Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Robotic Systems Lab
Prof. Dr. Marco Hutter

Semester Thesis

MPC-feedback Trajectory Optimization for Wheeled-legged Robots

Spring Term 2020

Supervised by:
Marko Bjelonic
Ruben Grandia

Author:
Chao Ni

Declaration of Originality

I hereby declare that the written work I have submitted entitled

MPC-feedback Trajectory Optimization for Wheeled-legged Robots

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

Author(s)

Chao	Ni

Student supervisor(s)

Marko	Bjelonic
Ruben	Grandia

Supervising lecturer

Marco	Hutter

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

The above written work may be tested electronically for plagiarism.

Zurich, 6.12.2020
Place and date

Chao Ni
Signature

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

Intellectual Property Agreement

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

1. Intellectual Property Rights

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

2. Settlement of Disagreements

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

Zurich, 6.12.2020	[illegible]
Place and date	Signature

Contents

Abstract

Wheeled-legged robots can cope with tough terrains in an energy-efficient way with the help of wheels, while it also preserves its ability to negotiate complex terrains through the presence of the leg. To achieve various tasks in different terrains, trajectory optimization (TO) is required to serve as guidance. Furthermore, a model predictive control (MPC) feedback planner is utilized to track this reference trajectory.

In this thesis, we model the robot as a single rigid body, and the customized TOWR is used to handle the wheeled-legged robot trajectory optimization problem. The tracking reference is converted into joint space via inverse kinematics and be fed into MPC as guidance. We achieve this by adding the trajectory to the cost term of the MPC. A whole-body controller is used to generate torque commands for the real robots. The framework is verified on simulation and hardware. We show that such a framework can be modularized and TOWR can be replaced by other optimizers. MPC can help correct the infeasibility of the trajectory to make it physically practicable, as well as smooth the trajectory to avoid wild motions. Such a trajectory-correction scheme can be further explored to realize online trajectory-computing and tracking.

Acronyms and Abbreviations

MPC	model predictive control
WBC	whole body controller
TO	trajectory optimization
OCP	optimal control problem
NLP	nonlinear programming
TOWR	trajectory optimization for walking robots
SCP	simulation control playground

Chapter 1

Introduction

1.1 Motivation

Legged robots, such as ANYmal [1] have recently shown great ability to cope with challenging terrains, such as forest exploration, slope, stairs, and also provides capability in extreme environments. While it allows the robot to exhibit complex motions, it is inefficient in speed and also energy-consuming. Wheeled-legged robots, on the other hand, can cope with tough terrains at a higher speed with the help of wheels, while at the same time, preserve its ability to negotiate complex terrains through the presence of the leg. To explore these two benefits requires the robot to accurately generate the trajectory and in the meantime, track the trajectory with online algorithms. What's more, in the real world, the robot also has to cope with uncertainties from the environment, especially from the estimation error of the terrain. If the robot ignores the error and goes all the way along the generated trajectory, it may end up with unluckiness due to some violations of constraints such as dynamic constraints. To this end, model predictive control (MPC) has recently appeared on stage. It takes the perturbed state into consideration and looks ahead for a long horizon to smooth the trajectory. In this thesis, we will explore the possibility of bridging trajectory optimization and MPC-feedback control, and apply it on a customized ANYmal equipped with wheels.

1.2 Related Work

Equipping a quadrupedal robot with a wheel is not a new approach. A steerable robot that can conduct hybrid locomotion is developed recently [2]. However, its leg can only move in the sagittal plane and lacks the flexibility of a legged robot. Multiple levels of abstraction of trajectory planning for driving-stepping locomotion is also proposed [3] but is still restricted to a single gait motion. A customized ANYmal robot with non-steerable wheels [4][5] is also developed and able to conduct highly dynamic and complex motions.

Trajectory optimization for the wheeled-legged robots essentially shares a lot in common with legged robots. They differ in the way the constraints are enforced at the wheels when they are in contact with the ground. To simplify the real robot case, some assumptions are usually required to reduce the computation complexity. Two techniques are often used in solving the trajectory optimization: 1. decompose the problem into some subproblems which are computationally easier to solve; 2. using heuristics to obtain part of the solution. An example using both techniques is the policy-regularized model predictive control [6] for the quadrupedal robot *MIT Cheetah* [7]. In this paper, the authors introduce heuristics to guide the optimization, while penalizing the deviation from it. [8] provides another approach by optimizing over swing times along the trajectory. Heuristics comes in the form of an initial guess of trajectory and an estimated time steps required to reach the target.

The gait of the robot is usually hard to design and requires hand-tuning. This problem can also be translated into finding a sequence of contact and non-contact phase for each foot (wheel). Mix-integer optimization can solve the foot-placement problems and there are several approaches that apply this algorithm on the real robots [9], [10]. In practice, such problems decompose the motion of the robot into a foot-placement optimization task and a torso trajectory computation. [8] optimizes the gait and torso trajectory simultaneously, but is prone to local optimum and therefore, heavily depend on a good initialization. *SkaterBots* [11] shows a similar approach that couples the torso trajectory and gait generation into the same framework.

While inheriting from the trajectory optimization for legged robots, research on TO for wheeled-legged robots is also active, and due to its novelty, the planning method for a highly dynamic, complex hybrid motion is limited. A whole-body motion controller equipping a non-steerable robot [5] and a whole-body walking excavator [12] are proposed based on the optimization framework verified on a proven phase-dependent optimization method [8].

Within robotics, there is an increasing interest in MPC. Modern approaches normally provide model predictive control by solving a nonlinear optimization problem and a control sequence over the receding horizon can be determined. It has been proven efficient in many applications and areas, such as autonomous racing [13] and legged locomotion [14][15][16]. MPC coupled with the whole-body controller has been recently explored in [17]. This paper utilizes MPC to guide the torque behavior of a low-level controller and shows the possibility of bridging low update-rate MPC and high-frequency torque commands.

1.3 Contributions

In this thesis, we bridge the trajectory optimizer and the online MPC-feedback controller. The controller connects to a whole body controller (WBC) and will control the wheeled-legged robot. We utilize the MPC framework proposed in [17] and TO adapted from [8]. We verify the framework by implementing on the real robot and address the importance of MPC in stabilizing the robot as well as smoothing the trajectory. We also generate the trajectory for tough terrains and propose the possibility of having a relaxed trajectory in order to make the whole process online.

1.4 Thesis Outline

In Chapter 2 we will formulate the trajectory optimization for the wheeled-legged robot and introduce how to couple this with an MPC framework. In Chapter 3 we will give some implementation details and analysis of the results. We will conclude the thesis in Chapter 4.

Chapter 2

Formulation

2.1 Framework Overview

In this section, the framework that incorporates TO and MPC is provided. We will generate trajectories offline and use MPC to track these trajectories. Firstly a human user will command a message to a smart initialization module of TO. Based on the terrain information and the target position as well as an estimation of the trajectory horizon, the smart initialization module will initialize the trajectory in a proper way. This is due to that TOWR, which will be used as our main trajectory optimizer in this project, utilizes an NLP solver and can find a local optimum efficiently. The trajectory generated will be translated into joint space and fed into the MPC controller together with current measured robot states. MPC runs at a frequency of 20 Hz and will compute the desired control plan for a high-frequency low-level controller. The final torque is generated by drivers and given to the real robot. The detailed overview is shown in Fig. 2.1. The trajectory optimization part is computed offline and MPC computing is completely online. We will also explore the possibility of making the whole process online at the end of the report.

Besides, as depicted by a brown dash-box, the trajectory optimizer can be replaced by any other optimizer. For example, Simulation control playground (SCP) [11] has been validated on this framework as well.

2.2 Trajectory optimization

This section formulates the TO problem for wheeled-legged robots and reformulates it as an NLP problem. Normally an NLP is usually computing demanding considering our dynamic is highly nonlinear and with high dimension. We address this issue by coping with a feasibility problem instead of minimizing a cost in the usual sense. The goal of our motion optimizer is therefore to solve the Optimal Control Problem (OCP) as follows:

$$\begin{aligned} &\text{find} && \boldsymbol{x}(t), \dot{\boldsymbol{x}}(t) && (2.1)\\ &\text{subject to} && \boldsymbol{x}(0) = \boldsymbol{x}_0, \;\; \boldsymbol{x}(T) = \boldsymbol{x}_f, && (2.2)\\ & && \boldsymbol{h}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \ddot{\boldsymbol{x}}(t)) \geq \boldsymbol{0}, && (2.3)\\ & && \boldsymbol{g}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \ddot{\boldsymbol{x}}(t)) = \boldsymbol{0}, && (2.4) \end{aligned}$$

where $\boldsymbol{x}(t)$ is the set of decision variables, and usually given by the position and orientation of the CoM, wheels contact points' positions, and contact forces. User will input the initial state $\boldsymbol{x}_0$, final state $\boldsymbol{x}_f$ and the time duration T. In our setting,

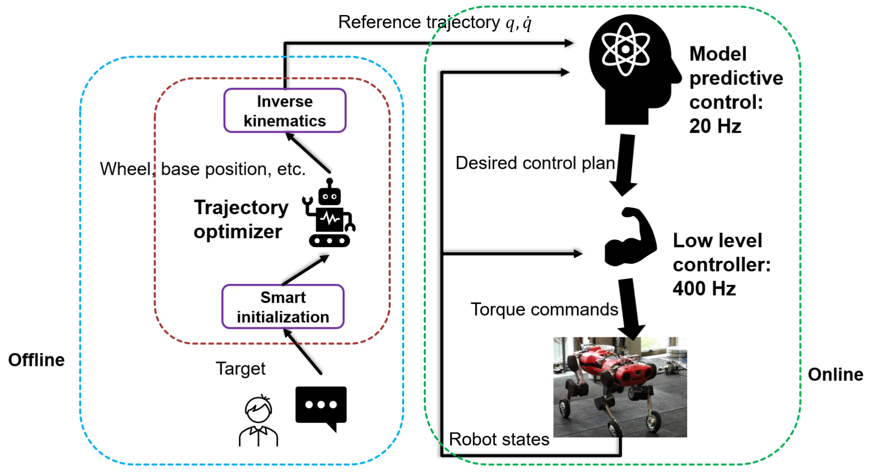

Figure 2.1: Overview of the MPC-feedback trajectory optimization framework. The human operator will command a message first to the TO, with smart initialization, TO will generate a feasible trajectory that will be fed into the MPC controller after passing through inverse kinematics. MPC uses this joint space trajectory as well as the measured robot state to compute the desired control plan. It will be fed into a low-level controller and a torque will then be given to the real robot.

we actually have a desired average speed towards the destination and hence the time duration should adjust to it together with the target position after driving over the terrain.

2.2.1 Direct Collection

We transcribe OCP into a Nonlinear Programming (NLP) by employing Direct Collection Shooting, where the continuous time-variant state is parameterized by a cubic polynomial and defined within a variable time duration ΔT. Therefore, the total time duration can be split into $T = [\Delta T_1, \Delta T_2, ...\Delta T_N]$ and $\sum_{i=1}^{N} \Delta T_i = T$. On each duration, a cubic can fully describe the state with Hermite parametrization.

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \tag{2.5}$$

where $a_i = f(x_k, \dot{x}_k, x_{k+1}, \dot{x}_{k+1}, \Delta T_i), \ \ \forall\, k \in [0, N-1]$. The equivalence between the state and the cubic polynomial allows us to optimize the state of the robot directly instead of the polynomial coefficients. This gives us the convenience to formulate the optimization and also helps simplify the implementation of NLP. The reason why we use the direct collection method is that we can hence apply nonlinear programming methods and it allows us to formulate and utilize the NLP solver much efficiently with the broad solver options available.

2.2.2 NLP formulation

In this section, we will illustrate the NLP problem in detail and explore its components. The formulation is shown as following equations. The right superscript denotes the coordinate component of the vector and the left superscript denotes the reference frame: $\mathcal{I}$ indicates the inertial frame, $\mathcal{B}$ indicates the base frame, $\mathcal{W}_i$ indicates the i^{th} fixed wheel frame, and $\mathcal{C}_i$ denotes the wheel contact frame, originate on the contact point on the ground. The formulation differs from the pointed end effector [8] in that the velocity of the end effector is no longer zero due to the existence of the wheel; instead, we include a rolling constraint to ensure consistency with the locomotion.

$$
\begin{array}{ll}
\text{find } {}^{I}\boldsymbol{r}(t) \in \mathbb{R}^3 & \text{(CoM linear position)} \\
\quad {}^{I}\boldsymbol{\theta}(t) \in \mathbb{R}^3 & \text{(base euler angles)} \\
\quad \text{for every wheel } \boldsymbol{i}: & \\
\quad \Delta T_{i,1}, ..., \Delta T_{i,N} \in \mathbb{R} & \text{(phase durations)} \\
\quad\quad {}^{I}\boldsymbol{p}_i(t) \in \mathbb{R}^3 & \text{(wheels' motions)} \\
\quad\quad {}^{I}\boldsymbol{f}_i(t) \in \mathbb{R}^3 & \text{(wheels' forces)} \\
\text{s.t. } [{}^{I}\boldsymbol{r}, {}^{I}\boldsymbol{\theta}](0) = [{}^{I}\boldsymbol{r}_0 \ , {}^{I}\boldsymbol{\theta}_0 \ ] & \text{(initial state)} \\
\quad [{}^{I}\boldsymbol{r}, {}^{I}\boldsymbol{\theta}](T) = [{}^{I}\boldsymbol{r}_g \ , {}^{I}\boldsymbol{\theta}_g \ ] & \text{(goal state)} \\
\quad \boldsymbol{F}_d({}^{I}\boldsymbol{r}, {}^{I}\boldsymbol{\theta}, {}^{I}\boldsymbol{p}_i \ , {}^{I}\boldsymbol{f}_i \ ) = \mathbf{0} & \text{(dynamic model)} \\
\quad \text{for every wheel } \boldsymbol{i}: & \\
\quad\quad {}^{I}\boldsymbol{p}_i \ \in \mathcal{R}_i({}^{I}\boldsymbol{r}(t), {}^{I}\boldsymbol{\theta}(t)) & \text{(kinematic model)} \\
\quad\quad \text{if wheel in rolling:} & \\
\quad\quad\quad {}^{I}\boldsymbol{p}_i^z \ = h_{terrain}({}^{I}\boldsymbol{p}_i^{x,y}(t)) & \text{(terrain height)} \\
\quad\quad\quad {}^{C_i}\boldsymbol{f}_i^z(t) \geq 0 & \text{(normal force)} \\
\quad\quad\quad {}^{C_i}\boldsymbol{f}_i^x(t) \leq f_{max} & \text{(maximum torque)} \\
\quad\quad\quad {}^{C_i}\dot{\boldsymbol{p}}_i{}^y(t) = 0 & \text{(rolling constraint)} \\
\quad\quad\quad {}^{I}\boldsymbol{f}_i^z(t) \in \mathcal{F}(\mu, \boldsymbol{n}, {}^{I}\boldsymbol{p}_i^{x,y}(t)) & \text{(friction cone)} \\
\quad\quad \text{if wheel in step:} & \\
\quad\quad\quad {}^{I}\boldsymbol{f}_i(t) = \mathbf{0} & \text{(no force in air)} \\
\quad\quad \displaystyle\sum_{j=1}^{N} \Delta T_{i,j} = T & \text{(total duration)}
\end{array}
$$

Dynamic and kinematic constraints

The robot is approximated by a single rigid-body whose mass is centered at the torso. We assume the mass of the legs is negligible compared to the base, which simplifies the dynamics of the system. This assumption is reasonable in that most robots have all the hardware components, such as the battery, the sensors, and the embedded electronics located at the base.
Based on this simplified model, the robot's dynamic equations are

$$m {}^{I}\ddot{\boldsymbol{r}}(t) = \sum_{i=1}^{4} {}^{I}\boldsymbol{f}_i(t) - m {}^{I}\boldsymbol{g}, \tag{2.6}$$

$$\mathbf{I} {}^{I}\dot{\omega}(t) + {}^{I}\omega(t) \times \mathbf{I} {}^{I}\omega(t) = \sum_{i=1}^{4} {}^{I}\boldsymbol{f}_i(t) \times ({}^{I}\boldsymbol{r}(t) - {}^{I}\boldsymbol{p}_i(t)), \tag{2.7}$$

where ${}^{I}\omega(t)$ denotes the angular velocity of the base in the inertial frame, m denotes the mass of the robot. Noticed that here the orientation is described by Euler angle, however, there are also other methods available such as quaternions and the transformation detail can be found in any robotics book.

Contact constraints

- **Terrain constraints**
  To make the robot move efficiently, we will assume the most of the time the robot would be in the driving phase, which means we need to impose additional physical constraints on the wheel's motion and contact forces. Firstly, all wheels have to be in contact with the terrain, therefore satisfying the constrains:
  $$ {}^{I}p_i^z(t) = h_{terrain}({}^{I}\boldsymbol{p}_i^{x,y}(t)), \tag{2.8}$$
  where $h_{terrain}$ contains the terrain information and is a function of the ground plane coordinate.

- **Non-negative constraint** Being in contact with the ground also implicitly adds the non-negative constraints for the normal contact forces:
  $$ {}^{I}\boldsymbol{f}_{n_i}(t) = {}^{C_i}\boldsymbol{f}_i^z(t) \geq 0. \tag{2.9}$$

- **Non-slippery constraint** The non-slippery constraint remains the same as the legged robot case, we approximate the Coulomb friction cone by a friction pyramid, which linearizes the constraint and speed up the computation.
  $$ -\mu^{I}\boldsymbol{f}_{n_i}(t) \leq {}^{C_i}\boldsymbol{f}_i^x(t) \leq \mu^{I}\boldsymbol{f}_{n_i}(t), \tag{2.10}$$
  $$ \mu^{I}\boldsymbol{f}_{n_i}(t) \leq {}^{C_i}\boldsymbol{f}_i^y(t) \leq \mu^{I}\boldsymbol{f}_{n_i}(t). \tag{2.11}$$

- **Rolling constraint** In the driving phase, due to that our robot's wheel is non-steerable, we also need to enforce a rolling constraint:
  $$ {}^{C_i}\dot{\boldsymbol{p}}^y(t) = 0. \tag{2.12}$$

2.2.3 Initialization

Due to the non-linearity of our formulation, it is important to have a proper initialization. Some general points need to be taken into consideration.

Gait generation

Gait can be chosen by human command and is realized by initialing the robot state to have a corresponding phase duration. Fig. 2.2 shows an initialization of pace motion. It is a hybrid motion that the robot can simultaneously drive and step. The generated solution may not be identical because the phase duration is also an optimized variable and will be changed. However, the change is slight and the pace gait is preserved because NLP will find the solution around the initialization point.

Discontinuous terrains

For discontinuous terrains, the problem arises that the gradient of the constraints is theoretically not well defined in the discontinuous point. Typically we set it as zero and it doesn't make much difference as a finite accountable number of points won't be selected in the discretized case. Therefore the terrain constraints have to be satisfied in the initialization. One way to handle this issue is to use a steep slope to approximate the stair.

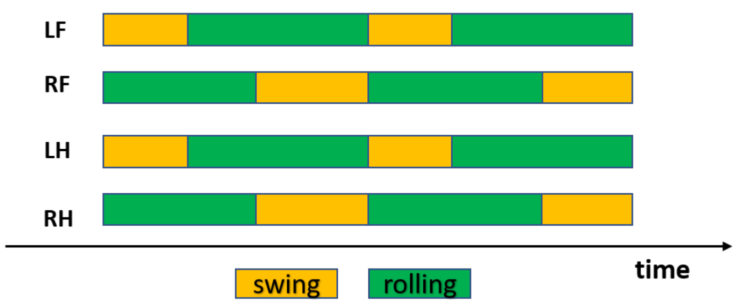

Figure 2.2: An initialization example of pace gait. LF (left front leg) and LH have identical phase duration and RF and RH have identical phase duration. There is an overlap in the rolling phase which means the robot is completely driving while in other cases it is conducting a hybrid motion.

2.2.4 Inverse kinematics

In TOWR, we consider a simplified single rigid body and ignore the joints. However, for MPC which will be discussed in the next section, it requires base positions, velocities, and joint positions as input, we then need to pass the trajectory into an inverse kinematics solver to generate the trajectory in the generalized joint space.

2.3 MPC-feedback Control

To track the generated trajectory despite the disturbance, both from parameter space and the real world noise, MPC is used in this project. MPC will further generate a control plan to be fed into low level controller and generate the torques required to control the real robot.

2.3.1 General MPC formulation

The general MPC is to find the optimal control sequence $u(\cdot)$ to minimize the cost along the a time period.

$$\min_{u(\cdot)} \Phi(\boldsymbol{x}(T)) + \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt, \tag{2.13}$$

where $\boldsymbol{x}(t)$ is the state and $\boldsymbol{u}(t)$ is the input at time instance t. $L(\cdot)$ is a time-varying cost function, which in many cases is approximated by a quadratic cost function. The goal is to find the optimal control that minimizes this cost subject to the following system dynamics, initial condition, and some equality and inequality constraints:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2.14}$$
$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{2.15}$$
$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) = 0 \tag{2.16}$$
$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t) \geq 0. \tag{2.17}$$

2.3.2 Feedback MPC

The above problem is an open-loop prediction and has no feedback inputs. We add the feedback gain to compensate the deviation from the optimal state trajectory, and use the policy $\boldsymbol{\pi}(\boldsymbol{x}, t)$ as input to the WBC.

$$\boldsymbol{\pi}(\boldsymbol{x}, t) = \boldsymbol{u}^*(t) + \boldsymbol{K}(t)(\boldsymbol{x}(t) - \boldsymbol{x}^*(t)), \tag{2.18}$$

where $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$ are state trajectory and optimal control respectively. The techniques used in choosing a proper feedback gain $\boldsymbol{K}$ is illustrated in detail in [17]

Chapter 3

Results

In this chapter, we will in detail describe how we conduct the experiment and analyze the results.

3.1 Implementation

3.1.1 Inverse kinematics

We use an iterative method to compute the joint angles, The pseudo-code is illustrated as follows. We use the line search method to find the maximum step size at each iteration.

Algorithm 1: Iterative inverse kinematics

```
Result: Joint angles $q^*$
initialization $q = q_{default}$;
while Not Converged do
    1. Compute Jacobi matrix;
    2. Line Search to find optimal step size;
    3. Update $q$ using gradient method;
end
```

The complete code can be found at the Bitbucket repo.

3.1.2 Initialization

Swing phase

One important point when initializing the wheel positions is to make sure the swing phase is a proper value to be fed into MPC as MPC explicitly set the swing height, and therefore a too large or small swing phase would cause a problem in interpolating the swing phase. An example of trotting motion is depicted in Fig. 3.1.

Discontinuous stair

As illustrated in the second chapter, when it comes to a discontinuous terrain, the initialization is crucial to get a feasible solution. A general approach of walking over a stair for a quadrupedal robot would be stepping over one leg at a time. In order to make sure that all terrain constraints are satisfied in initialization, we tune the speed of the robot to $1.4m/s$ and set the initial phase duration as $\{0.48, 0.3, 1.62\}$, $\{0.52, 0.3, 1.58\}$, $\{1.68, 0.3, 0.42\}$ and $\{1.72, 0.3, 0.38\}$ for LF, RF, LH and RH leg respectively. An illustration is described in Fig. 3.2.

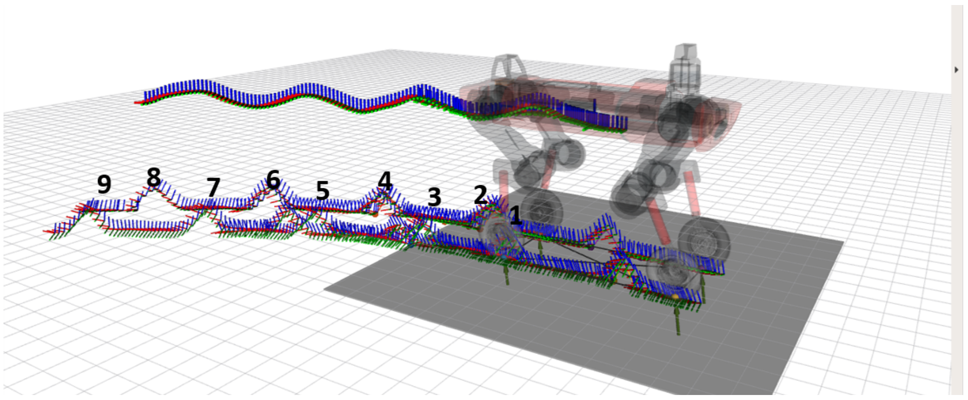

Figure 3.1: An example of phase duration: trot motion. This motion has a total duration of 2.4 seconds and has 9 separate phases for the LF leg. The swing phase is about 0.33 seconds.

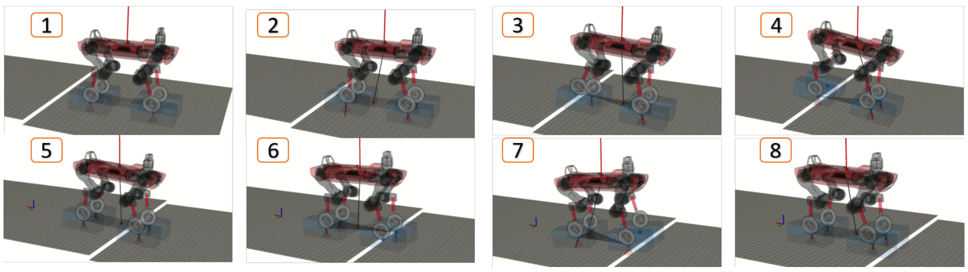

Figure 3.2: An example of a feasible stepping motion in term of terrain constraints. The robot starts with a driving, after it senses the stair, it lifts its leg in the order of LF, RF, LH, RH. While stepping over the stair, the robot is simultaneously driving.

3.2 MPC Optimization Illustration

In this project, all trajectories are set to have a horizon of 2.4 seconds and have a sampling frequency of 50 Hz. MPC has a horizon of 1 second and will be implemented at 20 Hz.

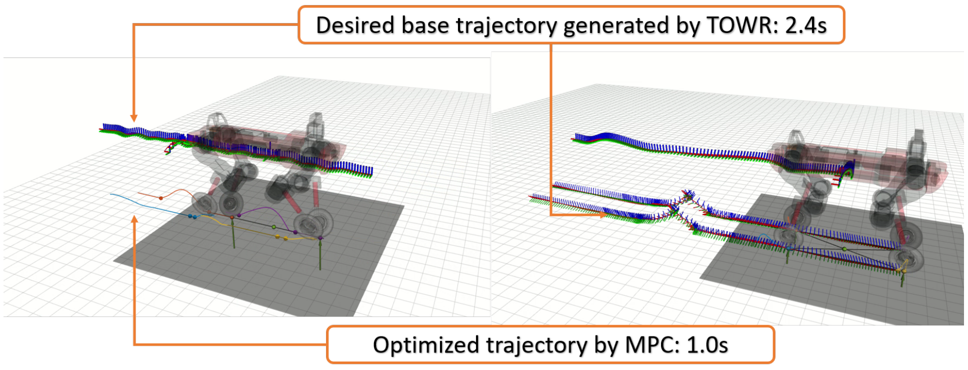
Desired base trajectory generated by TOWR: 2.4s
Optimized trajectory by MPC: 1.0s

Figure 3.3: Comparison between TOWR trajectory and the MPC-optimized trajectory. On the left is a snapshot from trotting motion and on the right is the gap motion. The desired trajectory is generally longer than the predictive horizon. No disturbance is introduced in MPC computing, the MPC-optimized trajectory is solely dependent on the parameters used in the MPC framework, which are tuned based on the ANYmal robot. Therefore, the MPC-tracking performance is out of the scope of this project (practically this tracking is very well).

3.3 Simulation

The simulations were carried out in the robot simulation environment using the full rigid body dynamics of the ANYmal robot. In order to verify the tracking performance of our MPC-feedback controller, Fig. 3.4 shows the desired motions compared with the measured positions with jumping over gap motion. MPC-feedback controller can well track the desired trajectory both for base position and wheel positions. The average tracking error along the whole trajectory is 6 cm for the base. The top-right picture depicts that the measured pitch angle is smoothed compared to the desired orientation as MPC punishes wild motions by adding a cost function about it. In this way, MPC can quotient out some wild motions and make the computed trajectory practically more feasible. The bottom-left picture of this picture also shows the initial velocity of the robot is not zero. It is due to the fact that MPC looks several steps ahead and blend between transitions.

Fig. 3.5 shows that the contact forces for each wheel are well tracked under the MPC-feedback controller. In all four subfigures, there is a period when the contact force is zero, which shows that the corresponding wheel is in the air and performs a jump.

Fig. 3.6 compares the wheel position in z-direction between the desired and the measured trajectory. MPC explicitly sets the swing height to 10 cm while the desired swing height is dependent on the NLP solution.

We did some simulation about other motions, and plots are included in the Appendix. A

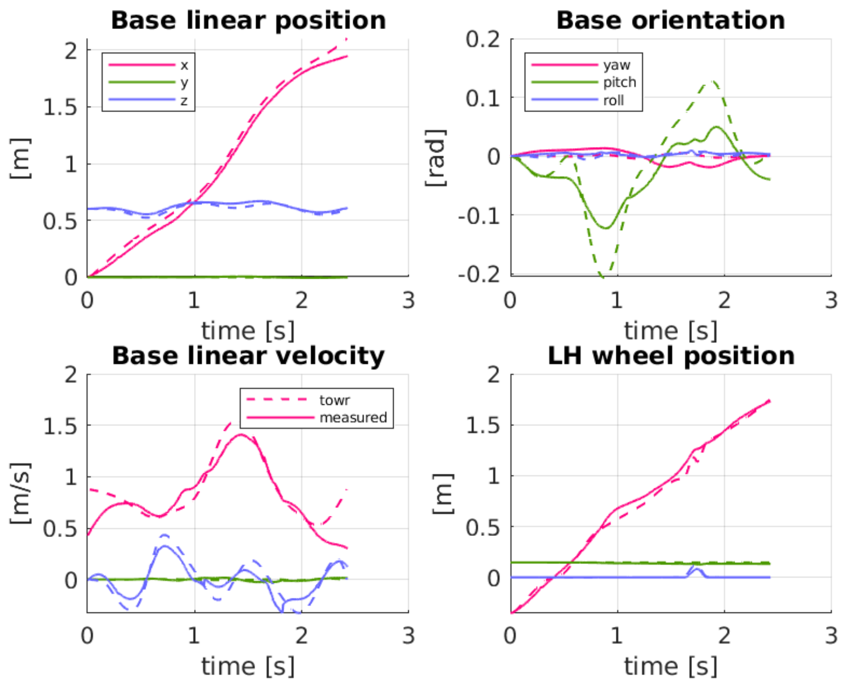

Figure 3.4: The desired trajectory computed by TOWR (dashed line) and the measured positions (full line). In the top-right picture, measured trajectory is smoother than the desired one when the pitch angle is evened. In bottom-left picture, initial velocity of base is blended between transitions.

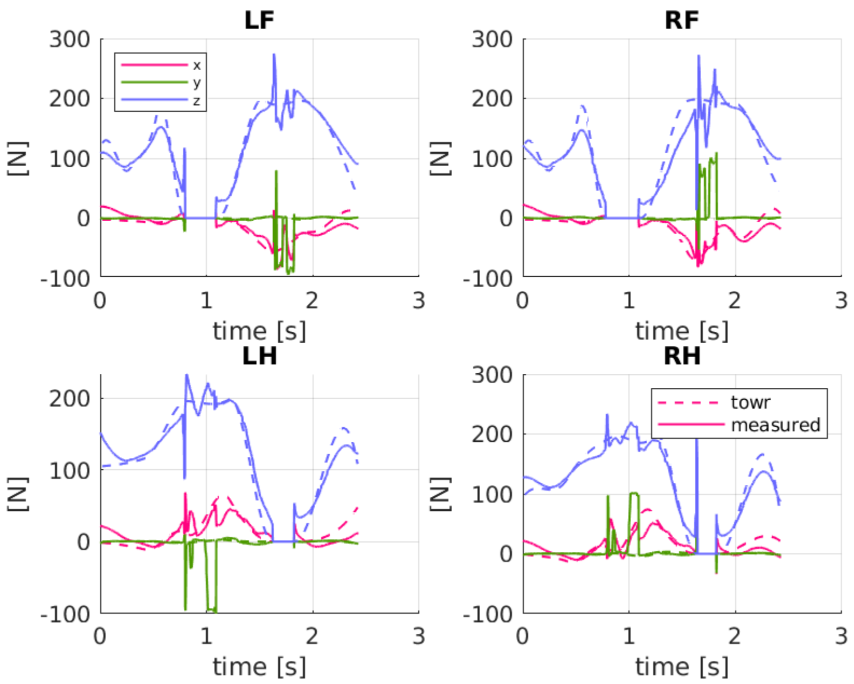

Figure 3.5: The desired contact forces computed by TOWR (dashed line) and the measured contact forces (full line). MPC-feedback controller is able to track the contact forces as well.

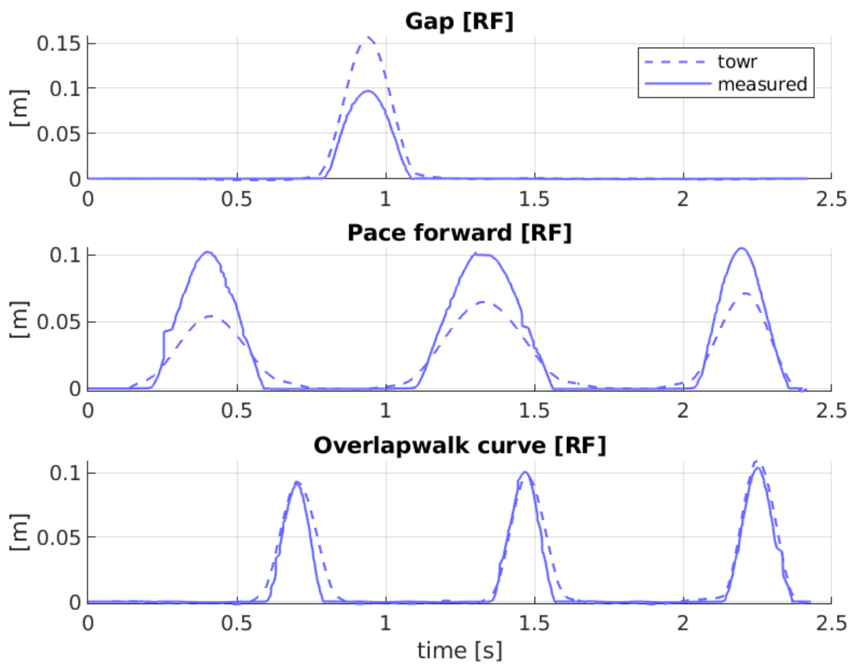

Figure 3.6: The desired wheel position and the measured wheel position in the z-direction for different motions. Measured swing height is always close to 10 cm while the desired one is relatively flexible depending on the motion.

3.4 Hardware Experiment

We conducted experimental tests on different gait motions. Fig. 3.7 and 3.8 show snapshots of the robot conducting pace curve and gap gait. The robot is able to track these trajectories despite the difficulty. The robot sometimes don't have enough power due to hardware limitations, but basically, it proves the correctness of our MPC-feedback TO algorithm.

Fig. 3.9 shows some unexpected situation of the gap motion. The base has a drift in the y-direction and the roll and yaw angle show some strange behaviors. While the robot basically tracks the motion as shown in Fig. 3.8, we also analyze the reason why such an error would happen.

Fig. 3.10 gives us some insights. When the front legs were in the air, the right hind leg had an average force of $153N$ in a 0.8 seconds window while the left hind wheel had $139N$, therefore the robot was pushed towards the positive y-direction. On the other hand, when the hind legs were in the air, the left front wheel had an average $262N$ contact force while the right front wheel had $99N$, then it pushed the robot back. This explains the drift as well as the behavior of yaw and roll angles. Fig. 3.10 also shows that the mass we used for computing the trajectory was half of the true value. While it would introduce some errors, it won't make much difference for the following reason:

- TOWR trajectory is dependent on how we impose constraints. Fig. A.6 shows an example when we take less strict feasibility checks. Under a 0.04-second check step, this trajectory was feasible. However, it was not the case if we pick more checkpoints. The trajectory serves as guidance and the resulting optimized trajectory doesn't necessarily align with that, especially when we relax the feasibility constraints.

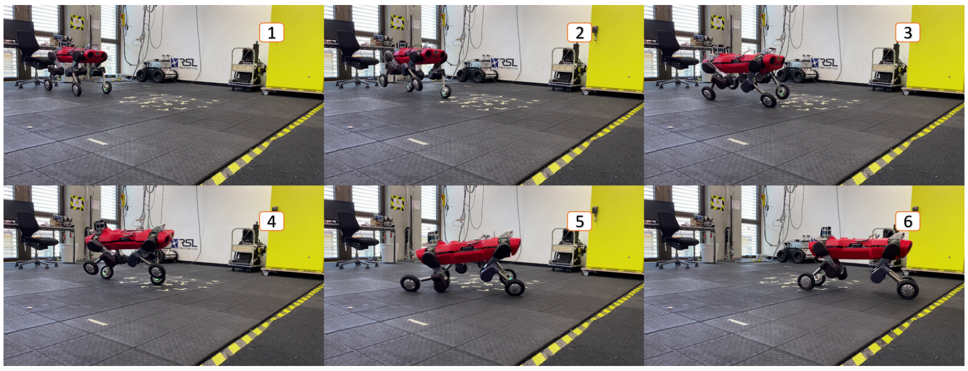

Figure 3.7: Snapshots of the robot conducting a pace curve motion. It was able to track the desired trajectory. When the wheel attached the ground from the air, it would suddenly increase the torque and sometimes overreach the maximum the drive could achieve.

3.5 Tracking Performance

We define the tracking error as the mean Euclidean distance between the desired trajectory and the measured trajectory along the whole time period. We exclude the drift in y direction for the hardware experiment trot forward as this was mainly

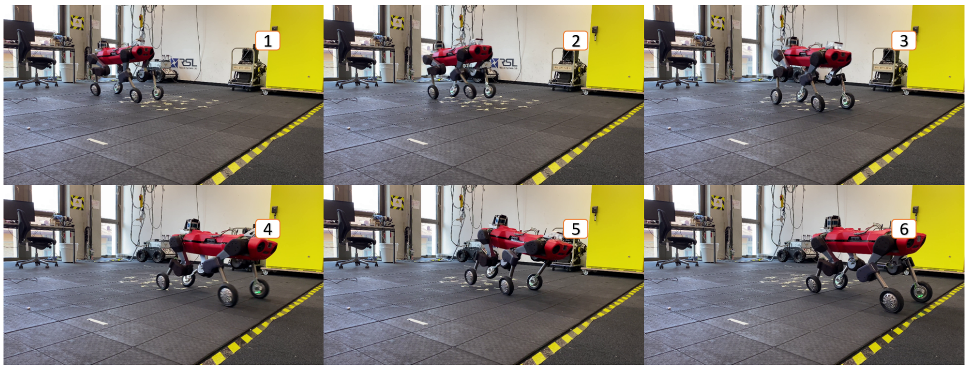

Figure 3.8: Snapshots of the robot conducting a gap jumping motion. A slight drift towards positive y direction can be observed; the robot turned to its left a bit and then turned back.

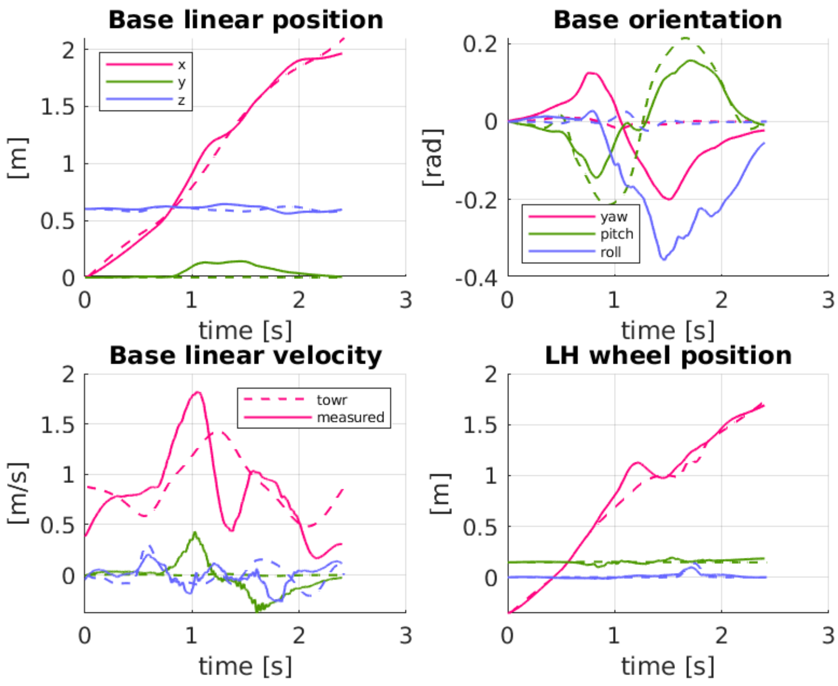

Figure 3.9: The desired trajectory computed by TOWR (dashed line) and the measured positions (full line). In the top-left picture, the base position has an unexpected drift in y direction; Its base orientation also behaves strangely in yaw angle and roll angle.

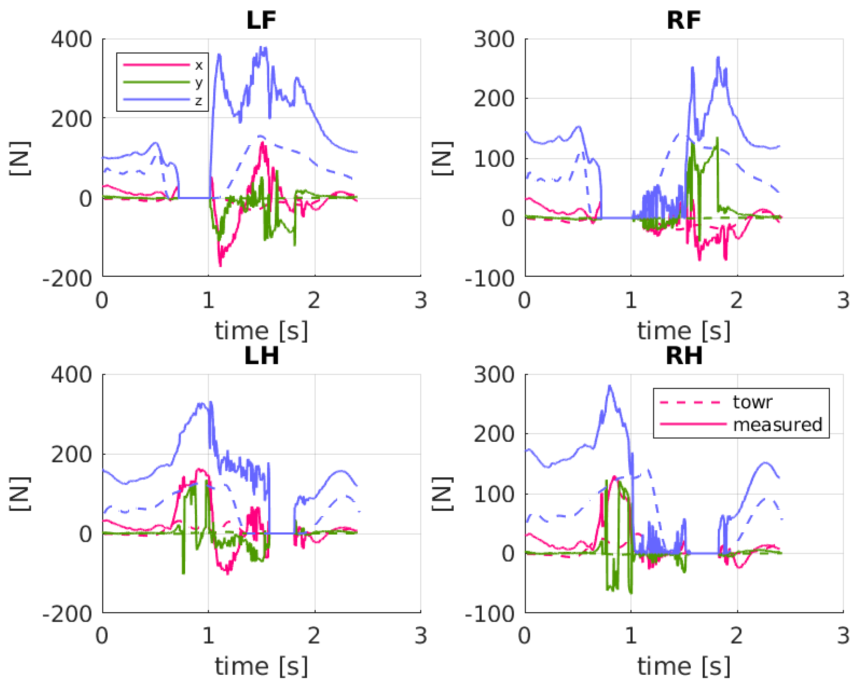

Figure 3.10: Comparison in contact forces for the gap experiment. The desired contact force is generally below the measured force as the mass we used in TOWR was off the true value. When the front legs are in the air, the right hind leg has a bigger contact force and pushes the robot towards the positive y-direction; when the hind legs are in the air, the left front leg has bigger power and pushes the robot back.

due to hardware problems as can be seen from Fig. A.5. The tracking performance is summarized in Tab. 3.1

	Mean Euclidean Distance [m]
Gap (real robot)	0.0862
Trot (real robot)	0.0642
Trot curve (real robot)	0.1759
Gap	0.0600
Pace forward	0.0749
Overlap walk curve	0.0588
Trot forward	0.0717

Table 3.1: Tracking Performance of Motions (Base)

3.6 Discussion

In the experiment, we elaborate on the tracking performance when equipped with MPC-feedback controller. We top this feedback onto a low-level controller and it is still unclear if the WBC alone will have the same performance. The WBC runs at a high frequency of 400 Hz and MPC's frequency is 20 Hz. One resulting issue is that we need to feed the desired control plan of MPC to WBC a couple of times until a new control plan is generated.

When we skip the MPC part and feed the generated trajectory from TOWR directly to WBC, the story is different. There are some more concerns about the trajectory:

Feasibility check

WBC assumes the given trajectory is feasible. In other words, if the trajectory is not feasible, then WBC will also try to track this false trajectory, which will lead to instability. This factor imposes a higher accuracy requirement for the feasibility of the trajectory. For TOWR, this means we need to check the feasibility of more time steps and this will highly increase the computational time.

Smoothness

Fig. 3.11 depicts the comparison between desired and measured contact forces when we use TOWR and WBC without MPC feedback. There are more peaks and the change between transitions is sudden and abrupt. The attached video also shows the movement is stiff and looks like falling anytime.

Tracking performance

Because WBC can't measure the unexpected noise and disturbance from the real model, it behaves some drift when the robot is moving. In this specific case, the tracking error is 0.1308m while this can be decreased to 0.0717m if MPC is equipped.

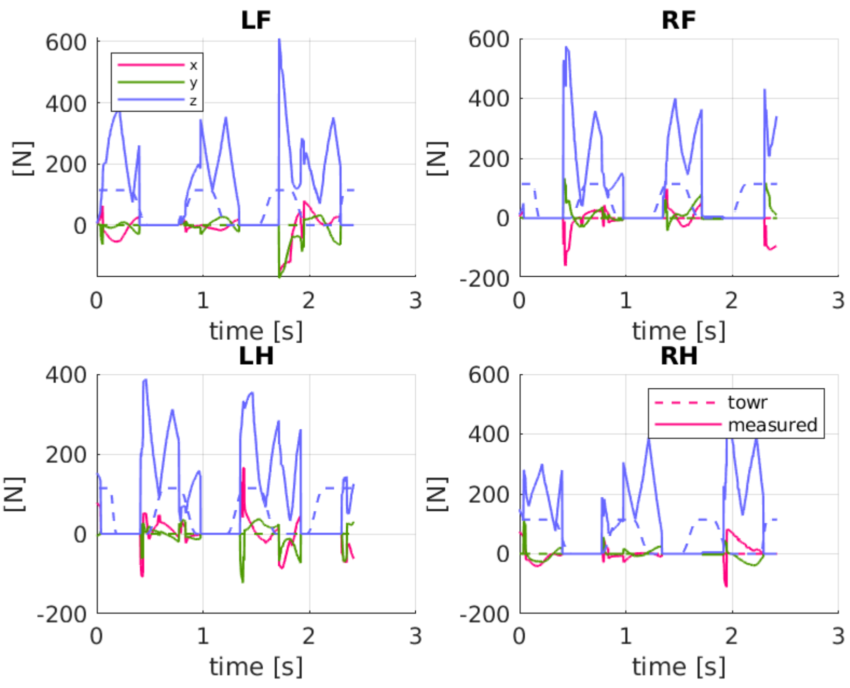

Figure 3.11: Comparison in contact forces for the trot simulation with TOWR and WBC. The measured contact force behaves abruptly between transitions and results in the robot motion stiff. WBC also doesn't take into consideration the perturbed noise and therefore there is drift in the lateral direction as can be seen from the attached video.

Chapter 4

Outcomes

4.1 Conclusion

In this thesis, we proposed to bridge the offline TO and MPC-feedback controller to generate highly dynamics, hybrid motions on a wheeled-legged robot. This framework is built on the top of a WBC and we verified our framework both on simulation and hardware experiments.

This thesis shows the great potential MPC-feedback trajectory optimization has in various tough tasks. We verified on the flat ground, gap ground with various hybrid motions. We also generated a discontinuous walking-over stair trajectory which is ready to be tested on the MPC framework. Introducing MPC-feedback would blend between transitions and smooths the trajectory, which in turn corrects the feasibility of the trajectory. MPC also handles disturbance in parameters and real-world noise and guarantees high performance in tracking. With MPC equipped, a theoretically infeasible trajectory could also be fed into controllers and has a great opportunity for successful tracking. The possibility of a more natural way to merge TO and MPC and make the whole trajectory optimization and tracking process more integrated is being raised in this thesis.

4.2 Outlook

As mentioned above, looking into the future excites lots of possibilities and challenges. There are also immediate improvements that can be done on short notice.

Follow-up experiments

As shown in Chapter. 3. Some hardware experiments experienced issues and don't output ideal results. A new drive is expected to equip the wheel and some more experiments should be done for various motions to get more valid results.

Incorporating terrain information in MPC

Currently, we used a flat ground for all motions in MPC, and a more complex terrain is expected to be overcome in the future. Discontinuous stair needs to be convexitized in order that an MPC problem can be solved efficiently.

Trajectory relaxation

Here trajectory relaxation refers to the relatively relaxed constraints imposed to check the feasibility of the trajectory. We have shown that some trajectories that are not theoretically feasible can also be used as guidance for MPC. After MPC's correction, an optimized trajectory would be generated. This can be a potential direction to achieve online trajectory optimization and tracking. A similar concept also appears in [18] where trajectory relaxation is used to compute inverse kinematics efficiently.

Learning based TO

If a trajectory relaxation is accepted, then learning-based TO would be a candidate for new optimizers because of its natural early stopping feature and domain-dependent framework.

Bibliography

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

[2] T. Klamt and S. Behnke, "Anytime hybrid driving-stepping locomotion planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4444–4451.

[3] ——, "Planning hybrid driving-stepping locomotion on multiple levels of abstraction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1695–1702.

[4] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-body mpc and online gait sequence generation for wheeled-legged robots," *arXiv preprint arXiv:2010.06322*, 2020.

[5] M. Bjelonic, C. D. Bellicoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, and M. Hutter, "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2116–2123, 2019.

[6] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4102–4109.

[7] M. Raibert, *Legged Robots That Balance.* Cambridge, MA: MIT Press, 1986.

[8] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[9] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2017.

[10] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[11] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[12] E. Jelavic and M. Hutter, "Whole-body motion planning for walking excavators," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2292–2299.

[13] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[14] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 577–584.

[15] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[16] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3346–3351.

[17] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," *arXiv preprint arXiv:1905.06144*, 2019.

[18] K. Dufour and W. Suleiman, "On inverse kinematics with nonlinear criteria: Trajectory relaxation," in *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*, 2018, pp. 102–107.

Appendix A

Experiment Plots

A.1 Simulation plots

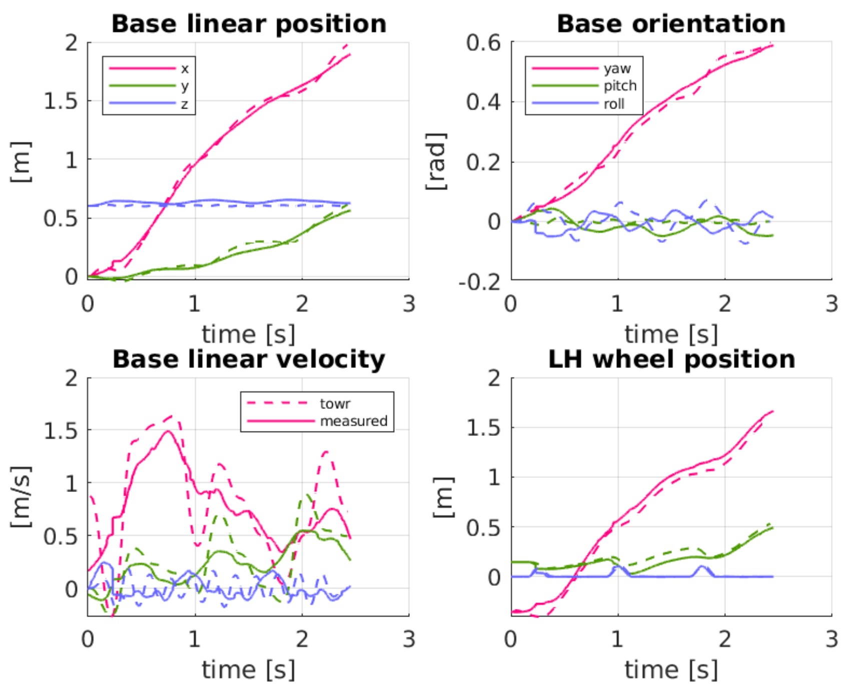

Figure A.1: Overlap walk curve: trajectory

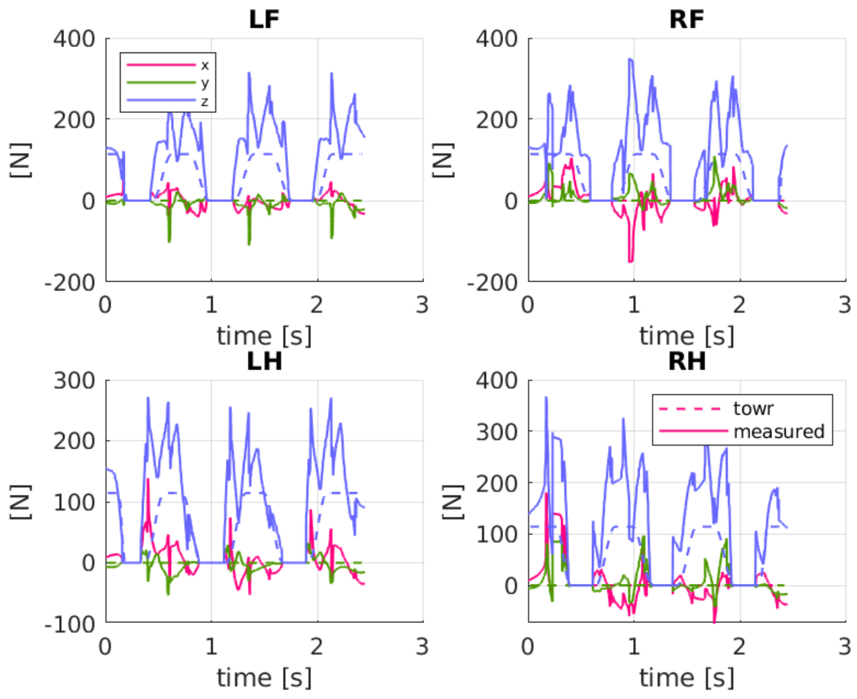

Figure A.2: Overlap walk curve: contact force

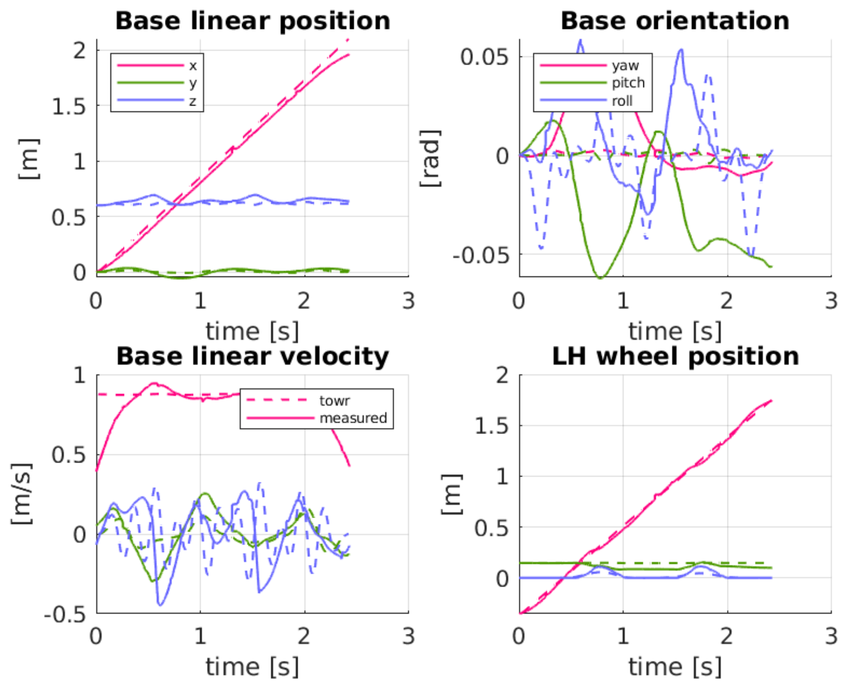

Figure A.3: Pace forward: trajectory

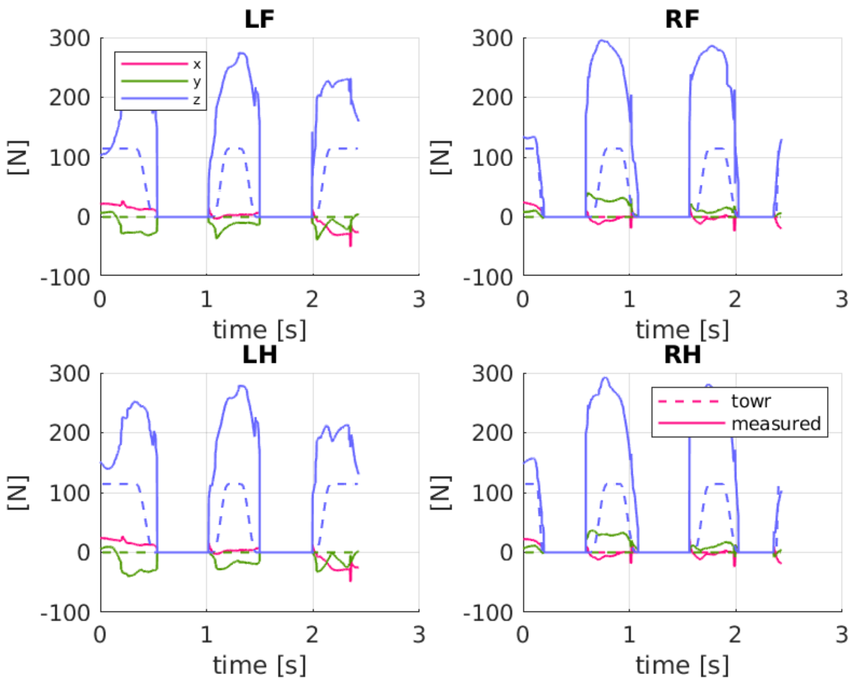

Figure A.4: Pace forward: contact force

A.2 Hardware plots

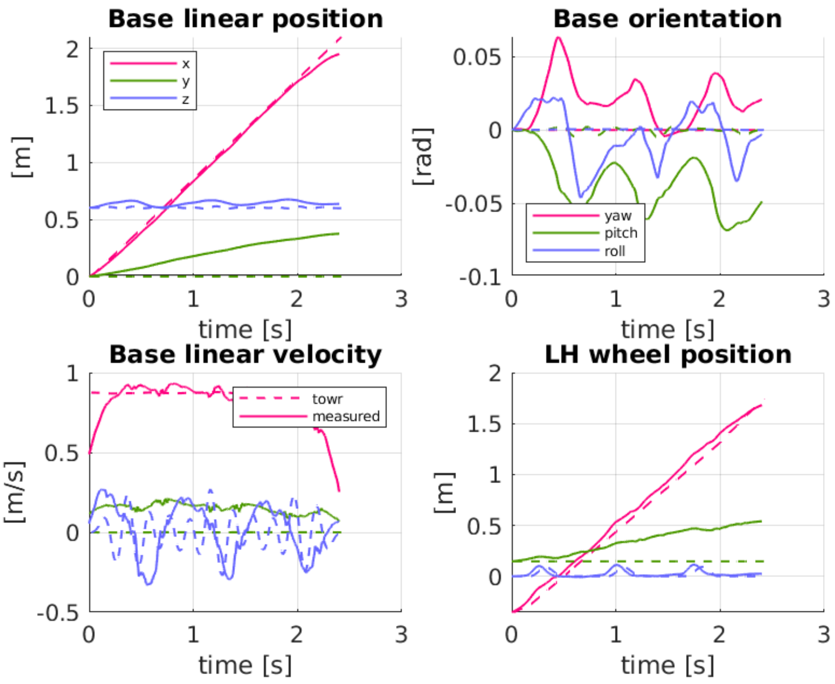

Figure A.5: Trot forward (hardware): trajectory

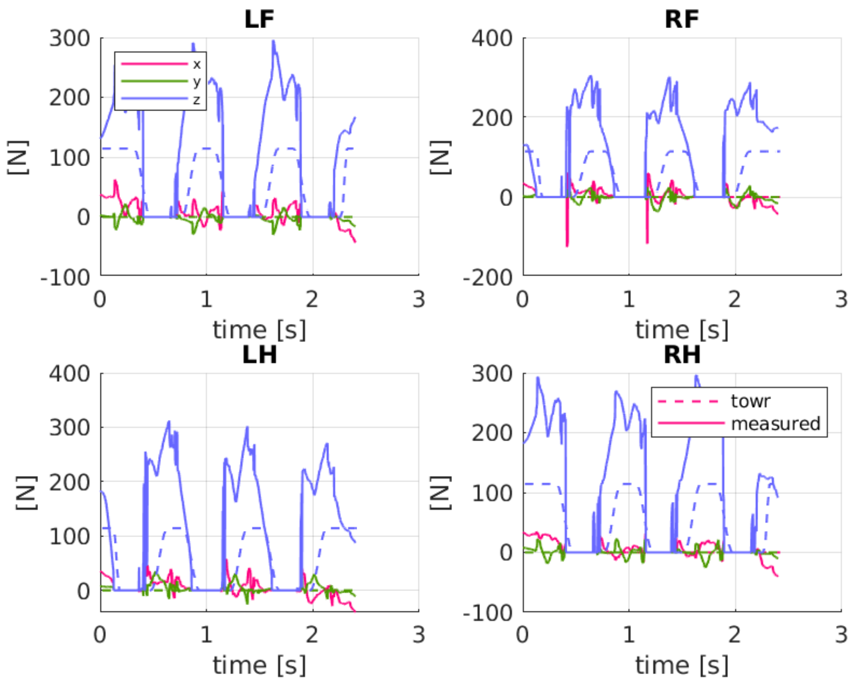

Figure A.6: Trot forward (hardware): contact force

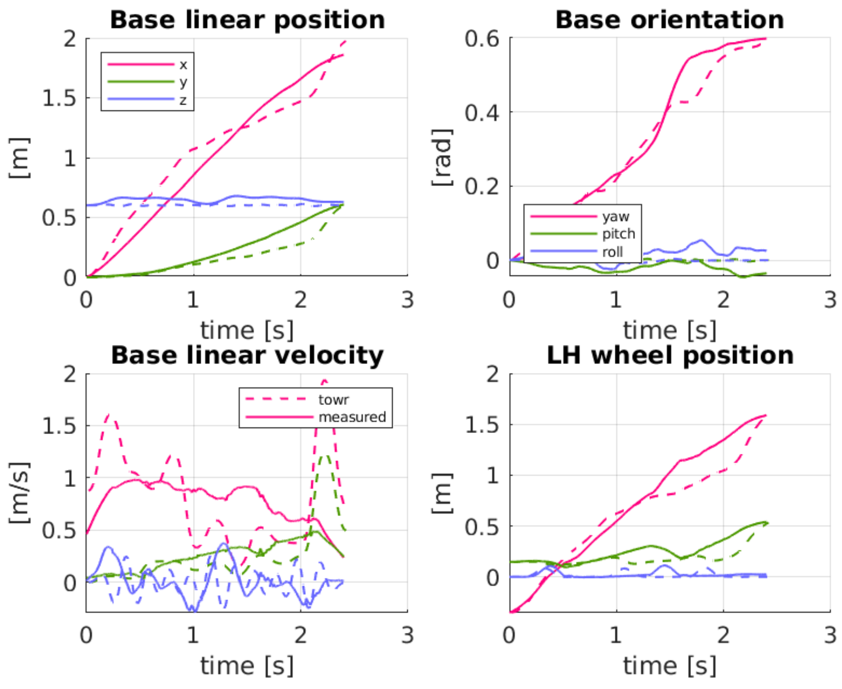

Figure A.7: Trot curve (hardware): trajectory

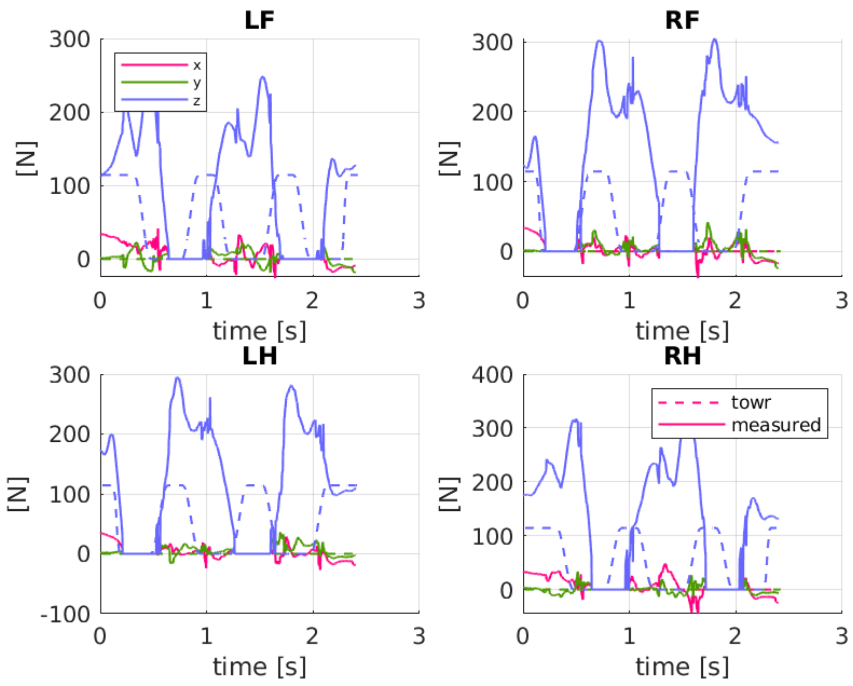

Figure A.8: Trot curve (hardware): contact force

Appendix B

Listings

B.1 Video listings

All videos used in this thesis are hosted at: https://1drv.ms/f/s!AnVR8jEfSpl0hKoEBfgQNpr73CChEQ

B.2 Plotting code

Plotting matlab files are stored at: https://1drv.ms/u/s!AnVR8jEfSpl0hKoOwBOjE94QhbfxrA?e=Yh9A1D

B.3 Trajectory bag file

Sample trajectories are stored at: https://bitbucket.org/leggedrobotics/ocs2_to/src/master/