaling Heuristics

By default, we define a list of penalty weights ρ_i sampled logarithmically from 10^{-3} to 10^3 and set the initial value to $\rho = 0.1$.

During the solve, we compute a nominal penalty parameter from (3.12) and use the closest value in the predefined list by choosing the corresponding weight W and bias b.

Algorithm 1 ReLU-QP

Given: ϵ, ρ_i, check_interval, max_iters
Offline:
Scale problem
Compute KKT inverse, W and b for each $\rho \in \rho_i$
Online:
Update b, $\tilde{c}$, and $\tilde{d}$ with initial conditions
$v = [y;\ z;\ \lambda]$
for i = 1:max_iters **do**
    $v \leftarrow \text{clamp}(Wv + b, \tilde{c}, \tilde{d})$
    if mod(i, check_interval) == 0 **then**
        W, b, $\rho \leftarrow$ rho_update(v)
        if residuals(v) $< \epsilon$ **then** break
        end if
    end if
end for
return y

Preconditioning

We use a modified Ruiz equilibration heuristic [70] for ill-conditioned problems. Similar to OSQP, [77], we scale both the constraints and cost to improve the convergence rate.

Convergence Checking

The conventional wisdom that ADMM iterations are computationally inexpensive relative to computing convergence criteria (3.10) and (3.11) still holds for our solver. We check the convergence criteria every 25 iterations by default.

Software Implementation

We implemented the ReLU-QP solver in two popular machine-learning libraries: Flux.jl [30] and PyTorch [62]. Both implementations demonstrate similar performance, especially on large problems where the solve time is dominated by low-level GPU operations on large matrices. Open-source implementations of ReLU-QP can be found at: https://github.com/RoboticExplorationLab/ReLUQP.jl.

3.4 Dense Model-Predictive Control

We solve a condensed form of the MPC problem (3.13) to take advantage of existing GPU-accelerated dense matrix operations from standard machine-learning libraries. A common technique for condensing MPC problems is to eliminate the states as variables and only optimize over the controls given an initial condition x_0. This idea is similar to single shooting methods where the solver only optimizes over dynamically feasible trajectories, resulting in fewer decision variables. However, for open-loop unstable systems, condensing the MPC problem in this way also introduces severe numerical ill-conditioning [79].

We find that the modified Ruiz equilibration is often insufficient for addressing ill-conditioning in these problems. Instead, we use an infinite-horizon LQR feedback controller as an effective preconditioner, expressing the controls as $u_k = -Kx_k + \Delta u_k$, where K is the LQR gain matrix and Δu_k are the new decision variables in the MPC problem. This modification has been shown to eliminate the ill-conditioning associated with condensed MPC problems and improve solver convergence [53, 69].

In summary, we define the following transformations: :

$$
\begin{aligned}
y &= S\tilde{y} + Mx_0, \qquad \bar{A} = A - BK, \\
\tilde{y} &= \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_N \end{bmatrix}, \qquad M = \begin{bmatrix} -K \\ \bar{A} \\ \vdots \\ -K\bar{A}^{N-1} \\ \bar{A}^N \end{bmatrix}, \\
S &= \begin{bmatrix} I & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \\ -K\bar{A}^{N-2}B & -K\bar{A}^{N-3}B & \dots & I \\ \bar{A}^{N-1}B & \bar{A}^{N-2}B & \dots & B \end{bmatrix}.
\end{aligned}
$$

The Hessian, gradient, and constraint matrices of the "direct" form are then modified

as follows:

$$\bar{H} = S^T H S, \qquad \bar{g} = S^T H M x_0, \qquad \bar{G} = GS,$$
$$\bar{c} = c - GMx_0, \qquad \bar{d} = d - GMx_0.$$

The rows of G corresponding to the dynamics constraints are eliminated in this formulation as they are included in the new cost Hessian.

3.5 Simulation Experiments

In this section, we present the results of benchmarking experiments demonstrating ReLU-QP on both random, dense QPs and simulated closed-loop MPC problems, including balancing the Atlas humanoid on one foot and performing a whole-body pick-up motion on a quadruped robot equipped with a six-degree-of-freedom arm.

All experiments were performed on a desktop equipped with an Intel i7-12700K CPU with 64 GB of memory and an NVIDIA GeForce RTX 3080 GPU with 10 GB of VRAM. For OSQP [77] and (dense) ProxQP [7], we report internal solver timings and use the same convergence tolerances across all solvers to ensure fair comparisons. Unless otherwise specified, we solve each problem three times and average the solution times. All benchmarks are run in Julia [42] with the Flux.jl [30] implementation of ReLU-QP.

3.5.1 Random Dense QPs

We first benchmark on random dense QPs with both equality and inequality constraints. We generated problems for ten different decision-variable dimensions (n) logarithmically spaced from 10 to 2000. The number of equality and inequality constraints in each problem are both $n/4$. For each problem size, we randomly generate ten dense QPs and solve each problem until the residuals are below 10^{-6}. Note that, except for tolerance settings, default parameters are used for all solvers. Timing results are shown in Fig. 3.2. Compared to OSQP [77] and ProxQP [7], our method provides competitive solve times on medium-sized problems ($n = 100$) and is up to 30 times faster on large ($n = 2000$) problems.

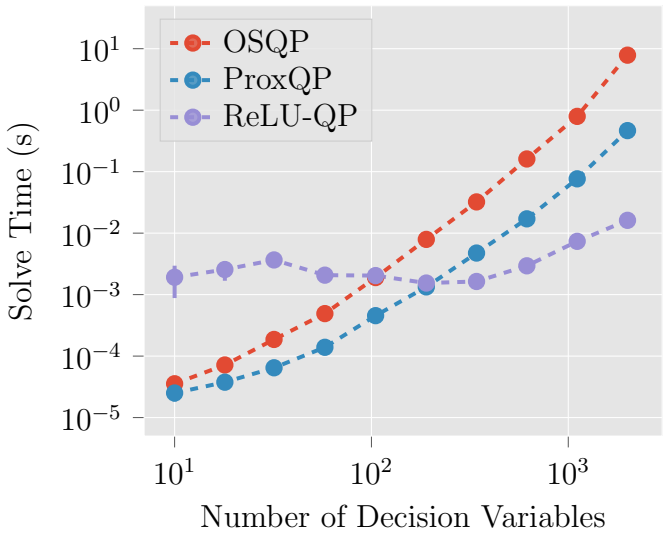

Figure 3.2: Solve times for random dense QPs with both equality and inequality constraints demonstrating that ReLU-QP is up to 30 times faster than OSQP and ProxQP on large problems.

3.5.2 Random Linear MPC Problems

We randomly generate controllable linear systems with control dimensions ranging from 10 to 50, where the state dimensions are always chosen to be triple the control dimensions, by sampling the A and B matrices in (3.13). We then use MPC to drive each system to the origin under control limits from a random initial state, making sure the initial states are large enough to activate the constraints for a significant portion of the trajectory. For OSQP and ReLU-QP, we perform one solver iteration per MPC step. For ProxQP, we solve to 10^{-2} tolerance, which normally corresponds to 1 outer iteration. All solvers are warm started with the primal and dual solutions from the previous MPC step. Solve times are shown in Fig. 3.1, where ReLU-QP is an order of magnitude faster on high-dimensional problems.

Note that all experiments in Fig. 3.1 solve the LQR-preconditioned dense MPC formulation, except *OSQP-Direct*, which solves the "direct" MPC formulation (3.13) using a sparse linear system solver (QDLDL) [77] that efficiently exploits matrix sparsity.

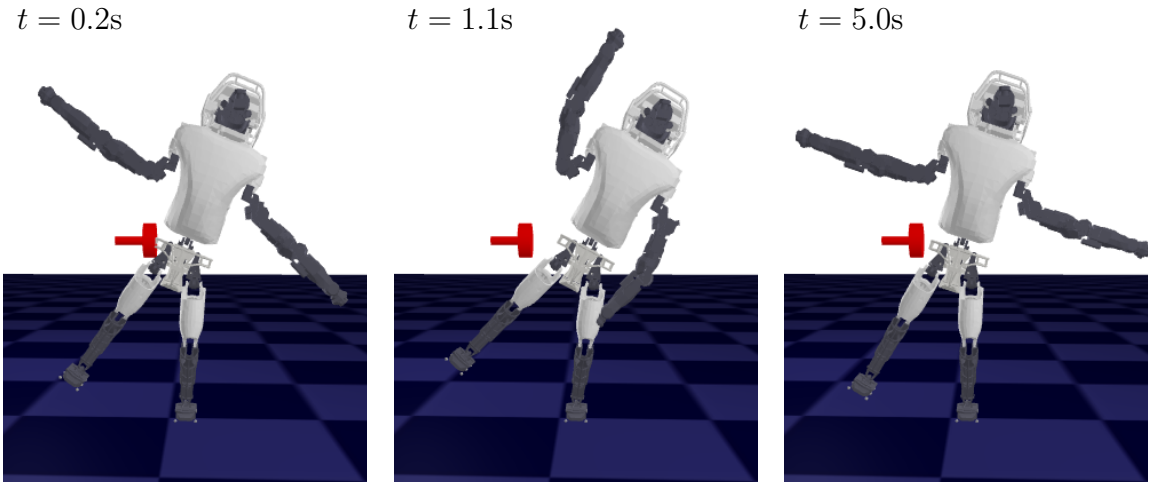

Figure 3.3: An Atlas humanoid robot balancing on one foot subject to control limits and disturbances. ReLU-QP successfully solves the task at up to 2600 Hz.

3.5.3 Atlas Balance

We use MPC to stabilize a full-body model of the Atlas humanoid robot balancing on one foot subject to initial velocity disturbances and control limits. The model has 58 states and 29 control inputs, and we linearize about a nominal reference pose. We benchmark our solver against OSQP and ProxQP using the preconditioned condensed formulation, with MPC horizons of 0.3s, 0.4s, and 0.5s. We warm start each solver by solving an initial problem instance to 10^{-6} tolerances. After the initial solve, we run OSQP and ReLU-QP for 2 iterations per MPC step and ProxQP to 10^{-4} tolerances – both of which were empirically found to be the coarsest and fastest settings that could successfully stabilize the robot.

Timing results for each horizon length are shown in Table 3.1. ReLU-QP demonstrates speed improvements of up to 20 times and can successfully stabilize the non-linear Atlas model at up to 2600 Hz, introducing the possibility of using whole-body MPC on high-dimensional robots without lower-level joint space or impedance controllers.

Table 3.1: Average MPC solve frequencies for a whole-body Atlas stabilizing task. ReLU-QP demonstrates 10-20 times speed ups.

MPC Horizon	OSQP	ProxQP	ReLU-QP	speed up
0.3 s	206 Hz	192 Hz	**2599** Hz	**10x**
0.4 s	104 Hz	153 Hz	**1925** Hz	**13x**
0.5 s	65 Hz	66 Hz	**1348** Hz	**20x**

3.5.4 Whole-Body Pick-up Motion on a Quadruped with an Arm

We used MPC to control a quadruped pick-up motion with a 6 DoF arm. The model has 52 states, 20 control inputs, and 12 foot-contact force variables, and was linearized around a standing pose. The problem is shown below, where u contains the controls and foot forces. C is the linearized pinned-foot constraint and $\mathcal{U}$ represents the feasible set for u which includes the pyramidal friction cone, normal force, and torque limit constraints, totaling 52 constraints per knot point in the dense formulation.

$$\begin{aligned} \underset{x_{0:N},\, u_{0:N-1}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + \frac{1}{2} x_N^T Q_N x_N \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \quad \forall k \in 0, \ldots, N-1, \\ & C x_k = 0 \qquad \forall k \in 1, \ldots, N, \\ & u_k \in \mathcal{U} \qquad \forall k \in 1, \ldots, N-1 \end{aligned} \tag{3.18}$$

We benchmarked ReLU-QP against ProxQP and OSQP, with ReLU-QP and ProxQP solving the preconditioned condensed formulation and OSQP solving the sparse direct formulation. Each solver was warm-started with ReLU-QP and OSQP running for 15 iterations per MPC step, which were empirically found to be the coarsest and fastest settings that could achieve the task. However, even when solving to a tolerance of 10^{-6} ProxQP was unable to stabilize the system. The results of the benchmarks are shown in Table 3.2. ReLU-QP was able to run over two times faster than OSQP with a frequency of 834 Hz, compared to 325 Hz for OSQP.

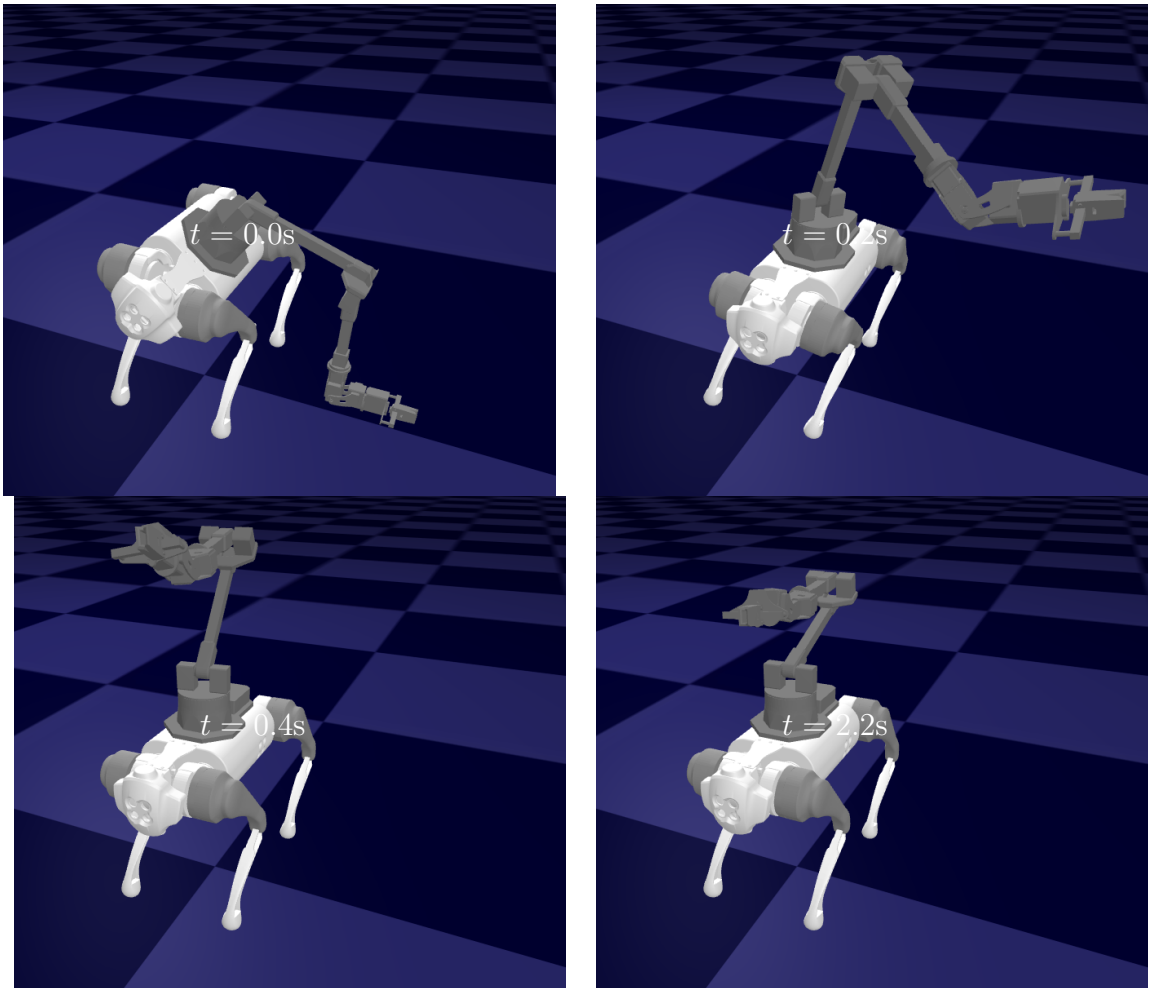

Figure 3.4: A Go1 robot equipped with a 6 DoF arm performing a pick-up motion using linear MPC. ReLU-QP solves the problem at 834 Hz.

Table 3.2: Average MPC solve frequencies for a whole-body quadruped model with an arm picking up an object from the ground. ReLU-QP demonstrates a 2× speed up over OSQP.

MPC Horizon	OSQP	ProxQP	ReLU-QP	speed up
0.3 s	325 Hz	– Hz	**834** Hz	**2.6x**

3.6 Discussion and Conclusions

This paper presents ReLU-QP, a GPU-accelerated quadratic programming solver capable of solving medium-to-large problem instances at real-time rates. Our solver excels at quickly solving large, dense QPs to coarse tolerances, and the LQR-preconditioned condensed MPC formulation enables high-performance model-predictive control on complex, high-dimensional systems.

3.6.1 Limitations

The primary limitation of our method is the need to pre-compute weight matrices offline, which prevents online updating of dynamics matrices or solving nonlinear problems with sequential quadratic programming. We also note that the speed-ups achieved on the quadruped example were less than on Atlas, which we attribute to the large number of constraints in the problem and ReLU-QP's current inability to leverage matrix sparsity, unlike OSQP.

3.6.2 Future Work

Several directions remain for future work: First, we plan to extend ReLU-QP to better exploit problem sparsity. Second, we will investigate indirect linear-system solvers like conjugate gradient to avoid pre-computing inverses offline and enable online updates and nonlinear SQP methods. Finally, we plan to investigate the possibility of initializing neural network control policies using ReLU-QP before performing additional policy tuning using reinforcement-learning techniques.

Chapter 4

Conclusion

In this thesis, we make two key contributions. In the first part, we introduced the first-ever framework for imitating human and animal motions from casual, everyday videos for legged robots. Our method enabled motion transfer from a human to a humanoid robot in simulation and from cat and dog motions to a quadruped robot on hardware. In the second part, we developed a quadratic programming solver using only GPU-friendly operations. This solver leads to an order-of-magnitude speed improvement across several benchmarks.

Many exciting future research directions remain. First, developing a reliable operation-space controller through contact remains an open problem for legged robots. Even in the teleoperation setting, the current pose alone is generally not sufficient for robots to effectively plan through contact. Better prediction of demonstrator indent and contact sequence can significantly improve teleoperation performance. Second, despite the success of human pose estimation from labeled MoCap datasets, reliable contact and pose estimation from vision alone is still difficult in the general case. We provided initial promising results using differentiable physics [88] to regularize the reconstruction problem, but much more work is necessary for both tooling and algorithms to fully take advantage of physics simulation. Finally, much has been discussed in this thesis about CPU versus GPU computation, but recent developments in shared-memory computing platforms may quickly render this conversation irrelevant. One can simply leverage the CPU for CPU-friendly routines and the GPU for easily parallelizable operations without any consideration for data-sharing overhead.

Designing optimization solvers to take advantage of this form of edge computation will be critical in the next several years.

Bibliography

[1] Ipopt: Documentation. URL https://coin-or.github.io/Ipopt/. 2.2.2

[2] Behcet Acikmese and Scott R. Ploen. Convex programming approach to powered descent guidance for mars landing. 30(5):1353–1366. ISSN 0731-5090. doi: 10.2514/1.27553. URL https://doi.org/10.2514/1.27553. Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.27553. 3.1

[3] MOSEK ApS. MOSEK Optimization Toolbox for MATLAB. URL https://docs.mosek.com/10.1/toolbox.pdf. 3.1

[4] Behçet Açıkmeşe, John M. Carson, and Lars Blackmore. Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem. *IEEE Transactions on Control Systems Technology*, 21(6):2104–2113, November 2013. ISSN 1558-0865. doi: 10.1109/TCST.2012.2237346. Conference Name: IEEE Transactions on Control Systems Technology. 3.1

[5] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. In *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-8-5. doi: 10.15607/RSS.2022.XVIII.026. URL http://www.roboticsproceedings.org/rss18/p026.pdf. 2.2.4

[6] Praneet C Bala, Benjamin R Eisenreich, Seng Bum Michael Yoo, Benjamin Y Hayden, Hyun Soo Park, and Jan Zimmermann. Openmonkeystudio: Automated markerless pose estimation in freely moving macaques. *BioRxiv*, pages 2020–01, 2020. 2.2.1

[7] Antoine Bambade, Sarah El-Kazdadi, Adrien Taylor, and Justin Carpentier. PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond. June 2022. URL https://inria.hal.science/hal-03683733. 3.1, 3.2.2, 3.5, 3.5.1

[8] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation.

ISBN 978-0-9923747-6-1. doi: 10.15607/RSS.2020.XVI.064. URL http://www.roboticsproceedings.org/rss16/p064.pdf. (document), 2.2.1, 2.2.3, 2.1, 2.2.4, 2.4.4, 2.6, 2.5

[9] Arun L. Bishop, John Z. Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. ReLU-QP: A GPU-accelerated quadratic programming solver for model-predictive control. In *IEEE International Conference on Robotics and Automation*, 2024. URL http://arxiv.org/abs/2311.18056. 1

[10] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae 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)*, pages 2245–2252. IEEE. ISBN 978-1-5386-8094-0. doi: 10.1109/IROS.2018.8593885. URL https://ieeexplore.ieee.org/document/8593885/. 2.2.2, 2.2.3, 3.1

[11] Boston Dynamics. More parkour atlas, . URL https://www.youtube.com/watch?v=_sBBaNYex3E. 2.2.2, 2.2.3

[12] Boston Dynamics. No time to dance | boston dynamics, . URL https://www.youtube.com/watch?v=6U-bI3On1Ww. 2.2.3, 2.2.4

[13] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, July 2011. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000016. URL https://www.nowpublishers.com/article/Details/MAL-016. Publisher: Now Publishers, Inc. 3.2.2, 3.2.2

[14] Simon Le Cleac'h, Taylor Howell, Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, Mac Schwager, and Zachary Manchester. Fast contact-implicit model-predictive control, 2021. URL https://arxiv.org/abs/2107.05616. 2.2.3, 2.3, 2.3.3, 2.3.3

[15] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. doi: 10.1109/IROS.2018.8594448. ISSN: 2153-0866. 3.1

[16] Luke Drnach and Ye Zhao. Robust trajectory optimization over uncertain terrain with stochastic complementarity. 6(2):1168–1175. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2021.3056064. URL https://ieeexplore.ieee.org/document/9343725/. 2.2.2

[17] Luke Drnach, John Z. Zhang, and Ye Zhao. Mediating between contact feasibility and robustness of trajectory optimization through chance complementarity con-

straints. 8. ISSN 2296-9144. URL https://www.frontiersin.org/articles/10.3389/frobt.2021.785925. 2.2.2

[18] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive Active Steering Control for Autonomous Vehicle Systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, May 2007. ISSN 1558-0865. doi: 10.1109/TCST.2007.894653. Conference Name: IEEE Transactions on Control Systems Technology. 3.1

[19] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, December 2014. ISSN 1867-2957. doi: 10.1007/s12532-014-0071-1. URL https://doi.org/10.1007/s12532-014-0071-1. 3.1

[20] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. page 16. 2.2.3

[21] Pontus Giselsson and Stephen Boyd. Metric selection in fast dual forward–backward splitting. *Automatica*, 62:1–10, December 2015. ISSN 0005-1098. doi: 10.1016/j.automatica.2015.09.010. URL https://www.sciencedirect.com/science/article/pii/S0005109815003611. 3.2.2

[22] Pontus Giselsson and Stephen Boyd. Linear Convergence and Metric Selection for Douglas-Rachford Splitting and ADMM. *IEEE Transactions on Automatic Control*, 62(2):532–544, February 2017. ISSN 1558-2523. doi: 10.1109/TAC.2016.2564160. Conference Name: IEEE Transactions on Automatic Control. 3.2.2

[23] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33, September 1983. ISSN 1436-4646. doi: 10.1007/BF02591962. URL https://doi.org/10.1007/BF02591962. 3.1

[24] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018. 2.2.1

[25] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com. 3.1

[26] B. S. He, H. Yang, and S. L. Wang. Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities. *Journal of Optimization Theory and Applications*, 106(2):337–356, August 2000. ISSN 1573-2878. doi: 10.1023/A:1004603514434. URL https://doi.org/10.1023/A:1004603514434. 3.2.2

[27] Taylor A. Howell, Simon Le Cleac'h, J. Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable physics engine for robotics, . URL http:

//arxiv.org/abs/2203.00806. 2.2.2, 2.3.2

[28] Taylor A. Howell, Simon Le Cleac'h, Kevin Tracy, and Zachary Manchester. CALIPSO: A differentiable solver for trajectory optimization with conic and complementarity constraints, . URL http://arxiv.org/abs/2205.09255. 2.2.2

[29] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679, November 2019. doi: 10.1109/IROS40897.2019.8967788. ISSN: 2153-0866. 3.1

[30] Mike Innes. Flux: Elegant machine learning with Julia. *Journal of Open Source Software*, 3(25):602, May 2018. ISSN 2475-9066. doi: 10.21105/joss.00602. URL https://joss.theoj.org/papers/10.21105/joss.00602. 3.3.3, 3.5

[31] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2.2.1

[32] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5614–5623, 2019. 2.2.1

[33] Dongho Kang, Simon Zimmermann, and Stelian Coros. Animal gaits on quadrupedal robots using motion matching and model-based control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8500–8507. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9635838. URL https://ieeexplore.ieee.org/document/9635838/. 2.2.1, 2.1, 2.2.4

[34] Dongho Kang, Flavio De Vincenti, Naomi C Adami, and Stelian Coros. Animal motions on legged robots using nonlinear model predictive control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11955–11962. IEEE, 2022. 2.2.1

[35] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 international conference on robotics and automation (ICRA)*, pages 6295–6301. IEEE, 2019. 2.2.2

[36] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control, September 2019. URL http://arxiv.org/abs/1909.06586. arXiv:1909.06586 [cs]. 2.2.3, 2.5.2

[37] Sunwoo Kim, Maks Sorokin, Jehee Lee, and Sehoon Ha. Human mo-

tion control of quadrupedal robots using deep reinforcement learning. In *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-8-5. doi: 10.15607/RSS.2022.XVIII.021. URL http://www.roboticsproceedings.org/rss18/p021.pdf. 2.2.4

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2.3.1

[39] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020. 2.3.1

[40] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5253–5263, 2020. 2.2.1

[41] Jonas Koenemann, Felix Burget, and Maren Bennewitz. Real-time imitation of human whole-body motions by humanoids. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2812. IEEE, 2014. 2.2.4

[42] Twan Koolen and Robin Deits. Julia for robotics: simulation and real-time control in a high-level programming language. May 2019. doi: 10.1109/ICRA.2019.8793875. 3.5

[43] Basil Kouvaritakis and Mark Cannon. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer International Publishing, Cham, 2016. ISBN 978-3-319-24851-6 978-3-319-24853-0. doi: 10.1007/978-3-319-24853-0. URL http://link.springer.com/10.1007/978-3-319-24853-0. 3.1

[44] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. 40(3):429–455. ISSN 1573-7527. doi: 10.1007/s10514-015-9479-3. URL https://doi.org/10.1007/s10514-015-9479-3. 3.1

[45] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016. 2.2.2

[46] Suryansh Kumar, Yuchao Dai, and Hongdong Li. Superpixel soup: Monocular dense 3d reconstruction of a complex dynamic scene. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1705–1717, 2019. 2.3.1

[47] Tianyu Li, Jungdam Won, Sehoon Ha, and Akshara Rai. FastMimic: Model-based

motion imitation for agile, diverse and generalizable quadrupedal locomotion. URL http://arxiv.org/abs/2109.13362. version: 1. 2.1, 2.2.4

[48] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 2.2.1

[49] Haimin Luo, Teng Xu, Yuheng Jiang, Chenglin Zhou, QIwei Qiu, Yingliang Zhang, Wei Yang, Lan Xu, and Jingyi Yu. Artemis: articulated neural pets with appearance and motion synthesis. *arXiv preprint arXiv:2202.05628*, 2022. 2.2.1

[50] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC). 2.2.2, 2.3.1

[51] Zachary Manchester, Neel Doshi, Robert J Wood, and Scott Kuindersma. Contact-implicit trajectory optimization using variational integrators. *The International Journal of Robotics Research*, 38(12-13):1463–1476, 2019. 2.2.2, 2.3.2, 2.3.2, 2.5

[52] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 2.2.2, 2.3.2

[53] Ian McInerney, Eric C. Kerrigan, and George A. Constantinides. Horizon-Independent Preconditioner Design for Linear Predictive Control. *IEEE Transactions on Automatic Control*, 68(1):580–587, January 2023. ISSN 1558-2523. doi: 10.1109/TAC.2022.3145657. Conference Name: IEEE Transactions on Automatic Control. 3.4

[54] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2.2.1, 2.3.1

[55] Shin'ichiro Nakaoka, Atsushi Nakazawa, Fumio Kanehiro, Kenji Kaneko, Mitsuharu Morisawa, Hirohisa Hirukawa, and Katsushi Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *The International Journal of Robotics Research*, 26(8): 829–844, 2007. 2.2.4

[56] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, July 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2800124. Conference Name: IEEE Robotics and Automation Letters. 3.1

[57] Chuong Nguyen, Lingfan Bao, and Quan Nguyen. Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control. URL http://arxiv.org/abs/2204.01147. 2.2.3

[58] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, pages 11453–11464, 2021. 2.3.1

[59] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2e edition, 2006. 3.1

[60] Jorge Nocedal and Stephen J. Wright. Quadratic Programming. In *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, pages 448–492. Springer, New York, NY, 2006. ISBN 978-0-387-40065-5. doi: 10.1007/978-0-387-40065-5_16. URL https://doi.org/10.1007/978-0-387-40065-5_16. 3.2.1

[61] Diego Pardo, Michael Neunert, Alexander Winkler, Ruben Grandia, and Jonas Buchli. Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion. In *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-3-0. doi: 10.15607/RSS.2017.XIII.042. URL http://www.roboticsproceedings.org/rss13/p42.pdf. 2.2.2, 2.3.2

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Pages: 8024–8035 Publication Title: Advances in Neural Information Processing Systems 32 original-date: 2016-08-13T05:26:41Z. 3.3.3

[63] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: example-guided deep reinforcement learning of physics-based character skills. 37(4):1–14. ISSN 0730-0301, 1557-7368. doi: 10.1145/3197517.3201311. URL https://dl.acm.org/doi/10.1145/3197517.3201311. 2.1, 2.2.4

[64] Nancy S Pollard, Jessica K Hodgins, Marcia J Riley, and Christopher G Atkeson. Adapting human motion for the control of a humanoid robot. In *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, volume 2, pages 1390–1397. IEEE, 2002. 2.2.4

[65] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory

optimization of rigid bodies through contact. 33(1):69–81. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364913506757. URL http://journals.sagepub.com/doi/10.1177/0278364913506757. 2.2.2, 2.3.2

[66] Matthew J. Powell, Eric A. Cousineau, and Aaron D. Ames. Model predictive control of underactuated bipedal robotic walking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5121–5126. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139912. URL http://ieeexplore.ieee.org/document/7139912/. 2.2.3

[67] Jintasit Pravitra, Kasey A. Ackerman, Chengyu Cao, Naira Hovakimyan, and Evangelos A. Theodorou. L1-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7661–7666, October 2020. doi: 10.1109/IROS45743.2020.9341154. ISSN: 2153-0866. 3.1

[68] Stéphane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Gauss' least constraints principle and rigid body simulations. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 517–522, Washington, DC, United States, 2002. doi: 10.1109/ROBOT.2002.1013411. URL https://hal.science/hal-01147672. 3.1

[69] J. A. Rossiter, B. Kouvaritakis, and M. J. Rice. A numerically robust state-space approach to stable-predictive control strategies. *Automatica*, 34(1):65–73, January 1998. ISSN 0005-1098. doi: 10.1016/S0005-1098(97)00171-4. URL https://www.sciencedirect.com/science/article/pii/S0005109897001714. 3.4

[70] Daniel Ruiz. A scaling algorithm to equilibrate both rows and columns norms in matrices. January 2001. 3.2.2, 3.3.3

[71] Michel Schubiger, Goran Banjac, and John Lygeros. GPU acceleration of ADMM for large-scale quadratic programming. *Journal of Parallel and Distributed Computing*, 2020. 3.1

[72] P.O.M. Scokaert and J.B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1169, August 1998. ISSN 1558-2523. doi: 10.1109/9.704994. Conference Name: IEEE Transactions on Automatic Control. 3.2.3

[73] Krishna Shankar, Joel W. Burdick, and Nicolas H. Hudson. A Quadratic Programming Approach to Quasi-Static Whole-Body Manipulation. In H. Levent Akin, Nancy M. Amato, Volkan Isler, and A. Frank van der Stappen, editors, *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, Springer Tracts in Advanced Robotics, pages 553–570. Springer International Publishing, Cham, 2015. ISBN 978-3-319-16595-0. doi: 10.1007/978-3-319-16595-0_32. URL

https://doi.org/10.1007/978-3-319-16595-0_32. 3.1

[74] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. URL http://arxiv.org/abs/2202.10448. 2.2.4

[75] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. URL http://arxiv.org/abs/2208.07860. 2.2.3

[76] Andrew Spielberg, Brandon Araki, Cynthia Sung, Russ Tedrake, and Daniela Rus. Functional co-optimization of articulated robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5042, May 2017. doi: 10.1109/ICRA.2017.7989587. 3.1

[77] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, December 2020. ISSN 1867-2957. doi: 10.1007/s12532-020-00179-2. URL https://doi.org/10.1007/s12532-020-00179-2. 3.1, 3.2.2, 3.2.2, 3.2.2, 3.2.2, 3.3.3, 3.5, 3.5.1, 3.5.2

[78] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. 39(15):2673–2691. ISSN 1097-0207. doi: 10.1002/(SICI)1097-0207(19960815)39:15⟨2673::AID-NME972⟩3.0.CO;2-I. 2.2.2

[79] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832), 2019. 3.4

[80] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 2.2.1

[81] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022. 2.2.1

[82] Carnegie Mellon University. Carnegie-mellon mocap database. http://mocap.cs.cmu.edu, 2024. Accessed: 2024-07-30. 2.2.1

[83] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, May 2017. doi: 10.1109/ICRA.2017.7989202. 3.1

[84] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, March 2006. ISSN 00255610. doi: 10.1007/s10107-004-0559-y. URL https://login.proxy.library.cmu.edu/login?qurl=https://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=19168876&site=eds-live&scope=site. Publisher: Springer Nature. 3.1

[85] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. *Advances in neural information processing systems*, 32, 2019. 2.2.1, 2.3.1

[86] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. BANMo: Building animatable 3d neural models from many casual videos. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2853–2863. IEEE. ISBN 978-1-66546-946-3. doi: 10.1109/CVPR52688.2022.00288. URL https://ieeexplore.ieee.org/document/9880012/. 2.2.1, 2.3.1, 2.3.1, 2.3.1, 2.3.1, 2.5

[87] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15980–15989, 2021. 2.2.1, 2.3.1

[88] Gengshan Yang, Shuo Yang, John Z. Zhang, Zachary Manchester, and Deva Ramanan. Physically plausible reconstruction from monocular videos. In *ICCV*, 2023. 2.3.1, 2.3.1, 2.5, 4

[89] Qingfeng Yao, Jilong Wang, Shuyu Yang, Cong Wang, Hongyin Zhang, Qifeng Zhang, and Donglin Wang. Imitation and adaptation based on consistency: A quadruped robot imitates animals from videos using deep reinforcement learning. URL http://arxiv.org/abs/2203.05973. 2.1, 2.2.4

[90] Jae Shin Yoon, Zhixuan Yu, Jaesik Park, and Hyun Soo Park. Humbi: A large multiview dataset of human body expressions and benchmark challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):623–640, 2021. 2.2.1

[91] Leiming Yu, Abraham Goldsmith, and Stefano Di Cairano. Efficient Convex Optimization on GPUs for Embedded Model Predictive Control. In *Proceedings of the General Purpose GPUs*, pages 12–21, Austin TX USA, February 2017. ACM. ISBN 978-1-4503-4915-4. doi: 10.1145/3038228.3038234. URL https://dl.acm.org/doi/10.1145/3038228.3038234. 3.1

[92] John Z. Zhang, Shuo Yang, Gengshan Yang, Arun L. Bishop, Swaminathan

Gurumurthy, Deva Ramanan, and Zachary Manchester. Slomo: A general system for legged robot motion imitation from casual videos. In *IEEE Robotics and Automation Letters*, 2023. URL https://ieeexplore.ieee.org/abstract/document/10246373. 1

[93] Ziyi Zhou, Bruce Wingo, Nathan Boyd, Seth Hutchinson, and Ye Zhao. Momentum-aware trajectory optimization and control for agile quadrupedal locomotion. URL http://arxiv.org/abs/2203.01548. 2.2.3