NAIST-IS-DD1561003

Doctoral Dissertation

Hierarchical Model Predictive Control for Real-time Whole-body Control of a Humanoid Robot

Koji Ishihara

March 15, 2018

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Koji Ishihara

Thesis Committee:

Professor Kenji Sugimoto	(Supervisor)
Professor Tsukasa Ogasawara	(Co-supervisor)
Associate Professor Jun Morimoto	(Co-supervisor)
Associate Professor Takamitsu Matsubara	(Co-supervisor)
Assistant Professor Masaki Ogura	(Co-supervisor)
Assistant Professor Taisuke Kobayashi	(Co-supervisor)

Hierarchical Model Predictive Control for Real-time Whole-body Control of a Humanoid Robot*

Koji Ishihara

Abstract

Humanoid robots have been promoted for working in such real environments as extremely hazardous situations instead of humans. However, their agility remains problematic because it is so inferior to that of human beings. Even though robotic capabilities are rapidly advancing, the robot operation is dozens of times slower than a human demonstration.

The low speed of task execution can be partially attributed to two factors: a control strategy with multiple high-level controllers and the lack of a strong actuator. Therefore we aim to develop an ideal framework, where a unified whole-body controller plans motion trajectories under the full-body dynamics of a humanoid robot and the desired whole-body motions are executed with a strong actuation system composed of different types of actuators. In this thesis that anticipates the development of such a framework, we address two problems related to a unified controller and an actuation system. Moreover, we develop an estimation framework of task goals from human demonstrations because designing appropriate control objectives is a crucial but labor-intensive issue to generate robot motions with a unified controller.

As a unified whole-body controller, we adopt a real-time optimal control approach known as Model Predictive Control (MPC), which is useful for effectively deriving optimal policies to achieve many task goals. However, a real-time MPC

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1561003, March 15, 2018.

for a humanoid robot is deemed impractical since MPC is computationally intensive. To cope with this problem, we propose a MPC method that has a hierarchical optimization procedure based on a singularly perturbed system of a humanoid robot to ease MPC's computation burden. We evaluate our proposed MPC in a simple toy problem and a simulated humanoid robot and show that it successfully reduces the computation time without significantly degrading the control performance. We evaluate our approach in a real humanoid robot.

The development of a single light, compliant, yet strong actuator remains challenging due to the limitations of current actuation technology. Thus, a hybrid actuation system is a good alternative for such a light and strong actuator. If the distribution of the desired torque to each actuator is properly determined, the hybrid actuator system will provide high actuation performance from the complementary behavior of each actuator. To achieve this, we derive a two-stage control scheme for a pneumatic-electric hybrid actuator system based on its singularly perturbed system to decide the optimal torque distribution in real-time. We achieve our proposed control scheme with a hierarchical MPC. We evaluate our two-stage MPC on a forearm robot in tracking control problems and demonstrate that our approach successfully finds a torque distribution strategy in real-time.

Although it is important to specify informative control objectives that are sufficient to generate human-like fast behaviors with our hierarchical MPC, they must be designed through a time-consuming trial-and-error parameter tuning process. Although using captured human movements might be a useful approach to estimate a robot's control objectives, we cannot directly do so since the dynamics of humans and humanoid robots are not identical. To address this problem, we propose an estimation framework of control objectives from human demonstrations. In our proposed approach, after converting the human demonstrations into feasible motions for the robot, we extract human movement skills as objective functions with Inverse Optimal Control (IOC) and show that control objectives for jumping and squatting can be learned from captured human movements.

Keywords:

whole-body motion generation, optimal control, inverse optimal control, humanoid robot, hybrid actuator system

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Generating multiple whole-body motions of a humanoid robot is one of the most challenging control problems in robotics. Future humanoid robots are expected to work in dangerous environments instead of humans. However, it remains difficult to generate human-like behaviors, especially in terms of speed [1]. This problem has been demonstrated in a series of competitions called the DARPA Robotics Challenge (DRC) [2], whose primary goal is to encourage the development of versatile robots that can assist humans for disaster-response operations. In DRC, robots are required to execute such complex tasks as opening a door or an industrial gate valve, and walking on uneven terrain. The winning team's task execution, however, was about 20 times slower than a human demonstrator [1]. We can partially attribute their lack of agility to two factors.

The first factor is the current control strategy in which several restricted sub-motions are successively executed to achieve a complex task. Many teams that participated in DRC adopted the following approach. A designer specified several high-level controllers that determined behaviors under kinematic models or low-dimensional dynamics [3, 4, 5, 6], and the robot accomplished tasks by sequentially changing the designed high-level controllers. From the perspective of the task execution's speed, a robot must generate motions while planning whole-body states to efficiently achieve complex tasks. However, generating such whole-body motions is generally hard since both the kinematic and low-dimensional dynamical models significantly restrict generable movements. Instead, several restricted sub-motions are planned with the models, and the robot accomplishes complex tasks by successively performing them.

The ideal solution is to include the full-body dynamics of a humanoid robot in high-level controllers. Since such whole-body controllers share identical dynamics and whole-body state space, a controller can determine future plans while taking into account the influences of the other high-level controllers. As a result, they can be switched or combined to generate whole-body motions that efficiently accomplish the complex task. However, this approach has another problem: designing whole-body controllers that execute motor commands for a large number of degrees of freedom (DOF) is generally labor-intensive. Thus, we believe a unified optimization-based controller, which derives optimal control inputs under the constraints of its own whole-body dynamics, is the best alternative for motion generation in a diverse set of tasks. Since various motions can be automatically derived by specifying simple high-level task goals, the framework of full-body optimal control is a powerful approach. For many-DOF robot control, the full-body optimal control approach has been scrutinized [7, 8, 9]. An online optimal control approach known as Model Predictive Control (MPC) is especially suitable for our purpose. Since a finite optimal control problem is solved at each time-step, MPC can derive control policies under a situation in which robots must adaptively generate their motions by changing control objectives [10, 11, 12]. The robot can generate a smooth transient trajectory from one motion to another motion online to complete a task.

The second factor is actuation. In DRC, after falling, almost no robots were able to get back up without human intervention [1]. Since the development of a strong actuator with a high power-to-weight ratio is quite challenging due to its technical difficulties, robots are still too heavy to recover from falls using their arms [13]. The robots were controlled to remain statically stable but they just engaged in slow motions to prevent disastrous falls. Robots obviously lack a light actuator that can generate large torque.

To address the limitations of current actuation technology, hybrid actuated robots have been developed. By selecting each size or type of actuator to drive each joint based on the required torques, the total weight can be reduced [14]. Otherwise, by combining several actuators to drive one DOF joint, a hybrid actuation design can have a high power-to-weight ratio since different types of actuators have various dynamic properties [15, 16, 17]. For example, a pneumatic artificial muscle (PAM) can generate large torque, but it cannot generate high frequency torques due to the nature of the slow time constant of electric air valves. A small electric motor can compensate the slow PAM behaviors so that the hybrid actuator system can achieve higher control frequency bandwidth.

1.2 Research Motivation and Objective

To avoid the low agility problem attributed to control strategy and actuation, we develop a whole-body control framework where a unified MPC controller plans whole-body motion trajectories based on the full-body dynamics of a humanoid robot, and then the planning trajectories are executed with hybrid actuation systems.

However, MPC is a computationally intensive method since a large optimization problem must be solved at each time-step when the full-body dynamics of a humanoid robot are used as constraints. Since an MPC approach has been deemed unsuitable for real-time control of a high-dimensional system, a task-specific, low-dimensional approximated model such as an inverted pendulum model has been used in MPC frameworks for generating balanced and biped controllers [18, 19, 20]. However, it is difficult and time-consuming to construct a simple model for every task, and significantly reduced models restrict generable movements [21]. The inverted pendulum model has difficulty generating motions in which both feet of the robot leave the ground, such as jumping and flipping. Therefore, we must develop a MPC method that can be utilized in a more general context of motion generation as well as motions that are generated in real-time.

We thus propose a hierarchical MPC approach that aims for real-time humanoid robot control. By transforming humanoid robot dynamics into a singularly perturbed system that contains slow and fast sub-dynamics, control policies are derived through optimizations that are solved hierarchically. Specifically, in an upper layer, a coarse whole-body trajectory is decided under original full-body dynamics through an optimization with large time-steps. In a lower layer, the coarse trajectory is modified through an optimization with a fine-time resolution under low-dimensional dynamics. We extract the fast sub-dynamics as the low-dimensional dynamics from the singularly perturbed system. A humanoid robot's singularly perturbed system is newly derived to realize our proposed hierarchical

MPC.

A crucial issue of a hybrid actuator system is the torque distribution problem, where a controller is necessary to decide the amount of torque required for each actuator to produce a given desired torque. To solve this problem, a Macro-Mini control scheme [22, 23] and an optimal control approach [24] have been developed. In the Macro-Mini control scheme, the output torque of the slow actuator (PAM) is decided first and then the output of the fast actuator (electric motor) is decided based on the torque tracking error between the desired torque and the torque generated by the PAM. The sum of the PAM and electric motor outputs are applied to the robot as a current control input. On the other hand, the optimal control approach can automatically derive a series of optimal torque distributions to accomplish the entire task in accordance with such optimization criteria as a minimum energy criterion. However, the computational cost of the optimization is too high to derive torque distributions within the short control period of an actuator system.

We propose a control scheme that determines the optimal torque distributions for a hybrid actuator system in real-time based on the hierarchical MPC approach. In our proposed approach, similar to the Macro-Mini control scheme, a two-stage control scheme is adopted where the PAM torques are computed in the first stage, and then the electric motor's outputs are derived in the second stage to achieve the control objectives. We consider the hybrid actuator system a singularly perturbed system and extract fast sub-dynamics for a two-stage control scheme. The output torques are optimized using the original and fast dynamics by the hierarchical MPC.

The control framework's schematic is depicted in Fig. 1.1. If suitable control objectives are specified, the unified controller of the hierarchical MPC plans whole-body motion trajectories and decides the torque distributions so that the target trajectories are executed with the hybrid actuation system. However, designing control objectives through which the robot can plan human-like behaviors in real-time is time-consuming. Therefore they must be specified by trial and error since the cost (or reward) functions are not always obvious that connect the complex behaviors of a humanoid robot and the task goals. We develop an estimation framework in which the control objectives of the hierarchical MPC are

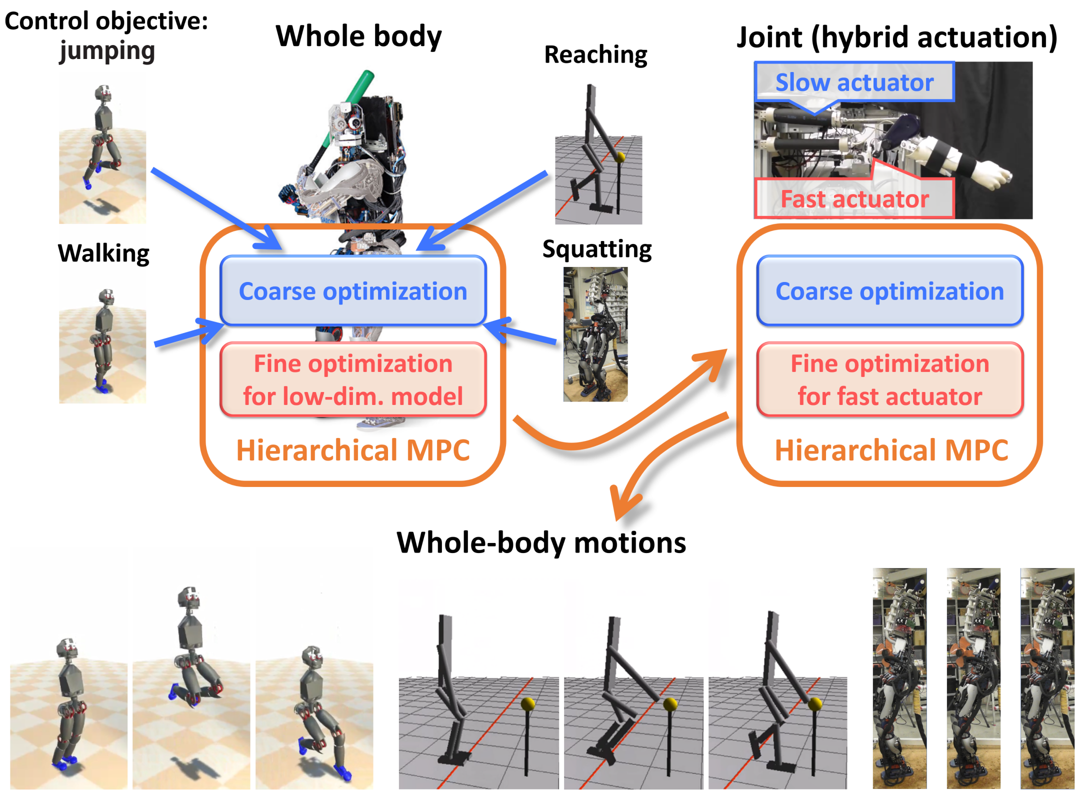


Figure 1.1: Overview of our ideal control framework for improving agility: As a unified controller, MPC plans whole-body motion trajectories using a humanoid robot's full-body dynamics. Planning trajectories are executed with strong actuators (hybrid actuation systems). For the framework's development, we address three problems related to the unified controller, hybrid actuation systems, and control objectives by proposing hierarchical MPC and an estimation framework of control objectives.

learned for real-time motion generation. For human-like behaviors, an approach that uses captured human movements to design a humanoid robot's control objectives might be fruitful because humans and humanoid robots share similar body structure.

However, since the dynamics of humans and robot models are not exactly identical, we cannot directly copy human joint angle trajectories to construct the control objectives. One possible approach is using specially designed movement features to convert human movements into robot behaviors [25]. In such a framework, however, these movement features need to be carefully hand-tuned for each specific movement task. Since this approach is unsuitable for generating a wide variety of robot movements, we focus on the time axis as common features among all the movements. To convert different types of infeasible motion data for a humanoid robot into feasible ones, we adopt a time warping technique, in which the time lines of human movements are modified to speed up or slow down the movements [26].

The cost functions representing the control objectives (objective functions) are estimated from the modified human movements using an Inverse Optimal Control (IOC) method [27], [28], [29]. Although the standard IOC's effectiveness is restricted in low-dimensional systems because many standard IOC methods are required to repeatedly solve a forward optimal control problem in the learning process, a local IOC method was recently developed to handle high-dimensionality in which a locally approximated likelihood is maximized to match human demonstrations [30]. For an offline optimal control approach, this method actually learned an objective function of biped locomotion from human demonstrations [31]. We explore the use of this local IOC method to estimate the control objectives for an online optimal control approach: the hierarchical MPC.

1.3 Dissertation Overview

The rest of this thesis is organized as follows.

Chapter 2 introduces our proposed hierarchical MPC: Coarse-Fast MPC. We evaluate it in a simulated 3D humanoid model and show that it can reduce the computational time without significantly degrading the control performances. As a result, we successfully generated in real-time eight humanoid robot motions including such complex behaviors as jumping and flipping. We also compared our approach with a conventional whole-body controller in terms of the required time to achieve a whole-body reaching task to show that motion planning with full-body dynamics results in fast motion execution. In addition, we evaluated our approach on a real robot experiment and show that very simple motions (standing and slow squatting) can be generated with the Coarse-Fast MPC.

Chapter 3 presents the proposed control scheme for a hybrid actuator system. The two-stage control scheme with a hierarchical MPC is called a two-stage MPC. We investigated the hybrid actuator model as a singularly perturbed system, derived a two-stage optimization strategy, and demonstrated that our method derives a torque distribution strategy for the hybrid actuator by confronting the difficulty of solving the real-time optimal control problem.

Chapter 4 describes our proposed framework that estimates the control objectives for the hierarchical MPC from human movements. We evaluated our proposed method by applying our proposed framework to a humanoid robot model and show that two different movements (jumping and squatting) can be generated with different objective functions estimated by IOC. We demonstrate that both movements can be generated in real-time with our hierarchical MPC approach.

Chapter 5 concludes this thesis with a discussion and outlines future expectations.

2 Hierarchical Model Predictive Control

2.1 Introduction

Motion generations for various tasks under the constraints of their own dynamics and environmental conditions are required for such versatile robots as humanoid and exoskeleton robots (e.g., in DARPA Robotics Challenge [2]). An optimization problem to find the controller under these conditions is considered an optimal control problem. In general, optimal control problems cannot be analytically solved for high-dimensional nonlinear systems. Moreover, in many cases, numerically solving these problems requires a great deal of computational resources. Therefore, trajectory-based optimal control methods, which find the optimal policy around a trajectory and can be applicable to high-dimensional nonlinear systems, are becoming a popular approach for many-DOF robot control [7, 9, 32]. However, most trajectory-based optimal control methods use time-indexed trajectory and are susceptible to external disturbances. In such a case, an online trajectory optimization approach known as Model Predictive Control (MPC) can be useful since it can effectively provide a feedback policy online by solving an finite-horizon optimal control problem at each control time-step. Furthermore, MPC can derive control policies under a situation in which agents must adaptively generate their actions by changing objectives and environments (e.g., changing the location of target objects, changing the behavioral intention of collaborators or users, dealing with unexpected circumstances). MPC methods have been applied to chemical plants [33] or ship-maneuvering problems [10] whose dynamics are sufficiently slow and smooth. Since a sufficient amount of time for calculating computationally intensive optimal control problems can be used in slowly responding systems, MPC is commonly used in such domains. MPC applications to simulated robot models have been demonstrated [11].

MPC, however, is not a suitable approach for the real-time control of a high-dimensional system that quickly changes its behavior due to its computationally intensive method. Thus, a simple low-dimensional approximated model such as the inverted pendulum model has been used in MPC frameworks for generating balancing and biped controllers [18, 20]. However, it is difficult to construct a simple model for every task, and a significantly reduced model restricts generable movements [21]. To generate whole-body motions with such a restricted model is generally hard, and this reduces a humanoid robot's agility. Therefore, developing a MPC method, which can be utilized in a more general context of motion generation, is desirable to reduce the computational time.

In this chapter, we propose a hierarchical MPC method that aims for real-time humanoid robot control. By transforming the full-body humanoid robot dynamics into a singularly perturbed system that contains slow and fast sub-dynamics, an optimal control policy can be derived through the hierarchical MPC with reduced computational time, where the behavior of slow sub-dynamics is optimized with a large time-step for coarse optimization but with a long evaluation period, and the behavior of fast sub-dynamics is optimized with a small time-step for fine optimization but with a short evaluation period (Fig. 2.1). The hierarchical MPC structure is a good alternative for easing MPC's computational burden; the controlled system is decomposed into subsystems with which more than one optimization problem is solved while exchanging information among local MPCs [34]. For a chemical process, a composite slow-fast MPC design was demonstrated for singularly perturbed systems that can reduce the computational time [35]. A composite slow-fast MPC approach has been applied to a simulated single link flexible joint manipulator [36]. However, for a highly nonlinear system such as a humanoid robot, it is generally difficult to derive an analytical solution of fast sub-dynamics for computing slow dynamics. Thus, we cannot straightforwardly apply the previously proposed slow-fast MPC approach. In this study, we propose a novel approach in which both the original full dynamics and fast dynamics are considered in hierarchical optimization.

In our experiments, a humanoid robot tries to generate eight movements:

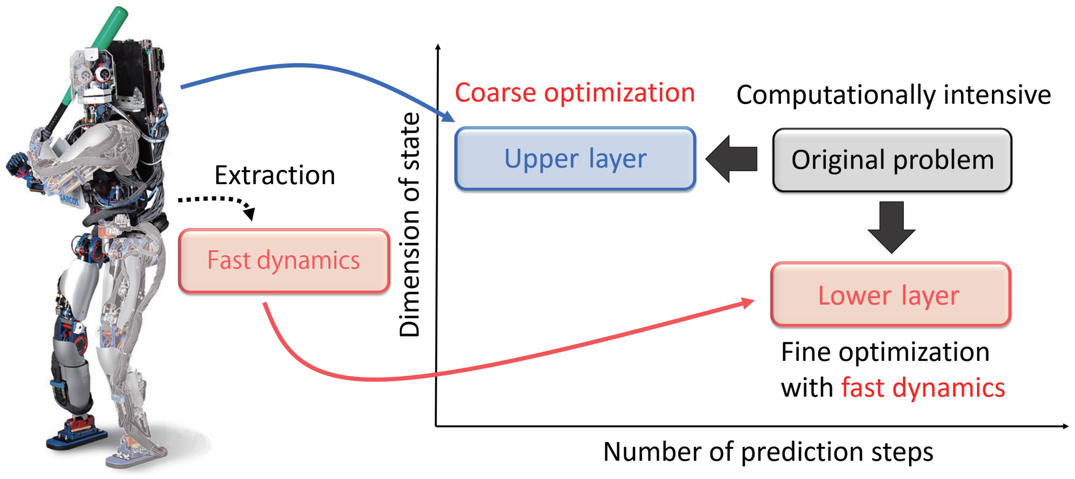


Figure 2.1: Schematic diagram of proposed (Coarse-Fast) MPC method: We decompose original optimal control problem into two smaller problems since it is computationally intensive to be solved in MPC framework.

standing, walking, running, sitting and on a chair, and three kinds of jumping that including a half turn and flip motions. Some motions are required to predict the long state trajectories since myopic behaviors fail to generate target movements (e.g., although the generation of walking is desired, a robot may fall forward without its legs spread apart because of a short prediction horizon). However, due to the high-dimensionality of humanoid robots, the hierarchical MPC approach is inadequate to reduce the computational time when many steps are predicted for the robots. Therefore, we combine the hierarchical MPC with a simple warm-start technique that enables the optimizer to start the optimization with good initial points from the beginning. We reveal that all eight motions were successfully generated in real-time with the new hierarchical MPC approach and compare our approach with a conventional full-body controller in terms of the required time to generate a whole-body reaching motion. Since the hierarchical MPC can plan whole-body motions under full-body dynamics, a walking to approach a target position and a reaching motion are simultaneously generated and improve the execution's speed. In addition, our approach is evaluated on a real robot experiment. We show that very simple motions (standing and slow squatting) can be generated with the hierarchical MPC.

The rest of this chapter is organized as follows. Section 2.2 explains a standard MPC problem. Section 2.3 introduces our proposed methods: Coarse-Fast MPC and Warm-Starting Coarse-Fast MPC. Section 2.4 shows how our proposed MPCs works with a toy example. Section 2.5 presents how we apply our proposed method to a humanoid robot model. We also discuss computational complexity when the proposed method is applied to the humanoid robot. Section 2.6 describes the task goals and the experimental setting for simulation experiments as well as their results. Section 2.7 explains the experimental settings for a real robot experiment and shows its experimental result.

2.2 Model Predictive Control

In MPC at each control period, a finite-horizon optimal control problem is solved to find an optimal control sequence that minimizes the accumulated cost over a finite future horizon. We evaluate the objective function based on a state trajectory and a control sequence estimated using a nonlinear system model:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ &= \mathbf{x}_k + \Delta t \mathbf{f}_d(\mathbf{x}_k, \mathbf{u}_k), \end{aligned} \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ respectively denote the state vectors and the control inputs. Δt denotes a step size (a control period). The accumulated cost is defined:

$$J(\mathbf{x}_k, \mathbf{U}_k) = \sum_{t=k}^{k+N-2} \ell(\mathbf{x}_t, \mathbf{u}_t) + \ell_T(\mathbf{x}_{k+N-1}), \tag{2.2}$$

where $\mathbf{U}_k \equiv \{\mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{k+N-2}\}$ is the control sequence. $\ell(\mathbf{x}, \mathbf{u})$ is the immediate cost, and $\ell_T(\mathbf{x}) = \ell(\mathbf{x}, \mathbf{0})$ is the terminal cost.

The optimal control sequence is defined:

$$\mathbf{U}_k^\star \equiv \arg\min_{\mathbf{U}_k} J(\mathbf{x}_k, \mathbf{U}_k), \tag{2.3}$$

where * indicates the optimized variables. The problem of minimization over an entire control sequence can be reduced to a sequence of minimizations over a single control input by the Dynamic Programming Principle. Define value function V as

$$V_k(\mathbf{x}_k) \equiv \min_{\mathbf{U}_k} J(\mathbf{x}_k, \mathbf{U}_k). \tag{2.4}$$

The total cost can be rewritten recursively with the following value:

$$\begin{aligned} V_k(\mathbf{x}_k) &= \min_{\mathbf{u}_k}[\ell(\mathbf{x}_k, \mathbf{u}_k) + V_{k+1}(\mathbf{x}_{k+1})] \\ &= \min_{\mathbf{u}_k} Q_k(\mathbf{x}_k, \mathbf{u}_k). \end{aligned} \tag{2.5}$$

By setting $V_{k+N-1}(\mathbf{x}_{k+N-1}) \equiv \ell_T(\mathbf{x}_{k+N-1})$, the optimal input at each time-step is derived backwards in time.

Once optimal control sequence $\mathbf{U}_k^\star$ is derived, the first few elements (usually only first element $\mathbf{u}_k^\star$) of the optimal control sequence are applied to the robot. Since the optimal control sequence is computed at each time-step based on the robot's current state and the current definition of the cost function, MPC can cope with external disturbances and adaptively derive control policies even with dynamically changing cost functions. After a new state is measured at subsequent time-step $k+1$, optimal control sequence $\mathbf{U}_{k+1}^\star$ is derived and the robot is controlled in the same manner.

Although the optimal control sequence can be derived if the control input at each time-step is found that minimizes function Q, it cannot be analytically derived for nonlinear systems in general. Instead, to derive a locally optimal controller, we use a Differential Dynamic Programming (DDP) [37] approach called iterative Linear Quadratic Regulator (iLQR) [8]. DDP and iLQR have been used in the MPC context [11, 38]. In iLQG, given nominal $(\mathbf{x}_t, \mathbf{u}_t)$ at time-step t, the Q-function is approximated with a second order Tylor expansion of Q around $(\mathbf{x}_t, \mathbf{u}_t)$:

$$\begin{aligned} Q_t(\mathbf{x}_t + \delta\mathbf{x}_t, \mathbf{u}_t + \delta\mathbf{u}_t) &\approx Q_t(\mathbf{x}_t, \mathbf{u}_t) \\ &+ \frac{1}{2}\begin{bmatrix} 1 \\ \delta\mathbf{x}_t^\top \\ \delta\mathbf{u}_t^\top \end{bmatrix}^\top \begin{bmatrix} 0 & Q_{\mathbf{x}t}^\top & Q_{\mathbf{u}t}^\top \\ Q_{\mathbf{x}t} & Q_{\mathbf{xx}t} & Q_{\mathbf{xu}t} \\ Q_{\mathbf{u}t} & Q_{\mathbf{ux}t} & Q_{\mathbf{uu}t} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}. \end{aligned} \tag{2.6}$$

Here, $(\mathbf{x}_t + \delta\mathbf{x}_t, \mathbf{u}_t + \delta\mathbf{x}_t)$ represents the perturbations around $(\mathbf{x}_t, \mathbf{u}_t)$ pair. The

Q-function derivatives are given by:

$$\begin{aligned}
Q_{\mathbf{x}t} &= \ell_{\mathbf{x}t} + \mathbf{f}_{\mathbf{x}t}^\top V_{\mathbf{x}t+1} && (2.7)\\
Q_{\mathbf{u}t} &= \ell_{\mathbf{u}t} + \mathbf{f}_{\mathbf{u}t}^\top V_{\mathbf{x}t+1} && (2.8)\\
Q_{\mathbf{xx}t} &= \ell_{\mathbf{xx}t} + \mathbf{f}_{\mathbf{x}t}^\top V_{\mathbf{xx}t+1}\mathbf{f}_{\mathbf{x}t} && (2.9)\\
Q_{\mathbf{ux}t} &= \ell_{\mathbf{ux}t} + \mathbf{f}_{\mathbf{u}t}^\top V_{\mathbf{xx}t+1}\mathbf{f}_{\mathbf{x}t} && (2.10)\\
Q_{\mathbf{uu}t} &= \ell_{\mathbf{uu}t} + \mathbf{f}_{\mathbf{u}t}^\top V_{\mathbf{xx}t+1}\mathbf{f}_{\mathbf{u}t}. && (2.11)
\end{aligned}$$

The optimal control modification that minimizes Eq. (2.6) with respect to $\delta\mathbf{u}_t$ can be derived analytically:

$$\begin{aligned}
\delta\mathbf{u}^\star &= -Q_{\mathbf{uu}t}^{-1}(Q_{\mathbf{u}t} + Q_{\mathbf{ux}t}\delta\mathbf{x}_t)\\
&= \mathbf{k}_t + \mathbf{K}_t\delta\mathbf{x}_t.
\end{aligned} \tag{2.12}$$

By plugging the input into Eq. (2.6), the second order local model of the value function is given:

$$\begin{aligned}
V_k(\mathbf{x}_t + \delta\mathbf{x}_t) &\approx V(\mathbf{x}_t) + \Delta V_t\\
&+ \frac{1}{2}\begin{bmatrix} 1 \\ \delta\mathbf{x}_t^\top \end{bmatrix}^\top \begin{bmatrix} 0 & V_{\mathbf{x}t}^\top \\ V_{\mathbf{x}t} & V_{\mathbf{xx}t} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x}_t \end{bmatrix},
\end{aligned} \tag{2.13}$$

where ΔV_t,$V_{\mathbf{x}t}$ and $V_{\mathbf{xx}t}$ are given by

$$\begin{aligned}
\Delta V_t &= -\tfrac{1}{2}Q_{\mathbf{u}t}^\top Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{u}t} && (2.14)\\
V_{\mathbf{x}t} &= Q_{\mathbf{x}t} - Q_{\mathbf{ux}t}^\top Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{u}t} && (2.15)\\
V_{\mathbf{xx}t} &= Q_{\mathbf{xx}t} - Q_{\mathbf{xu}t}Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{ux}t}. && (2.16)\\
& && (2.17)
\end{aligned}$$

Given initial state $\mathbf{x}_k$ and inputs $\mathbf{U}_k = \mathbf{U}^{init}$, new control inputs around the nominal trajectory $(\mathbf{x}_k, \mathbf{U}_k)$ can be obtained recursively backward and forward in time from the above equations. From time-step $k + N - 1$, the value function is set to $V_{k+N-1}(\mathbf{x}_{k+N-1}) = \ell_T(\mathbf{x}_{k+N-1})$ or given by Eqs. (2.15) or (2.17), and the Q-function derivatives are computed with Eqs. (2.8) and (2.11). Policy $\{\mathbf{k}, \mathbf{K}\}$ in the optimal control modifications is obtained by Eq. (2.12). The new control input at time-step t is given from $t = k$ to $k + N - 2$:

$$\begin{aligned} \tilde{\mathbf{x}}_k &= \mathbf{x}_k && (2.18) \\ \tilde{\mathbf{u}}_t &= \mathbf{u}_t + \mathbf{k}_t + \mathbf{K}_t(\tilde{\mathbf{x}}_t - \mathbf{x}_t) && (2.19) \\ \tilde{\mathbf{x}}_{t+1} &= \mathbf{f}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t). && (2.20) \end{aligned}$$

Set the new trajectory $(\tilde{\mathbf{x}}_k, \tilde{\mathbf{U}}_k)$ as the nominal one and repeat the above computation until it converges to the locally optimal trajectory for the original nonlinear system. The regularization to keep $Q_{\mathbf{uu}t}$ a positive definite matrix and the line search procedure are improved to use iLQG in the MPC context [11]. We employed both changes to benefit from fast convergence.

At next time-step $k + 1$, the solver starts to compute optimal sequence $\mathbf{U}^{\star}_{k+1}$ with a warm-start technique in which previous solution $\mathbf{U}^{\star}_{k}$ is used as a starting point of the current optimization. The technique is often used to speed up the optimizer [11, 39]. Suppose the previously computed trajectory is $\check{\mathbf{U}}_k = \mathbf{U}^{\star}_{k}$:

$$\check{\mathbf{U}}_k \equiv \{\check{\mathbf{u}}_k, \check{\mathbf{u}}_{k+1}, \cdots, \check{\mathbf{u}}_{k+N-2}\}. \qquad (2.21)$$

We can initialize the initial control sequence with the time-shifted trajectory of $\check{\mathbf{U}}_k$. In the case of shifting one time-step, the initial sequence is

$$\mathbf{U}^{init}_{k+1} \leftarrow \{\check{\mathbf{u}}_{k+1}, \check{\mathbf{u}}_{k+2}, \cdots, \check{\mathbf{u}}_{k+N-2}, \check{\mathbf{u}}_{k+N-2}\}. \qquad (2.22)$$

2.3 Proposed Method

2.3.1 Problem Decomposition

If a plant has a high-dimensional state and input variables but sometimes needs to generate fast movements, the above finite optimal control problem of a conventional MPC must be solved within a very short time period to compute a large number of input variables. A humanoid robot corresponds to such a high-dimensional system.

Utilizing the distributed or hierarchical MPC structures is a feasible alternative to ease MPC's computational burden [34]. The computational time can be reduced with these structures because the optimal control problems are formulated based on the property of each subsystem and the problem size becomes smaller than the original one. A detailed explanation about the computational complexity of MPC (a gradient-based optimization algorithm such as iLQR) is described in Section 2.5.

A time-scale decomposition technique, especially a two-time-scale decomposition technique, is widely used to make subsystems [40]. We focus on a two-time-scale decomposition for the following singularly perturbed systems:

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{h}(\mathbf{y}, \mathbf{z}, \mathbf{u}^{\mathbf{y}}, \mathbf{u}^{\mathbf{z}}) \\ \varepsilon \dot{\mathbf{z}} &= \mathbf{g}(\mathbf{y}, \mathbf{z}, \mathbf{u}^{\mathbf{z}}), \end{aligned} \tag{2.23}$$

where $\mathbf{y} \in \mathbb{R}^{n_y}$, $\mathbf{z} \in \mathbb{R}^{n_z}$, $\mathbf{u}^{\mathbf{y}} \in \mathbb{R}^{m_y}$ and $\mathbf{u}^{\mathbf{z}} \in \mathbb{R}^{m_z}$ respectively denote the vectors of slow, fast state variables, and control inputs for slow and fast states. ε is a small positive parameter.

To extract a slow subsystem, we set $\varepsilon = 0$ in Eq. (2.23):

$$\dot{\mathbf{y}} = \mathbf{h}(\mathbf{y}, \mathbf{z}, \mathbf{u}^{\mathbf{y}}, \mathbf{u}^{\mathbf{z}}) \tag{2.24}$$

$$0 = \mathbf{g}(\mathbf{y}, \mathbf{z}, \mathbf{u}^{\mathbf{z}}). \tag{2.25}$$

Assume that Eq. (2.25) possesses a unique root:

$$\hat{\mathbf{z}} = \hat{\mathbf{g}}(\hat{\mathbf{y}}, \hat{\mathbf{u}}^{\mathbf{z}}). \tag{2.26}$$

Substituting Eq. (2.26) into Eq. (2.24), a slow subsystem is obtained:

$$\dot{\hat{\mathbf{y}}} = \mathbf{h}(\hat{\mathbf{y}}, \hat{\mathbf{g}}(\hat{\mathbf{y}}, \hat{\mathbf{u}}^{\mathbf{z}}), \hat{\mathbf{u}}^{\mathbf{y}}, \hat{\mathbf{u}}^{\mathbf{z}}) = \hat{\mathbf{h}}(\hat{\mathbf{y}}, \hat{\mathbf{u}}^{\mathbf{y}}, \hat{\mathbf{u}}^{\mathbf{z}}). \tag{2.27}$$

We now extract the fast subsystem. We introduce the deviation of state $\boldsymbol{\zeta} = \mathbf{z} - \hat{\mathbf{z}}$ and control variables $\boldsymbol{\nu} = \mathbf{u}^{\mathbf{z}} - \hat{\mathbf{u}}^{\mathbf{z}}$, and the singularly perturbed system in Eq. (2.23) is rewritten with new time-scale $\varepsilon\tau = t$. Finally, we set $\varepsilon = 0$ in the resulting system and obtain:

$$\frac{d\hat{\mathbf{y}}}{d\tau} = 0 \tag{2.28}$$

$$\frac{d\boldsymbol{\zeta}}{d\tau} = \mathbf{g}(\hat{\mathbf{y}}, \boldsymbol{\zeta} + \hat{\mathbf{z}}, \boldsymbol{\nu} + \hat{\mathbf{u}}^{\mathbf{z}}) = \hat{\mathbf{g}}(\boldsymbol{\zeta}, \boldsymbol{\nu}), \tag{2.29}$$

and the system of Eq. (2.29) is the fast subsystem.

In optimal control theory for singularly perturbed systems, composite optimal control structures have been widely studied [40, 41, 42]. A similar structure is also proposed in the MPC context where one MPC regulates the slow subsystem and the other MPC regulates the fast subsystem in a hierarchical or distributed manner [35]. The controller derived through optimizations with the two subsystems is expected to perform as the original system's optimal controller.

2.3.2 Hierarchical Composite Slow-Fast MPC Design

Here we introduce a hierarchical composite MPC design to ease the computational burden. Given total cost J^s, the following slow optimization problem is solved:

$$\min_{\hat{\mathbf{U}}_k^{\mathbf{y}}, \hat{\mathbf{U}}_k^{\mathbf{z}}} J_k^s(\hat{\mathbf{y}}_k, \hat{\mathbf{z}}_k, \hat{\mathbf{U}}_k^{\mathbf{y}}, \hat{\mathbf{U}}_k^{\mathbf{z}}) \tag{2.30a}$$

$$\text{s.t.} \quad \hat{\mathbf{y}}_{k+1} = \hat{\mathbf{y}}_k + \Delta t_s \hat{\mathbf{h}}(\hat{\mathbf{y}}_k, \hat{\mathbf{u}}_k^{\mathbf{y}}, \hat{\mathbf{u}}_k^{\mathbf{z}}), \tag{2.30b}$$

where $\hat{\mathbf{U}}_k^{\mathbf{y}} \equiv \{\hat{\mathbf{u}}_k^{\mathbf{y}}, \hat{\mathbf{u}}_{k+1}^{\mathbf{y}}, \cdots, \hat{\mathbf{u}}_{k+N_s-1}^{\mathbf{y}}\}$ and $\hat{\mathbf{U}}_k^{\mathbf{z}} \equiv \{\hat{\mathbf{u}}_k^{\mathbf{z}}, \hat{\mathbf{u}}_{k+1}^{\mathbf{z}}, \cdots, \hat{\mathbf{u}}_{k+N_s-1}^{\mathbf{z}}\}$ as a slow and fast control sequence. Δt_s is a coarse-time-step ($\Delta t_s > \Delta t$), and N_s is a horizon for the first optimization ($N_s \Delta t_s = N \Delta t$). The fast state is computed with Eq. (2.26). The first optimization problem is smaller than the original one in terms of the state dimension as well as the prediction horizon.

To generate a fast movement, the coarse input sequence derived in the upper-layer optimization must be further optimized with a fine-time resolution. Then the fine optimization's objective is to refine the upper-layer optimization result. In the lower-layer optimization, we use shorter time-step Δt_f rather than coarse-time-step Δt_s. The horizon for the fine optimization is defined as N_f. The second optimization problem is smaller than the original one in terms of the input and state dimensions since a reduced-order subsystem only considered the constraints.

Optimal slow and fast control sequences $\hat{\mathbf{U}}_k^{\mathbf{y}^\star}$, $\hat{\mathbf{U}}_k^{\mathbf{z}^\star}$ are obtained after the first optimization. Optimal slow and fast state trajectories $\hat{\mathbf{Y}}^\star$ and $\hat{\mathbf{Z}}^\star$ are also derived. We define the deviation of a fast state from the optimal fast state in the first optimization as $\boldsymbol{\zeta} = \mathbf{z} - \hat{\mathbf{z}}^\star$, the control input difference as $\boldsymbol{\nu} = \mathbf{u}^{\mathbf{z}} - \hat{\mathbf{u}}^{\mathbf{z}^\star}$, and a sequence of the deviations of control inputs as $\mathbf{N}_j \equiv \{\boldsymbol{\nu}_j, \boldsymbol{\nu}_{j+1}, \ldots, \boldsymbol{\nu}_{j+N_f-2}\}$, and formulate the fine optimization problem to modify the first optimization's results:

$$\min_{\mathbf{N}_j} J_j^f(\hat{\mathbf{Y}}^\star, \hat{\mathbf{Z}}^\star, \hat{\mathbf{U}}^{\mathbf{y}^\star}, \boldsymbol{\zeta}_j, \mathbf{N}_j) \tag{2.31a}$$

$$\text{s.t.} \quad \boldsymbol{\zeta}_{k+1} = \boldsymbol{\zeta}_k + \frac{1}{\varepsilon}\Delta t_f \hat{\mathbf{g}}(\boldsymbol{\zeta}_j, \boldsymbol{\nu}_j), \tag{2.31b}$$

where

$$\begin{array}{rcl} \hat{\mathbf{Y}}^\star & \equiv & \{\hat{\mathbf{y}}_j^\star, \hat{\mathbf{y}}_{j+1}^\star, \dots, \hat{\mathbf{y}}_{j+N_f-1}^\star\} \\ \hat{\mathbf{Z}}^\star & \equiv & \{\hat{\mathbf{z}}_j^\star, \hat{\mathbf{z}}_{j+1}^\star, \dots, \hat{\mathbf{z}}_{j+N_f-1}^\star\} \\ \hat{\mathbf{U}}^{\mathbf{y}^\star} & \equiv & \{\hat{\mathbf{u}}_j^{\mathbf{y}^\star}, \hat{\mathbf{u}}_{j+1}^{\mathbf{y}^\star}, \dots, \hat{\mathbf{u}}_{j+N_f-2}^{\mathbf{y}^\star}\} \\ \hat{\mathbf{U}}^{\mathbf{z}^\star} & \equiv & \{\hat{\mathbf{u}}_j^{\mathbf{z}^\star}, \hat{\mathbf{u}}_{j+1}^{\mathbf{z}^\star}, \dots, \hat{\mathbf{u}}_{j+N_f-2}^{\mathbf{z}^\star}\} \\ . & & \end{array} \tag{2.32}$$

Horizon N_f is set to $\Delta t_f N_f < \Delta t_s N_s$ in our evaluations because the controller for the fast subsystem modifies the robot's short-term effect.

Finally, the control input for the fast state at time j is composed of two control variables:

$$\mathbf{u}_j^\star = \begin{bmatrix} \mathbf{u}_j^{\mathbf{y}^\star} \\ \mathbf{u}_j^{\mathbf{z}^\star} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}}_j^{\mathbf{y}^\star} \\ \hat{\mathbf{u}}_j^{\mathbf{z}^\star} + \boldsymbol{\nu}_j^\star \end{bmatrix}. \tag{2.33}$$

2.3.3 Modified Hierarchical MPC Design: Coarse-Fast MPC

If the nonlinear algebraic equations in Eq. (2.25) can be solved, the above hierarchical approach will significantly reduce MPC's computational time. However, it is often difficult to solve the equations analytically or even numerically for a humanoid system due to high nonlinearity. As a result, the slow dynamics can't correctly predict the slow state. Therefore, the MPC methods fail to compute optimal control sequences. Thus, we cannot straightforwardly apply the above slow-fast MPC approach. Instead, in this study, we handle the first problem as a coarse optimization given objective function J^c and the original system dynamics:

$$\min_{\tilde{\mathbf{U}}_k} J_k^c(\tilde{\mathbf{x}}_k, \tilde{\mathbf{U}}_k) \tag{2.34a}$$

$$\text{s.t.}\quad \tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \Delta t_c \mathbf{f}_d(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_k), \tag{2.34b}$$

where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{U}}_k \equiv \{\tilde{\mathbf{u}}_k, \tilde{\mathbf{u}}_{k+1}, \dots, \tilde{\mathbf{u}}_{k+N-2}\}$ denote a state and a control sequence for the coarse optimization. Δt_c is a coarse-time-step ($\Delta t_c > \Delta t$). Since the optimal controller corresponds to a result that solves the original optimal control problem of Eq. (2.3) with coarse-time-step Δt_c and horizon N_c, in this chapter we call an MPC with this optimal control problem a Coarse MPC and an MPC that solves the original problem a Fine MPC.

The lower-layer optimization is the same as Eq. (2.31). For fast optimization, we first convert the optimal state and the input trajectories derived in coarse optimization. We transform the optimal trajectories described in the original state and the action space into those described in slow and fast state-action space: from the trajectories of $\tilde{\mathbf{x}}^\star$ and $\tilde{\mathbf{u}}^\star$ into the trajectories of $\tilde{\mathbf{y}}^\star$, $\tilde{\mathbf{z}}^\star$, $\tilde{\mathbf{u}}^{\mathbf{y}^\star}$ and $\tilde{\mathbf{u}}^{\mathbf{z}^\star}$. Then we use a zero-order hold model to keep the optimized states and the control inputs for $M = \Delta t_c/\Delta t_f$ time-steps to match the optimal trajectories with the time-scale of the fast subsystem:

$$\begin{array}{rcl}
[\hat{\mathbf{y}}^\star_j, \hat{\mathbf{y}}^\star_{j+1}, \dots, \hat{\mathbf{y}}^\star_{j+M-1}] & \leftarrow & [\tilde{\mathbf{y}}^\star_k, \tilde{\mathbf{y}}^\star_k, \dots, \tilde{\mathbf{y}}^\star_k] \\
[\hat{\mathbf{z}}^\star_j, \hat{\mathbf{z}}^\star_{j+1}, \dots, \hat{\mathbf{z}}^\star_{j+M-1}] & \leftarrow & [\tilde{\mathbf{z}}^\star_k, \tilde{\mathbf{z}}^\star_k, \dots, \tilde{\mathbf{z}}^\star_k] \\
[\hat{\mathbf{u}}^{\mathbf{y}^\star}_j, \hat{\mathbf{u}}^{\mathbf{y}^\star}_{j+1}, \dots, \hat{\mathbf{u}}^{\mathbf{y}^\star}_{j+M-1}] & \leftarrow & [\tilde{\mathbf{u}}^{\mathbf{y}^\star}_k, \tilde{\mathbf{u}}^{\mathbf{y}^\star}_k, \dots, \tilde{\mathbf{u}}^{\mathbf{y}^\star}_k] \\
[\hat{\mathbf{u}}^{\mathbf{z}^\star}_j, \hat{\mathbf{u}}^{\mathbf{z}^\star}_{j+1}, \dots, \hat{\mathbf{u}}^{\mathbf{z}^\star}_{j+M-1}] & \leftarrow & [\tilde{\mathbf{u}}^{\mathbf{z}^\star}_k, \tilde{\mathbf{u}}^{\mathbf{z}^\star}_k, \dots, \tilde{\mathbf{u}}^{\mathbf{z}^\star}_k],
\end{array} \tag{2.35}$$

where subscript $j = (k-1)M + 1$ indicates the time index of the fine-time resolution and k denotes the time index of the coarse-time resolution.

Using the optimal inputs for upper-layer problems $\tilde{\mathbf{u}}^{\mathbf{y}^\star}$ and $\tilde{\mathbf{u}}^{\mathbf{z}^\star}$, the control input for the fast state at time j is composed of two control variables:

$$\mathbf{u}^\star_j = \begin{bmatrix} \mathbf{u}^{\mathbf{y}^\star}_j \\ \mathbf{u}^{\mathbf{z}^\star}_j \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{u}}^{\mathbf{y}^\star}_j \\ \tilde{\mathbf{u}}^{\mathbf{z}^\star}_j + \boldsymbol{\nu}^\star_j \end{bmatrix}. \tag{2.36}$$

Figure 2.1 shows a schematic diagram of the proposed method. Since the original optimal control problem (original problem in Fig. 2.1) is computationally intensive to solve in MPC, we decompose it into two optimal control problems, as shown in Eqs. (2.34) and (2.31). After solving one optimal control problem in Eq. (2.34) with coarse optimization, the results are modified by solving the other problem, Eq. (2.31).

2.3.4 Input Initialization for Warm-Starting

Warm-starting the optimizer greatly reduces the computational time since optimization starts with a good initial point based on a previously computed trajectory. However, the technique with the time-shifted solutions described in Section 2.2 works from the next optimization procedure to find optimal inputs $\mathbf{U}_2^\star$ after finding initial optimal input sequence $\mathbf{U}_1^\star$. Therefore we must give the optimizer a good initial trajectory $\mathbf{U}_1^{init}$ to warm-start from the beginning.

Combining offline and online optimization methods achieves either fast computation or good performances since online optimization iteratively improves the pre-computed approximate optimal results [43, 44]. If MPC is warm-started from offline computations at every time-step, each optimization process will probably converge after very few iterations. However, we cannot straightforwardly combine both methods because of the high-dimensionality of a humanoid robot since storing all the results of the offline optimizations for many tasks is no longer practically possible.

Thus we adopt a simple input initialization method that combines both optimizations; the initial input sequence is optimized offline, and then the online trajectory optimization starts with the optimal initial inputs to warm-start the solver from the beginning. The initialization is especially quickened to compute initial optimal input sequence $\mathbf{U}_1^\star$. From the next time-step, the time-shifted trajectory is used as the optimization's initial points.

Figure 2.2 explains the warm-start technique. To warm-start the online computation from the beginning, the initial input sequence is optimized offline. In online optimization, after solving one optimal control problem in Eq. (2.34) with the coarse-time resolution in the upper layer, the results are modified by solving the other problem in Eq. (2.31) in the lower layer. Algorithm 1 summarizes our proposed method.

2.4 Numerical Example

In this section, we consider a regulator problem for a nonlinear singularly perturbed system to explore our approach and present its experimental settings in simulations and results. We apply three methods, the coarse MPC, the fine MPC,

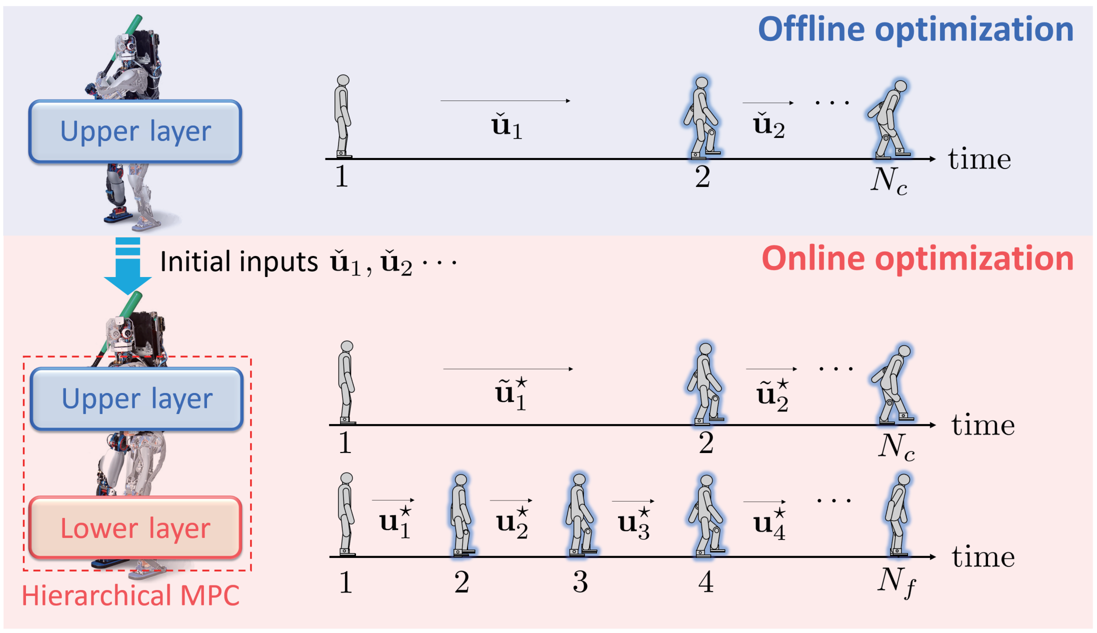


Figure 2.2: Simple warm-start technique: an input initialization approach: Only initial inputs are optimized offline to warm-start solver from beginning of online computation. In online optimization, decomposed problems are solved in hierarchical manner to derive robot's optimal policies.

Algorithm 1 Warm-Starting Coarse-Fast MPC.

```
1: Input initialization: Initialize $\tilde{\mathbf{U}}_1^{init} \leftarrow \arg\min_{\tilde{\mathbf{U}}_1} J_1^c$.
2: for $k = 1$ to $T$ (terminating time-step of experiment) do
3:     Coarse optimization: Eq. (2.34) with $\tilde{\mathbf{U}}_k^{init}$.
4:     Time shift to generate $\tilde{\mathbf{U}}_{k+1}^{init}$ from $\tilde{\mathbf{U}}_k^\star$.
5:     Fast optimization: Eq. (2.31) with $\mathbf{N}_k^{init}$ initialized all elements to be 0.
6:     Compose new inputs: $\mathbf{U}_k^\star$ with Eq. (2.36).
7:     Control robot with first few inputs of $\mathbf{U}_k^\star$
8: end for
```

Table 2.1: Time-steps and horizons used in numerical example.

	Step size (ms)	Horizon
Fine	$\Delta t = 1$	$N = 400$
Coarse	$\Delta t = 10$	$N = 40$
Coarse-fast	$\Delta t_c = 10$, $\Delta t_f = 1$	$N_c = 40$, $N_f = 30$
Slow-fast	$\Delta t_c = 10$, $\Delta t_f = 1$	$N_s = 40$, $N_f = 30$

and the slow-fast MPC in addition to the proposed coarse-fast MPC, and compare the state transitions of the slow and fast states. We demonstrate that the proposed method can handle singularly perturbed systems in which the equilibrium points of the fast dynamics do not uniquely exist or whether they are just difficult to analytically derive.

2.4.1 Experimental Settings

We consider the following two-dimensional nonlinear singularly perturbed system:

$$\begin{aligned} \dot{y} &= -y + z - u_1 - u_2 \\ \varepsilon \dot{z} &= -yz - \cos z - u_1 + u_2 \quad , \end{aligned} \tag{2.37}$$

where $\varepsilon = 0.1$. In this system, depending on the state and control variables, the plant does not have a unique root (Eq. (2.26)) (e.g., if $y = 0$, $\hat{z} = \cos^{-1}(-u_1 - u_2)$). Or a unique root of the fast dynamics exists, but a numerical method such as Newton's method is necessary to obtain it (e.g., if $y = 1.0, u_1 = 0, u_2 = 0$). This causes an incorrect prediction of the slow state with a slow subsystem.

By defining a state vector as $\mathbf{x} = [y, z]^\top$ and an input vector as $\mathbf{u} = [u_1, u_2]^\top$, we consider a regulator problem in which the following total cost is minimized in each time-step:

$$J_k = \sum_{i=k}^{k+N} (\mathbf{x}_i - \mathbf{x}^d)^\top Q (\mathbf{x}_i - \mathbf{x}^d) + \sum_{i=k}^{k+N-1} \mathbf{u}_i^\top R \mathbf{u}_i, \tag{2.38}$$

where $\mathbf{x}^d$ is the desired state and Q and R are the weighting matrices. We set $\mathbf{x}^d = [y^d, z^d]^\top = [1.0, 2.0]^\top$, $Q = \mathrm{diag}(1.0, 1.0)$, and $R = \mathrm{diag}(0.01, 0.01)$ in the experiment.

The total cost for the optimal control problem in Eq. (2.34) is given by:

$$J_k^c = \sum_{i=k}^{k+N_c} (\mathbf{x}_i - \mathbf{x}^d)^\top Q(\mathbf{x}_i - \mathbf{x}^d) + \sum_{i=k}^{k+N_c-1} \mathbf{u}_i^\top R\mathbf{u}_i. \tag{2.39}$$

This optimal control problem is the same as the original one except for coarse-time-step Δt_c and horizon N_c. To modify the derived optimal control sequence, we determined the total cost for the second optimization at time-step k as follows:

$$J_k^f = \sum_{i=k}^{k+N_f} (\mathbf{x}_i - \mathbf{x}^d)^\top Q(\mathbf{x}_i - \mathbf{x}^d) + \sum_{i=k}^{k+N_{2nd}-1} \boldsymbol{\nu}_i^\top R_\nu \boldsymbol{\nu}_i. \tag{2.40}$$

Here $\boldsymbol{\nu} = \mathbf{u} - \hat{\mathbf{u}}^\star$, and $\hat{\mathbf{u}}^\star$ denotes the optimal control input computed through the first optimization. We set weighting matrix $R_\nu = R = \mathbf{diag}(0.01, 0.01)$.

We compared the result of our approach with three methods: the fine MPC, the coarse MPC, and the slow-fast MPC described in Section 2.3. In the slow-fast MPC, we used identical total cost $J^s = J^c$ for the first optimization in Eq. (2.30). Table 2.1 shows the time-steps and the horizon used in each method. Five simulations were carried out in each one. We measured the series of the computational times required for deriving each optimal control sequence in a simulation and computed the maximum computational times in each execution. We then averaged the values. All simulations were performed by a Core i7-4770, 3.4 GHz computer. The simulation duration was 0.4 s, and the sampling time was 1.0 ms in all the methods. Regarding the maximum number of iterations and termination criteria for iLQR, the values used in all the methods were 50 and 1.0×10^{-6}.

2.4.2 Results

Figures 2.3 and 2.4 respectively show the state transitions of slow state y and fast state z. The state transition result obtained by the fine MPC solves the original optimal control problem. As shown in Fig. 2.3, the slow state transition acquired with the coarse MPC and the coarse-fast MPC are similar to the fine MPC result. By contrast, as shown in Fig. 2.4, the fast state transition obtained by the slow-fast MPC and the coarse-fast MPC resemble the fine MPC result. Since the coarse optimization only considers the optimal control problem in Eq.

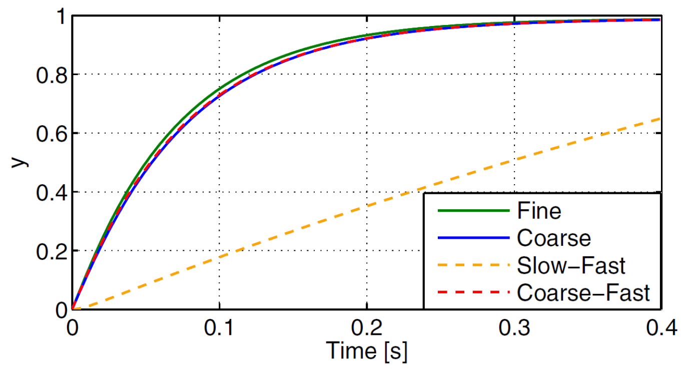


Figure 2.3: State transitions of slow state y: Since slow subsystem fails to correctly predict slow state trajectories, slow-fast MPC can't generate an optimal controller for the slow state, which does not converge to desired state. State trajectory achieved with coarse-fast MPC resembles the fine MPC result.

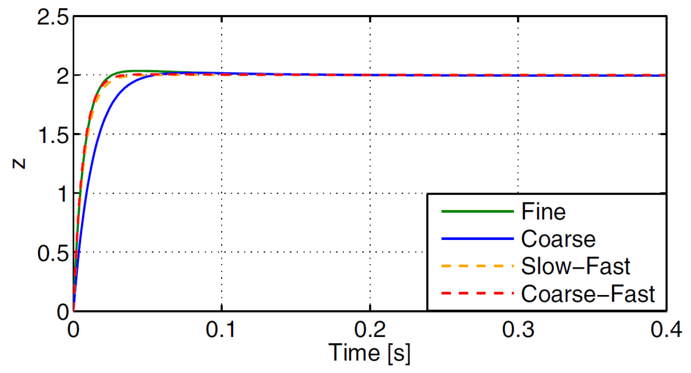


Figure 2.4: State transitions of fast state z: Since a course time-step is only used in coarse MPC, derived controller can't be sufficiently optimized for fast state. Coarse-fast MPC achieved similar state trajectory to fine MPC result.

(2.34), the derived controller can't be optimized enough for the fast state. On the other hand, the equilibrium points of the fast dynamics cannot be derived in a particular state or from control variables in the plant, and the slow subsystem fails to predict the correct slow state trajectories. Thus, the slow-fast MPC can't generate an optimal controller for the slow state, which does not converge to the desired state. In the proposed method, since the slow dynamics optimization is approximated with coarse optimization and the controller is refined with the second optimization, good control performance is achieved that resembles a fine MPC.

Figure 2.5 shows the maximum computational time required for computing an optimal control sequence in one time-step in the fine and coarse-fast MPCs. Each value is the average of five simulations. The error bars represent the standard deviations. The black dashed line in Fig. 2.5 represents the sampling time used in this experiment. Since the sampling time is 1.0 ms, the optimization must be finished within 1.0 ms for real-time implementation. As shown in Fig. 2.5, the computational time of the coarse-fast MPC is shorter than the sampling time. The proposed method uses a large time-step in the first optimization and reduces the number of control variables optimized in the first step. Moreover, since the fast state is evaluated with a short period in the second optimization, just a little extra computational time is necessary. Therefore, only our proposed method can achieve real-time optimal control without degrading the control performance.

2.5 Application to a Humanoid Model

In this section, our approach is applied to a humanoid robot model to demonstrate that versatile movements can be generated in real-time with our proposed method.

2.5.1 System Dynamics

We consider a seven-link humanoid robot with a torso link and three links in each leg. Its height and total weight are 1.59 m and 47.0 kg. The dynamics of the robot's movements in three-dimensional space are given by the following

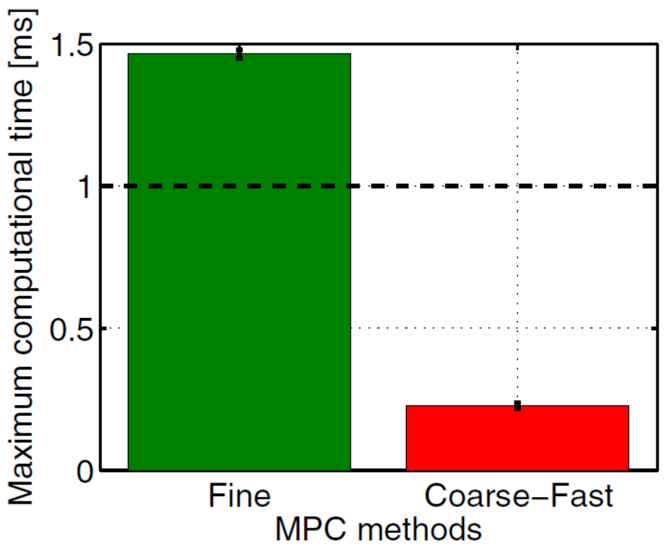


Figure 2.5: Maximum computational time required to derive an optimal control sequence in each method: Since computational time of coarse-fast MPC is shorter than sampling time (dashed line), our approach achieves real-time optimal control.

equations of motion:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{c}(\mathbf{q}, \mathbf{v}) + \mathbf{u} \tag{2.41a}$$

$$\dot{\mathbf{q}} = \mathbf{v}, \tag{2.41b}$$

where $\mathbf{M}$ is the inertia matrix and $\mathbf{c}$ is the vector of such total external forces as gravity and coriolis. $\mathbf{q}$ is the generalized positions (base link positions, attitude, and joint angles), $\mathbf{v}$ is their velocities, and $\mathbf{u}$ is the applied control. State $\mathbf{x}$ of the state-space model consists of the following positions and velocities: $\mathbf{x} = [\mathbf{q}^\top, \mathbf{v}^\top]^\top$.

In our simulator, we compute inertia matrix $\mathbf{M}$ and a part of external forces $\mathbf{c}$ using the Composite Rigid Body and Recursive Newton Euler algorithms [45, 46]. Moreover, a smooth contact model and its solver [11, 47] are used to compute the contact forces. We adopted those algorithms for their computational efficiency.

2.5.2 Transformation into a Singularly Perturbed System and Fast Dynamics Extraction

To apply our proposed method to a humanoid robot, we must extract the fast dynamics from the whole-body dynamics. This can be achieved by transforming the humanoid robot into a singularly perturbed system. We introduce a new singularly perturbed system of a humanoid robot, where we decomposed the inertia matrix relative to base link $\mathbf{M}_{\text{base}}$ and the other $\mathbf{M}_{\text{leg}}$:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & O \\ O & O \end{bmatrix} + \begin{bmatrix} O & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} = \mathbf{M}_{\text{base}} + \mathbf{M}_{\text{leg}}. \tag{2.42}$$

We can assume that matrix $\mathbf{M}_{\text{base}}$ is only related to the accelerations of the base link or to both its linear and angular accelerations. If the former assumption is adopted, $\mathbf{M}_{11}$ equals a diagonal matrix of the total mass. The total weight is much bigger than each element of $\mathbf{M}_{\text{leg}}$. Thus, inertia $\mathbf{M}_{\text{leg}}$ can be clearly considered a perturbation matrix of $\mathbf{M}_{\text{base}}$. In this study, however, we adopt the latter assumption. This eases the computational burden of the proposed method since fast dynamics has lower dimensionality. Since the elements of new inertia matrix $\mathbf{M}_{\text{base}}$ may include all the mass parameters, the matrix becomes relatively large. Thus, with a positive small parameter ε, the equations of motion in Eq. (2.41a) become

$$(\mathbf{M}_{\text{base}} + \varepsilon\mathbf{Q})\dot{\mathbf{v}} = \mathbf{c} + \mathbf{u}, \tag{2.43}$$

where $\mathbf{Q} = \mathbf{M}_{\text{leg}}/\varepsilon$ and ε is defined as $||\mathbf{M}_{\text{leg}}||/||\mathbf{M}_{\text{base}}||$. We obtain the following singularly perturbed system by multiplying a matrix $(\mathbf{I} - \varepsilon\mathbf{S})$ on both sides of Eq. (2.43):

$$\begin{bmatrix} \mathbf{M}_{11} & \varepsilon\mathbf{Q}_{12} \\ O & \varepsilon\mathbf{Q}_{22} - \varepsilon^2\mathbf{Q}_{21}\mathbf{M}_{11}^{-1}\mathbf{Q}_{12} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}}_{\text{slow}} \\ \dot{\mathbf{v}}_{\text{fast}} \end{bmatrix} = (\mathbf{I} - \varepsilon\mathbf{S})(\mathbf{c} + \mathbf{u}), \tag{2.44}$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{S}$ is the following matrix:

$$\mathbf{S} = \begin{bmatrix} O & O \\ \mathbf{Q}_{21}\mathbf{M}_{11}^{-1} & O, \end{bmatrix} \tag{2.45}$$

and matrix $\mathbf{Q}$ is partitioned into four blocks:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix}. \tag{2.46}$$

Let velocities $\mathbf{v} = [\mathbf{v}_{\text{base}}^\top, \mathbf{v}_{\text{leg}}^\top]^\top$, where slow $\mathbf{v}_{\text{slow}}$ and fast $\mathbf{v}_{\text{fast}}$ variables are represented as $[\mathbf{v}_{\text{slow}}, \mathbf{v}_{\text{fast}}]^\top = [\mathbf{v}_{\text{base}}, \mathbf{v}_{\text{leg}}]^\top$. In addition to the velocities, we also assume that their generalized positions are slow and fast states. Thus, slow variables $\mathbf{y}$ and fast variables $\mathbf{z}$ are denoted as:

$$\mathbf{y} = \begin{bmatrix} \mathbf{q}_{\text{slow}} \\ \mathbf{v}_{\text{slow}} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{\text{base}} \\ \mathbf{v}_{\text{base}} \end{bmatrix}, \ \mathbf{z} = \begin{bmatrix} \mathbf{q}_{\text{fast}} \\ \mathbf{v}_{\text{fast}} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{\text{leg}} \\ \mathbf{v}_{\text{leg}} \end{bmatrix}. \tag{2.47}$$

Now we extract a fast subsystem from the singularly perturbed system. In our proposed method, we use the subsystem to modify the results of the upper-layer optimization. Thus we consider the dynamics of the deviation between the fast state and the optimal fast state derived from upper-layer optimization and define the deviation as follows:

$$\boldsymbol{\zeta} = \begin{bmatrix} \zeta_q \\ \zeta_v \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{\text{fast}} - \hat{\mathbf{q}}_{\text{fast}}^\star \\ \mathbf{v}_{\text{fast}} - \hat{\mathbf{v}}_{\text{fast}}^\star \end{bmatrix}. \tag{2.48}$$

We partition $(\mathbf{c}+\mathbf{u})$ on the right side of Eq. (2.44) into slow variables $\mathbf{p}_y$ and fast variables $\mathbf{p}_z$:

$$\mathbf{c}+\mathbf{u}=\mathbf{p}=\begin{bmatrix}\mathbf{p}_y\\ \mathbf{p}_z\end{bmatrix}. \tag{2.49}$$

After introducing time-scale $\sqrt{\varepsilon}\tau = t$ and rewriting Eq. (2.44) with setting $\varepsilon \to 0$, we obtain a fast subsystem that is described in new time-scale τ and restore the fast subsystem to original time-scale t. Finally, the subsystem takes the following form:

$$\dot{\mathbf{v}}_{\text{slow}} = \mathbf{0} \tag{2.50a}$$

$$\varepsilon \mathbf{Q}_{22}\dot{\boldsymbol{\zeta}}_v = \mathbf{p}_z. \tag{2.50b}$$

Equation (2.50) describes the dynamics of the fast states under fixed slow variables. Since control inputs have no effect on slow states, the following discretized system can be obtained:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \Delta t_f \hat{\mathbf{h}}(\mathbf{y}_k) \tag{2.51a}$$

$$\tag{2.51b}$$

$$\boldsymbol{\zeta}_{k+1} = \boldsymbol{\zeta}_k + \frac{1}{\varepsilon}\Delta t_f \hat{\mathbf{g}}(\mathbf{y}_k, \boldsymbol{\zeta}_k, \boldsymbol{\nu}_k), \tag{2.51c}$$

where the time evolution of the slow states does not depend on the control inputs. Therefore, in the lower-layer optimization of Eq. (2.31), the reduced-order subsystem of Eq. (2.51c) is the only constraints that are considered.

In an original work that derived a singularly perturbed system of a biped robot [48], inertia matrix $\mathbf{M}$ was decomposed into the inertia matrix of the center-of-mass (CoM) and another based on the fact that the total kinetic energy of rigid body dynamics can be separated into the kinetic energy associated with the motion of CoM and the motion relative to it. In this study for faster computation, our system was derived based on an assumption that the states of the base link approximate the motion of CoM. Note that this approximation is often used in biped robot control (e.g., [18]).

2.5.3 Computational Complexity

In one iteration of a gradient-based optimization algorithm such as iLQR, the most computationally intensive part is to compute the derivatives of the dynamics. Deriving analytical differentiations is difficult because humanoid dynamics are too complex. We approximate them using finite-differencing, which requires that the dynamics be evaluated many times. The computational complexity of finite-differencing is $O(n)$ where n is the dimension of the state. Therefore, the total complexity is $O(Nn^4)$ because the derivatives are evaluated along a given trajectory (horizon N) and a single call to the dynamics function is $O(n^3)$ [49]. In practice, we can efficiently compute the dynamics using the branch-induced sparsity of inertia matrix $\mathbf{M}$. However, dynamics computation remains expensive.

In the proposed method, we solved the original optimal control problem of Eq. (2.3) with coarse-time-step Δt_c (horizon N_c) in the first optimization; the complexity is $O(N_c n^4) < O(Nn^4)$. In the second optimization, the problem in Eq. (2.31) is solved with low-dimensional dynamics: $O(N_f n_f^4) < O(Nn^4)$. Therefore, since two smaller problems are solved in the proposed method, the computational time is reduced.

2.6 Simulation Experiment

2.6.1 Experimental Task and Cost Function Design

In the first experiment, the task was to generate eight movements: standing (I), walking (II), running (III), sitting on a chair and standing (IV), low jumping (V), jumping with a half turn (VI), high jumping (VII), and doing a double flip (VIII). For versatile motion generation, we chose state and control cost functions and adopted weighted quadratic functions for the state-cost and control-cost terms.

In upper-layer optimization, the state-cost terms have parameters for attitude $\mathbf{a} \equiv \{\phi, \theta, \xi\}$, height p_z, and velocity $\mathbf{v} \equiv \{v_x, v_y, v_z\}$ of the base link. Various movements were generated by setting these parameters. We defined the XZ-plane as the sagittal plane, the YZ-plane as the frontal plane, and ϕ, θ, and ξ respectively as the roll, pitch, and yaw rotations. The roll, pitch, and yaw were counterclockwise rotations about the x-, y-, and z-axes.

We provide the form of each cost term in Chapter 4 (Section 4.3.1). Briefly, the state-cost consists of six terms. The first term forced the robot to regulate the attitude of the base link to $\mathbf{a}$ rad. If the torso link stands upright, each angle is 0 rad. In practice, rotations $\mathbf{a}$ were transformed into quaternions to compute the cost since we designed this term using unit quaternions. The second term penalized the vertical position of the base to p_z m. The third term forced the vertical and horizontal velocity of the base link to be $\mathbf{v}$ m/s. The fourth and fifth terms were penalties on the angles between the vertical axis and a line connecting each foot and the base. The fourth term forced the robot's legs to be spread apart. The fifth term prevented self-collision. The sixth term was the quadratic penalties on the angular velocities of the base link's attitude. The control-cost was the quadratic penalties on the control inputs.

Table 2.2 shows the parameter settings: I: standing, II: walking, and III: running. In cases I, II, and III, the parameters were the same except for forward velocity v_x.

Table 2.3 shows the settings: IV: sitting on a chair and standing, V: low jumping, VI: jumping with a half turn, VII: high jumping, and VIII: doing a double flip. We changed some parameters in each of the four or six phases. Specifically, for case IV, the first and second phases were for sitting on a chair and the others were for standing on it. For cases V to VII, the first phase corresponded to the preparations before initiating the jumping and flipping motions. The target motions were executed in the second phase. We designed a third phase for the landing motions so that the robot keeps it balance after the jumping and flipping motions. The fourth phase was for standing upright. In case VIII, the target motion was a double flip. Hence, the phases for motion execution and landing were repeated from the second phase. The last phase was for standing upright.

In the lower-layer optimization, the state-cost was a weighted quadratic function to minimize the deviation between the fast state and the optimal fast state derived from the first optimization:

$$\ell_{fast}^{state} = w_{fast}\boldsymbol{\zeta}^\top \boldsymbol{\zeta}, \tag{2.52}$$

where w_{fast} is a weight parameter. The control-cost was the quadratic penalty on the deviation of control inputs as with the state-cost. Since an optimal trajectory is derived with a coarse-time-step in the first optimization, the trajectory must

Table 2.2: Parameter settings in state-cost terms for generating motions I to III.

	Attitude $\mathbf{a}$ (ϕ, θ, ξ) (rad)	Height p_z (m)	Velocity $\mathbf{v}$ (v_x, v_y, v_z) (m/s)
I	0, 0, 0	0.85	0, 0, 0
II	0, 0, 0	0.85	1.2, 0, 0
III	0, 0, 0	0.85	2.0, 0, 0

be modified for more fine-time resolution. Thus, we designed cost functions to find a new trajectory for the fine time-step around the trajectory derived in the first optimization.

In the second experiment, the task generated a whole-body reaching motion. In addition to the cost functions for versatile motion generation, we designed a state-cost for reaching, which is a penalty on the difference between target reaching position p_{reach} and robot's hand position p_{hand}:

$$\ell_{reach} = w_{reach}(p_{reach} - p_{hand})^\top (p_{reach} - p_{hand}), \tag{2.53}$$

where w_{reach} is the weight parameter. The target position for the reaching motion is $p_{reach} = [0.70, -0.12, 0.70]^\top$. We set attitude $\mathbf{a} = [0, 0, 0]^\top$, height $p_z = 0.82$, and velocity $\mathbf{v} = [0, 0, 0]^\top$ for the other state-cost functions.

2.6.2 Simulation Settings

Our proposed method was applied to a humanoid robot to generate various movements such as running, flipping, and whole-body reaching motions. We set the sampling time to 30 ms. For real-time motion generation, an optimization at each time-step must be finished within the sampling time. Since we set the time-step of the simulator to 1.0 ms, modeling error occurs. All simulations were performed in a C language programming environment by an Intel Xeon Processor E5-2697 v3, 2.6 GHz computer. In iLQR, derivatives of dynamics and costs were computed in parallel with ten computer threads using Open Multi-Processing (OpenMP) [50].

In the first experiment, we show the effectiveness of the optimization of the initial inputs by measuring the maximum computational times with and without the

Table 2.3: Parameter settings in state-cost terms for generating motions IV to VIII.

		Phase 1	Phase 2	Phase 3	Phase 4
IV	Attitude $\mathbf{a}$ (ϕ, θ, ξ) (rad)	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Height p_z (m)	0.65	0.65	0.85	0.85
	Velocity $\mathbf{v}$ (v_x, v_y, v_z) (m/s)	-1.5, 0, 0	0, 0, 0	1.5, 0, 0	0, 0, 0
		Phase 1	Phase 2	Phase 3	Phase 4
V	Attitude $\mathbf{a}$ (ϕ, θ, ξ) (rad)	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Height p_z (m)	0.85	1.8	0.75	0.85
	Velocity $\mathbf{v}$ (v_x, v_y, v_z) (m/s)	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
		Phase 1	Phase 2	Phase 3	Phase 4
VI	Attitude $\mathbf{a}$ (ϕ, θ, ξ) (rad)	0, 0, 0	0, 0, π	0, 0, π	0, 0, π
	Height p_z (m)	0.85	1.8	0.75	0.85
	Velocity $\mathbf{v}$ (v_x, v_y, v_z) (m/s)	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
		Phase 1	Phase 2	Phase 3	Phase 4
VII	Attitude $\mathbf{a}$ (ϕ, θ, ξ) (rad)	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Height p_z (m)	0.0	1.9	0.75	0.85
	Velocity $\mathbf{v}$ (v_x, v_y, v_z) (m/s)	2.0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
		Phase 1	Phase 2, 4	Phase 3, 5	Phase 6
VIII	Attitude $\mathbf{a}$ (ϕ, θ, ξ) (rad)	0, 0, 0	0, 1.5π, 0	0, 2.0π, 0	0, 2.0π, 0
	Height p_z	0.0	1.9	0.75	0.85
	Velocity $\mathbf{v}$ (v_x, v_y, v_z) (m/s)	2.0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0

warm-start technique. If the initialization was not executed, we set all of the initial inputs to zero. We also compared our approach's results with a conventional method in different settings: Warm-Starting Coarse MPC and Warm-Starting Fine MPC, in terms of control performance and the computational time. Since the Slow-Fast MPC failed to control the simple nonlinear singularly perturbed system in the numerical example in Section 2.4, we did not apply it to the humanoid robot in this experiment. We carried out five simulations and averaged

Table 2.4: Time-steps and horizons used in humanoid robot experiment.

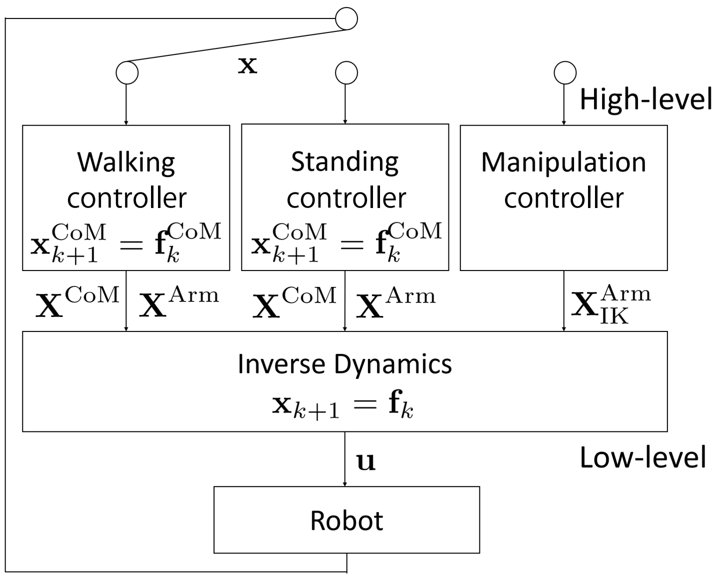
	Step size (ms)	Horizon
Fine	$\Delta t = 10$	$N = 30$
Coarse	$\Delta t = 30$	$N = 10$
Coarse-Fast	$\Delta t_c = 30$, $\Delta t_f = 10$	$N_c = 10$, $N_f = 10$

the series of maximum computational times. We applied all the methods to a humanoid robot whose state dimensionality n was 29. The robot's fast variables in Eq. 2.47 have $n_f = 16$ dimensions. Table 2.4 shows the time-step and the horizon used in each method. We decided the length so that the robot can predict enough long-term future state trajectories to generate target motions because a short horizon results in myopic behaviors. The maximum number of iteration and termination criteria for iLQR was respectively 6 and 5.0×10^{-2} in all the methods.

In the second experiment, we show that a whole-body reaching motion can be generated with our proposed approach in real-time. We compared our approach's result with a whole-body controller based on conventional controllers [51, 52] in terms of the task execution's speed. We applied the hierarchical MPC and the conventional whole-body controller to a humanoid robot with a right arm. Its state dimensionality n is 31. The state dimension of the fast variables is $n_f = 18$.

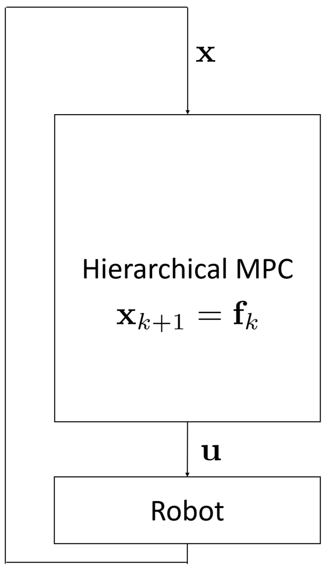
We set the time-step to $(\Delta t_c, \Delta t_f) = (30, 10)$ and the horizon to $(N_c, N_f) = (10, 10)$ for the hierarchical MPC. The maximum number of iterations and the termination criteria for iLQR was respectively 4 and 5.0×10^{-1}.

Based on conventional whole-body controllers [51, 52], we designed a whole-body controller as a conventional method. The diagrams of the conventional whole-body controller and our hierarchical MPC are represented in Fig. 2.6. For walking, given a desired center of pressure (CoP) trajectory $\mathbf{X}^{\text{CoP}}$, a desired CoM trajectory $\mathbf{X}^{\text{CoM}}$ is decided in a high-level controller (the walking controller) under CoM dynamics $\mathbf{f}^{\text{CoM}}$ using a trajectory optimization method (iLQR in our implementation). For standing, the same optimization as the walking controller is performed except that a desired CoP trajectory is designed for double support. The arm's desired angle trajectory consists of initial angle $\mathbf{X}^{\text{Arm}} = \{\mathbf{x}^{\text{Arm}}_{\text{init}}, \mathbf{x}^{\text{Arm}}_{\text{init}}, \cdots\}$



$\mathbf{x}$
High-level
Walking controller
$\mathbf{x}^{\mathrm{CoM}}_{k+1} = \mathbf{f}^{\mathrm{CoM}}_{k}$
Standing controller
$\mathbf{x}^{\mathrm{CoM}}_{k+1} = \mathbf{f}^{\mathrm{CoM}}_{k}$
Manipulation controller
$\mathbf{X}^{\mathrm{CoM}}$
$\mathbf{X}^{\mathrm{Arm}}$
$\mathbf{X}^{\mathrm{CoM}}$
$\mathbf{X}^{\mathrm{Arm}}$
$\mathbf{X}^{\mathrm{Arm}}_{\mathrm{IK}}$
Inverse Dynamics
$\mathbf{x}_{k+1} = \mathbf{f}_k$
$\mathbf{u}$
Low-level
Robot

(a) conventional whole-body controller



(b) hierarchical MPC

Figure 2.6: Control diagrams of conventional whole-body controller and hierarchical MPC: In conventional whole-body controller, high-level controllers plan behaviors under low-dimensional dynamics and kinematic models, and then robot accomplishes tasks by sequentially changing or blending the designed high-level controllers. In hierarchical MPC, full-body dynamics are used for motion planning and execution.

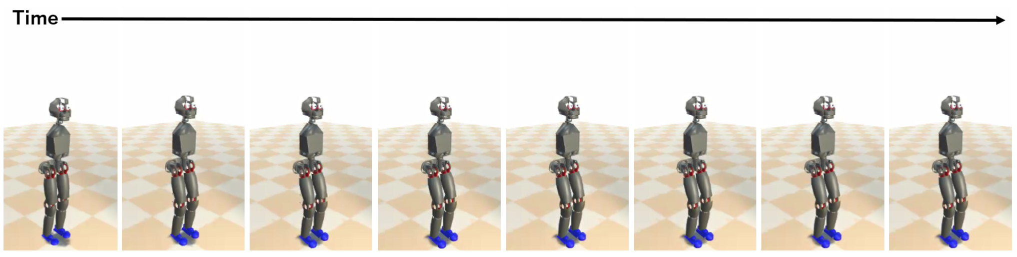
for the walking controller and the current angle for the standing controller. For the manipulation controller, the desired trajectories to reach a target position are computed using Inverse Kinematics (IK): $\mathbf{X}_{\mathrm{IK}}^{\mathrm{Arm}}$. Inverse Dynamics (ID) decides joint torques $\mathbf{u}$ under the full-body dynamics (we use a simple notation $\mathbf{f}$ for $\mathbf{f}(\mathbf{x}, \mathbf{u})$) to achieve the desired CoM and arm trajectories. Several targets and constraints such as tracking a swing leg trajectory and keeping the friction and CoP are also considered in the ID module.

2.6.3 Simulation Results

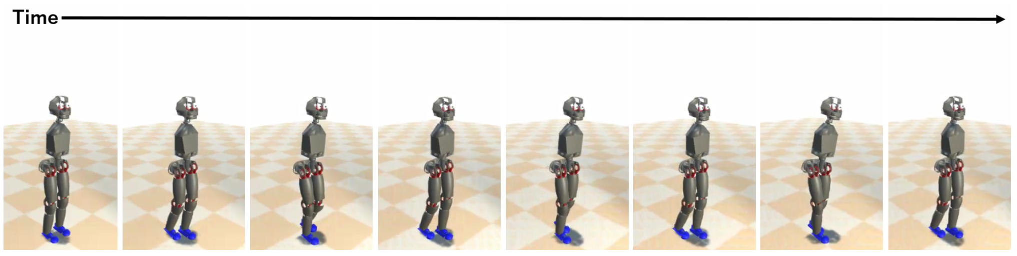
Versatile Motion Generation

Figure 2.7 shows the generated movements with the proposed method: Warm-Starting Coarse-Fast MPC with parameter settings I to IV. In cases I, II, and III, the robot tried to achieve target horizontal velocity v_x of the base link while standing the base link upright and keeping its position around 0.85 m height. Depending on the target velocity, the robot generated standing, walking, and running motions. The double swing phase accounted for 20% of the simulated duration in the running motion in Fig. 2.7c. We set the chair's height to 0.65 m and forced the robot to maintain the base's vertical position at the same height in case IV. The robot sat on the chair to keep the position. After the second phase, the robot stood up since target position p_z is higher.

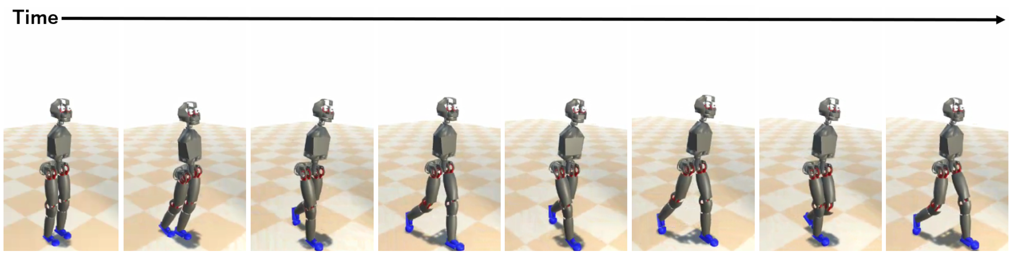
The clear advantage of using full-body dynamics in MPC is complex, and dynamic behaviors can be generated just by specifying simple high-level task descriptions. Fig. 2.8 represents the complex behaviors generated in real-time with our proposed method. Since the robot tries to raise up its torso from a standing position in case V, jumping is generated. In addition to case V, the robot was forced to half-rotate its base link (i.e., $\xi = \pi$ rad) when the parameter setting is case VI. This results in a half turn while jumping. The robot jumps higher in case VII since we set the parameters to include preparation before the jumping began. In the first phase, the robot bends its torso back, and then using the reaction force, it tried to reach a higher position than in case V. In case VIII, the robot also tried to rotate its body to generate a jumping movement (case VII). As a result, the first flip motion was generated. Since we designed



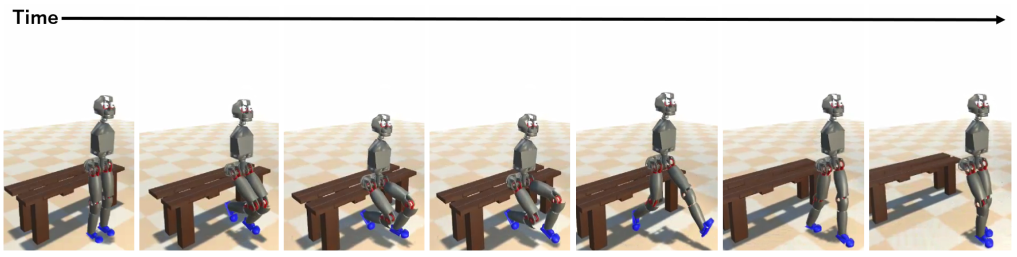
(a) standing



(b) walking

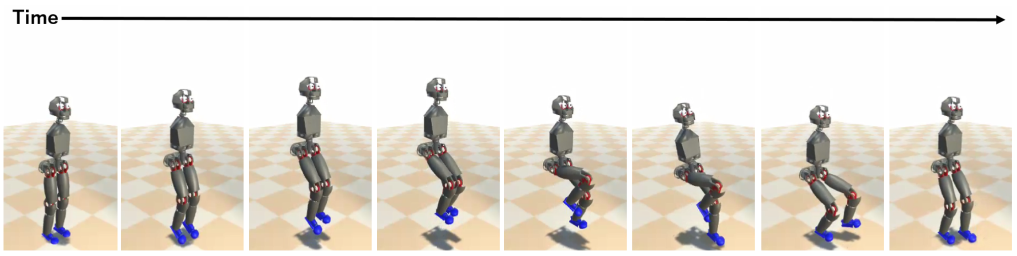


(c) running

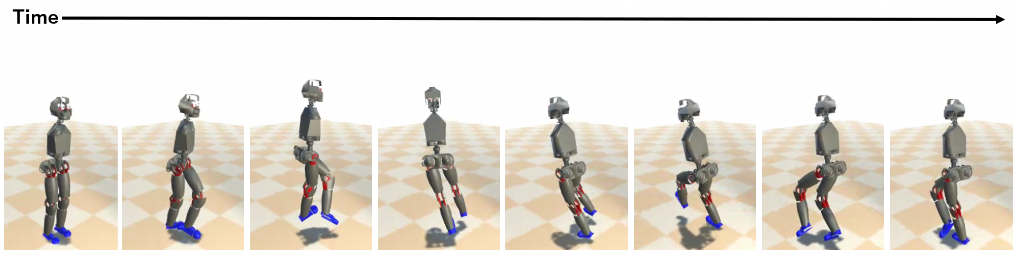


(d) sitting on a chair and standing from chair

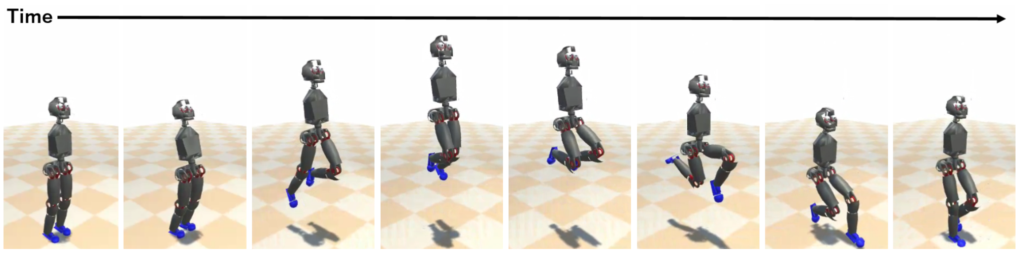
Figure 2.7: Real-time movement generation with Warm-Starting Coarse-Fast MPC in cases from I to IV: At each time-step, all motions are generated within sampling time: 30 ms.



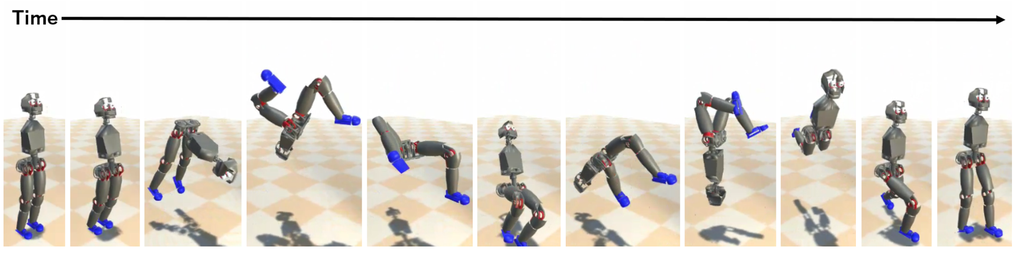
(a) low jumping



(b) jumping with half turn



(c) high jumping



(d) double flip

Figure 2.8: Real-time movement generation with our proposed method in cases V to VIII: All four motions were successfully generated in real-time.

the state-cost so that flipping and landing motions were generated again, the second flip was accomplished. Generating these motions is difficult just using the inverted pendulum model because the motions have a certain moment in which both feet are off the ground.

Figure 2.9 represents the effectiveness of the warm-start technique described in Section 3.2. If the initial inputs are not optimized (Coarse-Fast MPC was only applied to the robot), the computational time of the initial optimization exceeds the sampling time and corresponds to the maximum computational time. On the other hand, the computational time is significantly reduced when we combined the warm-start technique with Coarse-Fast MPC. As a result, the maximum computational time becomes lower than the sampling time.

We compared Warm-Starting Coarse-Fast MPC with Warm-Starting Fine MPC and Warm-Starting Coarse MPC in terms of control performances and computational times. In this case, we did not truncate the MPC computations at the sampling time. Fig. 2.10 shows the accumulated cost during each simulated duration and represents the control performances of each method. We normalized all the costs with the total cost of the Fine MPC. Thus, its costs are all one in Fig. 2.10. As examples of the generated movements with the conventional MPC approach, the generated running patterns and double flip motions with Coarse MPC and Fine MPC are shown in Fig. 2.11. In this case, the optimizations are truncated at the sampling time. The robot did not generate stable running and flipping motions with both methods.

Figure 2.12 shows the maximum computational time required for computing an optimal control sequence in one time-step. Each value is the average of five simulations. Although a computation for deriving an optimal control sequence must be finished within 30 ms, all of the maximum computational times of the Warm-Starting Fine MPC are more than twice as much as the sampling time, but no results of the Warm-Starting Coarse-Fast MPC were exceeded (Fig. 2.12). The maximum computational time averaged over eight motions of the Warm-Starting Coarse-Fast MPC was 27.0 ms. This value is about one-third as large as that of the Warm-Starting Fine MPC: 79.0 ms. As described in Section 2.5.3, MPC's computational complexity mainly depends on the state dimension: $O(Nn^4)$. Since the number of state dimensions can be reduced in the lower-layer

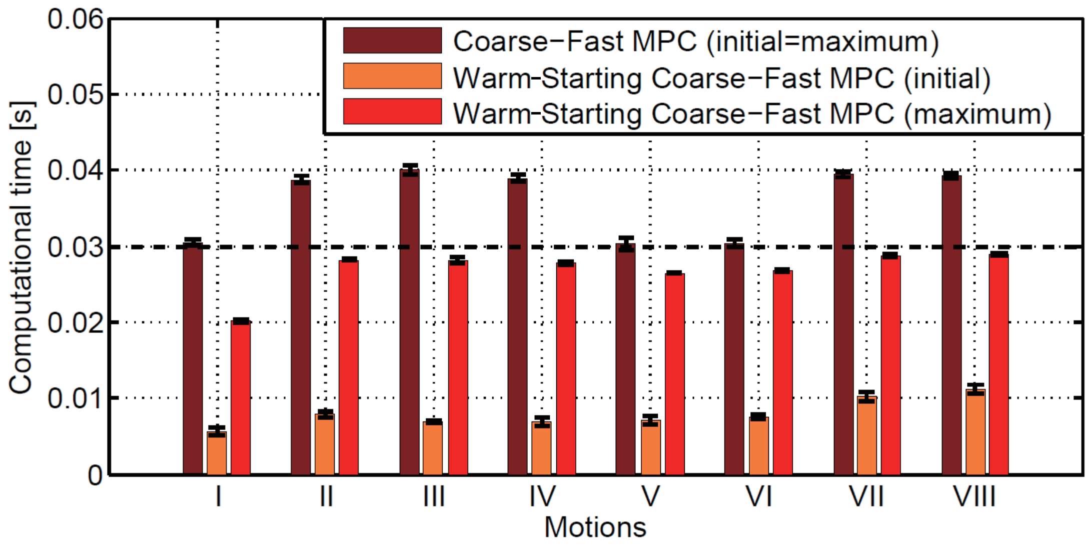


Figure 2.9: Effectiveness of input initialization: Maximum computational time exceeds sampling time (dashed line) when initial inputs are not optimized before control. After combining Warm-Start Technique with Coarse-Fast MPC, computational time of initial optimization is significantly reduced (orange bar). All motions are generated with our proposed method.

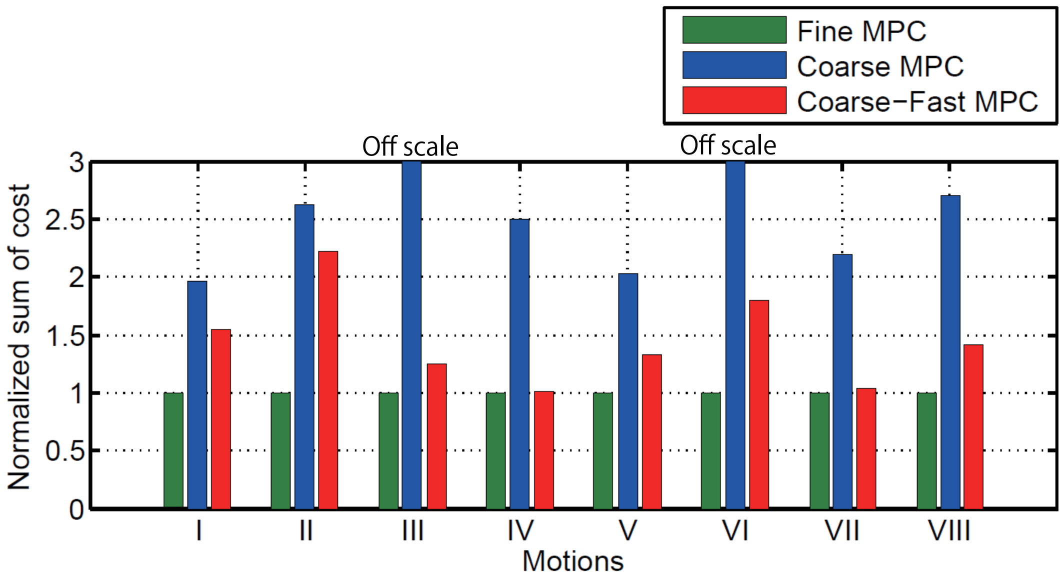
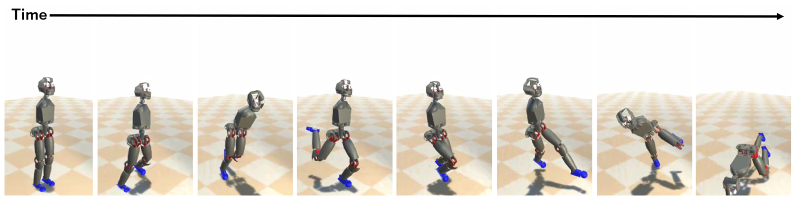
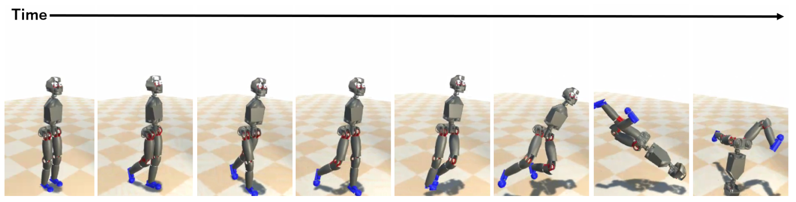


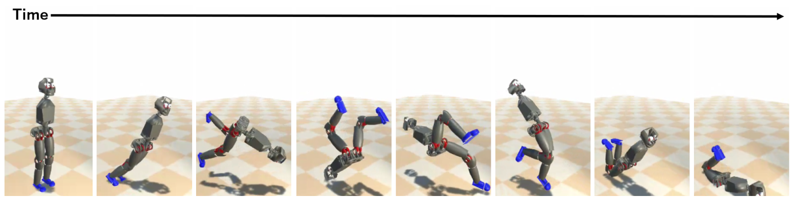
Figure 2.10: Comparison of control performances: Since normalized total costs are shown, costs of Warm-Starting Fine MPC are all one where lower value represents better performance. Each accumulated cost was computed with each generated motion trajectory. Performances of Warm-Starting Coarse MPC are the worst in all motions. Since the coarse optimization result is modified in the second optimization in our proposed method, the Warm-Starting Coarse-Fast MPC performances improved.



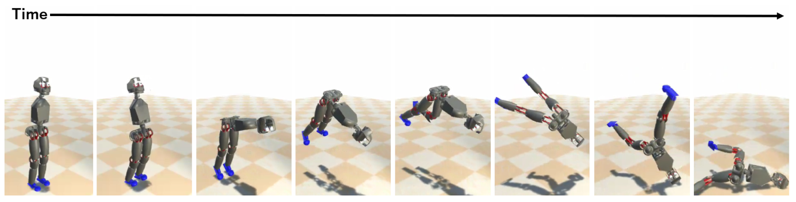
(a) generated running with Warm-Starting Fine MPC



(b) generated running with Warm-Starting Coarse MPC



(c) generated double flip motion with Warm-Starting Fine MPC



(d) generated double flip motion with Warm-Starting Coarse MPC

Figure 2.11: Real-time motion generations with Warm-Starting Fine and Warm-Starting Coarse MPCs: If we truncate computational times at sampling time, running and flipping motions are not generated with Fine MPC (e.g., figures (a) and (c)) because there is not enough time to compute optimal motions. Although two motions are not generated with just the coarse optimization (e.g., figures (b) and (d)), by refining them with the second optimization, they are successfully generated with proposed method.

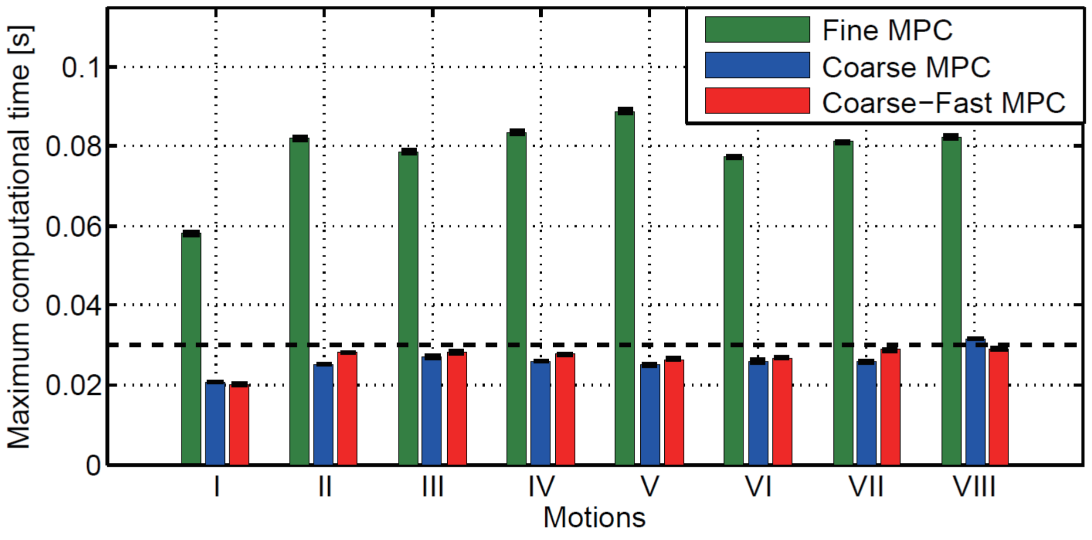


Figure 2.12: Maximum computational time required to derive optimal control sequence in each method: Maximum computational time of Warm-Starting Fine MPC is more than twice sampling time (dashed line), but no Warm-Starting Coarse-Fast MPC results are over the sampling time. For real-time robot motion generation with proposed method, we truncated the optimization when computational time reached the sampling time. Since truncation has no influence, all eight motions were successfully generated in real-time with proposed method.

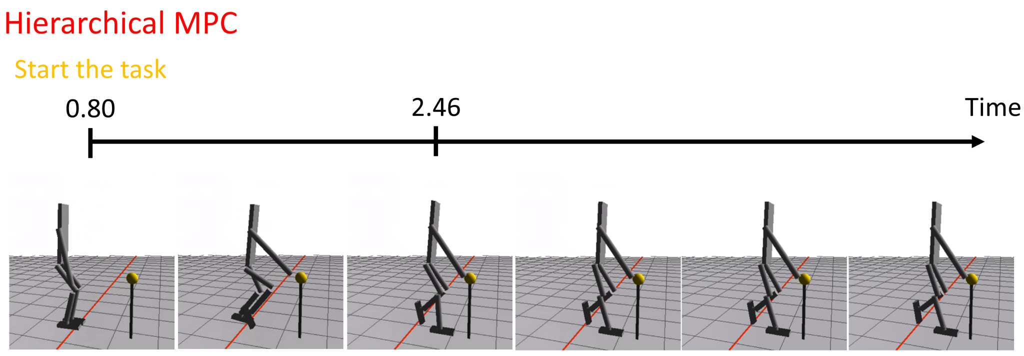
optimization, the computational complexity of our proposed MPC can be approximated as $O(N_c n^4)$, which is the complexity of the upper-layer optimization. In the experiments, we set horizon $N = 30$ for the Warm-Starting Fine MPC and $N_c = 10$ for the Warm-Starting Coarse-Fast MPC. Therefore, the computational time was reduced by one-third.

Required Time for Whole-body Reaching

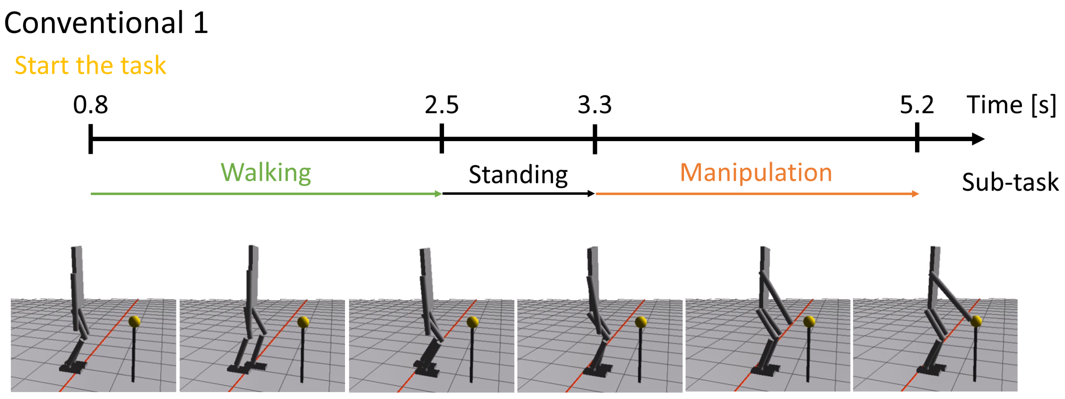
Figure 2.13a shows the generated whole-body reaching motion with our proposed hierarchical MPC (Warm-Starting Coarse-Fast MPC). The target position is indicated as a yellow ball. The robot moved forward while controlling its arm to reach the target. The task started from 0.8 s, and the distance between the target and hand position was minimized at 2.46 s. Since the averaged maximum computational time was 29.2 ms, which is less than the sampling period of 30 ms, the whole-body reaching motion was successfully generated in real-time.

Figure 2.13b represents the result of the conventional full-body controller. Whole-body reaching was accomplished through three sub-tasks: walking, standing, and manipulation. Since the robot was close to the target in 1.66 s when we applied MPC, we designed the center of pressure trajectory for the left and right steps per 0.8 s (including a double support phase for 0.1 s) for the walking controller. After maintaining standing posture for 0.8 s with the standing controller, the manipulation sub-task was initiated from 3.3 s. To generate reaching, CoM's desired trajectory was derived with the standing controller, and that of the arm was given by the manipulation controller. Since the robot rotated its arm with an average velocity of -0.269 rad/s until the hand reached the target by MPC, we adjusted the manipulation speed to be close to the value for the conventional controller. The actual averaged velocity was -0.269 rad/s. Finally, the robot successfully reached the target reaching position at 5.2 s.

Using the same conventional controller, we tried to generate whole-body reaching motions in different settings by changing the time at which the manipulation was initiated. In the first case, the manipulation was started right after walking (Conventional 2 in Fig. 2.14a). In the second case, walking and manipulation were simultaneously started, where the desired trajectory of the CoM and the arm were decided by the walking and manipulation controllers (Conventional 3



(a) generated whole-body reaching with Warm-Starting Coarse-Fast MPC



(b) generated whole-body reaching with conventional full-body controller (Conventional 1)

Figure 2.13: Generated whole-body reaching: Yellow ball indicates target reaching position. (a) Robot simultaneously executed walking and manipulation and reached target position at 2.46 s with the proposed hierarchical MPC. (b) Using conventional full-body controller, the robot sequentially walked, stood, and manipulated the ball. Distance between target and hand position was minimized at 5.2 s.

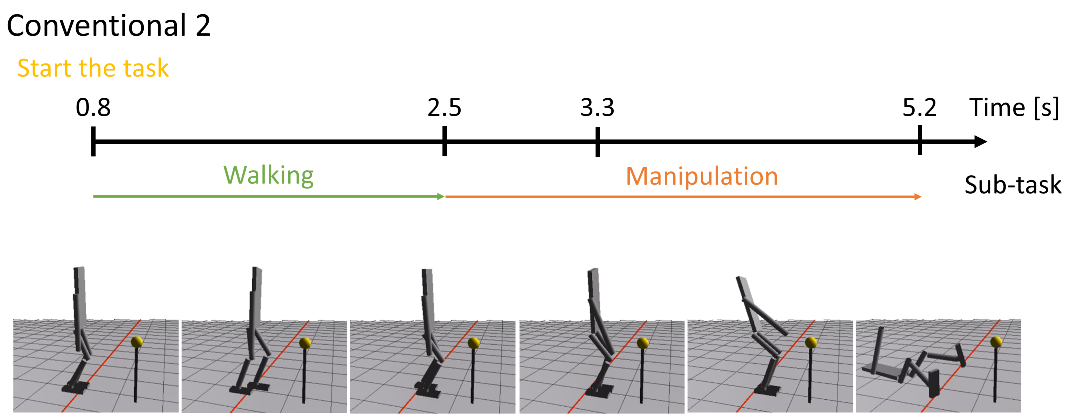
in Fig. 2.14b). If the manipulation were started earlier, the robot could complete the task more quickly. The robot, however, did not reach the target position and fell over after initiating the arm movements (Fig. 2.14)). The standing duration (0.8 s) in Conventional 1 was decided by increasing the time by 0.1 s from the setting of Conventional 2. The robot reached the target while keeping its balance after standing for 0.8 s. The distances between the target and hand positions at 5.2 s for all methods are represented in Fig. 2.15.

We attribute these results to the restricted representation of the humanoid robot in the high-level controllers. Since the walking controller of the conventional whole-body controller only considers the CoM dynamics, it cannot determine future plans while taking into account the influences of the arm motions. In contrast, since the manipulation controller for the arm trajectories ignores the CoM dynamics, the planning of one controller acts as if they are disturbances for the other controllers. The disturbances caused fatal falls in this experiment. Since simultaneously moving the CoM and arm is difficult, the standing sub-task had to be executed for 0.8 s after walking to rest the CoM's motion. This resulted in sequential execution of the three sub-tasks. On the other hand, MPC does consider the full-body dynamics of the humanoid robot for motion planning. Therefore, walking and manipulation were simultaneously executed with our approach and improved the slow speed of the robot operation. By comparing cases in which the robot successfully reached its target, our proposed approach generated a reaching motion more than twice as fast as the conventional controller; the required time of the hierarchical MPC to minimize the distance between the target and the hand was 1.66 s and that of the conventional whole-body controller was 4.40 s.

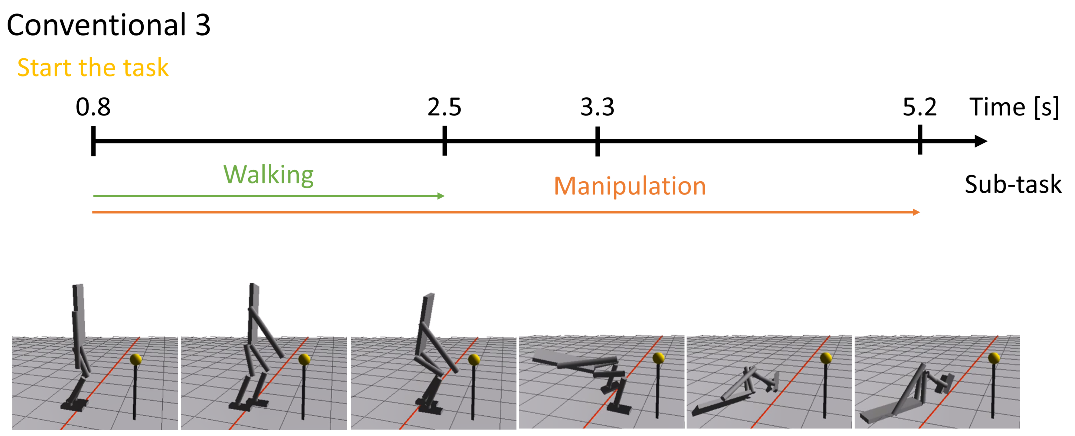
2.7 Real Robot Experiment

2.7.1 System Identification

In addition to the simulation experiment, we evaluated our method on a real robot whose lower extremity is CB-i [53], which is a hydraulically actuated humanoid robot. Several control approaches, such as passivity-based full-body force control



(a) generated whole-body reaching with conventional full-body controller (Conventional 2)



(b) generated whole-body reaching with conventional full-body controller (Conventional 3)

Figure 2.14: Generated whole-body reaching where robot initiated manipulation at different times: (a) Manipulation was started immediately after walking (from 2.5 s). (b) Robot simultaneously tried to execute walking and manipulation from 0.8 s. Robot failed to reach the target in both cases. Since two high-level controllers for walking and manipulation cannot determine future plans while taking into account the influences of each other's motion, dynamically changing or blending controllers is generally difficult.

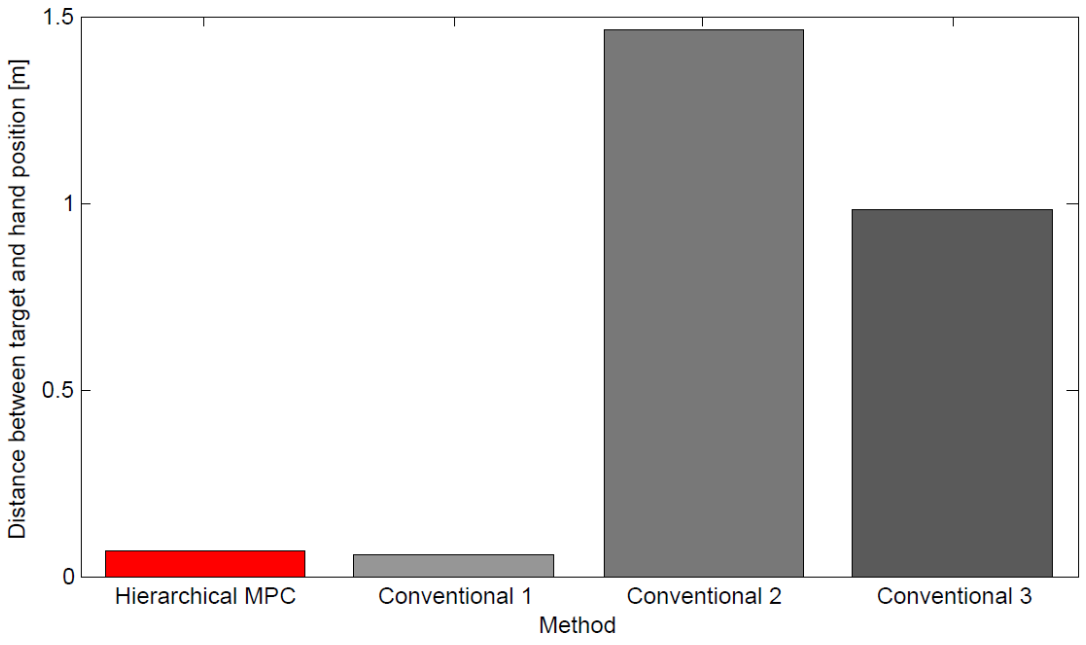


Figure 2.15: Distances between target and hand position of robot at 5.2 s in whole-body reaching task: In Conventionals 2 and 3, distances increased due to fatal falls.

[54] and biologically inspired locomotion control [55], were performed on this robot in the past. This is the first time that a real-time full-body optimal control approach has been applied to it.

In simulation experiments, we used CAD data to predict the motion trajectories of the humanoid robot. However, we found that they are not reliable because they don't include the inertial parameters of either the components of the hydraulic actuators or the new parts of the torso. We estimated the mass, the CoM position, and the inertia matrix of each body segment using a real robot's motion data.

The equations of motion of a robot manipulator can be written in a linear regression form with respect to the inertial parameters [56, 57]. By using the equations as an inverse dynamics model, a system identification problem can be formulated, where output torques are computed using the inverse dynamics model and then the inertial parameters are optimized to minimize the differences between the observed and computed torques. Since the identification problems can be solved efficiently, the approach can be applied to high-dimensional systems such as a human being and a humanoid robot [58, 59]. However, the estimated parameters for a humanoid robot with the inverse dynamics model are not always accurate for forward dynamics since the robot is a highly nonlinear system. We consider a problem formulation using the forward dynamics model in which the accumulated cost over finite-horizon T is minimized:

$$\min_{\Phi} \sum_{t=1\cdots T} \|\hat{\mathbf{x}}_t - \mathbf{x}_t(\Phi)\|, \tag{2.54}$$

where $\mathbf{x}_t(\Phi)$ is the predicted state using the forward dynamics model at time-step t. Variables Φ include the viscous friction coefficient of each joint and the parameters of the smoothed contact dynamics [11] as well as the inertial parameters (the mass, the CoM position, and the inertia matrix of each body segment). In our implementation, we minimized Eq. (2.54) with a gradient-based optimization: L-BFGS [60] in minFunc [61]. A similar approach was developed [62], where inertial parameters were estimated and sensors were calibrated at the same time, but we did them separately. We collected motion data $\hat{\mathbf{x}}$ using position tracking control with fine-tuned PID parameters.

2.7.2 State Estimation

Since future state trajectories are predicted from estimated current states in MPC, the control performances of the controller greatly depend on estimation accuracy. On the other hand, a fast estimation is another crucial issue for real-time model-based control. Therefore, fast and accurate state estimation is an essential part for real-time full-body optimal control.

Internal sensors such as joint encoders and an inertial measurement unit (IMU) can be used with a high sampling rate. Although the state of a humanoid robot can be partially measured with them, the position and velocity of the base link of the robot are not directly observable. Thus, a whole-body state estimator based on an unscented Kalman filter (UKF) was developed [63], where candidates of the current state are computed with the full-body dynamics model through forward dynamics. To obtain the current estimation, a weighted average of the sample points is combined with the current observations of the inertial sensors. Although using full-body dynamics is too expensive for real-time state estimation, they exploit their fast and accurate physics engines [47, 64]. Since our simulator can evaluate the dynamics fast enough for the UKF approach, we adopted it as the state estimator of our real robot.

However, since we sometimes experienced estimation divergences of the base position and the velocity, we utilized the measurements of an accelerometer in an IMU. The sensor measures the three-axis acceleration of the base link. There are several works on the state estimation of the base link of a legged robot with accelerometer measurements. A base state estimator with a Kalman filter (KF) was developed [65, 66]. The KF approach can model the bias terms of the accelerometer because the drift effect must be reduced. A complementary filter combined the leg kinematics and the IMU data [67]. This approach can be applied to a more general context of motion generation where a supporting foot is not fixed to the ground. We used a base state estimator based on KF because no force sensors were available to measure the ground reaction forces. They are required to adaptively adjust the crossover frequency of the complementary filter. The estimation result of the base state estimator and the UKF's weighted average were used as each other's observations. In both estimators, we used generalized Rodrigues parameters to perturb the attitude of the base link (quaternion) [68].

2.7.3 Cost Function Design and Experimental Setting

We performed three tasks in our real robot experiment. In the first task, the robot tried to maintain its initial hip position with hierarchical MPC. The second task's control objective was to stand upright. The third task was to demonstrate that different motions can be generated with MPC by changing the control objectives. We switched the control objective from standing to squatting. The warm-start technique (explained in Section 3.2) was not used in this experiment. Instead we applied Coarse-Fast MPC to the robot as the hierarchical MPC. Before initiating each task execution, the robot maintained its initial standing posture by a PID controller, which is switched from the PID to the hierarchical MPC in 10 seconds.

Weighted quadratic functions were adopted for the state-cost and control-cost terms as in the simulation experiment. The weights were constant through the experiments. The state-cost forces the robot to achieve a target hip position while minimizing the linear and angular velocities of the base link and the joints. The control-cost consists of two terms. The first term is a quadratic penalty on the control inputs. The second forces the robot to smoothly change the control inputs since sudden jumps in the control inputs can cause undesired oscillations on it. For squatting, we used a sinusoidal trajectory as the target hip positions. We used the same cost function for the lower-layer optimization as in the simulation experiment.

We set the sampling time to 12 ms. The time-step and the horizon for the upper-layer optimization were 12 ms and 10, respectively. We selected the control joints of the humanoid robot to have state dimensionality $n = 29$. The robot's fast variables in Eq. 2.47 became $n_f = 16$ dimensions. For the lower-layer optimization, they are 6 ms and 10. All the computations in the state estimation and motion generation were executed by an Intel Xeon Processor E5-2697 v3, 2.6 GHz computer. The maximum number of iterations and termination criteria for iLQR were respectively 6 and 5.0×10^{-2}. In iLQR, the derivatives of the dynamics and the costs were computed in parallel with ten computer threads using Open Multi-Processing (OpenMP) [50]. As each joint level controller, we implemented a PID controller with load velocity feedback to compute the valve commands while compensating the actuator dynamics [69, 70]. The control period of the joint level controllers was 6 ms.

2.7.4 Experimental Results

Figure 2.16 shows the generated motions for the first task. Although the robot's controller was switched from PID to the hierarchical MPC during the task execution, the robot successfully maintained its initial hip position. The maximum computational time was 13.30 ms, which is slightly over the sampling period (12 ms). However, since only 0.84% (7/833) of the computations exceeded the limit, the overruns only slightly influenced the optimization result. Fig. 2.17 represents a series of motions of CB-i to stand upright from the initial posture in which it bent its legs. Since a finite optimal control problem was solved at each time-step in MPC, the robot generated smooth transient trajectories from the initial hip positions to the targets online. When the control objective was switched from standing to squatting in the third task, the robot changed its behavior smoothly to achieve the new objective (Fig. 2.18). In the third task, the maximum computational time was 12.02 ms, but only 0.12% (1/833) of the computations exceeded the sampling period. The above results demonstrate the advantage of our hierarchical MPC for versatile motion generation of a humanoid robot because the whole-body motions were automatically generated by changing the control objectives.

2.8 Summary

In this chapter, we proposed a Warm-Starting Hierarchical MPC approach for humanoid robot control. In the proposed method, we combined a hierarchical MPC approach named Coarse-Fast MPC with a simple warm-start technique. To warm-start the optimizer from the beginning, we optimized the initial inputs offline. Online computation began with the offline optimization result as good starting points. The upper layer of the hierarchical MPC derived an optimal control sequence of whole-body dynamics with a coarse-time resolution. In the lower layer, the derived control sequence was further optimized on extracted low-dimensional dynamics with fine-time resolution.

We evaluated our proposed MPC in a simulated 3D seven-link biped model. By transforming the humanoid robot dynamics into a singularly perturbed system, we extracted a fast subsystem to apply the proposed method. The robot fell

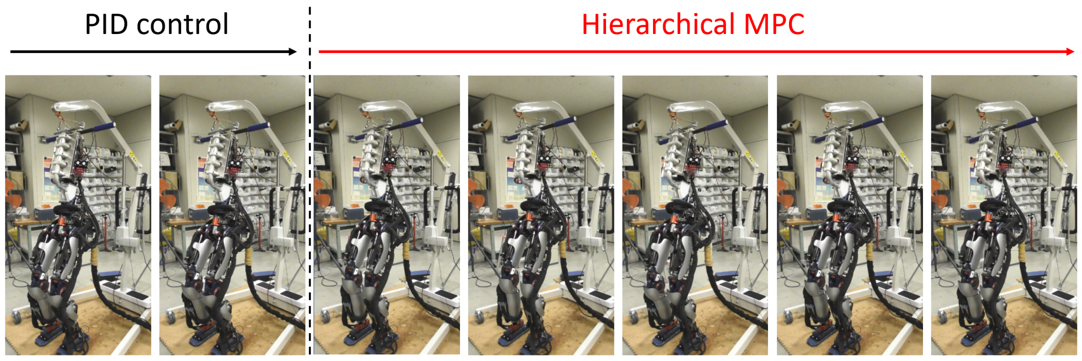


Figure 2.16: Motion generation with our proposed method: Coarse-Fast MPC for the first task. Robot successfully kept initial hip position after switching controller from PID to Coarse-Fast MPC.

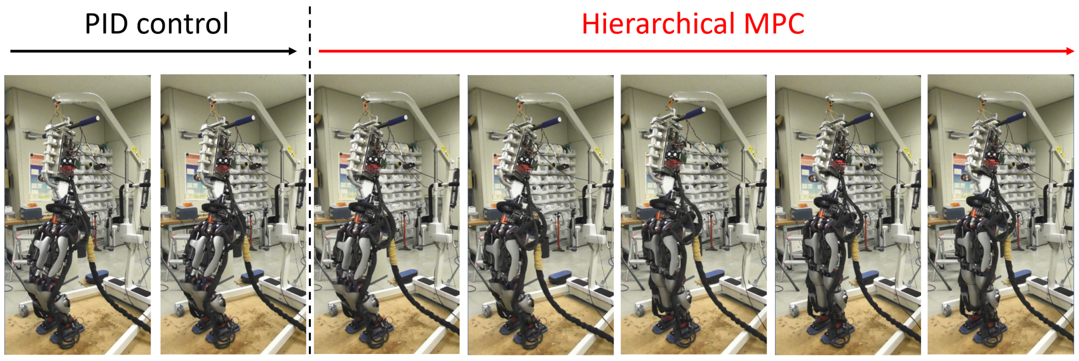


Figure 2.17: Snapshots of motion generation result with Coarse-Fast MPC for second task: Robot successfully generated series of motions to stand upright from initial posture where it bent its legs. Our proposed method derived smooth transient trajectory from initial hip position to the target online.

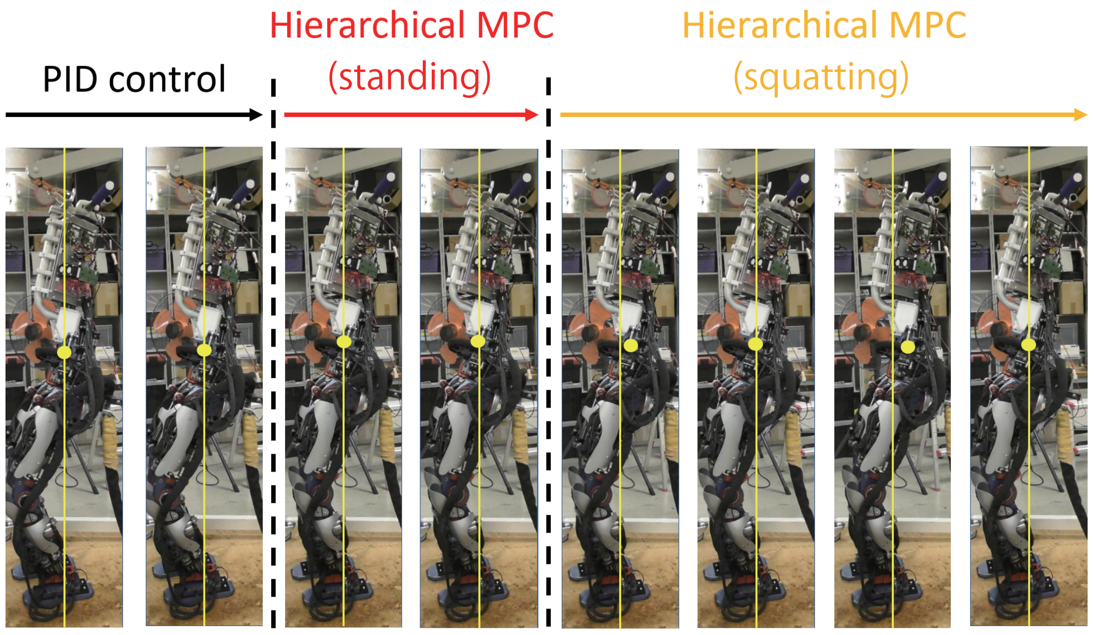


Figure 2.18: Generated motion with Coarse-Fast MPC in third task: motion generation in changing controllers and control objectives. Yellow line indicates initial hip position. Solid yellow circle denotes current hip position. Robot generated slow squatting motion when control objectives were switched from standing to squatting. This result demonstrates that different motions were generated with MPC by changing the control objectives on a real robot.

over when the motions were generated with conventional MPC settings. When using the Warm-Starting Fine MPC, the method required a lot of time to compute the optimal policies. Thus, few optimizations converged if we truncated the computation at the sampling time for real-time control. On the other hand, the computational time was reduced with Warm-Starting Coarse-Fast MPC. Since the controller derived by the coarse optimization was refined with the lower-layer optimization, better motions were also generated in all cases I to VIII than the Warm-Starting Coarse MPC. Therefore, real-time motion generations were achieved without significantly degrading the control performances with our proposed method: Warm-Starting Coarse-Fast MPC.

We compared our hierarchical MPC with a conventional full-body controller in terms of the required time to generate a whole-body reaching motion. Since the hierarchical MPC planned whole-body motions under full-body dynamics, a walking to approach a target position and a reaching motion were simultaneously generated and improved the task execution's speed.

We also evaluated our approach on a real robot experiment. Since the target motions were very simple (standing and slow squatting), they were successfully generated with Coarse-Fast MPC without the input initialization approach. When the control objectives were switched from standing to squatting, our method generated a smooth transient trajectory that connected them online.

In this chapter, we manually designed the parameters of the cost functions and estimated them from human demonstrations using the Inverse Optimal Control (IOC) approach [31, 71]. In Chapter 4, we explore a framework that combines our proposed MPC approach with IOC. With the approach, we can easily design cost functions for various motions. Future work will apply our proposed MPC to a real robot to generate more dynamic movements such as walking and jumping.

3 Two-Stage Control Scheme for a Hybrid Actuator System

3.1 Introduction

Although robotics technologies have rapidly improved and are widely used in industry and daily life, it remains difficult for robots to generate human-like fast and flexible movements to work in cluttered environments [2] or to assist human behaviors [72]. One reason that deters the development of such robot systems is that robots lack a light, compliant but strong actuator that works like the human muscles. As a matter of practice, electric motors with high reduction gears, which are dominantly used as robot actuators, tend to be heavy and rigid for generating large torques. Although such muscle-like actuators as a pneumatic artificial muscle (PAM) have similar properties to human muscles, control latency due to the air flow is inevitable since PAM uses air pressure to generate joint torque. Therefore, PAM has not been widely used in robot systems.

On the other hand, humans use multiple muscles to efficiently generate a joint movement. For example, at least the biceps and triceps are involved in the one degree of freedom of an elbow joint movement. In the biceps, there are two different bundles: biceps long head and biceps short head. The triceps has three different bundles: triceps long head, triceps lateral head, and triceps medial head [73]. Similarly, from a technical point of view, robots might be able to use multiple actuators to generate a joint motion. To explore this possibility, hybrid actuation systems have been developed and implemented in robot systems [17, 23, 74]. However, how to distribute the desired torque to each actuator has not been scrutinized. One possible approach to cope with this torque distribution problem is using an optimal control method [24, 75]. The optimal control framework provides a powerful tool for robot motion generation in a diverse set of tasks and applications to actuator control [76, 77] since optimal motor commands can be derived under the constraint of their own dynamics by specifying simple high-level task goals. However, solving an optimal control problem at each control time step has not been considered as a practical approach due to the large computational burden.

In this chapter, we propose a computationally efficient method to derive an optimal control strategy for a hybrid actuation system composed of multiple actuators, where each actuator has different dynamical properties. We derive a new singularly perturbed system of hybrid actuator dynamics to subdivide the original control problem into smaller subproblems so that the optimal control outputs for each actuator can be derived at each control time step. The singularly perturbed system has been intensively studied to describe fluid dynamics [78, 79], which shows different properties in the regions close and far from the wall due to its viscosity. Inspired by these kinds of studies, it was found that even robot systems, which have particular dynamic properties such as a biped robot or a flexible robot manipulator, can be represented as singularly perturbed systems [80, 81].

A method that derives optimal control outputs at each control time step is known as Model Predictive Control (MPC). MPC methods have been applied to such slow dynamical systems as chemical plants [33] or ship maneuvering problems [10]. However, directly applying MPC to the real-time control of such faster dynamics as robot systems is unrealistic. In this study, we newly derived a singular perturbed system for a hybrid actuator and applied MPC to the subdivided control problems. Based on an idea of using MPC for a simulated system that includes different time scale dynamics [35], we developed a two-stage MPC framework for the hybrid actuation system, in which a two-stage control scheme is derived based on the singular perturbed system of the hybrid actuator and the hierarchical MPC in Chapter 2 decides the optimal torque distributions. We showed that our proposed method was able to successfully control the real robot. Both optimization stages correspond to either the extracted fast components of the singularly perturbed system or the original dynamics that include a slow dynamical component. Then for the fast dynamical component, we propose short-term optimization with fine-control time resolution. Since a robot can quickly change its behavior with the fast dynamical component, the long-term optimization is not necessary. Instead, we can utilize computational resources to generate precise movements with higher time resolution. On the other hand, for the original dynamics, we address long-term optimization with coarse-time resolution. Since a robot cannot quickly change its behavior with the original dynamics that includes the slow dynamical component, we need to consider the long-term optimization problem. However, for the original dynamics, we can use the optimization problem with lower time resolution that requires less computation since we can rely on the optimization of the fast dynamical component to generate movements for fine-time resolution.

To evaluate the torque distribution performance of our proposed approach, we applied the two-stage optimization procedure to our pneumatic-electric hybrid actuator system, where PAM dominantly generates joint torques but is assisted by a small and lightweight electric motor for precise movements (Fig. 3.1). We first derived a singularly perturbed system for the hybrid actuation system based on the difference of the dynamic properties between PAM and the electric motor. From this theoretical analysis, we found that only the lower-dimensional subspace of the original system needs to be considered in the fast dynamical component and that only the electric motor outputs affect the behavior of the lower-dimensional subspace. Our method properly derived a torque distribution strategy for the hybrid actuation system by solving a real-time optimal control problem and successfully reduced the computation time without significantly deteriorating the control performance. In the proposed approach, after deriving the optimal input voltages for the air valve and the motor driver with coarse-time resolution, only the optimal sequence of the input voltage for the motor driver was further optimized with fine-time resolution under the lower-dimensional dynamics. For a comparison, we also applied two standard implementations of the MPC method with two different time resolutions each: optimal control strategies with fine-time resolution and coarse-time resolution to our hybrid actuator system. The coarse strategy corresponds to the optimization process for the original dynamics of our proposed method. Therefore, we can evaluate the effectiveness of the optimization for the fast dynamics component by comparing the tracking results of the coarse strategy and our proposed approaches. To the best of our knowledge, for the first time in the literature, a real robot with a hybrid actuator system is proposed as a singularly perturbed system and is successfully controlled in real-time with the new two-stage optimal control method.

The rest of our paper is organized as follows. Section 3.2 introduces our proposed two-stage optimal control method. Section 3.3 describes about the tasks, their goals and experimental setting. Section 3.4 shows the experimental results.

3.2 Proposed Method

3.2.1 Fast Dynamics of a Pneumatic-Electric Hybrid Actuator System

We, first, derive a singularly perturbed system for a hybrid actuator composed of a pneumatic artificial muscle and a small lightweight electric motor. By investigating the time constant of the first-order air pressure dynamics as a time scale to extract the fast dynamical component from the hybrid actuation system, we find that only the electric input affects the fast dynamical component.

The motion equations of a forearm robot with the hybrid actuator are expressed as follows:

$$\begin{aligned} T_p\dot{P} &= -k_{vp}P + v_p \\ I\ddot{\theta} &= c(\theta, \dot{\theta}) + \tau_p(P, \theta) + \tau_m(v_m), \end{aligned} \tag{3.1}$$

where I is the inertia and c is the vector of the total external forces, such as gravity and friction forces. The output torques generated with the pneumatic actuator and the electric motor are τ_p and τ_m, respectively. The transformation coefficient of the pressure to a voltage scale is denoted as k_{vp}. The time constant parameter of the air valve is represented as T_p.

The joint angle and angular velocity are denoted as θ and $\dot{\theta}$, P stands for the PAM pressure, and v_m and v_p are the input voltages for the electric motor and PAM. State $\mathbf{x}$ of the state-space model consists of the positions, the velocities, and the PAM pressure: $\mathbf{x} = [\theta, \dot{\theta}, P]^\top \in \mathbb{R}^3$. The input vector is composed of the input voltages: $\mathbf{u} = [v_m, v_p]^\top \in \mathbb{R}^2$. We adopted a smooth friction model [82] and a second-order pneumatic actuator model [75].

In general, the PAM pressure does not change as fast as the electrical motors since the air flow is much slower than the electrical current. Air pressure P is slower than joint angle θ and its velocity $\dot{\theta}$ is affected by the electric motor as a result of the relatively large value of time constant parameter T_p. To extract the fast dynamics, we define ε in Eq. (2.23) as $\varepsilon = 1/T_p$ and use the deviations of the fast states and the control input from the optimal fast ones in the coarse optimization as $\zeta_1 = \theta - \tilde{\theta}^\star$, $\zeta_2 = \dot{\theta} - \tilde{\dot{\theta}}^\star$ and $\nu = v_m - \tilde{v}_m^\star$. By rewriting Eq. (3.1) with the defined variables and setting $\varepsilon \to 0$, we obtain the following fast subsystem:

$$\dot{P} = 0. \tag{3.2a}$$

$$I\dot{\zeta}_2 = c(\zeta_1 + \tilde{\theta}^\star, \zeta_2 + \tilde{\dot{\theta}}^\star) + \tau_p(\tilde{P}^\star, \zeta_1 + \tilde{\theta}^\star) + \tau_m(\nu + \tilde{v}_m^\star) \tag{3.2b}$$

(3.2c)

Equation (3.2) describes the dynamics of the fast states under the fixed slow variables.

3.2.2 Two-stage Optimal Control Scheme

From the derived singularly perturbed system for the hybrid actuation system, we found that only the lower-dimensional subspace of the original system needs to be considered in the fast dynamical component and that only the electric motor outputs affect the behavior of the lower-dimensional subspace. Based on this theoretical analysis, we derived a two-stage optimal control scheme where each optimization stage corresponds to either the fast component of a singularly perturbed system of Eq. (3.2) or the original dynamics of Eq. (3.1).

The optimal torque distributions are decided with our proposed hierarchical MPC in Chapther 2. For the fast dynamical component, we used a short-term optimization with fine-control time resolution. Since a robot can quickly change its behavior with the fast dynamical component, we can disregard the long-term optimization problem and utilize computational resources to generate precise movements with fine-time resolution. Since control inputs do not only affect the slow states, the discretized reduced-order subsystem of Eq. (3.2c) is considered as a constraint in the fine optimization of the hierarchical MPC (Eq. (2.31) in Chapter 2). On the other hand, for the original dynamics, we address long-term optimization with coarse-time resolution. Since a robot cannot quickly change its behavior with the original dynamics that includes slow dynamical component, we need to consider the long-term optimization problem. However, for the original dynamics, we can use the optimization problem with lower time resolution that requires less computation since we can rely on the optimization of the fast dynamical component to generate movements with fine-time resolution. Fig. 3.2 shows the proposed two-stage control scheme: the two-stage MPC.

MPC's computation time depends on prediction length N, the dimensionality of state space n, and dimensionality of control input m. We thus divided the original control problem into two smaller problems in terms of the prediction horizon and the state and input dimensions.

3.3 Experiments

We evaluated our proposed method on our forearm robot with the hybrid actuation system (Fig. 3.1) in tracking control problems. For the target joint trajectories we used different sinusoidal patterns in terms of frequencies at 0.25, 0.5, and 1.0 Hz with a peak-to-peak amplitude of ± 30 deg. From the derived singularly perturbed system, we applied the two-stage optimization approach to the slow and fast dynamical components of the actuator system. We used one PAM actuator and one electric motor of our forearm robot to evaluate our proposed approach.

In this experiment, the cost function for the optimal control method was composed of four terms: error between the desired and current joint angles, penalties on the PAM pressure, control cost for joint torques due to the PAM and the electric motor outputs, and control cost for input voltages of the air valve and the motor driver. Specifically, the robot minimized the total cost of the immediate cost function ℓ accumulated along the predicted trajectory and the terminal cost function ℓ_T, where

$$\ell = w_\theta(\theta - \theta^d)^2 + w_P \mathrm{smin}(P - P_0, 0)^2 + w_\tau(\tau_p^2 + \tau_m^2) + w_v(v_p^2 + v_m^2), \tag{3.3a}$$

and

$$\ell_T = w_\theta(\theta - \theta^d)^2 + w_P \mathrm{smin}(P - P_0, 0)^2. \tag{3.3b}$$

smin(a, b) is a smooth approximation to min(a, b). In this experiment, we utilized the following function:

$$\mathrm{smin}(a, b) = -\frac{\sqrt{(a-b)^2 + \gamma^2} - a - b}{2}, \tag{3.4}$$

where γ is a constant to regulate the smoothness. We used $\gamma = 0.1$ in the experiment. The target angle at a time step is denoted as θ^d. If the PAM pressure falls below P_0, the wire connecting the PAM and the robot loses its tension and the pneumatic actuator cannot generate any torque. The second term on the right-hand side in Eq. (3.3) is the penalty on it. In the proposed method, the same cost function is used in both stages except the control costs. In the second stage, the immediate cost includes the deviation of the fast control input instead of the input voltages. We set the each weight as $w_\theta = 40$, $w_P = 30$, $w_\tau = 0.1$, $w_v = 0.002$ and limited the domains of the control variables, as in $0 \leq v_p \leq 5.0$ and $-5.0 \leq v_m \leq 5.0$. For comparisons presented in Fig. 3.4, we varied the weight for the voltage input as $w_v = 0.02, 0.002$ and 0.0002. The forearm robot weighed 2.7 kg. The sampling and control periods were 4.0 ms. Optimal input voltage command to the air valve of the PAM and the motor driver of the small lightweight electric motor needed to be derived within the control period.

In this experiment, we also applied two standard implementations of the MPC method each to compare the results with our proposed method. In the fine optimal control strategy, the optimal control sequence Eq. (2.3) for a fine time step is derived while a coarse time step is used in the coarse control strategy. Table 3.1 shows the time steps and the horizon used in each method. To investigate the effect of the time step size for the standard MPC implementations, we varied the time step size as 4.0 ms and 8.0 ms for the fine strategies and as 20.0 ms and 40.0 ms for the coarse strategies. At each control step, we solved finite optimal control problems with a 200-ms time horizon.

Method	Step size (ms)	Horizon
Fine 4ms	$\Delta t_f = 4$	$N = 50$
Fine 8ms	$\Delta t_f = 8$	$N = 25$
Coarse 20ms	$\Delta t_c = 20$	$N = 10$
Coarse 40ms	$\Delta t_c = 40$	$N = 5$
Two-stage	$\Delta t_f = 4$, $\Delta t_c = 20$	$N_f = 20$, $N = 10$

Table 3.1: Experimental setups of time step size and horizon length.

All the optimal controllers presented in the study were derived through a C language programming environment on an Intel Xeon Processor E5-2697 v3, 2.6-GHz computer. The maximum number of iteration and termination criteria for iLQR were 50 and 5.0×10^{-4} respectively in all the methods. In iLQR, we computed the derivatives of the dynamics and the costs in parallel with ten computer threads using Open Multi-Processing (OpenMP) [50].

3.4 Results and Discussion

3.4.1 Computational Times and Control Performances

We first compared the fine and coarse optimal control strategies with our proposed method in terms of computational times and control performances. Fig. 3.3a shows the maximum computation times required to derive an optimal control sequence to track the sinusoidal target trajectories with frequencies of 0.25, 0.5, and 1.0 Hz. We showed the average maximum computation times of five experimental trials. The 4.0-ms control period is depicted by the dashed black line. Although a calculation process to derive an optimal control sequence must be terminated within 4.0 ms for real-time control, all of the maximum computational times of the fine strategies exceeded the threshold, but not the coarse and proposed strategies (Fig. 3.3a). These results indicate that optimal control calculation with fine-time resolution cannot be an option for the real-time control of a hybrid actuator system.

Figure 3.3b shows the average accumulated cost for a 10-s duration of five tracking control trials. The accumulated cost was computed with the cost functions in Eq. (3.3). *Fine 4ms* strategy resulted in the largest total cost for all the target frequencies. This is mainly due to the large deviation from the target trajectory. This large deviation occurred since *Fine 4ms* strategy requires a large amount of computation time and was unable to finish the calculation within the control period. Although *Fine 8ms* strategy shows better performance than *Fine 4ms*, control outputs were not able to be derived in real time. If the computation exceeds the period, the control is delayed in the real robot experiment. The delay results in degradation of the control performances. Therefore, the fine strategies showed the large total costs. The coarse strategies and proposed strategy showed similar tracking performance for the target sinusoidal trajectory of 0.25 Hz. On the other hand, for the coarse strategies, the total cost gradually increased for the target trajectories with higher frequencies. Although *Coarse 20ms* strategy shows the best performance in all the standard MPC implementations, our proposed approach shows even better performance than *Coarse 20ms* for all the target frequencies

Figure 3.4 represents the accumulated cost for the three different weights of the voltage input cost: $w_v = 0.02, 0.002$ and 0.0002. Here, we utilized the target trajectory with frequency of 1.0 Hz. The proposed approach shows the best control performance regardless of the weight settings. The above results indicate that our two-stage optimal control framework with a newly derived singularly perturbed system successfully worked for a real hybrid actuation system.

3.4.2 Generated Trajectories

Here, we show the generated joint angle profiles for the target sinusoidal trajectories of 0.25 and 1.0 Hz using the three different optimal control strategies in Fig. 3.5. We depicted the results of the proposed method with those of the worst and the best implementations of the standard MPC in terms of the comparison depicted in Fig. 3.3, where the worst one corresponds to *Fine 4ms* strategy and the best one corresponds to *Coarse 20ms* strategy. The target trajectories are represented by dashed orange lines. We conducted five experimental trials for each strategy and depicted the five trajectories generated using each optimal control method.

As shown in Fig. 3.5a, for the 0.25-Hz sinusoidal target pattern, the generated movements using the fine strategy frequently deviated from the target trajectory. Even with the coarse strategy, small fluctuations were observed. On the other hand, with our proposed strategy, the generated movements consistently followed the target trajectory. For the target trajectory with a 1.0-Hz sinusoidal pattern, the joint trajectories were not properly generated when we used either the fine or the coarse strategies (Fig. 3.5b).

We quantitatively compared the tracking errors of the three MPC approaches both for 0.25-Hz and 1.0-Hz sinusoidal target movements in Fig. 3.6. We again found that our proposed approach outperformed the standard MPC implementations.

Figure 3.7 shows the generated motions of our forearm robot for the 1.0-Hz target trajectory using the three optimal control strategies. The target trajectories are depicted by dashed orange lines. For the fine and coarse strategies, the generated joint motions significantly deviated from the target movements and failed to recover from failure situations after the large deviations (Fig. 3.7). The motions generated by the proposed method successfully followed the target trajectory without any large tracking errors.

The above results again clearly showed the advantage of using our two-stage optimal control strategy based on the singular perturbation method. Although we evaluated our approach on the 0.25-, 0.5- and 1.0-Hz sinusoidal trajectories, similar tracking performances were observed up to a target trajectory with 1.8-Hz frequency.

3.4.3 Torque Distributions

In the previous sections, we evaluated the control performances of our proposed optimal control strategy. Here, we show how it distributed the commanded torque to the PAM and the electric motor for precise target trajectory tracking by our proposed method. Figure 3.8a shows the contributions of each actuator to generate the joint torques. We showed the average distributions of five experimental trials. Error bars represent standard deviations. Our results indicate that the contribution of the electric motor was much smaller than that of the PAM in terms of the amount of generated torque.

We then compared the tracking performance of our proposed hybrid control strategy with that of the control approach, which only uses PAM on joint trajectory tracking tasks. For PAM control, we simply used the coarse optimal control method with the settings of *Coarse 20ms* strategy in Table 3.1, which is equivalent to the optimization procedure for the original dynamics in our two-stage optimization method. Figure 3.8b shows the averaged tracking errors of five experimental trials each of the proposed hybrid control method and the PAM-only approach. Error bars are standard deviations. The electric motor greatly improved the tracking performances although the amount of output torque was much smaller than that of PAM. Therefore, using multiple actuators to control a joint movement through real-time optimal control methods seems promising. Figure 3.9 depicts the generated joint torque profiles to show how the torque was distributed to each actuator for the target sinusoidal trajectories of 0.25- and 1.0-Hz frequencies. For both target frequencies, the PAM and the electric motor were cooperatively activated for the target tracking tasks. The role of each actuator was dynamically changed according to the frequency of the target trajectory. The electric motor showed its peak torque earlier than the PAM to generate the upward movements at each cycle of the target trajectory with 1.0-Hz frequency. The quicker torque response of the electric motor successfully compensated the slower PAM movement.

3.5 Summary

We applied our proposed optimal control approach to derive a torque distribution strategy for our hybrid actuation system that is composed of a pneumatic artificial muscle and small and lightweight electric motors. We derived a singularly perturbed system to extract the fast dynamics for a forearm robot with a hybrid actuator. This property resembles the human muscle system, which is also composed of muscle fibers that have different twitch speeds [73].

Optimal control methods are getting much attention as control algorithms for a wide variety of robots due to the recent rapid improvement of powerful computational resources [3, 7, 9, 18, 20]. However, a standard optimal control framework that can be applied to nonlinear systems only provides an optimized control sequence that is calculated in an offline manner. Since such a pre-designed control sequence is susceptible to external disturbances, these standard offline optimal control approaches are not directly applicable for robot control in real environments. On the other hand, an online optimal control approach, also known as Model Predictive Control (MPC), can be useful since it updates an optimal control sequence at each control time step based on the current situation and can cope with the external disturbances influencing a robot [10, 83]. Moreover, MPC can derive control policies under a situation in which robots need to generate motions adaptively with changing objectives, e.g., dynamically changing the location of a target object or a target motion indicated by a human [12, 21].

In our study, we proposed a two-stage optimal control strategy to reduce MPC's computational burden. We first derived the optimal control sequence for the original dynamics with coarse-time resolution and then optimized the short-term control sequence for the fast dynamical component with fine-time resolution. Since the extracted fast dynamical component of the forearm robot has lower-dimensional state space with lower-dimensional input, its computational burden was further reduced. Although application of the structured MPC based on a singularly perturbed system was previously explored on a simulated chemical plant model [35] and a biped model as in Chapter 2, in this study, we newly derived a singularly perturbed system for a hybrid actuator. To the best of our knowledge, ours is the first scheme that applied the two-stage MPC method to a real system and achieved successful results. As we presented in the experimental results, the computation time of the proposed approach was reduced much more than the required calculation time for solving the original optimal control problem. In our proposed two-stage MPC approach, after computing the optimal input voltages for the pneumatic actuator and the electric motor with coarse-time resolution, only the optimal sequence of the input voltages for the electric motor is re-optimized with fine-time resolution on fast dynamics. Thus, the proposed method successfully tracked the target trajectories in real-time without deterioration of the control performances (Fig. 3.3 and Fig. 3.4).

We empirically found that our proposed approach with properly selected time step sizes and horizons was able to generate stable and high control performance of the real hybrid actuation system. On the other hand, Lyapunov-based analyses for theoretically stable MPC have been studied [84, 85]. The near-optimality related to the singular perturbation parameter of a slow-fast MPC is analyzed in [35]. There exists a singular perturbation parameter ε' which decides whether behaviors of the slow and fast subsystems can approximate those of the original dynamics under appropriate assumptions. The value of the total cost for the original dynamics converges to the sum of the total costs of the slow and fast subsystems if a singular perturbation parameter ($\varepsilon \leq \varepsilon'$) is sufficiently small. Similarly, theoretical consideration about the stability and optimality for our two-stage MPC can be an important future research topic. Furthermore, future work will apply our proposed method to an exoskeleton robot with hybrid actuators [16, 17, 86]. Since the dynamics of a biped robot can be transformed into a singularly perturbed system [80], we might study a hierarchically combined singularly perturbed system for a bipedal exoskeleton robot with a hybrid actuator system. We will explore our approach on more versatile motion generation including non-periodic movements.

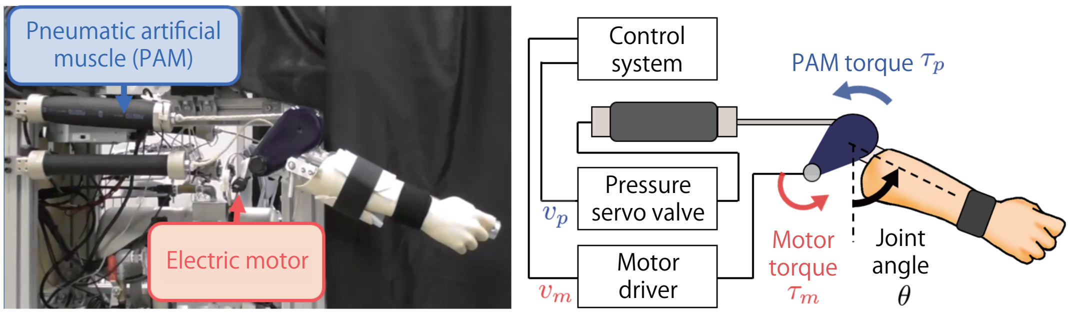


Figure 3.1: Forearm robot with pneumatic-electric hybrid actuator system: pneumatic artificial muscle dominantly generates torques but is assisted by small and lightweight electric motor for precise movements. v_p denotes voltage input for air valve, and v_m denotes voltage input for motor driver. We used one pneumatic artificial muscle and one electric motor.

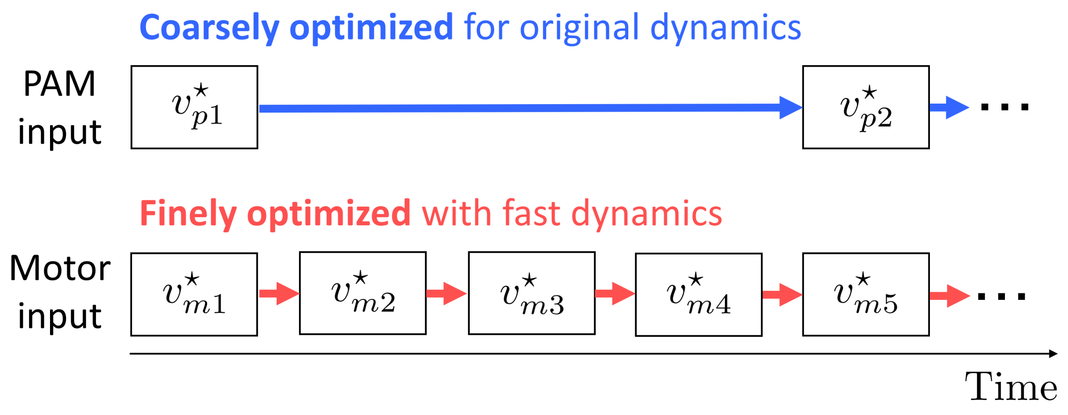


Figure 3.2: Two-stage optimization strategy: Each thread corresponds to either fast component of singularly perturbed system or original system. For fast dynamical component, we propose short-term optimization with fine-control time resolution. Since a robot can quickly change its behavior with fast dynamical component, long-term optimization problem is not necessary and we utilize computational resources to generate precise movements with higher time resolution. For original dynamics, we address long-term optimization with coarse-time resolution. Since a robot cannot quickly change its behavior with original dynamics that includes slow dynamical component, we need to consider long-term optimization problem. However, in two-stage optimization strategy, we can use optimization problem with lower time resolution that requires less computation since we can rely on optimization of fast dynamical component to generate movements for fine-time resolution. v_p denotes voltage input for air valve, and v_m denotes voltage input for motor driver.

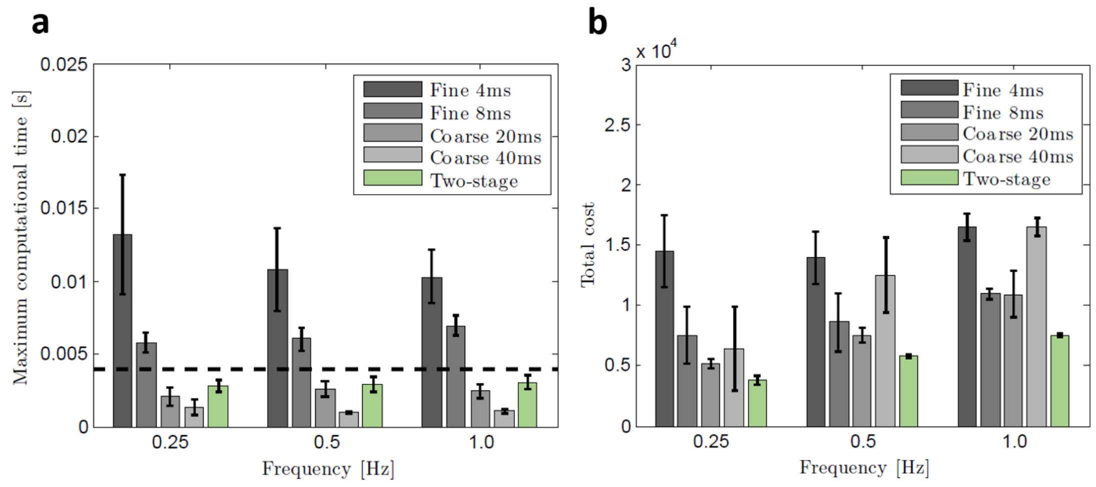


Figure 3.3: Computational times and control performances: (a) maximum computation times to derive optimal control sequence for target trajectories with frequencies of 0.25, 0.5, and 1.0 Hz. We show averaged maximum computation times of five experimental trials. 4.0-ms control period is depicted by dashed black line. (b) averaged accumulated cost for 10-s duration of five tracking control trials. *Fine 4ms* strategy resulted in largest total cost for all target frequencies. *Coarse 20ms*, *40ms* and proposed two-stage strategies showed similar tracking performances for target sinusoidal trajectory of 0.25 Hz. On the other hand, for *Coarse 20ms* and *40ms* strategies, total cost increased for target trajectories with higher frequencies. Error bars represent standard deviations.

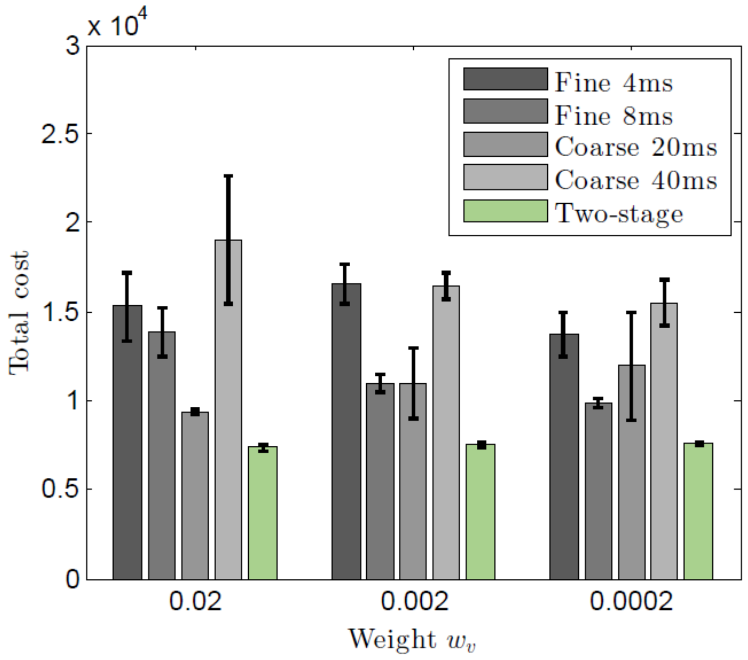


Figure 3.4: Control performances with three different weight of voltage input cost: $w_v = 0.02, 0.002$ and 0.0002. Accumulated costs for 10-s duration of five tracking control trials were compared. We utilized the target trajectory with frequency of 1.0 Hz. Our proposed approach showed best control performance regardless of weight settings. Error bars represent standard deviations.

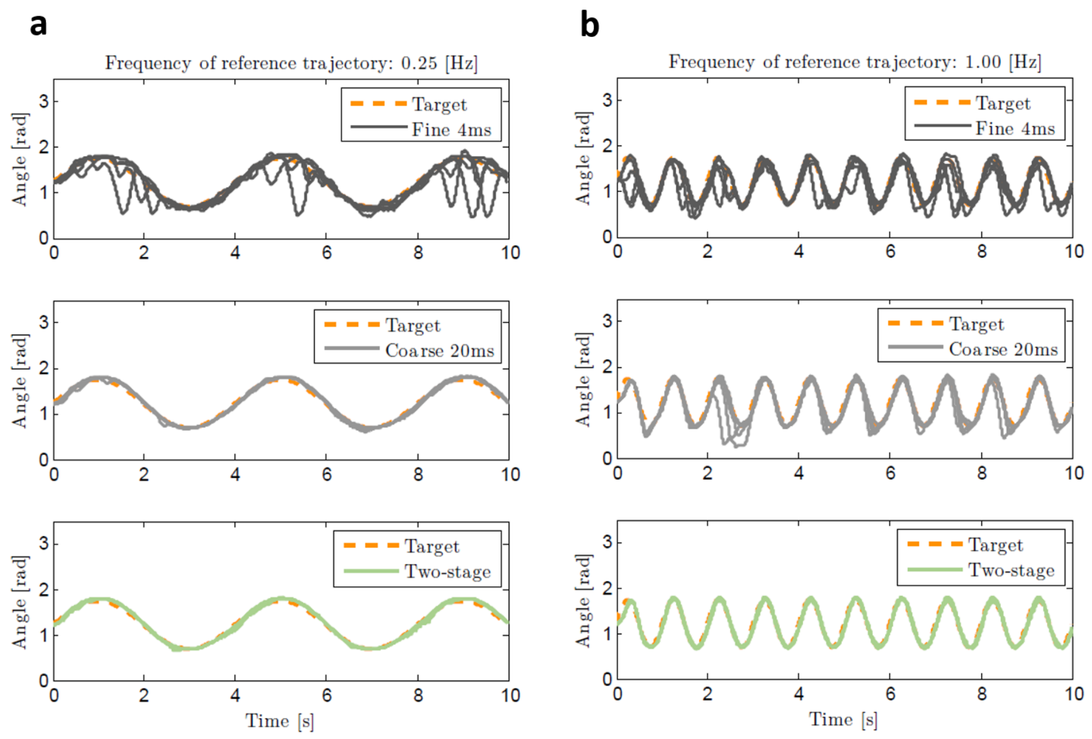


Figure 3.5: Generated joint angle profiles: Generated joint angle profiles are shown for target sinusoidal trajectories of 0.25 and 1.0 Hz using three different optimal control strategies. Target trajectories are represented by dashed orange lines. Since we conducted five experimental trials for each strategy, five trajectories are depicted for each optimal control method. (a) generated movements using the fine strategy for 0.25-Hz sinusoidal target patterns frequently deviated from target trajectory. Even with the coarse strategy, small fluctuations were observed. On the other hand, when we used our proposed strategy, generated movements consistently followed target trajectory. (b) for target trajectory with 1.0-Hz sinusoidal pattern, joint trajectories were not properly generated when we used either fine or coarse strategies. On the other hand, with proposed method, generated trajectories successfully followed target trajectory.

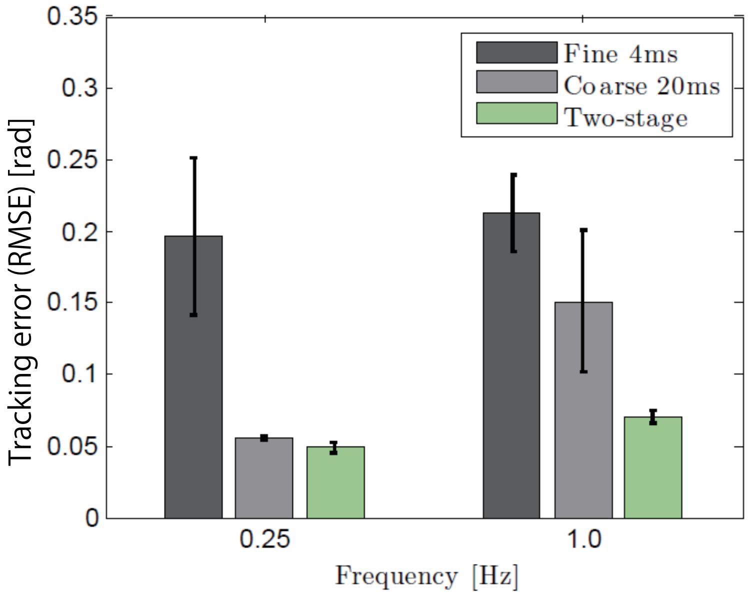


Figure 3.6: Averaged tracking errors of five experimental trials each of the fine, coarse and proposed strategies: We found that our proposed approach outperformed the standard MPC implementations. Error bars represent standard deviations.

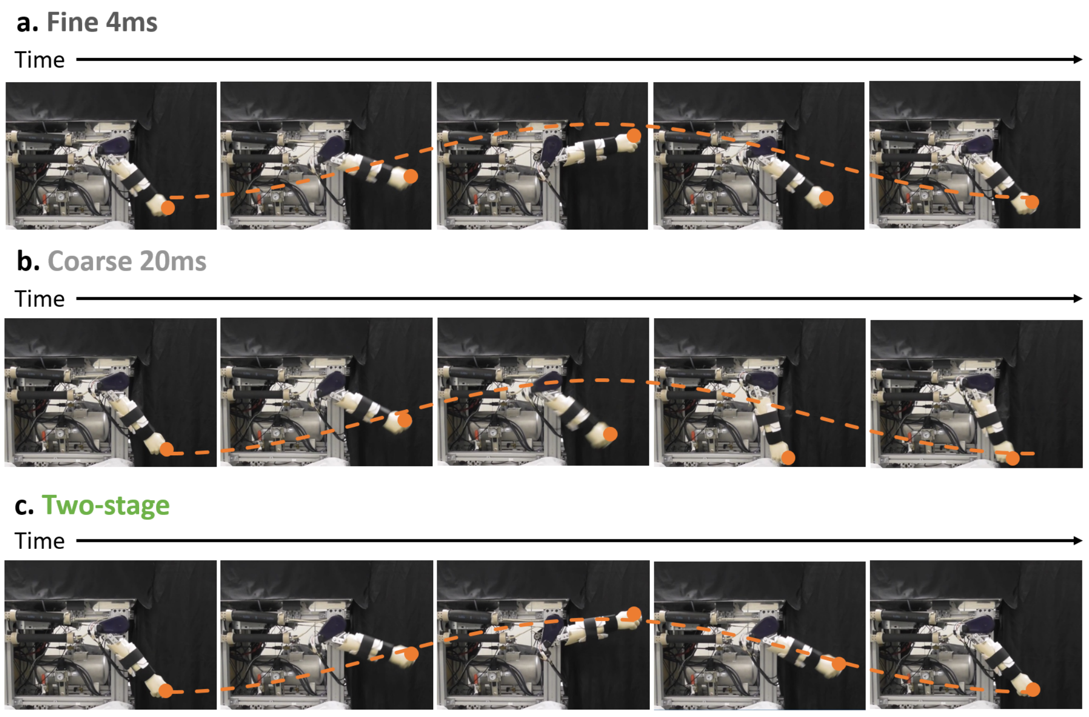


Figure 3.7: Snapshots of generated motions of our forearm robot for target trajectory of 1.0 Hz using three optimal control strategies: Target trajectories are depicted on snapshots by dashed orange lines. (a), (b) for *Fine 4ms* and *Coarse 20ms* strategies, generated joint motions significantly deviated from target movements and failed to recover from failure situations after large deviations. (c) motions generated by proposed method successfully followed target trajectory without any large tracking errors.

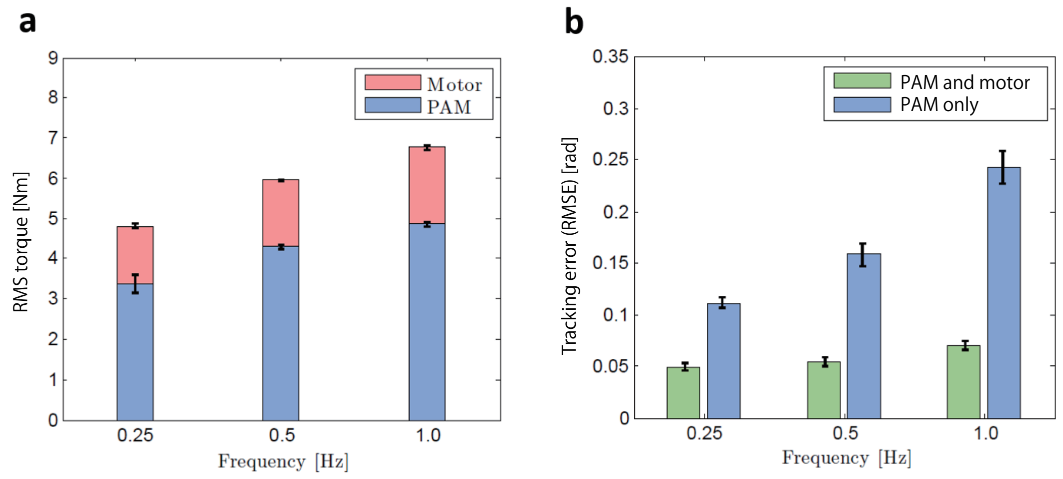


Figure 3.8: Torque distribution: (a) contributions of each actuator to generate joint torques. Contribution of electric motor was much small than that of PAM in terms of amount of generated torque; (b) tracking errors of proposed hybrid control method and PAM-only approach. Results clearly indicated that electric motor greatly improved tracking performances, even though amount of output torque was much smaller than that of PAM.

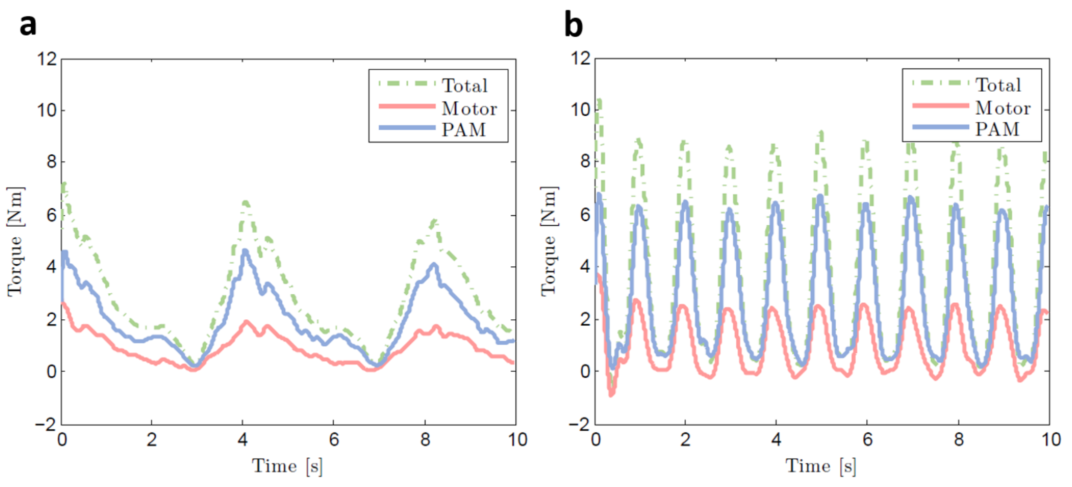


Figure 3.9: Generated joint torque profiles: Generated joint torque profiles presents how torque was dynamically distributed to each actuator for target sinusoidal trajectories of 0.25- and 1.0-Hz frequencies. For both target frequencies, PAM and electric motor were cooperatively activated for target tracking tasks. Moreover, the electric motor sometimes acted like antagonistic muscle in both cases for precise tracking by properly generating force to opposite direction of force generated by PAM.

4 Inverse Optimal Control for Hierarchical MPC

4.1 Introduction

The Optimal Control framework is a powerful approach for motion generation with a diverse set of tasks because such motions can be derived by specifying high-level task goals as objective functions. In practice, Model Predictive Control (MPC), also known as *online* trajectory optimization, is becoming a popular approach [11, 21] for generating a wide variety of robot movements due to the recent major improvements in computing performance as well as optimization algorithms. By repeatedly solving a finite-horizon optimal control problem at each time-step, MPC can effectively provide a feedback policy for even high-dimensional nonlinear systems.

However, for such high-dimensional robots as humanoid robots, designing the objective functions is time-consuming because suitable functions have to be selected by trial and error to connect the task goals and the behaviors of many-DOF robot systems. One possible way of dealing with this problem is to estimate the objective functions from the demonstrations of experts by Inverse Optimal Control (IOC), also known as Inverse Reinforcement Learning (IRL) methods [27, 30]. If an expert can operate an agent [87, 88] or kinesthetically demonstrate the desired trajectories [89], objective functions can be directly estimated from the demonstrations. For the whole-body motion generation of a humanoid robot, however, nobody can be the ideal expert except the robot itself, and it is not practical to perform a kinesthetic demonstration while keeping the robot's balance. Therefore, we learn the objectives using the human movements captured by the behaviors of experts since humans and humanoid robots share similar body structure. For *offline* optimization, an IOC method actually learned the objective functions for biped locomotion from human demonstrations [29], and humanoid robots successfully generated locomotion behaviors with offline calculation [31].

To generate motions with MPC, however, applying the conventional framework used for offline optimization is insufficient. This is because, in the conventional framework, the trade-off between the computational time and MPC's control performance is ignored. Since MPC is a computationally intensive method, for real-time motion generation, IOC should estimate an objective function with a coarse-time resolution. However, coarse objectives may fail to generate a humanoid robot's motions. Therefore, we propose a new framework to generate such motions with MPC using the estimated objective function by IOC and adopt a hierarchical MPC approach in Chapter 2 to cope with the trade-off. While IOC estimates the coarse-objective functions for upper-layer optimization in the hierarchical MPC, lower-layer optimization modifies the results of the first optimization. Therefore, we can achieve real-time motion generation without degrading the control performance. To estimate the coarse objectives for the upper layer in the hierarchical structure, human trajectories are reproduced with the coarse-time resolution before initiating the learning. Fig. 4.1 shows a schematic diagram of the proposed framework, each part of which is described in Section 4.2.

As experiments, we apply our proposed framework to a humanoid robot and show that two different movements, jumping and squatting, can be generated with different objective functions estimated with IOC. Furthermore, we show that both movements can be generated online by real-time calculation with our hierarchical MPC approach. The experimental settings and results are given in Section 4.3.

4.2 Proposed Framework

4.2.1 Reproducing Human Demonstrations

Since the kinematics and the dynamics of humans and robot models are not exactly identical, human demonstrations are not always feasible for robots. Ac-

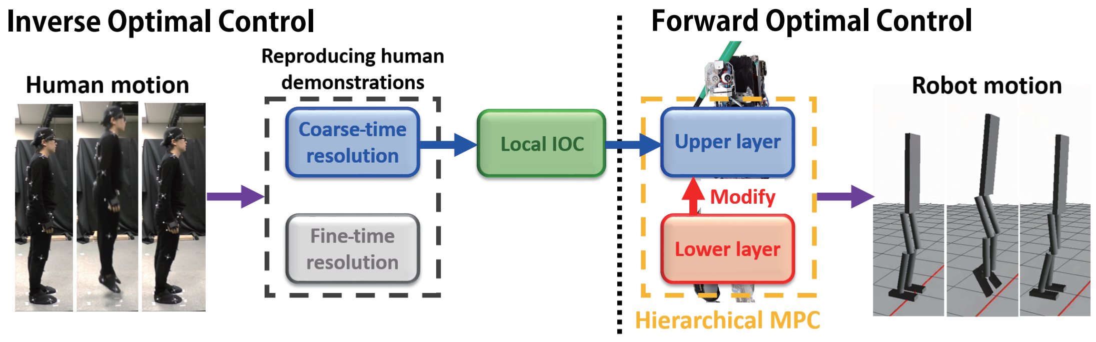


Figure 4.1: Schematic diagram of proposed framework.

cordingly, we cannot directly utilize human joint-angle trajectories to estimate a robot's objective. To deal with this problem, the human behaviors need to be transformed into feasible trajectories for it. Although complicated dancing behaviors were achieved with movement features to convert human movements into robot behaviors [25], this approach is unfavorable for generating a wide variety of robot movements since these movement features need to be carefully hand-tuned for each specific movement task. Therefore, we focus on the time axis as common features among all the movements and adopt the time warping technique. By changing the time lines of the human motions, we convert them to feasible ones for the robot. The time warping technique has been widely used in the past for speech recognition and pattern matching of human movements [90, 91]. It has also recently been successfully applied to motion adaptation [26]. They introduced a motion feasibility index that measures how far a human motion is from the feasible or infeasible region on a humanoid robot. Intuitively, a human pose is feasible if the robot can achieve it without slipping. We can quantify the feasibility by computing the distance between a contact wrench and a friction cone [92].

In the proposed framework, we first use the joint-angle mapping method [93] to modify the kinematic difference and then apply the time warping technique so that the infeasible motions closely approximate the feasible ones. After that, the modified trajectories are tracked with a conventional MPC to minimize the quadratic deviation between the modified joint angles and robot states to reproduce human motions on the robot.

Although tracking control was only used to construct human demonstrations in the previous work [31], we fixed the difference of the kinematics and dynamics before tracking the human trajectories, because in the proposed framework, tracking control is achieved with a coarse-time-step for learning the objective functions with a coarse-time resolution. In this case, the differences between the human and the robot may significantly affect the tracking performances.

4.2.2 Inverse Optimal Control

The objective function can be estimated from the reproduced human movements as the expert's demonstrations.

We consider nonlinear system dynamics described by:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ respectively denote the state and the control vectors.

Let $\mathbf{U}_1 \equiv \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{N-1}\}$ as a control sequence and define total cost $J(\mathbf{x}_1, \mathbf{U}_1)$:

$$J(\mathbf{x}_1, \mathbf{U}_1) = \sum_{i=1}^{N-1} \ell(\mathbf{x}_i, \mathbf{u}_i) + \ell_f(\mathbf{x}_N), \tag{4.2}$$

where $\ell(\mathbf{x}, \mathbf{u})$ is an immediate cost and $\ell_f(\mathbf{x})$ is a terminal cost.

Based on an assumption that an expert performs optimal state trajectories that minimize Eq. (4.2), IOC seeks a cost function to match the expert's behaviors. In particular, we provide features that represent cost function $\mathbf{f}_k : (\mathbf{x}_t, \mathbf{u}_t) \rightarrow \mathbb{R}$, as suggested by standard IOC approaches. Concretely, the cost is represented as a linear combination of weights and features:

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{w}^T \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) = \sum_k w_k \mathbf{f}_k(\mathbf{x}_t, \mathbf{u}_t). \tag{4.3}$$

Then weights $\mathbf{w}$ are estimated with IOC.

Since the captured human data include variant motion trajectories due to the probability of a sequence of actions or noise-contaminated signals, we employed the probabilistic IOC method. The expert's demonstrations are given by

$$\mathcal{D} = \{\{\mathbf{x}_1^1, \mathbf{U}_1^1\}, \{\mathbf{x}_1^2, \mathbf{U}_1^2\}, \cdots \{\mathbf{x}_1^M, \mathbf{U}_1^M\}\}, \tag{4.4}$$

where the number of demonstrated trajectories is denoted as M. Based on probabilistic model $p(\mathcal{D}|\ell) = \prod_m p(\mathbf{U}_1^m|\mathbf{x}_1^m, \ell) = \prod_m \prod_t p(\mathbf{u}_t^m|\mathbf{x}_1^m, \ell)$ that represents the expert's trajectories as a distribution, its likelihood is evaluated to estimate the costs. Since human behaviors are not always perfectly optimal, the trajectory (action) distribution should be wide enough to represent the uncertainty. Such a distribution can be derived based on an expected total cost augmented with entropy term $\mathcal{H}(p)$:

$$E_p[-J(\mathbf{x}_1, \mathbf{U}_1)] + \mathcal{H}(p), \tag{4.5}$$

where the entropy term $\mathcal{H}(p)$ is given by,

$$\mathcal{H}(p) = -\int p(\mathbf{u}_t|\mathbf{x}_t, \ell) \log p(\mathbf{u}_t|\mathbf{x}_t, \ell). \tag{4.6}$$

By maximizing Eq. (4.6), the action distribution at each time-step t is derived:

$$p(\mathbf{u}_t|\mathbf{x}_t, \ell) = \frac{1}{Z_t} e^{-Q_t(\mathbf{x}_t, \mathbf{u}_t)} = e^{-Q_t(\mathbf{x}_t, \mathbf{u}_t)} \left[\int e^{-Q_t(\mathbf{x}_t, \hat{\mathbf{u}}_t)} d\hat{\mathbf{u}}_t \right]^{-1}. \tag{4.7}$$

If the log likelihood of the maximum entropy model is maximized with respect to the cost in Q, the cost functions can be estimated. However, computing partition function Z requires that a forward optimal control problem be repeatedly solved in the learning process. Thus, it is intractable to apply the approach to a high-dimensional system such as a humanoid robot.

Action distribution is approximated with the Laplace approximation to avoid computing the partition function [30]. The model is approximated with a second order Tylor expansion of Q around $\mathbf{u}$:

$$Q(\hat{\mathbf{u}}_t) \approx Q(\mathbf{u}_t) + (\hat{\mathbf{u}}_t - \mathbf{u}_t)^\top Q_{\mathbf{u}t} + \frac{1}{2}(\hat{\mathbf{u}}_t - \mathbf{u}_t)^T Q_{\mathbf{uu}t}(\hat{\mathbf{u}}_t - \mathbf{u}_t). \tag{4.8}$$

We used simple notations $Q(\mathbf{u}_t)$ as $Q_t(\mathbf{x}_t, \mathbf{u}_t)$. After plugging the approximation into Eq. (4.7), the action distribution becomes:

$$\begin{aligned} p(\mathbf{u}_t|\mathbf{x}_t, \ell) &\approx e^{-Q(\mathbf{u}_t)} \left[\int e^{-Q(\mathbf{u}) - (\hat{\mathbf{u}}_t - \mathbf{u}_t)^\top Q_{\mathbf{u}t} - \frac{1}{2}(\hat{\mathbf{u}}_t - \mathbf{u}_t)^T Q_{\mathbf{uu}t}(\hat{\mathbf{u}}_t - \mathbf{u}_t)} d\mathbf{u}_t \right]^{-1} \\ &= \left[\int e^{\frac{1}{2} Q_{\mathbf{u}t}^\top Q_{\mathbf{uu}t}^{-1} Q_{\mathbf{u}t} - \frac{1}{2}(Q_{\mathbf{uu}t}(\hat{\mathbf{u}}_t - \mathbf{u}_t) + Q_{\mathbf{u}t})^\top Q_{\mathbf{uu}t}^{-1}(Q_{\mathbf{uu}t}(\hat{\mathbf{u}}_t - \mathbf{u}_t) + Q_{\mathbf{u}t})} d\mathbf{u}_t \right]^{-1} \\ &= e^{-\frac{1}{2} Q_{\mathbf{u}t}^\top Q_{\mathbf{uu}t}^{-1} Q_{\mathbf{u}t}} |-Q_{\mathbf{uu}t}|^{-\frac{1}{2}} (2\pi)^{\frac{d\mathbf{u}}{2}}. \end{aligned} \tag{4.9}$$

Finally, the cost is estimated by maximizing the sum of the model's log likelihood related to the cost:

$$\log p(\mathcal{D}|\ell) = \sum_{m=1}^{M}\sum_{t=1}^{T}(-\frac{1}{2}Q_{\mathbf{u}m,t}^{\top}Q_{\mathbf{uu}m,t}^{-1}Q_{\mathbf{u}m,t} + \frac{1}{2}\log|Q_{\mathbf{uu}m,t}|). \tag{4.10}$$

In our implementation, we divided a captured motion trajectory with T_{org} steps into N-step trajectories. Trajectory-length T is set to horizon N and $M = D(T_{org}/N)$. The number of demonstrations is denoted as D; we utilized $D = 40$ capture movements each for jumping and squatting in the experiment.

The gradient of the likelihood in Eq. (4.10) for time-step t with respect to cost parameter w_k is

$$\begin{aligned}\nabla_{w_k}\log p(\mathbf{u}_t|\mathbf{x}_t,\ell) = &-Q_{\mathbf{u}t}^{\top}Q_{\mathbf{uu}t}^{-1}[\nabla_{w_k}Q_{\mathbf{u}t}] + \frac{1}{2}Q_{\mathbf{u}t}^{\top}Q_{\mathbf{uu}t}^{-1}[\nabla_{w_k}Q_{\mathbf{uu}t}]Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{u}t}\\ &+ \frac{1}{2}\mathrm{tr}(Q_{\mathbf{uu}t}^{-1}[\nabla_{w_k}Q_{\mathbf{uu}t}]).\end{aligned} \tag{4.11}$$

Gradients $\nabla Q_{\mathbf{u}t}$ and $\nabla Q_{\mathbf{uu}t}$ are given by

$$\nabla Q_{\mathbf{u}t} \quad = \nabla_{w_k}\ell_{\mathbf{u}t} + f_{\mathbf{u}t}^{\top}[\nabla_{w_k}V_{\mathbf{x}t+1}] \tag{4.12}$$

$$\nabla Q_{\mathbf{uu}t} \quad = \nabla_{w_k}\ell_{\mathbf{uu}t} + f_{\mathbf{u}t}^{\top}[\nabla_{w_k}V_{\mathbf{xx}t+1}]f_{\mathbf{u}t}, \tag{4.13}$$

where $\nabla_{w_k}\ell_{\mathbf{u}t}$ and $\nabla_{w_k}\ell_{\mathbf{uu}t}$ are easily obtained when a cost function is in the form of Eq. (4.3):

$$\nabla_{w_k}\ell_{\mathbf{u}t} = \mathbf{f}_{\mathbf{u}tk} \tag{4.14}$$

$$\nabla_{w_k}\ell_{\mathbf{uu}t} = \mathbf{f}_{\mathbf{uu}tk}. \tag{4.15}$$

Based on Eqs. (2.16) and (2.17), the gradients of value functions V are

$$\begin{aligned}\nabla_{w_k}V_{\mathbf{x}t} =&\nabla_{w_k}Q_{\mathbf{x}t} - [\nabla_{w_k}Q_{\mathbf{xu}t}]Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{u}t} - Q_{\mathbf{xu}t}Q_{\mathbf{uu}t}^{-1}[\nabla w_k Q_{\mathbf{u}t}]\\ &+ Q_{\mathbf{xu}t}Q_{\mathbf{uu}t}^{-1}[\nabla_{w_k}Q_{\mathbf{uu}t}]Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{u}t}\end{aligned} \tag{4.16}$$

$$\begin{aligned}\nabla_{w_k}V_{\mathbf{xx}t} =&\nabla_{w_k}Q_{\mathbf{xx}t} - [\nabla_{w_k}Q_{\mathbf{xu}t}]Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{ux}t} - Q_{\mathbf{xu}t}Q_{\mathbf{uu}t}^{-1}[\nabla_{w_k}Q_{\mathbf{ux}t}]\\ &+ Q_{\mathbf{xu}t}Q_{\mathbf{uu}t}^{-1}[\nabla_{w_k}Q_{\mathbf{uu}t}]Q_{\mathbf{uu}t}^{-1}Q_{\mathbf{ux}t},\end{aligned} \tag{4.17}$$

where

$$\begin{aligned}
\nabla Q_{\mathbf{x}t} &= \nabla_{w_k}\ell_{\mathbf{x}t} + f_{\mathbf{x}t}^{\top}[\nabla_{w_k}V_{\mathbf{x}t+1}] && (4.18)\\
\nabla Q_{\mathbf{ux}t} &= \nabla_{w_k}\ell_{\mathbf{ux}t} + f_{\mathbf{u}t}^{\top}[\nabla_{w_k}V_{\mathbf{xx}t+1}]f_{\mathbf{x}t} && (4.19)\\
\nabla Q_{\mathbf{xx}t} &= \nabla_{w_k}\ell_{\mathbf{xx}t} + f_{\mathbf{x}t}^{\top}[\nabla_{w_k}V_{\mathbf{xx}t+1}]f_{\mathbf{x}t}. && (4.20)
\end{aligned}$$

Finally, the weight is updated with the gradient:

$$\nabla_{w_k}\log p(\mathcal{D}|\ell) = \sum_{m=1}^{M}\sum_{t=1}^{T}\nabla_{w_k}\log p(\mathbf{u}_t^m|\mathbf{x}_t^m,\ell). \tag{4.21}$$

A previous work [30] used a gradient-based optimization approach (LBFGS) while regularizing the Hessian Q_{uu} to be positive definite with the Augmented Lagrangian method. We used both approaches to learn the cost functions of the humanoid robot.

4.2.3 Forward Optimal Control

Finally, the estimated objectives are used in a hierarchical MPC method to generate the motions of a humanoid robot. In our hierarchical MPC approach explained in Chapter 2, total-cost J_c is minimized with coarse-time Δt_c in the upper layer:

$$\begin{aligned}
&\min_{\mathbf{U}_k^c} J_k^c(\mathbf{x}_k,\mathbf{U}_k^c) && (4.22)\\
&\text{s.t.}\ \ \mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t^c f_d(\mathbf{x}_k,\mathbf{u}_k^c), && (4.23)
\end{aligned}$$

where $\mathbf{u}^c$ denotes the vector of the control inputs. $\mathbf{U}^c$ denotes a control sequence for the first optimization. The weights in total cost J^c are estimated with IOC. In the second step, total cost J^f is defined to modify the results of the first optimization with fine-time-step Δt_f. An optimal control sequence is composed of the control sequences of both optimizations. Since the coarse-time-step is used in the upper-layer optimization, only the low-dimensional dynamics are considered to minimize total-cost J^f in the lower-layer optimization, and the computational time can be reduced without degrading the control performance (for further details, see Chapter 2). Therefore, we can cope with the trade-off between the computational time and the control performance.

4.3 Simulation Experiment

4.3.1 Cost Function Design and Simulation Settings

We consider a seven-link humanoid robot: a torso link and three in each leg. The robot's height and weight are 1.59 m and 47.0 kg. We applied our proposed framework to learn the objectives for the jumping and squatting movements and generated both of them with the hierarchical MPC. We utilized 40 capture movements each for jumping and squatting in the learning with IOC.

We adopt the same cost features that were used for the simulation experiment in Chapter 2. Control-cost $\ell_{control}$ was the quadratic penalties on joint torques $\boldsymbol{\tau}$:

$$\ell_{control} = \boldsymbol{\tau}^{\top}\mathbf{W}_{control}\boldsymbol{\tau}. \tag{4.24}$$

$\mathbf{W}_{control}$ is $\mathrm{diag}(w_{\mathrm{HAA}}, w_{\mathrm{HFE}}, w_{\mathrm{KFE}}, w_{\mathrm{AFE}}, w_{\mathrm{HAA}}, w_{\mathrm{HFE}}, w_{\mathrm{KFE}}, w_{\mathrm{AFE}})$. w_{HAA} and w_{HFE} are respectively the weights for the hip abduction-adduction and flection-extension. w_{KFE} is for the knee flection-extension, and w_{AFE} is for the ankle flection-extension. The state-cost consists of six terms. The first term, $\ell_{attitude}$, forced the robot to regulate the attitude of the base link to $\mathbf{a}$ rad. In practice, rotations $\mathbf{a}$ were transformed into quaternions $\mathbf{q}(\mathbf{a})$ to compute the cost since we designed this term using unit quaternions:

$$\ell_{attitude} = (\mathbf{q}(\mathbf{a}^r) - \mathbf{q}(\mathbf{a}))^{\top}\mathbf{W}_{attitude}(\mathbf{q}(\mathbf{a}^r) - \mathbf{q}(\mathbf{a})), \tag{4.25}$$

where $\mathbf{q}^r$ stands for the robot's actual position. The weight matrix $\mathbf{W}_{attitude}$ is a 4 by 4 diagonal matrix whose diagonal entries are represented as $w_{\mathrm{q}1}, w_{\mathrm{q}2}$, $w_{\mathrm{q}3}$ as weight parameters for the vector part of the quaternion, and $w_{\mathrm{q}4}$ for the scalar part of the quaternion. Second term ℓ_{height} penalized the vertical position of base p_z^r to be p_z m with weight w_{height}:

$$\ell_{height} = w_{height}(p_z^r - p_z)^2. \tag{4.26}$$

Third term $\ell_{velocity}$ forced the vertical and horizontal velocities of the base link to be $\mathbf{v}$ m/s:

$$\ell_{velocity} = (\mathbf{v}^r - \mathbf{v})^{\top}\mathbf{W}_{velocity}(\mathbf{v}^r - \mathbf{v}). \tag{4.27}$$

Weight matrix $\mathbf{W}_{velocity}$ has diagonal elements w_{vx}, w_{vy}, and w_{vz} for the velocities in the x, y, and z directions, respectively. The fourth and fifth terms are penalties on the angles between the vertical axis and the line connecting each foot and the base. The angles are described in Fig. 4.2 as α_R, α_L in the sagittal plane and β_R, β_L in the frontal plane. Fourth term ℓ_α forced the robot's legs to remain spread apart to maintain its balance:

$$\ell_\alpha = w_\alpha(\alpha_R^r - \alpha_L^r)^2. \tag{4.28}$$

With this term, the robot tries to maintain the position of the base link between its feet. Fifth term ℓ_β prevented a self-collision:

$$\ell_\beta = w_\beta((\beta_R^r - \beta_R^{init})^2 + (\beta_L^r - \beta_L^{init})^2), \tag{4.29}$$

where β_R^{init} and β_L^{init} indicate angles α and β at the initial position. Sixth term ℓ_ω was the quadratic penalties on the angular velocities of the attitude of base link $\boldsymbol{\omega}^r$:

$$\ell_\omega = \boldsymbol{\omega}^{r\top}\mathbf{W}_\omega\boldsymbol{\omega}^r. \tag{4.30}$$

$\mathbf{W}_\omega$ is a 3 by 3 diagonal matrix, whose diagonal entries are $w_{\omega x}, w_{\omega y}$, and $w_{\omega z}$.

We used the reproduced human demonstrations for target attitude $\mathbf{a}$, height p_z, and velocity $\mathbf{v}$ of the base link. The weights of the cost function are estimated by IOC. The weight parameters are

$$\mathbf{w} = \{w_1, \cdots, w_{13}\} = \{\mathbf{w}_{1,2,3,4}, \mathbf{w}_{5,6,7}, \mathbf{w}_{8,9,10}, \mathbf{w}_{11,12,13}\}, \tag{4.31}$$

where

$$\mathbf{w}_{1,2,3,4} = \{w_{\mathrm{HAA}}, w_{\mathrm{HFE}}, w_{\mathrm{KFE}}, w_{\mathrm{AFE}}\}, \tag{4.32}$$

$$\mathbf{w}_{5,6,7} = \{w_\alpha, w_\beta, w_{height}\}, \tag{4.33}$$

$$\mathbf{w}_{8,9,10} = \{w_{q1}, w_{q2}, w_{q3}\}, \tag{4.34}$$

$$\mathbf{w}_{11,12,13} = \{w_{vx}, w_{vy}, w_{vz}\}. \tag{4.35}$$

The weight parameters for the state- and control-costs were initialized with the values of 1.0 and 0.001 in both cases. The weights for the vector part of the quaternion were only estimated. We set w_{q4} to 0. We did not estimate the weights for the angular velocity of the attitude of the base link $w_{\omega x}, w_{\omega y}$ and $w_{\omega z}$. They are all zero in the experiment.

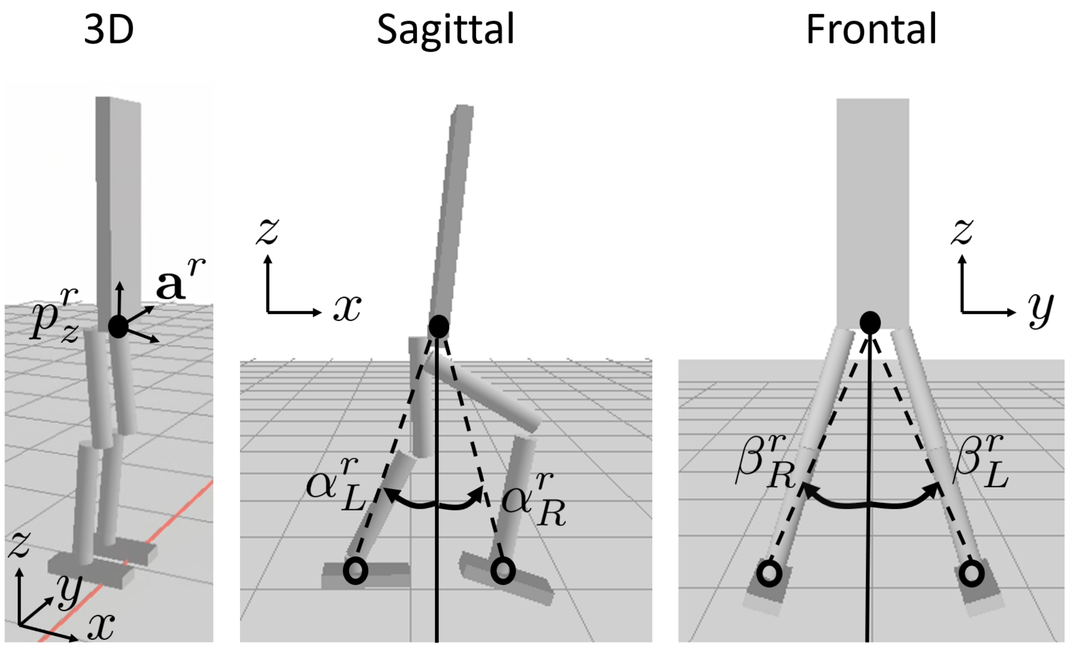


Figure 4.2: Definition of some variables of robot used in cost function.

4.3.2 Simulation Results

Figure 4.3 shows the estimated weights for squatting (pink bars) and jumping (red bars). The blue bars indicate the initial weight setting. The descriptions of weights $w_{1,\cdots,13}$ are explained in Eqs. (4.31) and (4.35). For each movement, different weights were learned from the human demonstrations. The demonstrated jumping movement and the learned behavior with the hierarchical MPC are illustrated in Fig. 4.4, and the demonstrated and learned squatting behaviors are shown in Fig. 4.5. The generated jumping and squatting motions with the initial weights are also represented as the results of the previous learning. The demonstrated behaviors were not utilized for the IOC learning. They were used as target attitude $\mathbf{a}$, height p_z, and velocity $\mathbf{v}$ of the base link for motion generation with the hierarchical MPC. A similar jumping movement was generated using our proposed framework. The robot also successfully generated squatting behavior. Here, since the robot did not have a joint at the torso link, it stood more upright than the human behavior.

The averaged maximum computational time over five motion generations for jumping was 26.3 ms, and the time for squatting was 26.5 ms. Since they were shorter than the sampling time of 30 ms, real-time control was achieved. We applied the hierarchical MPC, which can reduce the computational time, and thus real-time control was achieved.

4.4 Summary

In this chapter, we proposed a framework that combined IOC and MPC for humanoid control. In the proposed estimation framework, the humanoid robot's control objectives were learned by IOC with demonstrated human motions. The proposed framework addresses the difficulty of a humanoid robot's motion generation, since ideally we just capture the human behaviors. The objective functions are estimated with IOC, and then full-body motions that satisfy the objectives are automatically derived with MPC.

To evaluate our proposed method, we applied the proposed framework to a humanoid robot model and generated two different movements, jumping and squatting, with different objective functions estimated with IOC. Furthermore, we

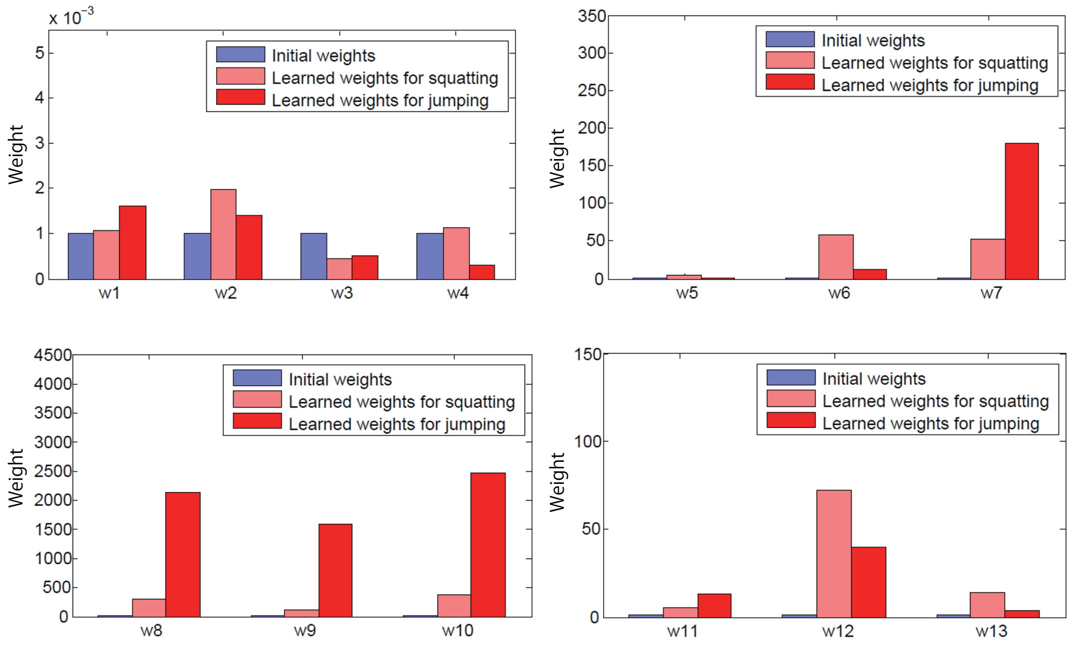
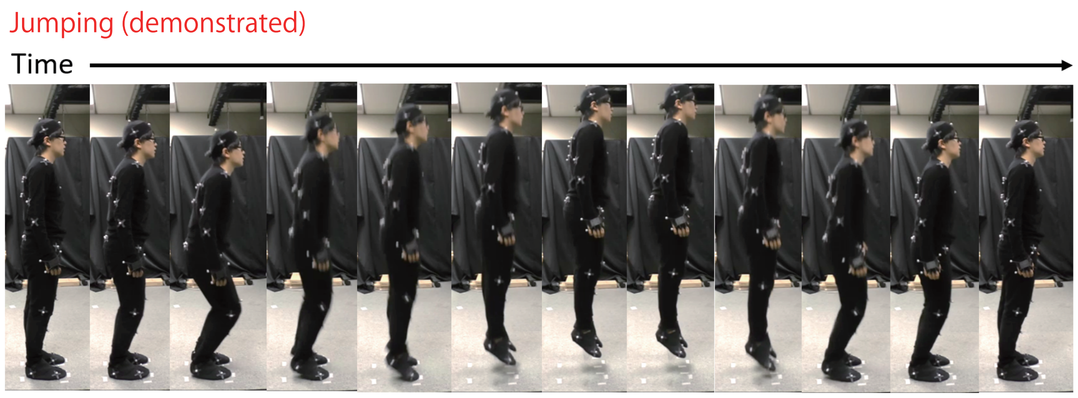
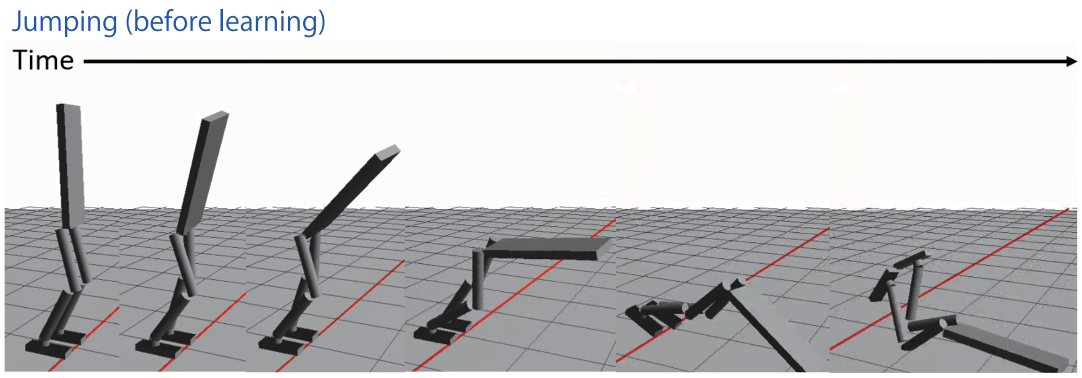


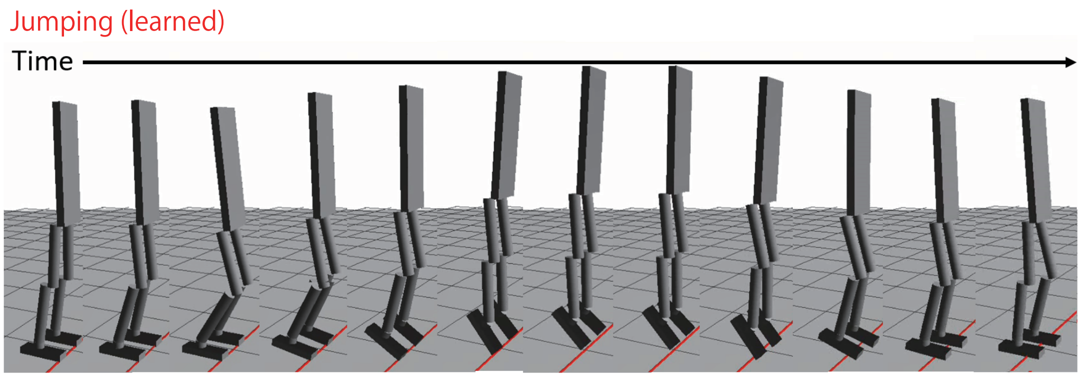
Figure 4.3: Initial and learned weights for squatting and jumping: For each movement, different weights were learned from human demonstrations. With estimated weights, robot successfully generated squatting and jumping motions in real-time with our proposed MPC.



(a) demonstrated jumping movement

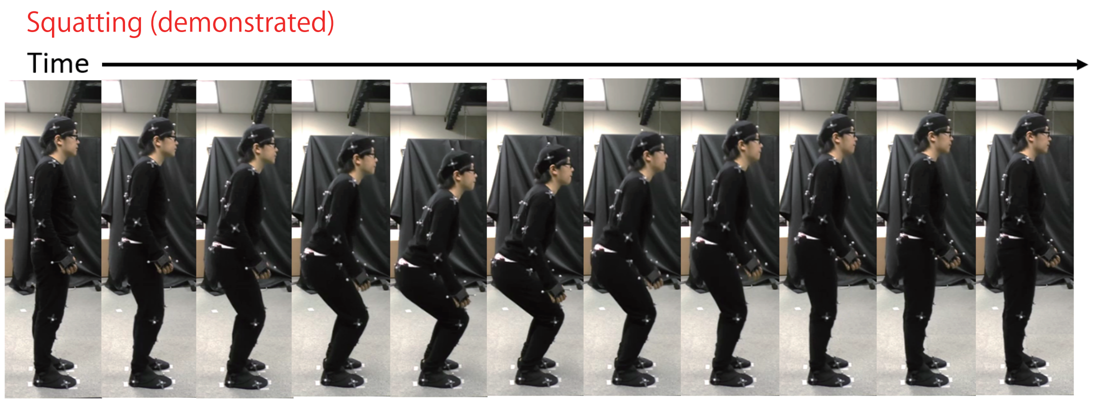


(b) initial jumping movement

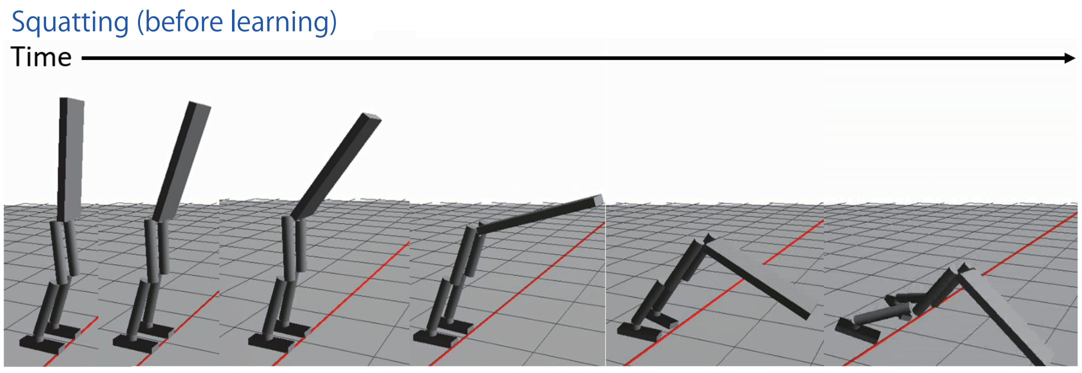


(c) learned jumping movement

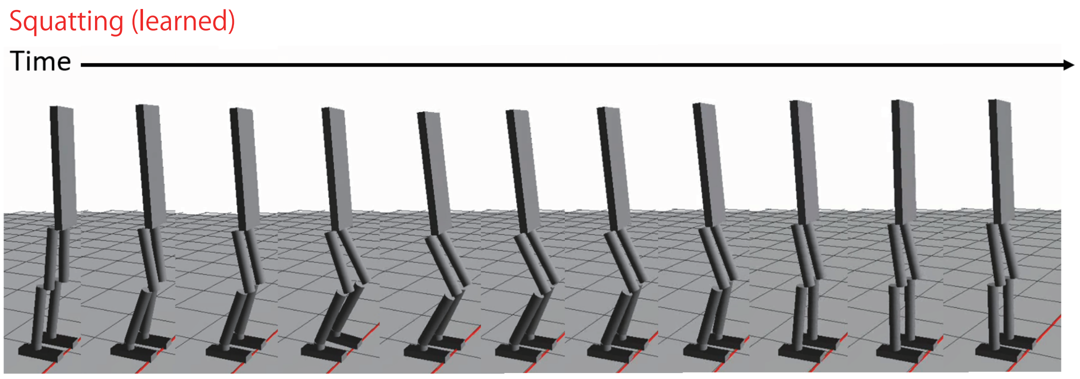
Figure 4.4: Demonstrated and learned jumping behaviors: Using proposed framework, robot successfully generated jumping behavior. Movement resembled the demonstrated jumping movement.



(a) demonstrated squatting



(b) initial squatting



(c) learned squatting

Figure 4.5: Demonstrated and learned squatting behaviors: Robot also generated squatting behavior. Since it did not have a joint at the torso link, the link stood upright.

showed that both movements can be generated in real time with our hierarchical MPC. Future work will apply our proposed method to control our real humanoid robot [53].

5 Conclusion

5.1 Summary

Although a humanoid robot can perform a diverse set of tasks in real environments due to the recent rapid growth in humanoid technology, the execution speed of such tasks is obviously an area that leaves much room for improvement. Humans can easily generate a wide variety of whole-body dynamic movements. On the other hand, it remains difficult for humanoid robots to achieve them. The lack of agility, which is a crucial issue to be solved before humanoid robots can engage in disaster response tasks instead of humans, can be partially attributed to two factors: the current control approach and actuation systems.

MPC is a candidate for an alternative control approach because a wide variety of whole-body robot motions can be derived by specifying high-level task goals as objective functions. The robot efficiently achieves each task's goal because motions can be planned under full-body dynamics without restricting generable movements. A hybrid actuator, which is composed of multiple actuators, is a desirable actuation system in terms of the power-to-weight ratio because even though it is light, it can still generate large torque. Therefore, we develop a control framework (Fig. 1.1) where a unified MPC controller plans whole-body motions based on full-body dynamics and the hybrid actuation system executes them. For developing a framework, this thesis addressed three problems related to MPC, the hybrid actuation system, and the control objectives.

MPC is a powerful approach for motion generation in a diverse set of tasks since such motions can be automatically derived just by designing simple high-level task goals. However, the motion generation of a humanoid robot with MPC has been considered impractical. Since a humanoid is a high-dimensional system that needs to generate fast movements based on the predictions of its long-term future, a large finite optimal control problem is required to be solved within a very short time period. To cope with this problem, in Chapter 2, we proposed an MPC method that combined a hierarchical MPC with a simple warm-start technique for the development for real-time humanoid robot control. In our proposed method, an optimizer was warm-started from the beginning of an optimization. An upper-layer optimization coarsely decided the whole-body motions. The detailed movements were derived in a lower layer based on low-dimensional fast dynamics extracted from a humanoid robot. We formulated two optimization problems to be smaller than the original one to ease MPC's computational burden. We evaluated our proposed MPC in a simulated 3D humanoid model and showed that it reduced the computational time without significantly degrading the control performances. As a result, eight humanoid robot motions were successfully generated in real-time, including such complex behaviors as jumping and flipping. In the whole-body reaching task, our proposed MPC was compared with a conventional whole-body controller in terms of the required time to execute the task. Since the high-level controllers in the conventional controller planned the motion trajectories of the low-dimensional dynamical and kinematic components of the humanoid robot, dynamically switching or blending the controllers to generate whole-body motions was difficult. On the other hand, since our MPC planned the motion trajectories under full-body dynamics, walking to approach the target position and the reaching motion were simultaneously generated. The generated whole-body reaching motion was about twice as fast as the conventional controller. In addition, we evaluated our approach on a real robot and demonstrated that slow squatting can be generated online by switching the control objective from standing.

Humans use multiple muscles to generate such joint movements as elbow motions. With multiple lightweight and compliant actuators, joint movements can also be efficiently generated. Similarly, robots can use multiple actuators to efficiently generate a one-DOF movement for which the desired joint torque must be properly distributed to each actuator. One approach to cope with this torque distribution problem is an optimal control method. However, solving the optimal control problem at each control time-step has not been deemed a practical approach due to its large computational burden. In Chapter 3, we proposed a computationally efficient method to derive an optimal control strategy for a hybrid actuation system where each actuator has different dynamical properties. We investigated a singularly perturbed system of the hybrid actuator model that subdivided the original large-scale control problem into smaller subproblems to derive the optimal control outputs for each actuator at each control time-step and applied our proposed method to our pneumatic-electric hybrid actuator system. Our method derived a torque distribution strategy for the hybrid actuator by dealing with the difficulty of solving real-time optimal control problems.

Designing objective functions for high-dimensional robots, such as humanoid robots, is time-consuming because appropriate objective functions have to be designed through trial and error. In Chapter 4, we derived a hierarchical architecture in both forward and inverse optimal control so that a policy can be derived in real-time using MPC. In the proposed hierarchical architecture, the control objectives for MPC were estimated by IOC, and the learned objectives generated the movements of a humanoid robot. By using captured human expert movements, human movement skills were transferred to a humanoid robot model through the estimated objective functions. To evaluate our proposed method, we applied the proposed framework to a humanoid robot model and showed that two different movements, jumping and squatting, can be generated with different objective functions estimated with IOC. Furthermore, we showed that both movements can be generated in real-time with our hierarchical MPC approach.

5.2 Future Work and Perspectives

Future work will apply our proposed hierarchical MPC to a real robot to generate more dynamic movements such as fast squatting, walking, and push recovery. We empirically found that by selecting time-step sizes and horizons, MPC's computational time was reduced and the humanoid robot successfully generated whole-body motions in real-time with our hierarchical MPC. However, how to properly select them is unclear in the current implementation.

As a theoretical consideration about an optimal horizon, under an appropriate assumption and cost functions, a condition has been derived with respect to the horizon under which the value function is a Lyapunov function [94]. The minimum value under it is an optimal horizon that minimizes MPC's computational burden. However, it is generally difficult for a humanoid robot to provide such a condition because it is a highly nonlinear system. As one possible approach to address this problem, we will develop a method to estimate the horizons and the time-step sizes from a set of motion data. By using the method, we can estimate the required horizon and the step sizes to achieve the demonstrated motions. An elegant approach might learn them as well as weight parameters by IOC because the optimal horizon condition would also be related to the cost functions [94]. By estimating all of the parameters for each hierarchical and conventional MPC, we can compare the computational time and the control performances.

Furthermore, we will evaluate the ideal control framework of Fig. 1.1 on an anthropomorphic robot with hybrid actuators [16, 17, 86]. One possible realization of such a framework is that first, the hierarchical MPC plans the whole-body motion trajectory, as in Chapter 2, and then the desired trajectory is tracked with a two-stage control scheme while distributing the torque to each actuator of the hybrid actuation system, as shown in Chapter 3. The control objectives for the hierarchical MPC are estimated from human demonstrations, as in Chapter 4, to avoid the labor of designing cost functions. However, a great deal of work remains to be done before an ideal control framework can be achieved for whole-body motion generation in a diverse set of tasks.

Although our proposed hierarchical MPC successfully eased MPC's computational burden, further improvements that reduce the computational time are necessary for better control performances or higher-dimensional robots. Since the control performances were task-dependently improved with the hierarchical structure, as shown in Chapter 2, we expect that an optimal policy can be derived effectively if an appropriate hierarchical structure were constructed for each task for further reducing MPC's computational time. Hence, developing a method where the hierarchical structure changes adaptively depending on the current task is one possible direction to generate versatile whole-body motions with a hierarchical MPC. Moreover, an efficient algorithm for forward dynamics computation has been proposed to achieve parallel computation with respect to the robot's links [95]. Since MPC's computational time greatly depends on the speed of the dynamics computation, parallel computation will contribute to its reduction. A different parallel computation scheme with respect to the horizon has already been implemented to compute the dynamics derivatives. Many threads are required if we perform both link- and horizon-wise computation. Therefore, we must develop a parallel algorithm in which the granularity of parallelism can be scheduled for future state trajectories (both the number of links and the horizon) to reduce the number of threads. In addition, the form of the constraints on the control should be appropriately selected. Control limits greatly affect the convergence property of an optimization algorithm [96]. For fast convergence, it is also important to model the robot's dynamics more accurately, for example, using an accurate contact model and its solver [97]. Due to the recent improvements of a nonlinear function approximator with a neural network, deep neural network dynamics [98, 99, 100] might be a good alternative to precisely predict the future state trajectories. However, a trade-off always exists between accuracy and computational speed.

Better system identification will further improve the overall performance. To achieve this, we need to seek better formulations or algorithms to estimate inertial parameters as well as contact forces [101]. More accurate state estimation will also benefit whole-body motion generation. Humanoids should be outfitted with many sensors. By combining all of the sensor information, a reliable state estimation can be achieved [102], for instance, with a tri-modal 3D range sensor named MultiSense SL [103].

Although several sophisticated PAM models have been proposed (e.g., [104, 105]), it remains challenging to operate them at a wide range of pressure levels because of their highly nonlinear behaviors. Thus, an accurate PAM model must be explored in either parametric or nonparametric ways to reduce the modeling errors to achieve better control performances for the two-stage control scheme of the hybrid actuation system.

Since the generated torque patterns of a human are not directly observable, human demonstrations are reproduced on the robot by tracking control before estimating the control objectives. The estimation accuracy greatly depends on the performances of the tracking control. For apprenticeship learning by observing, an IOC algorithm based on expectation-maximization (EM) has been developed that can learn objective functions with hidden actions [106]. However, IOC effectiveness with an EM-algorithm is restricted in low-dimensional systems with discrete state-action space. Developing an IOC method is needed that can estimate with hidden data, which can be applied to a high-dimensional system with continuous state-action space such as a humanoid robot.

Acknowledgements

There are many people I would like to thank for their help and support with my research. First, I would like to offer my special thanks to my direct supervisor Associate Professor Jun Morimoto for his continuous support and greatly helpful advice on my work. For accepting me as a student at ATR, which is a favorable environment to focus on research work, I would like to thank Professor Mitsuo Kawato. I would like to express my heartfelt gratitude to Professor Kenji Sugimoto for his kind instruction and massive support. I will never forget his help and encouragement in my hard time during my Ph.D. course. I also would like to thank other thesis committee members: Professor Tsukasa Ogasawara, Associate Professor Takamitsu Matsubara, Assistant Professor Masaki Ogura and Assistant Professor Taisuke Kobayashi, for their valuable comments and kind support.

I also would like to thank Professor Kazushi Ikeda, Doctor Saori Tanaka and Doctor Okito Yamashita, Affiliate Associate Professor Yuki Minami and Research Assistant Professor Cui Yunduan for their valuable and encouraging advices in student seminars at ATR and in regular meetings at NAIST. I am grateful to the secretary Hideko Hayashi, for assisting me to handle the paperwork at NAIST. I appreciate all students in Intelligent System Control Laboratory at NAIST for their kind help.

I am very grateful for the researchers in Department of Brain Robot Interface (BRI) at ATR, in no particular order: Doctor Eiji Uchibe, Doctor Tomoyuki Noda, Doctor Stefan Elfwing, Doctor Tatsuya Teramae, Doctor Giuseppe Lisi, Doctor Jun-ichiro Furukawa and Doctor Asuka Takai for their supports and advices. I am grateful to the secretaries Kazuko Davis and Naoko Terakawa, for assisting me in many different ways at ATR. I also thank research engineers for their assistance on the maintenance and programing. Furthermore, my thanks would go to all students in BRI and other departments at ATR for their kind help.

My thanks would also go to my beloved parents for their unconditional support and their encouragement. Finally, big thanks to our humanoid robot, CB-i, one of the most challenging robot in the world. To control the humanoid robot was an exceptional experience for me because I had chance to construct the control architecture from a very beginning part.

References

[1] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, C. Orlowski, The darpa robotics challenge finals: Results and perspectives, Journal of Field Robotics 34 (2), 2017, pp. 229–240.

[2] Defense Advanced Research Projects Agency, Darpa robotics challenge finals home, `http://www.theroboticschallenge.org/`, 2015.

[3] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, R. Tedrake, Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot, Autonomous Robots 40 (3), 2016, pp. 429–455.

[4] S. Kim, M. Kim, J. Lee, S. Hwang, J. Chae, B. Park, H. Cho, J. Sim, J. Jung, H. Lee, et al., Approach of team snu to the darpa robotics challenge finals, in: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, IEEE, 2015, pp. 777–784.

[5] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, et al., Team ihmc's lessons learned from the darpa robotics challenge trials, Journal of Field Robotics 32 (2), 2015, pp. 192–208.

[6] S. Feng, X. Xinjilefu, C. G. Atkeson, J. Kim, Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals, in: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, IEEE, 2015, pp. 1028–1035.

[7] J. Morimoto, G. Zeglin, C. G. Atkeson, Minimax differential dynamic programming: Application to a biped walking robot, in: Intelligent Robots

and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Vol. 2, IEEE, 2003, pp. 1927–1932.

[8] E. Todorov, W. Li, A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems, in: Proceedings of the 2005, American Control Conference, 2005., IEEE, 2005, pp. 300–306.

[9] D. Mitrovic, S. Nagashima, S. Klanke, T. Matsubara, S. Vijayakumar, Optimal feedback control for anthropomorphic manipulators, in: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, 2010, pp. 4143–4150.

[10] H. Seguchi, T. Ohtsuka, Nonlinear receding horizon control of an underactuated hovercraft, International journal of robust and nonlinear control 13 (3-4), 2003, pp. 381–398.

[11] Y. Tassa, T. Erez, 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.

[12] V. Kumar, Y. Tassa, T. Erez, E. Todorov, Real-time behaviour synthesis for dynamic hand-manipulation, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 6808–6815.

[13] C. G. Atkeson, B. Babu, N. Banerjee, D. Berenson, C. Bove, X. Cui, M. De-Donato, R. Du, S. Feng, P. Franklin, et al., What happened at the darpa robotics challenge, and why, submitted to the DRC Finals Special Issue of the Journal of Field Robotics.

[14] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, D. G. Caldwell, Design of hyq–a hydraulically and electrically actuated quadruped robot, Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 225 (6), 2011, pp. 831–849.

[15] D. Shin, I. Sardellitti, Y.-L. Park, O. Khatib, M. Cutkosky, Design and control of a bio-inspired human-friendly robot, The International Journal of Robotics Research 29 (5), 2010, pp. 571–584.

[16] T. Noda, N. Sugimoto, J. Furukawa, M.-a. Sato, S.-H. Hyon, J. Morimoto, Brain-controlled exoskeleton robot for bmi rehabilitation, in: Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on, IEEE, 2012, pp. 21–27.

[17] S.-H. Hyon, J. Morimoto, T. Matsubara, T. Noda, M. Kawato, Xor: Hybrid drive exoskeleton robot that can balance, in: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, 2011, pp. 3975–3981.

[18] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa, Biped walking pattern generation by using preview control of zero-moment point, in: Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, Vol. 2, IEEE, 2003, pp. 1620–1626.

[19] P.-B. Wieber, Trajectory free linear model predictive control for stable walking in the presence of strong perturbations, in: 2006 6th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2006, pp. 137–142.

[20] B. J. Stephens, C. G. Atkeson, Push recovery by stepping for humanoid robots with force controlled joints, in: 2010 10th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2010, pp. 52–59.

[21] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, E. Todorov, An integrated system for real-time model predictive control of humanoid robots, in: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), IEEE, 2013, pp. 292–299.

[22] M. Zinn, B. Roth, O. Khatib, J. K. Salisbury, A new actuation approach for human friendly robot design, The international journal of robotics research 23 (4-5), 2004, pp. 379–398.

[23] I. Sardellitti, J. Park, D. Shin, O. Khatib, Air muscle controller design in the distributed macro-mini (dm 2) actuation approach, in: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, IEEE, 2007, pp. 1822–1827.

[24] T. Matsubara, T. Noda, S.-H. Hyon, J. Morimoto, An optimal control approach for hybrid actuator system, in: Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on, IEEE, 2011, pp. 300–305.

[25] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, K. 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), 2007, pp. 829–844.

[26] Y. Zheng, K. Yamane, Adapting human motions to humanoid robots through time warping based on a general motion feasibility index, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2015, pp. 6281–6288.

[27] P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 1.

[28] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning., in: AAAI, Vol. 8, Chicago, IL, USA, 2008, pp. 1433–1438.

[29] K. Mombaur, A. Truong, J.-P. Laumond, From human to humanoid locomotion an inverse optimal control approach, Autonomous robots 28 (3), 2010, pp. 369–383.

[30] S. Levine, V. Koltun, Continuous inverse optimal control with locally optimal examples, in: Proceedings of the 29th International Conference on Machine Learning (ICML-12), 2012, pp. 41–48.

[31] T. Park, S. Levine, Inverse optimal control for humanoid locomotion, in: Robotics Science and Systems Workshop on Inverse Optimal Control and Robotic Learning from Demonstration, 2013.

[32] S. Schaal, C. G. Atkeson, Learning control in robotics, IEEE Robotics & Automation Magazine 17 (2), 2010, pp. 20–29.

[33] S. J. Qin, T. A. Badgwell, An overview of industrial model predictive control technology, in: AIChE Symposium Series, Vol. 93, New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997, pp. 232–256.

[34] R. Scattolini, Architectures for distributed and hierarchical model predictive control–a review, Journal of Process Control 19 (5), 2009, pp. 723–731.

[35] X. Chen, M. Heidarinejad, J. Liu, P. D. Christofides, Composite fast-slow mpc design for nonlinear singularly perturbed systems, AIChE Journal 58 (6), 2012, pp. 1802–1811.

[36] P. Sanila, J. Jacob, Composite fast-slow mpc design for flexible joint manipulator, in: International Conference on Advanced Trends in Engineering and Technology, 2014, pp. 23–29.

[37] D. H. Jacobson, New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach, Journal of Optimization Theory and Applications 2 (6), 1968, pp. 411–440.

[38] Y. Tassa, T. Erez, W. D. Smart, Receding horizon differential dynamic programming, in: Advances in neural information processing systems, 2008, pp. 1465–1472.

[39] Y. Wang, S. Boyd, Fast model predictive control using online optimization, IEEE Transactions on Control Systems Technology 18 (2), 2010, pp. 267–278.

[40] P. Kokotovic, H. K. Khalil, J. O'reilly, Singular perturbation methods in control: analysis and design, Vol. 25, Siam, 1999.

[41] J. Chow, P. Kokotovic, A decomposition of near-optimum regulators for systems with slow and fast modes, IEEE Transactions on Automatic Control 21 (5), 1976, pp. 701–705.

[42] D. Naidu, Singular perturbations and time scales in control theory and applications: an overview, Dynamics of Continuous Discrete and Impulsive Systems Series B 9, 2002, pp. 233–278.

[43] M. N. Zeilinger, C. N. Jones, M. Morari, Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization, IEEE transactions on automatic control 56 (7), 2011, pp. 1524–1534.

[44] T. Erez, Y. Tassa, E. Todorov, Infinite-horizon model predictive control for periodic tasks with contacts, Robotics: Science and systems VII, 2012, pp. 73.

[45] R. Featherstone, Rigid body dynamics algorithms, Springer, 2014.

[46] Y. Fujimoto, A. Kawamura, Simulation of an autonomous biped walking robot including environmental force interaction, IEEE Robotics & Automation Magazine 5 (2), 1998, pp. 33–42.

[47] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.

[48] S. Arimoto, F. Miyazaki, A hierarchical control scheme for biped robots, Journal of the Robotics Society of Japan 1 (3), 1983, pp. 167–175.

[49] T. Erez, Optimal control for autonomous motor behavior, Ph.D. thesis, Washington University, 2011.

[50] OpenMP ARB, Openmp, `http://www.openmp.org/`.

[51] S. Feng, X. Xinjilefu, W. Huang, C. G. Atkeson, 3d walking based on online optimization, in: Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on, IEEE, 2013, pp. 21–27.

[52] S. Feng, E. Whitman, X. Xinjilefu, C. G. Atkeson, Optimization-based full body control for the darpa robotics challenge, Journal of Field Robotics 32 (2), 2015, pp. 293–312.

[53] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, S. C. Jacobsen, Cb: A humanoid research platform for exploring neuroscience, Advanced Robotics 21 (10), 2007, pp. 1097–1114.

[54] S.-H. Hyon, G. Cheng, Gravity compensation and full-body balancing for humanoid robots, in: Humanoid Robots, 2006 6th IEEE-RAS International Conference on, IEEE, 2006, pp. 214–221.

[55] J. Morimoto, G. Endo, J. Nakanishi, G. Cheng, A biologically inspired biped locomotion strategy for humanoid robots: Modulation of sinusoidal patterns by a coupled oscillator model, IEEE Transactions on Robotics 24 (1), 2008, pp. 185–191.

[56] C. G. Atkeson, C. H. An, J. M. Hollerbach, Estimation of inertial parameters of manipulator loads and links, The International Journal of Robotics Research 5 (3), 1986, pp. 101–119.

[57] H. Kawasaki, Y. Beniya, K. Kanzaki, Minimum dynamics parameters of tree structure robot models, in: Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on, IEEE, 1991, pp. 1100–1105.

[58] K. Ayusawa, G. Venture, Y. Nakamura, Identifiability and identification of inertial parameters using the underactuated base-link dynamics for legged multibody systems, The International Journal of Robotics Research 33 (3), 2014, pp. 446–468.

[59] J. Jovic, A. Escande, K. Ayusawa, E. Yoshida, A. Kheddar, G. Venture, Humanoid and human inertia parameter identification using hierarchical optimization, IEEE Transactions on Robotics 32 (3), 2016, pp. 726–735.

[60] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming 45 (1), 1989, pp. 503–528.

[61] Mark Schmidt., minfunc: unconstrained differentiable multivariate optimization in matlab, `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`, 2005.

[62] S. Kolev, E. Todorov, Physically consistent state estimation and system identification for contacts, in: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, IEEE, 2015, pp. 1036–1043.

[63] K. Lowrey, J. Dao, E. Todorov, Real-time state estimation with whole-body multi-contact dynamics: A modified ukf approach, in: Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on, IEEE, 2016, pp. 1225–1232.

[64] T. Erez, Y. Tassa, E. Todorov, Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2015, pp. 4397–4404.

[65] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, R. Siegwart, State estimation for legged robots-consistent fusion of leg kinematics and imu, Robotics 17, 2013, pp. 17–24.

[66] X. Xinjilefu, S. Feng, W. Huang, C. G. Atkeson, Decoupled state estimation for humanoids using full-body dynamics, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 195–201.

[67] K. Masuya, T. Sugihara, Dead reckoning for biped robots that suffers less from foot contact condition based on anchoring pivot estimation, Advanced Robotics 29 (12), 2015, pp. 785–799.

[68] J. L. Crassidis, F. L. Markley, Unscented filtering for spacecraft attitude estimation, Journal of Guidance Control and Dynamics 26 (4), 2003, pp. 536–542.

[69] D. C. Bentivegna, C. G. Atkeson, J.-Y. Kim, Compliant control of a hydraulic humanoid joint, in: Humanoid Robots, 2007 7th IEEE-RAS International Conference on, Ieee, 2007, pp. 483–489.

[70] T. Boaventura, J. Buchli, C. Semini, D. G. Caldwell, Model-based hydraulic impedance control for dynamic robots, IEEE Transactions on Robotics 31 (6), 2015, pp. 1324–1336.

[71] D. Clever, K. Mombaur, An inverse optimal control approach for the transfer of human walking motions in constrained environment to humanoid robots, Proceedings of Robotics: Science and Systems.

[72] A. Ansari, C. G. Atkeson, H. Choset, M. Travers, A survey of current exoskeletons and their control architectures and algorithms (draft 4.0), `http://www.cs.cmu.edu/~cga/exo/survey.pdf`, 2015.

[73] E. Burdet, D. W. Franklin, T. E. Milner, Human robotics: neuromechanics and motor control, MIT Press, 2013.

[74] T. Noda, T. Teramae, B. Ugurlu, J. Morimoto, Development of an upper limb exoskeleton powered via pneumatic electric hybrid actuators with bowden cable, in: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE, 2014, pp. 3573–3578.

[75] T. Teramae, T. Noda, J. Morimoto, Optimal control approach for pneumatic artificial muscle with using pressure-force conversion model, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 4792–4797.

[76] D. Braun, M. Howard, S. Vijayakumar, Optimal variable stiffness control: formulation and application to explosive movement tasks, Autonomous Robots 33 (3), 2012, pp. 237–253.

[77] S. Haddadin, M. Weis, S. Wolf, A. Albu-Schäffer, Optimal control for maximizing link velocity of robotic variable stiffness joints, IFAC Proceedings Volumes 44 (1), 2011, pp. 6863–6871.

[78] P. Lagerstrom, R. Casten, Basic concepts underlying singular perturbation techniques, Siam Review 14 (1), 1972, pp. 63–120.

[79] M. Van Dyke, Perturbation methods in fluid mechanics, Parabolic Press, Incorporated, 1975.

[80] S. Arimoto, F. Miyazaki, A control theoretic study on dynamical biped locomotion, Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME 102, 1980, pp. 233–239.

[81] B. Siciliano, W. J. Book, A singular perturbation approach to control of lightweight flexible manipulators, The International Journal of Robotics Research 7 (4), 1988, pp. 79–90.

[82] C. Makkar, W. Dixon, W. Sawyer, G. Hu, A new continuously differentiable friction model for control systems design, in: Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on, IEEE, 2005, pp. 600–605.

[83] L. Wang, E. H. van Asseldonk, H. van der Kooij, Model predictive control-based gait pattern generation for wearable exoskeletons, in: Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on, IEEE, 2011, pp. 1–6.

[84] P. Mhaskar, N. H. El-Farra, P. D. Christofides, Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control, Systems & Control Letters 55 (8), 2006, pp. 650–659.

[85] A. Jadbabaie, J. Yu, J. Hauser, Unconstrained receding-horizon control of nonlinear systems, IEEE Transactions on Automatic Control 46 (5), 2001, pp. 776–783.

[86] B. Ugurlu, C. Doppmann, M. Hamaya, P. Forni, T. Teramae, T. Noda, J. Morimoto, Variable ankle stiffness improves balance control: experiments on a bipedal exoskeleton, IEEE/ASME Transactions on mechatronics 21 (1), 2016, pp. 79–87.

[87] P. Abbeel, A. Coates, M. Quigley, A. Y. Ng, An application of reinforcement learning to aerobatic helicopter flight, in: Advances in neural information processing systems, 2007, pp. 1–8.

[88] S. Levine, Z. Popovic, V. Koltun, Nonlinear inverse reinforcement learning with gaussian processes, in: Advances in Neural Information Processing Systems, 2011, pp. 19–27.

[89] M. Kalakrishnan, P. Pastor, L. Righetti, S. Schaal, Learning objective functions for manipulation, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 1331–1336.

[90] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE transactions on acoustics, speech, and signal processing 26 (1), 1978, pp. 43–49.

[91] L. Wang, W. Hu, T. Tan, Recent developments in human motion analysis, Pattern recognition 36 (3), 2003, pp. 585–601.

[92] Y. Zheng, C.-M. Chew, Distance between a point and a convex cone in n-dimensional space: Computation and applications, IEEE Transactions on Robotics 25 (6), 2009, pp. 1397–1412.

[93] K. Yamane, J. Hodgins, Control-aware mapping of human motion data with stepping for humanoid robots, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 726–733.

[94] S. E. Tuna, M. J. Messina, A. R. Teel, Shorter horizons for model predictive control, in: American Control Conference, 2006, IEEE, 2006, pp. 6–pp.

[95] K. Yamane, Y. Nakamura, Efficient parallel dynamics computation of human figures, in: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, Vol. 1, IEEE, 2002, pp. 530–537.

[96] Y. Tassa, N. Mansard, E. Todorov, Control-limited differential dynamic programming, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 1168–1175.

[97] E. Todorov, Implicit nonlinear complementarity: A new approach to contact dynamics, in: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, 2010, pp. 2322–2329.

[98] I. Lenz, R. A. Knepper, A. Saxena, Deepmpc: Learning deep latent features for model predictive control., in: Robotics: Science and Systems, 2015.

[99] A. Punjani, P. Abbeel, Deep learning helicopter dynamics models, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2015, pp. 3223–3230.

[100] A. Nagabandi, G. Kahn, R. S. Fearing, S. Levine, Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, arXiv preprint arXiv:1708.02596.

[101] N. Fazeli, R. Kolbert, R. Tedrake, A. Rodriguez, Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact, The International Journal of Robotics Research 36 (13-14), 2017, pp. 1437–1454.

[102] M. F. Fallon, M. Antone, N. Roy, S. Teller, Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing, in: Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on, IEEE, 2014, pp. 112–119.

[103] Carnegie Robotics, LLC, Multisense sl, `https://carnegierobotics.com/multisense-sl/`.

[104] T. Vo-Minh, T. Tjahjowidodo, H. Ramon, H. Van Brussel, A new approach to modeling hysteresis in a pneumatic artificial muscle using the maxwell-slip model, IEEE/ASME Transactions on Mechatronics 16 (1), 2011, pp. 177–186.

[105] B. Ugurlu, P. Forni, C. Doppmann, J. Morimoto, Torque and variable stiffness control for antagonistically driven pneumatic muscle actuators via a stable force feedback controller, in: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, 2015, pp. 1633–1639.

[106] K. Bogert, J. F.-S. Lin, P. Doshi, D. Kulic, Expectation-maximization for inverse reinforcement learning with hidden data, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1034–1042.

Publication List

Journal

[1] Koji Ishihara and Jun Morimoto, "An optimal control strategy for hybrid actuator systems: Application to an artificial muscle with electric motor assist", Neural Networks, 99C, pp. 92-100, 2018 (to appear).

International Conference

[1] Koji Ishihara and Jun Morimoto, "Real-Time Model Predictive Control with Two-Step Optimization Based on Singularly Perturbed System", 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids 2015), pp. 173-180, Nov. 3–5, 2015.
[2] Koji Ishihara and Jun Morimoto, "A Hierarchical Model Predictive Control Approach to Generate Biped Robot Movements in Real-time", IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR2016) Workshop, Dec. 12–16, 2016.
[3] Koji Ishihara and Jun Morimoto, "A Forward and Inverse Optimal Control Framework to Generate Humanoid Robot Movements with Hierarchical MPC", In Proceedings of the Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM), p. 36, Jun. 11-14, 2017.

Domestic Conference

[1] 石原 弘二, 寺前 達也, 野田 智之, 森本 淳, "特性の異なるアクチュエータを持つロボットのためのモデル予測制御", 第 32 回日本ロボット学会学術講演会 (RSJ2014), 2I2-06 (DVD), 九州産業大学, 9 月 4 日-6 日, 2014.

Patent

[1] 石原　弘二, 森本　淳, 野田 智之, 寺前 達也, "駆動システム", 特開 2016-53824. 2016-04-14.