공학석사 학위논문

로봇 시스템의 설계 및 동작 동시 최적화

Simultaneous Design and Motion Optimization for Robot Systems

2020년 8월

서울대학교 대학원

기계항공공학부

김 승 현

ABSTRACT

Simultaneous Design and Motion Optimization for Robot Systems

by

Seunghyun Kim

Department of Mechanical Engineering
Seoul National University

A robot design has the potential for numerous combinations of the components such as the actuators, links, joints, *etc*. Therefore, a process of finding a good design is a challenging problem even for the robot experts. To overcome this difficulty, we present an optimization framework for the morphological shape of a robot, considering its motion. Both the design and motion parameters can be simultaneously optimized for specific tasks by our methodology. In the space where the design and motion parameters are combined, our framework seeks the steepest direction that reduces the objective function on the constraint manifold. To

overcome the flaws of the previous studies, we utilize the recently discovered recursive differential dynamics, which informs of the analytic relationship between the variation of joint torques and design parameters, thus our framework brings faster and more accurate optimization results. We validate our optimization framework through two numerical experiments: the 2-R planar manipulator with a given end-effector trajectory and the quadruped robot with a locomotion task.

Keywords: Robot Design Optimization, Rigid Body Dynamics, Optimal Control, Legged Robot

Student Number: 2018-20883

Contents

List of Tables

List of Figures

1 Introduction

1.1 Design Optimization of Robotic Devices

Designing a high-performance robot is a highly exhausting procedure that needs to consider a huge number of parameters and underlying connections between them. Since it is hard to discover a golden rule to build a satisfactory design, the roboticists usually conduct experiments repeatedly with various possible designs and choose the best one among the rest. To overcome this difficult and tedious process, numerous studies about robot design optimization have been conducted. The design of a robot has diverse components such as topology, geometry, structure, inertia, compliance, actuator, etc. Among them, the geometry of the robot, *i.e.,* the lengths of the links or the axes of the joints, is often regarded as a critical factor due to the ease of alteration and large design space. Therefore, most robot design optimization studies set a design parameter as geometry, which is often called a kinematic design.

In the robot kinematic design, robots used to be considered that they need to

have versatility, not to perform some specific tasks. Therefore, the robot design optimization frameworks had also been devised to pursue this philosophy. Two of the most wide-spread robot design benchmarks involved with the above intention are workspace volume and dexterity [1, 2, 3, 4]. The former index denotes the volume of the region that can be reached by the end-effector of the robot, and it is desirable to have a large workspace volume for versatile operation. Second, dexterity implies an ability to generate motions in arbitrary directions. Investigating both of these performance criteria is essential in the robot design process if the robot designer wants their artifacts to be versatile. In the real application, however, robots usually execute only certain tasks for a long time. If we look inside the industrial sites, after the installation of the robots that are capable of versatile manipulation, they just continue to perform repetitive task such as pick and place, not the various operations. Even the legged robots which do not perform only one operation like the above example, there are few crucial tasks that mostly affect the ability of the robots. Therefore, during the optimization of the robot, it is important for the robots to take the frequently executing behaviors into consideration. A number of studies following this perspective, by which we denote task-specific robot design optimization, have been attempted to optimize a morphology of legged robots [5, 6], serial manipulators [7, 8], parallel robots [9], modular robots [10, 11], and cable-driven mechanisms [12] for a given task.

On the other hand, robot movements are becoming more and more dynamic. Parallel robots for high-speed manipulation have been devised and are widely installed in many industrial fields. One of the biggest obstacles for the parallel manipulators is reducing the shaking forces and moments, which produce unwanted vibration [13]. Furthermore, the mobile robots which had previously been in the research stage, are commercializing nowadays [14]. Since they have to be operated

by the mounted battery, ways to save energy must be examined. For these reasons, considering the dynamic performances such as energy consumption in the robot design process is becoming more and more important. In this thesis, we propose *the robot kinematic design optimization based on the dynamic performance criteria for the specific task.*

We can formulate the presented robot design optimization problem as an expanded version of a classical optimal control problem. A formal definition of the optimal control can be described as follows:

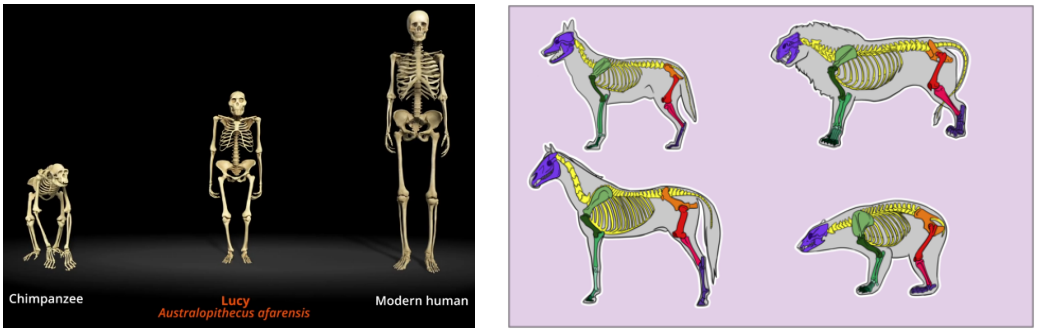
$$\begin{aligned}(x^*, \tau^*) = \underset{x,\,\tau}{\arg\min} \quad & J(x, \dot{x}, \tau) = \int_0^{t_f} r(t, x, \dot{x}, \tau; \rho) dt \\ \text{subject to} \quad & \dot{x} = f(t, x, \tau; \rho), \\ & g(t, x, \dot{x}, \tau; \rho) \leq 0, \\ & h(t, x, \dot{x}, \tau; \rho) = 0\end{aligned} \tag{1.1.1}$$

where x is a state, τ is an input, ρ is a design parameter, and t is a time variable. The terms J and (g, h), stand for the dynamic performance criteria and the constraints to accomplish given tasks, respectively. In the above problem, the design parameter ρ is considered to be fixed, *i.e.,* the mechanism cannot modify its shape but the input. In the robot design optimization, however, ρ has to be able to change, then Equation 1.1.1 becomes

$$\begin{aligned}(x^*, \tau^*, \rho^*) = \underset{x,\,\tau,\,\rho}{\arg\min} \quad & J(x, \dot{x}, \tau, \rho) = \int_0^{t_f} r(t, x, \dot{x}, \tau, \rho) dt \\ \text{subject to} \quad & \dot{x} = f(t, x, \tau, \rho), \\ & g(t, x, \dot{x}, \tau, \rho) \leq 0, \\ & h(t, x, \dot{x}, \tau, \rho) = 0,\end{aligned} \tag{1.1.2}$$

which implies this problem attempts to optimize not only the control but also the design of the robot.

Such a simultaneous optimization scheme can also be found in nature. Several studies have describes the interaction of the shape and motion in the evolution of animals [15, 16]. Figure 1.1 shows the morphological difference between species. It comes from the synergy of the body and the brain to adapt better to nature. As we can see in Figure 1.1a, human beings have been evolved to properly walk with two legs. Figure 1.1b shows the difference between each quadruped mammal. Each animal has optimized its own morphology to suit its own behavior.



(a) Anatomy of Primates (b) Anatomy of Mammals

Figure 1.1: Examples of simultaneous morphology and motion evolution in nature.

1.2 Limitations of Previous Works

Most similar research to our simultaneous design and motion optimization approach is [17]. They presented computational schemes to concurrently optimizing both the design and motion of a robot for a certain task. However, due to the complexity of the equations of motion, it gets harder to differentiate them analytically as the mechanism becomes more complicated. Hence, the previous studies including [17] adopted finite differentiation of the dynamic equations with respect

to several parameters. In the optimization process, the knowledge of analytic differentiation plays an important role for the good result. First, calculating the finite difference takes lots of time and computing power. This flaw becomes worse as the dimension becomes larger. Second, even the finite differentiation is well calculated, that gradient differs from the real one so that the result may be inaccurate and the process becomes slow.

Recently, recursive differential dynamics [18] which can analytically differentiate the equations of motion with respect to joint screws has been developed. They utilized the fact that joint screws can be locally parameterized. Joint screws can be regarded as design parameters which define the morphology of the robot. In this thesis, we use the recursive differential dynamics, thus can calculate each differential term analytically.

1.3 Main Contributions of This Thesis

This thesis proposes a simultaneous design and motion optimization framework for robot systems. We specifically focus on the kinematic design of the robot with the dynamic performance criteria. The optimization proceeds by focusing on the part that performs a particular action rather than various tasks.

Starting with the classical optimal control problem, our robot design optimization is formulated by including the design parameter to the optimization variable. Unlike other prior studies, we use analytic derivatives which can be computed by the recursive differential dynamics [18], and this results in fast and accurate optimization outcomes. The analytic gradients are calculated in a recursive manner, thus our framework has expandability to the complex robot structures.

The remainder of this thesis is organized as follows. In Chapter 2, we review

the basics of rigid body dynamics. Based on the matrix Lie group theory, the equations of motion of the serial manipulator and floating body system are described. Further, the recursive differential dynamics for the analytic differentiation of the dynamic equations with respect to the design parameter is introduced. In Chapter 3, we describe our design optimization scheme. Both design and motion parameters are concurrently optimized to reduce the given cost function with our method. Chapter 4 shows the optimization results and verification in the physics simulator for a 2-R planar manipulator with some given trajectories and a quadruped robot with locomotion task. Chapter 5 discusses the key properties, limitations, and possible extensions of our method.

2

Preliminaries

This chapter reviews some core concepts of our robot design optimization methodology. We first begin with Lie group theory in section 2.1. Then, we review the basic concepts of rigid body dynamics based on Lie group theory in sections 2.2. Recursive differential dynamics, which guides to compute the derivatives of the equations of motion with respect to the design parameters analytically, is stated in section 2.3.

2.1 Lie Group Theory

The kinematics and dynamics of serial chain robots can be represented using the product of exponentials (PoE) formula. In this paper, we establish our design optimization framework based on this formula. This section briefly reviews the basic concepts of Lie group theory to understand PoE formula. Further plentiful discussions about this concept are in [19, 20, 21].

2.1.1 $SO(3)$ and $SE(3)$

The three-dimensional Special Orthogonal Group, $SO(3)$ for brevity, is Lie group and represents the rotation of a rigid body in three-dimensional space. $SO(3)$ is the set of matrices as follows:

$$SO(3) = \{R \in \mathbb{R}^{3\times3} | RR^\top = I, \det(R) = 1\}. \tag{2.1.1}$$

Lie algebra of $SO(3)$, denoted by $so(3)$, is a set of 3×3 real skew-symmetric matrices:

$$so(3) = \{\Omega \in \mathbb{R}^{3\times3} | \Omega^\top + \Omega = 0\}. \tag{2.1.2}$$

An element of $so(3)$ can also be represented as a three-dimensional real vector. Let $\omega = (\omega_1, \omega_2, \omega_3)$ be an element of $\mathbb{R}^3$. The skew-symmetric representation of the given vector can be expressed as

$$[\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{2.1.3}$$

The three-dimensional Special Euclidean Group $SE(3)$ is also Lie group and denotes the rigid body motion in three-dimensional space. $SE(3)$ consists of 4×4 real matrices of the form

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, R \in SO(3), p \in \mathbb{R}^3. \tag{2.1.4}$$

The corresponding Lie algebra $se(3)$ is of the form

$$[S] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{2.1.5}$$

where $[\omega] \in so(3)$ and $v \in \mathbb{R}^3$. Similar to the case of $so(3)$, S is a six-dimensional real vector form of $S = (\omega^\top, v^\top)^\top$.

The Lie group and Lie algebra have a relationship of exponential mapping. Given $S = (\omega^\top, v^\top)^\top \in se(3)$, the corresponding Lie group $T \in SE(3)$ can be expressed as follows:

$$T = e^{[S]} = \begin{bmatrix} e^{[\omega]} & G(\omega)v \\ 0 & 1 \end{bmatrix}, \tag{2.1.6}$$

where $e^{[w]}$ is a matrix exponential and $G(w)$ is

$$G(w) = I + \frac{1}{2!}[\omega] + \frac{1}{3!}[\omega]^2 + \frac{1}{4!}[\omega]^3 + \cdots \tag{2.1.7}$$

If the screw is normalized, *i.e.,* $S = \hat{S}\theta = (\hat{\omega}^\top, \hat{v}^\top)^\top \theta$ where $||\hat{\omega}|| = 1$ and $\omega = \hat{\omega}\theta$, the equation 2.1.6 and 2.1.7 transform into

$$T = \begin{bmatrix} e^{[\hat{\omega}]\theta} & G(\hat{\omega}, \theta)\hat{v} \\ 0 & 1 \end{bmatrix}, \tag{2.1.8}$$

$$G(\hat{\omega}, \theta) = I\theta + (1 - \cos\theta)[\hat{\omega}] + (\theta - \sin\theta)[\hat{\omega}]^2.$$

The physical meaning of the exponential mapping from $se(3)$ to $SE(3)$ can be interpreted as a screw motion, *i.e.,* the rigid body transformation by the screw S. In more details, let T_a and T_b be the coordinate frames of $\{A\}$ and $\{B\}$ with respect to the reference frame $\{0\}$. The rigid body transformation from $\{A\}$ to $\{B\}$ can be represented by a screw $S = \hat{S}\theta = (\hat{\omega}^\top, \hat{v}^\top)^\top \theta$,

$$T_b = T_a e^{[\hat{S}]\theta},$$

$$e^{[\hat{S}]\theta} = \begin{bmatrix} e^{[\hat{\omega}]\theta} & (I - e^{[\hat{\omega}]\theta})q + h\theta\hat{\omega} \\ 0 & 1 \end{bmatrix}, \tag{2.1.9}$$

which indicates the frame $\{A\}$ rotates with respect to axis $\hat{\omega}$ passing through the point q by the angle θ, and translates by the vector $(I - e^{[\hat{\omega}]\theta})q + h\theta\hat{\omega}$. The scalar h represents the pitch and $\hat{v}$ is determined to be $\hat{v} = -\hat{\omega} \times q + h\hat{\omega}$.

2.1.2 Twists and Wrenches

Consider a moving frame whose trajectory is given by

$$T(t) = \begin{bmatrix} R(t) & p(t) \\ 0 & 1 \end{bmatrix} \in SE(3). \tag{2.1.10}$$

A generalized velocity or a twist can be defined as

$$V = T^{-1}\dot{T} = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}, \tag{2.1.11}$$

where $[\omega] = R^\top \dot{R}$ and $v = R^\top \dot{p}$. The twist is included in $se(3)$ and also can be represented in a six-dimensional vector:

$$V = \begin{bmatrix} \omega \\ v \end{bmatrix}. \tag{2.1.12}$$

$\omega, v \in \mathbb{R}^3$ denote the angular velocity and linear velocity of the moving frame, respectively.

A generalized force or a wrench acting on a rigid body can be defined as

$$F = \begin{bmatrix} m \\ f \end{bmatrix}, \tag{2.1.13}$$

where $m, f \in \mathbb{R}^3$ indicate the moment and force, respectively. The wrench F is known as an element of $se^*(3)$, the dual space of $se(3)$, since $F^\top V$ has a unit of work.

2.1.3 Adjoint Mappings

Given $T = (R, p) \in SE(3)$, the large adjoint mapping $\mathrm{Ad}_T : se(3) \to se(3)$ is defined as follows:

$$\mathrm{Ad}_T([\mathcal{V}]) = T[V]T^{-1}, \tag{2.1.14}$$

which can also be regarded as a linear operator of the form

$$\mathrm{Ad}_T(V) = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix}. \tag{2.1.15}$$

The physical meaning of a large adjoint mapping is a coordinate transformation. Let V_a and V_b denote the twist of $T(t) \in SE(3)$ with respect to the different reference frames. Then, they have the following relation:

$$V_b = [\mathrm{Ad}_{T_{ba}}]V_a \text{ and } V_a = [\mathrm{Ad}_{T_{ab}}]V_b. \tag{2.1.16}$$

Similarly, given $A = (\omega^\top, v^\top)^\top \in se(3)$, the small adjoint mapping $\mathrm{ad}_A : se(3) \to se(3)$ can be defined by

$$\mathrm{ad}_A(B) = [A][B] - [B][A] \quad \text{or} \quad \begin{bmatrix} [\omega_A] & 0 \\ [v_A] & [\omega_A] \end{bmatrix} \begin{bmatrix} \omega_B \\ \omega_A \end{bmatrix}. \tag{2.1.17}$$

2.2 Rigid Body Dynamics

This section describes the dynamics of rigid articulated bodies. Also, an algorithm for recursively calculating kinematic and dynamic elements of a serial chain is presented. Further detailed information about rigid body dynamics can be found in [19, 22, 23].

2.2.1 Dynamics of a Single Rigid Body

Before dealing with the dynamic equation of articulated bodies, we first introduce the equations of motion (EoM) of a single rigid body. Assume the body reference frame $\{c\}$ is attached to the center of mass (COM). Then, the equations of motion of a single rigid body is of the form

$$F_c = G_c\dot{V}_c - \mathrm{ad}^\top_{V_c}(G_cV_c), \tag{2.2.18}$$

where G_c, V_c, and F_c denote the generalized inertia, twist, and externally applied wrench with respect to the frame $\{c\}$, respectively. The generalized inertia G_c can be represented as

$$G_c = \begin{bmatrix} \mathcal{I}_c & 0 \\ 0 & I_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times6}, \tag{2.2.19}$$

where $\mathcal{I}_c \in \mathbb{R}_{3\times3}$ is a rotational inertia of the rigid body with respect to the center of mass frame, and $I_{3\times3}$ is the 3×3 identity matrix. Equation 2.2.18 can be identically expressed with respect to another body reference frame $\{b\}$ with coordinate transformation,

$$F_b = G_b\dot{V}_b - \mathrm{ad}^{\top}_{V_b}(G_bV_b), \tag{2.2.20}$$

and the coordinate transformation of each component,

$$V_b = \mathrm{Ad}_{T_{bc}}(V_c), \tag{2.2.21}$$

$$\dot{V}_b = \mathrm{Ad}_{T_{bc}}(\dot{V}_c), \tag{2.2.22}$$

$$F_b = \mathrm{Ad}^{\top}_{T^{-1}_{bc}}(F_c), \tag{2.2.23}$$

$$G_b = [\mathrm{Ad}_{T_{cb}}]^{\top}G_c[\mathrm{Ad}_{T_{cb}}]. \tag{2.2.24}$$

2.2.2 Dynamics of Open Chains

Taking as a point of departure the above dynamic equation of a single rigid body, recursive inverse dynamics algorithm can be derived for computing dynamic elements of each rigid body composing a serial open chain mechanism. Consider an n-dof serial manipulator whose base is fixed to the ground. Algorithm 1 describes Newton-Euler recursive inverse dynamics: given joint variables (θ, $\dot{\theta}$, $\ddot{\theta}$), compute (V_i, $\dot{V}_i$, F_i) of each body-attached frame and τ_i of each joint. The subscript i denotes the joint index. The recursive inverse dynamics algorithm consists of two parts: the forward and the backward iteration step. During the forward iteration,

each joint twist and derivative of twist is calculated, while in the backward iteration the wrenches and torques are computed from the end-effector to the base. More details and proof of the algorithm can be found in [19].

Algorithm 1 Recursive Inverse Dynamics

Input: $\theta,\ \dot{\theta},\ \ddot{\theta}$

Output: $V_i,\ \dot{V}_i,\ F_i,\ \tau_i$

Initialize: $V_0 = 0,\ \dot{V} = -g\ F_{n+1} = F_{ext}$
procedure - forward recursion
for $i = 1 : n$ **do**
    $T_{i-1,i} = e^{[A_i]\theta_i}$
    $V_i = A_i\dot{\theta}_i + [\mathrm{Ad}_{T_{i-1,i}^{-1}}]V_{i-1}$
    $\dot{V}_i = A_i\ddot{\theta}_i + [\mathrm{Ad}_{T_{i-1,i}^{-1}}]\dot{V}_{i-1} + [\mathrm{ad}_{V_i}]A_i\dot{\theta}_i$
end for
procedure - backward recursion
for $i = n : 1$ **do**
    $F_i = [\mathrm{Ad}_{T_{i,i+1}^{-1}}]^\top F_{i+1} + G_i\dot{V}_i - [\mathrm{ad}_{V_i}]^\top G_i V_i$
    $\tau_i = A_i^\top F_i$
end for

There exist a few more forms of equations of motion for a serial manipulator. The EoM of the mechanism can be determined in a closed-form:

$$\tau = M(\theta)\ddot{\theta} + C(\theta,\dot{\theta}) + J(\theta)^\top F_{ext}, \tag{2.2.25}$$

where $M(\theta)$, $C(\theta,\dot{\theta})$, and $J(\theta)$ denote the mass matrix, Coriolis and gravitational

forces, and Jacobian of the contact point, respectively. Moreover, the above dynamic equation can be reformulated from the fact that the joint torque has a linear relationship with the inertia:

$$\tau = Y(\theta, \dot{\theta}, \ddot{\theta})\Phi + J(\theta)^\top F, \tag{2.2.26}$$

where $Y \in \mathbb{R}^{n \times 10n}$ is the regressor and $\Phi \in \mathbb{R}^{10n}$ is the augmented vector of the link inertias. Note that in Equation 2.2.26, the kinematic and inertial parameters can be separated into Y and Φ.

2.2.3 Dynamics of Floating Bodies

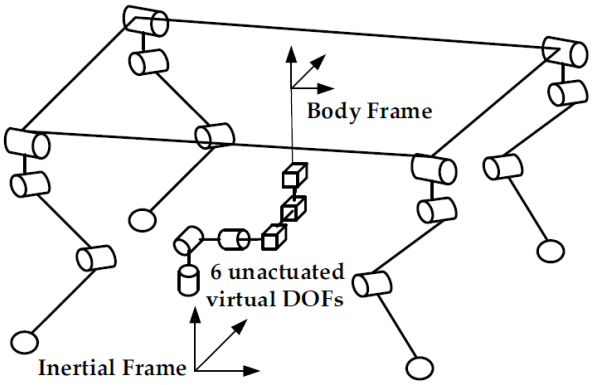


Figure 2.1: A floating body example: quadruped robot.

A Floating body indicates a rigid articulated body system whose base is not fixed to the ground. This arises when the legged structure robot becomes airborne. To describe its kinematic configuration, the base configuration should also be considered, *i.e.,* $(SE(3) \times q_r)$, where $SE(3)$ and q_r denote the configuration of the base and the joints of the floating system, respectively. This can be interpreted that the virtual 6-dof is added to the floating base from the world frame. One

of the most typical floating systems is the legged robot. It consists of several serial chains and Algorithm 1 can be applied to each serial chain to compute the joint torques. However, since the base is no longer fixed, one should carefully use it when initializing V_0 and $\dot{V}_0$.

Typically, the dynamics of the floating bodies can be express as follows:

$$M(q)\ddot{q} + C(q,\dot{q}) + J(q)^\top F = S^\top \tau, \tag{2.2.27}$$

where $q \triangleq [q_b^\top \ q_r^\top]^\top$ is the overall configuration of the floating system ($q_b \in \mathbb{R}^6$ and $q_r \in \mathbb{R}^n$ are the configuration of the base and the joints, respectively), and $S = [0_{n\times 6} \ I_{n\times n}]$ is the actuated joint selection matrix.

2.3 Recursive Differential Dynamics

In this section, a recursive algorithm for differentiating the equations of motion with respect to the kinematic parameters is presented. Given a twist $A \in se(3)$, its neighborhood $\hat{A} \in se(3)$ is known to be locally parametrized as

$$\hat{A} = \mathrm{Ad}_{e^{[\eta]}}(A), \tag{2.3.28}$$

for some $\eta \in se(3)$ [24]. Equation 2.3.28 can be differentiate with respect to η as follows:

$$\delta A = -[\mathrm{ad}_A]\delta\eta. \tag{2.3.29}$$

By Equation 2.3.29 and Algorithm 1, we can obtain the following recursive algorithm which computes the derivatives of the joint torques with respect to the joint screws and the joint variables. Note that compared to [18], the terms for the joint variables (θ,$\dot{\theta}$,$\ddot{\theta}$) to the derivatives are added, since we consider motions as well as designs for the performance of the robot. To simplify the notation, we use the

following abbreviations: $\mathrm{Ad}_i \coloneqq \mathrm{Ad}_{T_{i-1,i}}$, $\mathrm{Ad}_i^{-1} \coloneqq \mathrm{Ad}_{T_{i-1,i}^{-1}}$. The $[(i:j),(k:l)]$ sub-matrix of a matrix M is denoted by $[M]_{(i:j,k:l)}$.

Proposition 2.3.1 (Recursive Differential Dynamics [18])**.** The differential relationship between the joint torques, joint twists, and joint variables can be written

$$\delta\tau = S(\Phi)\delta\boldsymbol{x}, \tag{2.3.30}$$

where $\delta\boldsymbol{x} = \left[[\delta\eta_1, \delta\theta_1, \delta\dot{\theta}_1, \delta\ddot{\theta}_1]^\top, \cdots, [\delta\eta_n, \delta\theta_n, \delta\dot{\theta}_n, \delta\ddot{\theta}_n]^\top\right]^\top$ and $S(\Phi) \in \mathbb{R}^{n\times 9n}$ whose i-th row is given by

$$\begin{aligned} S_i &= -\mathcal{F}_i^\top[\mathrm{ad}_{A_i}^0] + A_i^\top R_i \\ [\mathrm{ad}_{A_i}^0] &\coloneqq \begin{bmatrix}\mathbf{0}_{6\times 9(i-1)} & [\mathrm{ad}_{A_i}] & \mathbf{0}_{6\times(9(n-i)+3)}\end{bmatrix}. \end{aligned} \tag{2.3.31}$$

R_i can be recursively calculated as follows:

$$\begin{cases} P_i = [\mathrm{Ad}_i^{-1}]P_{i-1} + P_i^0, \\ Q_i = [\mathrm{Ad}_i^{-1}]Q_{i-1} - [\mathrm{ad}_{A_i\dot{\theta}_i}]P_i + Q_i^0, \\ R_i = [\mathrm{Ad}_{i+1}^{-\top}]R_{i+1} - ([\mathrm{ad}_{V_i}]^\top G_i + [\mathrm{ad}^*_{G_iV_i}])P_i + G_iQ_i + R_i^0. \end{cases} \tag{2.3.32}$$

where $P_i^0,\ Q_i^0,\ R_i^0 \in \mathbb{R}^{6\times 9n}$ are zero-padded on both sides similar to $[\mathrm{ad}_{A_i}^0]$:

$$\begin{cases} \left[P_i^0\right]_{(:,9(i-1)+(1:6))} &= [\mathrm{Ad}_i^{-1}][\mathrm{ad}_{V_{i-1}}](I - \mathrm{Ad}_i) - [\mathrm{ad}_{A_i\dot{\theta}_i}], \\ \left[P_i^0\right]_{(:,9(i-1)+(7:8))} &= \begin{bmatrix}[\mathrm{Ad}_i^{-1}][\mathrm{ad}_{V_{i-1}}] & A_i\end{bmatrix}, \\ \left[Q_i^0\right]_{(:,9(i-1)+(1:6))} &= [\mathrm{Ad}_i^{-1}][\mathrm{ad}_{\dot{V}_{i-1}}](I - \mathrm{Ad}_i) - [\mathrm{ad}_{V_i}][\mathrm{ad}_{A_i\dot{\theta}_i}] - [\mathrm{ad}_{A_i\ddot{\theta}_i}], \\ \left[Q_i^0\right]_{(:,9(i-1)+(7:9))} &= \begin{bmatrix}[\mathrm{Ad}_i^{-1}][\mathrm{ad}_{\dot{V}_i}] & [\mathrm{ad}_{V_i}]A_i & A_i\end{bmatrix}, \\ \left[R_i^0\right]_{(:,9(i-1)+(1:6))} &= -[\mathrm{Ad}_{i+1}^{-1}]^\top[\mathrm{ad}^*_{F_{i+1}}](\mathrm{Ad}_{i+1}^{-1} - I), \\ \left[R_i^0\right]_{(:,9(i-1)+7)} &= -[\mathrm{Ad}_{i+1}^{-1}]^\top[\mathrm{ad}^*_{F_{i+1}}]A_{i+1}. \end{cases} \tag{2.3.33}$$

Initial conditions for the recursions are as follows:

$$\begin{cases} P_1 = -\begin{bmatrix}[\mathrm{ad}_{A_1\dot{\theta}_1}] & \mathbf{0}_{6\times1} & A_1 & \mathbf{0}_{6\times1} & \mathbf{0}_{6\times9(n-1)}\end{bmatrix}, \\ Q_1 = \begin{bmatrix}-[\mathrm{ad}_{A_1\ddot{\theta}_1}] + [\mathrm{Ad}_1^{-1}][\mathrm{ad}_{\dot{V}_0}](I - [\mathrm{Ad}_1]) & [\mathrm{Ad}_1^{-1}][\mathrm{ad}_{\dot{V}_0}]A_i & \mathbf{0}_{6\times1} & A_i & \mathbf{0}_{6\times9(n-1)}\end{bmatrix}, \\ R_n = -([\mathrm{ad}_{V_n}]^\top G_n + [\mathrm{ad}^*_{G_nV_n}])P_n + G_nQ_n. \end{cases} \tag{2.3.34}$$

For more detailed explanations and proof, see [18].

Proposition 2.3.1 provides an insight into how to differentiate the dynamics with respect to the design parameters. Due to the complex structure of the equations of motion, it was regarded as an impossible task to analytically differentiate EoM, especially with respect to the design parameters. Therefore, many studies have substituted them numerically; this step causes inaccuracy and lag in the optimization process. Proposition 2.3.1 can be utilized to calculate an analytic gradient, however, the derivatives in Proposition 2.3.1 are with respect to the joint screws, not the design parameters. We will bridge this gap later in Chapter 3. Consequently, the derivative of the joint torque with respect to the design parameters can be analytically determined by a chain rule so that the design parameters can be optimized to reduce the overall dynamic performances.

3

Simultaneous Design and Motion Optimization

In this chapter, we describe the robot design optimization framework considering its motion. Both design and motion parameters are concurrently optimized to reduce the given cost function with our method. We define the optimization problem as an expanded version of classical optimal control. The constraints are set for the robot to achieve given tasks. The core of our method is the recursive differential dynamics utilized to compute the analytic gradient of the joint torque.

3.1 Problem Definition

In what follows we assume a general design optimization problem can be formulated as follows:

$$\begin{aligned} \underset{\rho, x, \dot{x}, \tau}{\text{minimize}} \quad & J(\rho, x, \dot{x}, \tau) \\ \text{subject to} \quad & \dot{x} = f(t, x, \tau; \rho), \\ & g(t, x, \dot{x}, \tau; \rho) \leq 0, \\ & h(t, x, \dot{x}, \tau; \rho) = 0 \end{aligned} \tag{3.1.1}$$

We define the motion parameter as $m = (x, \dot{x})$ to contain all of the information about the joint trajectories enough to describe EoM. The types of robots that we consider in this thesis are serial manipulators or that consist of some serial chains. In these cases, $m = (\theta, \dot{\theta}, \ddot{\theta})$, where θ denotes the joint angles. The user may need to replace $(\theta, \dot{\theta}, \ddot{\theta})$ with proper variables for complex or higher-order mechanisms. For the serial manipulator, the joint torque τ is able to be computed by Algorithm 1, which gets $(\theta, \dot{\theta}, \ddot{\theta})$ of a serial manipulator and outputs joint torques. Hence, Equation 3.1.1 can be converted into the equivalent form:

$$\begin{aligned} \underset{\rho, m}{\text{minimize}} \quad & J(\rho, m) \\ \text{subject to} \quad & g(t, \rho, m) \leq 0, \\ & h(t, \rho, m) = 0. \end{aligned} \tag{3.1.2}$$

There are some cases that the joint torques cannot be completely determined with only ρ and m, *e.g.,* quadruped robots, because the contact forces are indefinite if there are more than three point contacts [25]. In this case, we add the parameter f_e which describes the external force to the optimization variables (ρ, m).

Our framework solves the optimization problem with a dynamic performance. In this thesis, we set the objective function as follows:

$$J(\rho, m) = \frac{1}{2} \int_0^{t_f} \tau^\top \tau dt, \tag{3.1.3}$$

which is widely called *effort* that captures the desire to reduce the applied joint torques during the task. The users can choose the objective functions they want to optimize. Since we can compute the derivative of the joint torque with respect to the design parameters, we can efficiently solve Problem 3.1.2 with various dynamic performance criteria.

3.2 Optimization Parameters

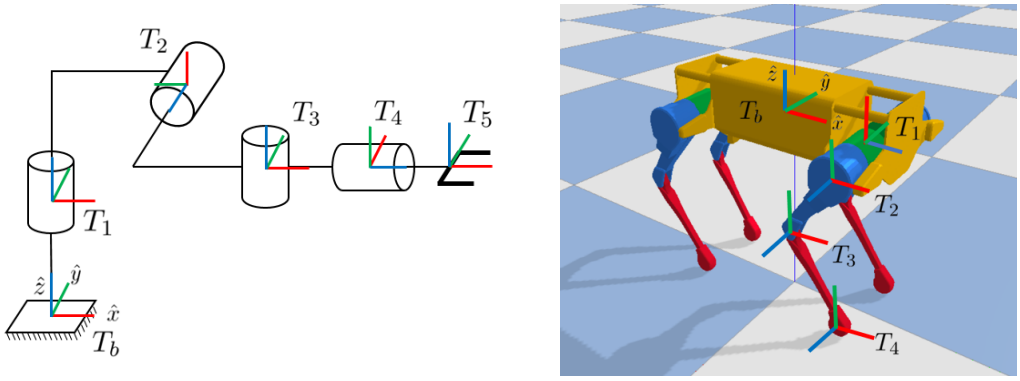
As already discussed, two essential kinds of parameters exist in our design optimization framework: i) *design parameters* and ii) *motion parameters*. In this section, we describe how to define these parameters.

3.2.1 Design Parameters

To fully define the kinematic design of a robot, the information of the joints such as its position and axis is needed. Among the way to describe them, one of the most popular methods is defining the design as a set of joint frames by which the users can obtain the kinematic composition of the robot. This manner is similar to the convention of URDF (Unified Robotics Description Format [26]), which is an XML specification to model multi-body systems such as robotic arms or legged robots.

Figure 3.1 show the examples of the kinematic design definition in our framework. First, the base frame should be defined, and the joint and end-effector frames can be described with respect to the base frame. Then, the design ρ can be described as a following set of $SE(3)$.

$$\rho \triangleq \{T_b, T_1, \cdots, T_{n_j}, T_1^{ee}, \cdots, T_{n_{ee}}^{ee}\}, \tag{3.2.4}$$



(a) Design definition: a serial chain

(b) Design definition: a quadruped

Figure 3.1: Examples of the design definitions of robots.

where T_b, T_i, and T_i^{ee} denote the frame of base, joint, and end-effector, respectively. n_j and n_{ee} are the number of joints and end-effectors. We set each component of $T = (R, p)$ as follows:

- p denotes the 3-dimensional position of the joint (motor) or the end-effector.
- Where $R = \begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \end{bmatrix}$, $\hat{z}$ denotes the joint axis.

Since we define the design parameters as a set of $SE(3)$, the feasible space of Equation 3.1.2 becomes the product space of matrix Lie group (design parameter) and vector space (motion parameter). In a gradient-based optimization of a matrix Lie group, the update rule of the optimization variables is

$$T \leftarrow e^{[\eta]}T \text{ or } T \leftarrow Te^{[\eta]}, \tag{3.2.5}$$

where η is the corresponding Lie algebra. Therefore, one question arises: *which is a better update rule?* To answer this question, recall Equation 2.3.28. In this equation, the local parametrization of the joint screw is formulated by the large adjoint group action whose physical meaning is the change of the reference frame.

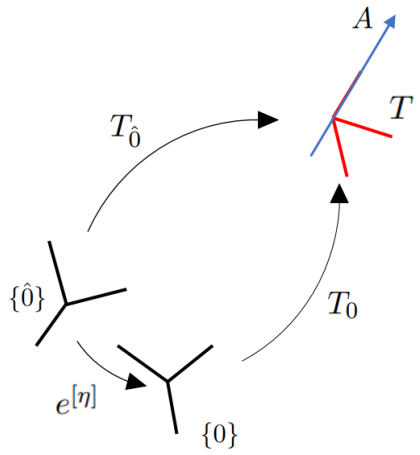


Figure 3.2: Transformation of the reference frame.

Figure 3.2 shows this situation. Assuming $\{0\}$ and $\{\hat{0}\}$ denote the different reference frames, the coordinate transformation of the screw $A \in se(3)$, which can be depicted by $T \in SE(3)$, is of the form

$$A_{\hat{0}} = [\mathrm{Ad}_{e^{[\eta]}}]A_0, \tag{3.2.6}$$

where $e^{[\eta]} = T_{\hat{0}0}$, *i.e.,* the transformation from frame $\{\hat{0}\}$ to $\{0\}$. The above equation is equivalent to Equation 2.3.28, and we can say that Equation 2.3.28 is same as the transformation of the reference frame by $e^{-[\eta]}$. Consequently, the infinitesimal change of T should be

$$\hat{T} = e^{[\eta]}T, \quad \eta \in se(3). \tag{3.2.7}$$

in order to follow the same view of Equation 2.3.28. Since the recursive differential dynamics departs from Equation 2.3.28, we can utilize the attractive results (the derivative of the joint torque with respect to the change of joint screw) in [18] with the given update rule. Mathematical supplements can be found in Appendix A.1.

3.2.2 Motion Parameters

In order to solve the optimization problem that contains the trajectory in the optimization variable, the parametrization of a joint trajectory is needed to change the problem into a tractable one. To do this, B-spline curve is used which has been widely used in many robotic researches [27, 28]. This section briefly reviews the concept of B-spline, and how the motion parameters can be expressed by B-spline. More detailed information can be found in [29].

$\theta(t) \in \mathbb{R}^d$ can be described as a weighted-sum of some points

$$\begin{aligned} \theta(t) = \theta(t; u_{1:n_k}, c_{1:n_c}) = \sum_{i=1}^{n_c} N_{i,p}(t) c_i, \\ u \in \mathbb{R}, c \in \mathbb{R}^d, p = n_k - n_c, t \in [0, t_f], \end{aligned} \tag{3.2.8}$$

where u, c, and N denote the knots, B-spline control points, and B-spline basis functions, respectively. p is the order of B-spline, n_k and n_c is the number of knots and control points, and t denotes the time variable. The B-spline basis function $N_{i,p}(t)$ is C^{p-2} continuity and can be recursively computed as follows [30]:

$$\begin{aligned} N_{i,p} &= \frac{t - u_i}{u_{i+p-1} - u_i} N_{i,p-1}(t) + \frac{u_{i+p} - t}{u_{i+p} - u_{i+1}} N_{i+1,p-1}(t), \\ N_{i,1} &= \begin{cases} 1 & \text{if } u_i \leq t < u_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3.2.9}$$

The velocity and acceleration $\dot{\theta}(t)$, $\ddot{\theta}(t)$ of the trajectory can be derived from Equation 3.2.8,

$$\dot{\theta}(t) = \sum_{i=1}^{n_c} \dot{N}_{i,p}(t) c_i, \quad \ddot{\theta}(t) = \sum_{i=1}^{n_c} \ddot{N}_{i,p}(t) c_i, \tag{3.2.10}$$

where the derivative of the B-spline basis function is

$$\dot{N}_{i,p}(t) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(t) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(t). \tag{3.2.11}$$

$\ddot{N}_{i,p}(t)$ can be derived by differentiating the above equation. The differentiation of θ, $\dot{\theta}$, and $\ddot{\theta}$ with respect to $\mathbf{c} = [c_1^\top, \cdots, c_{n_c}^\top]^\top$ can be represented by

$$\begin{aligned} \delta\theta(t) &= \boldsymbol{N}_p(t)\delta\mathbf{c}, \\ \delta\dot{\theta}(t) &= \dot{\boldsymbol{N}}_p(t)\delta\mathbf{c}, \\ \delta\ddot{\theta}(t) &= \ddot{\boldsymbol{N}}_p(t)\delta\mathbf{c}. \end{aligned} \tag{3.2.12}$$

where $\boldsymbol{N}_p(t) = [N_{1,p}(t), \cdots, N_{n_c,p}(t)]$. $\dot{\boldsymbol{N}}_p(t)$ and $\ddot{\boldsymbol{N}}_p(t)$ can be defined in a similar fashion.

Next, we describe the conversion from the trajectory to the B-spline control points. Let $\boldsymbol{\theta} = [\theta(t_1)^\top, \cdots, \theta(t_{n_t})^\top]^\top$ the collection of the points on the trajectory $\theta(t)$. n_t denotes the number of discrete time intervals, $t_1 = 0$, and $t_{n_t} = t_f$. The corresponding B-spline control point $\mathbf{c}$ can be computed in a least-square sense

$$E(\boldsymbol{c}) = \frac{1}{2}\sum_{j=1}^{n_t} || \sum_{i=1}^{n_c} N_{i,p}(t_j)c_i - \theta(t_j)||^2, \tag{3.2.13}$$

of which the solution becomes

$$\mathbf{c} = (A^\top A)^{-1} A^\top \boldsymbol{\theta}, \tag{3.2.14}$$

where A can be derived by the B-spline basis functions. Further details can be found in [30]. Consequently, in this thesis, the motion parameter is expressed as

$$m = (c_1, c_2, \cdots, c_{n_c})^\top, \tag{3.2.15}$$

by the B-spline control points.

3.2.3 Constraints

Constraints limiting design and motion parameters need to be formulated in order to achieve a given task. In this thesis, we first define the task constraint as the

trajectory of the end-effectors the robot should follow, similar to [17]. Examples for this kind of constraint can be easily found in many robotic applications, *e.g.,* pick-and-place, cutting, grinding *etc.* In the locomotion task of the legged robot, also, the trajectories of the feet are usually prescribed based on the step length, step height, and the period.

The differential kinematics of n-dof serial chain can be described in the form

$$\begin{aligned}(T^{-1}\delta T)^{\vee} &= \sum_{i=1}^{n}\{([\mathrm{Ad}_i]-[\mathrm{Ad}_{i+1}])\delta\eta_i + [\mathrm{Ad}_{i+1}]A_i\delta\theta_i\} + [\mathrm{Ad}_{n+1}]\delta\eta_{n+1}, \\ [\mathrm{Ad}_i] &\triangleq [\mathrm{Ad}_{M^{-1}e^{-[A_n]\theta_n}\cdots e^{-[A_i]\theta_i}}], [\mathrm{Ad}_{n+1}] = [\mathrm{Ad}_{M^{-1}}],\end{aligned} \tag{3.2.16}$$

where A_i, $\delta\eta_i$, $\delta\theta_i$ denote the screw, infinitesimal change of the design parameter, and the angle of i^{th} joint, respectively. If the current design and motion parameter pairs follow the given trajectory, the task constraint for trajectory can be presented as a linear equation of the form

$$A\begin{bmatrix}\delta\eta \\ \delta\theta\end{bmatrix} = 0, \tag{3.2.17}$$

where $\delta\eta = [\delta\eta_1^{\top}, \cdots, \delta\eta_{n+1}^{\top}]^{\top}$ and $\delta\theta = [\delta\theta_1, \cdots, \delta\theta_n]^{\top}$. More details can be found in Appendix A.3

Besides the above constraints which describe the trajectories of the end-effectors, the user can add more constraints if needed. For the quadruped locomotion task, the foot forces need to be physically valid; the normal forces are positive and large enough not to slip. Therefore, we added the well-known *friction pyramid* inequality constraints for the quadruped locomotion task as follows:

$$0 \le f_z,\ \ f_x \le \mu f_z,\ \ -f_x \le \mu f_z,\ \ f_y \le \mu f_z,\ \ -f_y \le \mu f_z, \tag{3.2.18}$$

where μ is Coulomb friction coefficient, and (f_x, f_y, f_z) denotes the force exerted to the foot from the ground. z is the normal direction. Furthermore, some of the

most frequently treated constraints are the joint limits: min/max angle, velocity, acceleration, *etc*. They can also be easily adopted by simple inequalities of the motion parameter.

3.2.4 Inertial Changes

To handle the inertial perturbation by changing the design parameter, parametrization for the inertial term is needed. In this thesis, we set the inertia of the actuators as a point mass. Also, the inertia of the link is defined to the thin rod connecting the positions of the adjacent actuators. We call this as a *design rule*. The users can define their own design rule for their application. Further details can be found in Appendix A.2.

Reset $\delta x = [\delta\eta^\top, \delta m^\top]^\top$ by using Proposition 2.3.1 and Equation 3.2.12 to convert $(\theta, \dot{\theta}, \ddot{\theta})$ into our motion parameter m defined by the B-spline control points. From Equation 2.2.26,

$$\delta\tau(t) = S\delta x + Y\delta\Phi, \tag{3.2.19}$$

since Proposition 2.3.1 assumes only the inertial terms are fixed. Under the definition of the inertial term, the derivative of Φ becomes

$$\delta\Phi = D\delta\eta. \tag{3.2.20}$$

Therefore, we can calculate the derivative of the joint torque as

$$\delta\tau(t) = (S + [YD \quad \mathbf{0}_{n_j \times n_m}])\delta x, \tag{3.2.21}$$

where n_j and n_m denotes the number of joints and motion parameters, respectively.

3.3 Optimization Algorithm Description

Our framework concurrently optimizes both design and motion parameters of the robot. Let x be the optimization variable, then x should contain both design and motion parameters so that $x = (\rho, m)$. There are some cases that the additional variables are needed, *e.g.,* quadruped robot with locomotion task (see Chapter 4.2). We also define $\delta x = (\delta\eta_1^\top, \cdots, \delta\eta_{n_\rho}^\top, \delta c_1^\top, \cdots, \delta c_{n_c}^\top)^\top$ to contain the local parametrization $\delta\eta$ of the design parameters.

We first discretize Problem 3.1.2. Let $(t_1, \cdots, t_{n_t})$ a set of discretized time indices, where $t_1 = 0$ and $t_{n_t} = T$. Then, the objective function can be approximated to

$$\frac{1}{2}\int_0^{t_f} \tau^\top \tau dt \approx \frac{1}{2}\boldsymbol{\tau}^\top \boldsymbol{\tau} \Delta t, \tag{3.3.22}$$

where $\boldsymbol{\tau} = [\tau_1^\top, \cdots, \tau_{n_t}^\top]^\top$, $\tau_i = \tau(t_i)$, and Δt is the time interval.

To solve our design optimization problem, the gradient-descent method is adopted which iteratively finds the direction to reduce the cost function. Since the optimization variable contains both Lie group (design parameter) and vector (motion parameter), one should be careful during the gradient update step. The update rule of the design parameter ρ is already described in Chapter 3.2.1. For the motion parameter, which is a vector in Euclidean space since parametrized by the B-spline control points, a general update rule $m \leftarrow m + \delta m$ can be adopted.

To deal with the constraints, the below strategies are applied:

- inequality constraints: the approximated l_1 exact barrier function is used [31]. Barrier function generates high value for the objective function where the objective variable is near or out of the boundary of the feasible region, in order not to violate the inequality constraints. The barrier function lift

these constraints to the objective function, which becomes

$$J(x) + \lambda \sum_{i=1}^{n_{ineq}} p_\epsilon(g_i(x)),$$
$$p_\epsilon(t) = \begin{cases} \frac{3}{2}\epsilon e^{t/\epsilon} - 2\epsilon & if\ t \leq 0, \\ t - \frac{1}{2}\epsilon e^{-t/\epsilon} & if\ t > 0, \end{cases} \tag{3.3.23}$$

where $\lambda, \epsilon > 0$.

- equality constraints: after calculating the steepest direction for reducing the value of the objective function without equality constraints, that gradient is projected to the null space of the linearized equality constraints from the current solution. This can be mathematically represented as follows:

$$grad \leftarrow N(N^\top N)^{-1} N^\top \times grad, \tag{3.3.24}$$

where $grad$ and N denote the steepest gradient and the linearized null space matrix of the equality constraint, respectively.

Algorithm 2 summarizes our simultaneous optimization framework. It gets the initial design and motion pair x_0 as inputs and finds the optimal parameters x^* while maintaining the feasibility. It consists of three large parts: calculating the gradients (line 2 and 3), updating the optimization variable (line 4, 5, and 6), re-projecting the optimization variable to the equality constraint (line 7).

First, the gradients are calculated. From Equation 3.2.21, the derivative of each joint torque can be computed by $\delta\tau_i = (S_i + [Y_i D \quad \mathbf{0}_{n_j \times n_m}])\delta x$. Let $\boldsymbol{S}$ be the stack of the preceding equation ($\delta\boldsymbol{\tau} = \boldsymbol{S}\delta x$). Then, we can calculate the gradient as

Algorithm 2 Simultaneous Design and Motion Optimization

Input: Initial parameter x_0, inequality constraints $g(x)$, equality constraints $h(x)$

Output: x^*

1: **while** not reach the terminal conditions **do**
2: $grad \leftarrow CalculateObjectiveGradient(x, g)$
3: $A \leftarrow \frac{\partial}{\partial x} h(x)$
4: $N \leftarrow null(A)$
5: $grad \leftarrow N(N^\top N)^{-1} N^\top \times grad$
6: $x \leftarrow UpdateVariables(x, grad, stepsize)$
7: $x \leftarrow EqualityConstraintProjection(x, h)$
8: **end while**

follows:

$$
\begin{aligned}
grad &= \Delta t \boldsymbol{\tau}^\top \boldsymbol{S} + \lambda \sum_{i=1}^{n_{ineq}} \frac{\partial p_\epsilon}{\partial g_i} \frac{\partial g_i}{\partial x}, \\
\frac{\partial p_\epsilon}{\partial g_i} &= \begin{cases} \frac{3}{2} e^{t/\epsilon} & if \ t \leq 0, \\ 1 + \frac{1}{2} e^{-t/\epsilon} & if \ t > 0, \end{cases}.
\end{aligned}
\tag{3.3.25}
$$

Second, x is updated to the direction of the gradient, which is the projected vector of what calculated in the previous step. For the projection, we linearize the equality constraint from the current x and conduct the projection to the null space of it. This procedure helps x not to recede from the constraint manifold. Each partial derivative of equality constraint can be found in Appendix A.3.

Third, x is projected to the equality constraint to adjust the numerical error generated from step 2 which comes from the linearization of the nonlinear function.

This procedure can be generally formulated as follows:

$$\begin{aligned} (x^*) = \ & \arg\min_{x} && ||h(x)||^2 \\ & \text{subject to} && g(x) \leq 0, \end{aligned} \tag{3.3.26}$$

starting from the resultant x of the first and second step of our framework. In this thesis, we fix ρ and solve the above problem only with m in order to maintain the design change. The above optimization problem can be replaced by a similar procedure; for example, we solve the inverse kinematics of the 2-R planar manipulator for this step. Solving the inverse kinematics is easier than the problem formulated as Problem 3.3.26.

4

Numerical Experiments

In this chapter, we verify our simultaneous design and motion optimization framework through numerical experiments carried out on two circumstances: the 2-R planar manipulator with given end-effector trajectories and the quadruped robot with locomotion task. The optimization algorithm was implemented using the MATLAB, and the results were certified on MuJoCo physics simulator.

4.1 2-R Planar Manipulator

4.1.1 Experimental Settings

To validate our design optimization framework, we first begin with a simple 2-R planar manipulator whose desired end-effector trajectories are given. Figure 4.1 shows the robot and the three given trajectories: circle, triangle, and square shapes. The diameter and the lengths of the sides of triangle and square are 1m each. The task is set to these trajectories that the end-effector should follow. The overall time horizon of each task is set to 4 seconds, and we pick the points on the trajectories

by 0.05 second time interval. Since the task trajectories are only positional (there is no designation for the rotation), the equality constraint projection step (line 7 in Algorithm 2) becomes the simple inverse kinematics problem. The manipulator is mounted on the x-y plane and two revolute joints whose axes are set to positive z-axis. Each length of the links, the mass of the motors, and linear density of the links are set to 1.5m, 1kg, and 0.1kg/m, respectively. We set the zero mass to the end-effector. The initial base position of the robot is placed at $(0, 0)$.

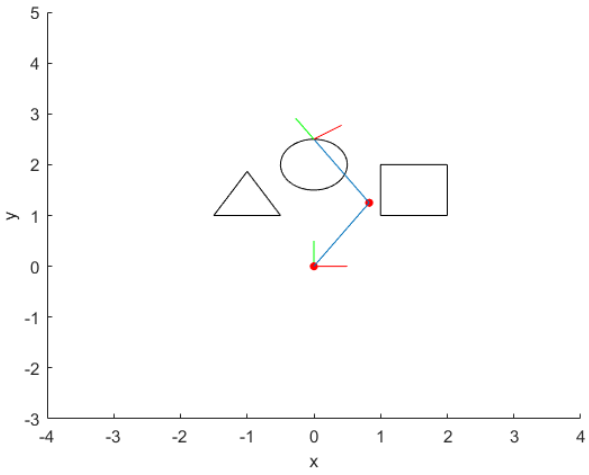


Figure 4.1: 2-R planar robot manipulator with three specified trajectories.

For the 2-R planar manipulator case, we fix the change of the design parameters except for the y-coordinates of the second motor and end-effector thus our framework can search the lengths of the links that fit well with the given task. The corresponding joint torques are computed by Algorithm 1 which get inputs the joint values $(\theta, \dot{\theta}, \ddot{\theta})$ calculated by the motion parameter m. The number of B-spline control points is 25 for each trajectory, therefore the dimension of overall optimization parameters is 152.

4.1.2 Optimization Results

Figure 4.2 shows the change of the objective function during the optimization process. The initial value of the effort (the objective function) is 978.649 and it converged to 789.453, about 19.3% reduction. The resultant lengths of the links is in Table 4.1. The length of the first link decreases from 1.5 to 0.994m the second link is lengthened from 1.5 to 1.884m. Since the mass of the second actuator is a dominant term, we infer that our framework tends to shorten the length of the link 1 to reduce the joint torques, and extends the link 2 in order to reach the trajectories. The tracking error, which is a root-mean-square value, maintains extremely small value thus we can conclude our methodology force both the design and motion or the robot not to violate given tasks.

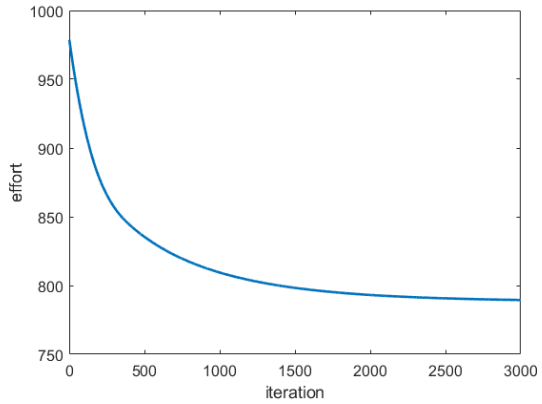


Figure 4.2: Effort during the optimization process.

Our framework adjusts the motion of the robot under the perturbation of the design to achieve the given tasks. Figure 4.3 shows the joint values before and after optimization, and we can check that the motion is changed by our framework. We also confirm the agreement of these motion trajectories by forward kinematics

Results	Before optimization	After optimization
Link 1 length (m)	1.5	0.994
Link 2 length (m)	1.5	1.844
Effort	978.649	789.453
Tracking error (m)	1.2412e-06	2.5739e-06

Table 4.1: Optimization result of the 2-R planar manipulator.

errors (RMS value) and visual check. The corresponding joint torques can be seen in Figure 4.4, which are suitably declined to reduce the overall effort.

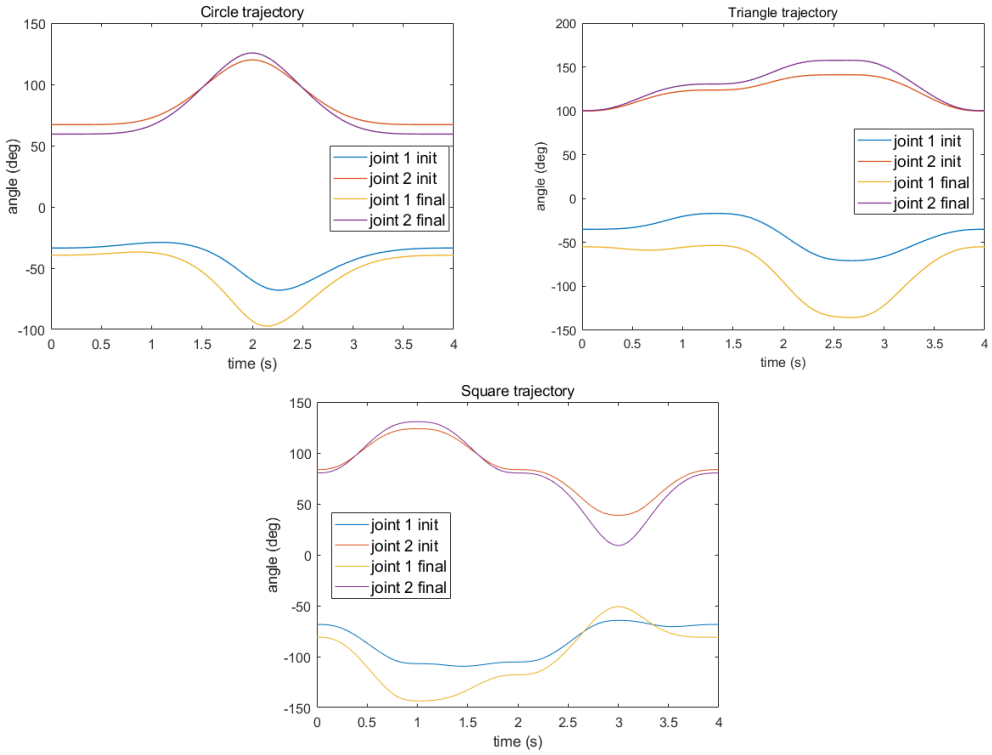


Figure 4.3: Joint angles of the 2-R manipulator of each task.

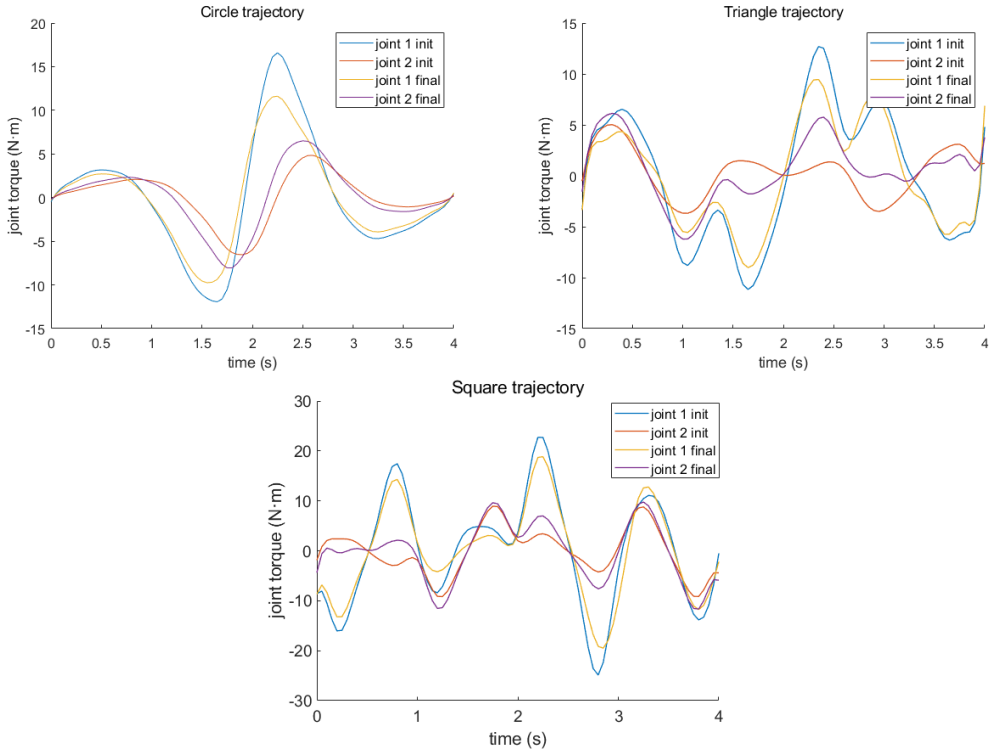


Figure 4.4: Joint torques of the 2-R manipulator of each task.

In our framework, the position of the base can also be included in the design parameter. We also experimented with the above situation, and the process results in the case that no inverse kinematics solution exists, *i.e.,* the robot cannot reach some points of the trajectories. The singular configuration of the rigid body structure has an advantage on the actuating force; the humans stand straight to support their weight using the bones. Since our methodology seeks the design to obtain better dynamic performance, the resultant morphology tends to be in a singular configuration under the output motion parameter. This episode might depend on the task we provided. The case with a fixed base position has not reached

the singular configuration. Since the singularity is the thing to avoid, the method to deal with this problem should be devised. One way to handle this problem is by restricting the design not to modify a lot. The designs of commercial robots are invented by the robot experts and they consider hundreds of factors for a good design. However, our framework concentrates on the dynamic performances and may harm other design criteria. We guess the design limitation is the compromise between the two design perspectives. We leave this for future work.

4.2 Quadruped Robot

Our second design optimization problem considers a quadruped robot with a locomotion task. The locomotion is the most important task for the quadruped robots since it is usually performed for the longest time. For this problem, the external forces, *i.e.,* foot forces need to be augmented to the optimization variable, since the design and motion cannot fully determine the actuator forces. If the quadruped robot has more than three point contacts, the contact forces become redundant and there are infinite possible combinations to be consistent with the whole movement of the robot system [25]. Therefore, we should add the external forces to the optimization variable, and it becomes $x = (x_{FR}, x_{FL}, x_{RR}, x_{RL})$ where the subscript denotes the foot index (front right, front left, rear right, rear left). Each $x_j \triangleq (\rho, m, f_e)$ where $j \in \{FR, FL, RR, RL\}$, $f_e = (f_e(t_1)^\top, \cdots, f_e(t_{n_t})^\top)^\top$ and each $f_e(t_i)$ contains three directional components (f_x, f_y, f_z). In addition, we add the equations of motion for the base of the quadruped to the equality constraints. From the fact that we utilize Newton-Euler inverse dynamics algorithm which in fact the equations of motion for the links, to compute the joint torques, the EoM of the base is needed to compute the gradient direction that is consistent with the

whole EoM of the robot system. Consequently, the augmented design optimization problem can be formulated as follows:

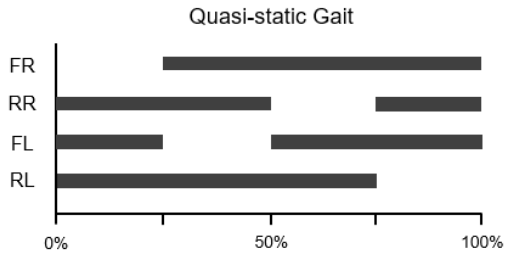
$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \frac{1}{2}\int_{t=0}^{T}(\tau_{FR}^{\top}\tau_{FR}+\cdots+\tau_{RL}^{\top}\tau_{RL})dt \\
\text{subject to} \quad & f_{e,j}^{z}(t_i)\leq 0, \\
& -f_{e,j}^{x}(t_i)+\mu f_{e,j}^{z}(t_i)\leq 0,\ \ f_{e,j}^{x}(t_i)+\mu f_{e,j}^{z}(t_i)\leq 0, \\
& -f_{e,j}^{y}(t_i)+\mu f_{e,j}^{z}(t_i)\leq 0,\ \ f_{e,j}^{y}(t_i)+\mu f_{e,j}^{z}(t_i)\leq 0, \\
& \frac{1}{2}|p_{des,j}(t_i)-p_j(t_i,\rho,m)|^2=0, \\
& \frac{1}{2}|G_b\dot{V}_b(t_i)-[\mathrm{ad}_{V_b}(t_i)]^{\top}G_bV_b(t_i)-\sum_{j=1}^{4}[\mathrm{Ad}_{T_{bj}^{-1}}]^{\top}F_{e,j}(t_i)|^2=0, \\
& (i=1,2,\cdots,n_t,\ \ j=1,2,3,4).
\end{aligned}
\tag{4.2.1}
$$

where n_t is the number of time instant, G_b and V_b respectively denote the inertia and twist of the base, T_{bj} is the coordinate transformation from the body frame b to the j^{th} foot frame, and $F_{e,j}=(0,0,0,f_{e,j}^{\top})^{\top}$ denotes the external wrench by the contact of each foot. The three inequality constraints denote the friction pyramid constraints, the forth and fifth equality constraints represent the task constraints and the EoM of the base, respectively.

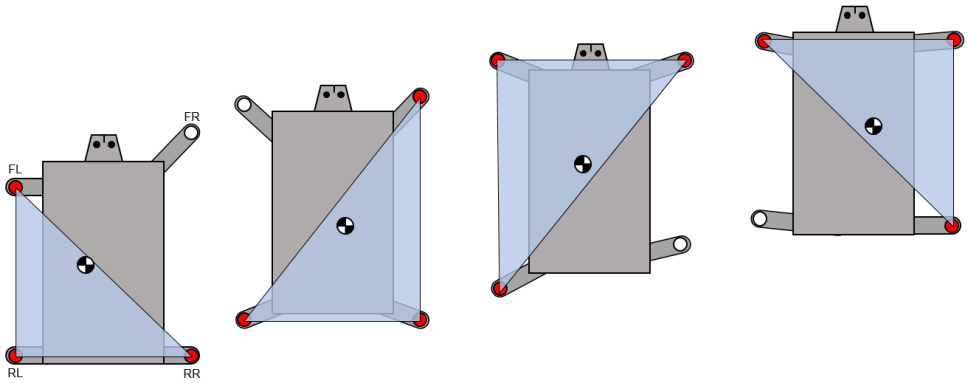
4.2.1 Experimental Settings

There are several kinds of locomotion patterns and among them, we adopt the quasi-static gait, which maintains the ZMP of the quadruped within the support polygon during the locomotion. Figure 4.5 shows how the quasi-static gait is comprised. The quadruped robot pushes the legs in the following order: front right (FR), front left (FL), rear right (RR), rear left (RL). The overall time horizon of the gait is set to 8 seconds, and the short 0.15-second four-leg support phase is

given to move the ZMP toward the next support polygon. Further details about the above gait pattern can be found in [32].



(a) Gait graph of the qausi-static gait.



(b) Support polygon during the quasi-static gait.

Figure 4.5: Quasi-static gait of the quadruped

For the numerical experiments, Laikago by Unitree Robotics is selected [33]. Laikago has four 3-dof legs that the axis of the first motor is parallel and the second and third motors are perpendicular to the front direction, respectively. The body mass and inertia are $15kg$ and $l(I_c) = (0.1062, 0.3406, 0.3906, 0, 0, 0)^\top$ with respect to the base frame, respectively. The masses of the motors are set to $1kg$, $1.5kg$, and $0.5kg$ in order close to the body. The linear density of the link is set

to $0.1kg/m$. The initial design parameters can be found in Appendix A.4.

We limit the design to be left-right symmetry and to modify only the lengths of the upper and lower legs during the optimization for practicality. The initial motion is generated using the method described in [34]. For the control of the quadruped robot in the physics simulator, the inverse dynamics control scheme in [23] is adopted which utilizes the PD signal of the joint trajectories and P signal of the external foot forces.

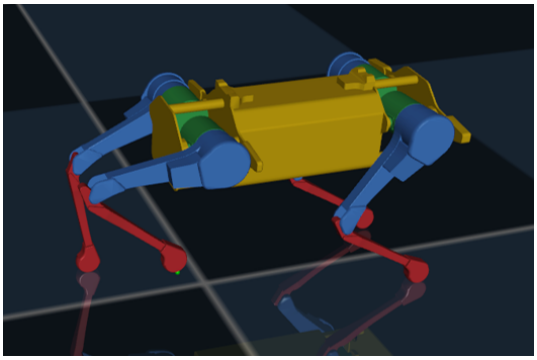


Figure 4.6: Laikago by Unitree Robotics (in MuJoCo simulator).

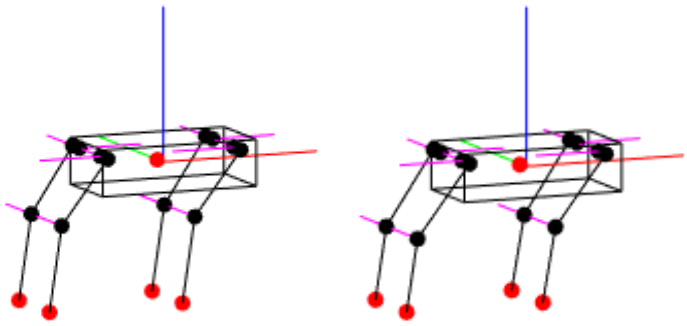
4.2.2 Optimization Results

Table 4.2 shows the optimization result of the quadruped locomotion. We can see that our framework successfully reduces the effort during the locomotion task. The lengths of the upper legs increase and the lower legs shorten to reach the desired foot trajectory. We can observe the joint torques in Figure 4.8 and 4.9, which verify the successful optimization results. The overall change of the design and the effort are 11.3% and 28.1%, respectively. This result implies even the small change in the design can result in much better improvement in the performance, and we can conclude that the design optimization step is essential in the robot

design process.

Results	Before optimization	After optimization
Front Upper leg length (m)	0.253	0.269
Front Lower leg length (m)	0.278	0.237
Rear Upper leg length (m)	0.253	0.278
Rear Lower leg length (m)	0.278	0.230
Effort (Matlab)	4477.644	3660.171
Effort (MuJoCo)	5599.212	4025.264

Table 4.2: Optimization result of the quadruped robot.



(a) Before optimization. (b) After optimization.

Figure 4.7: Morphology of the quadruped robot.

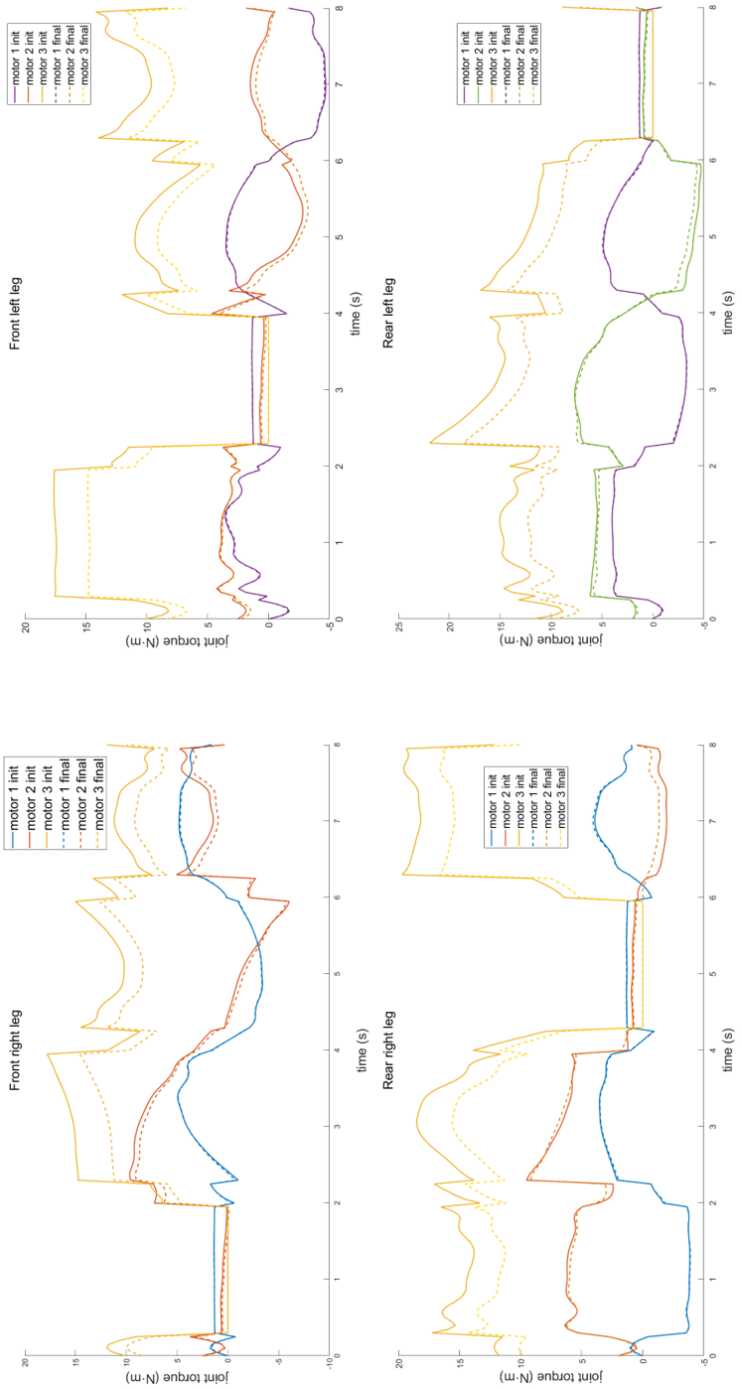


Figure 4.8: The quadruped robot optimization result: joint torques (Matlab).

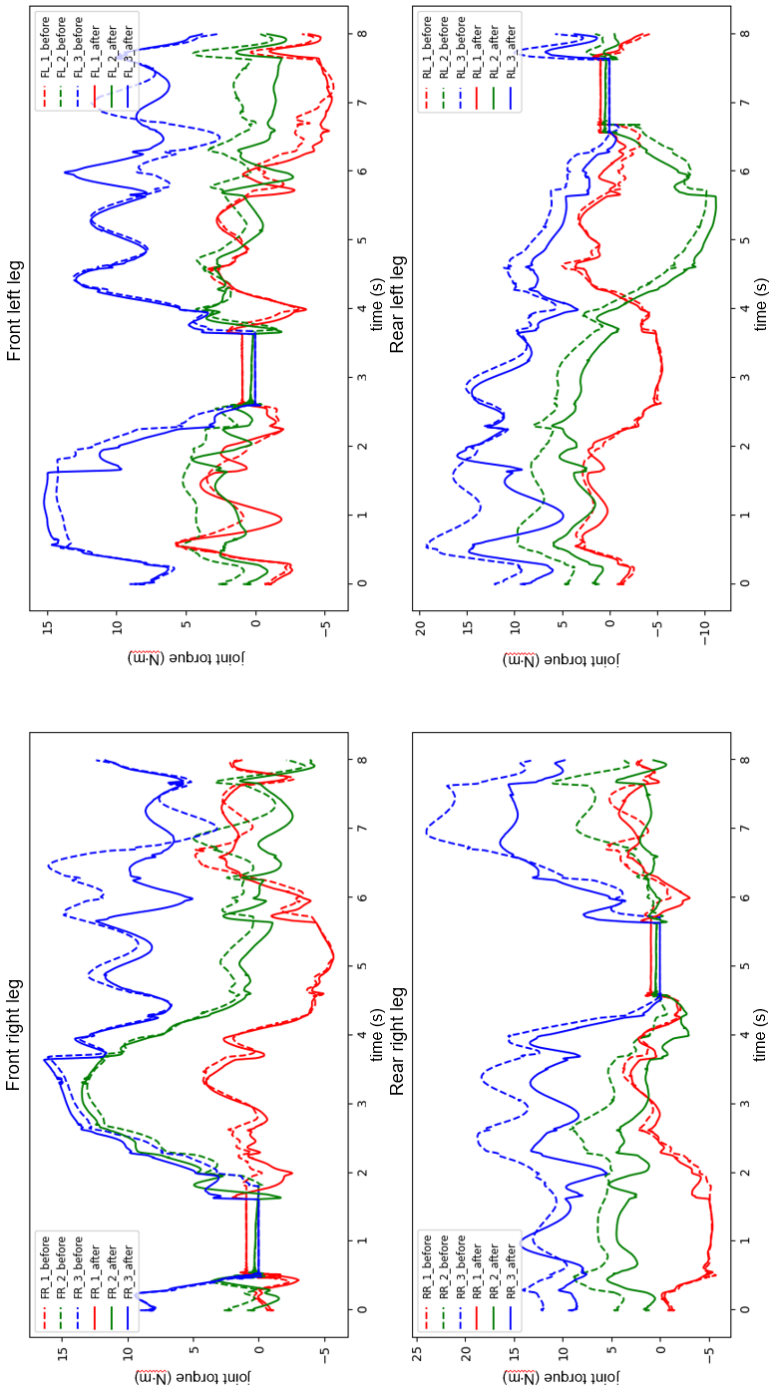
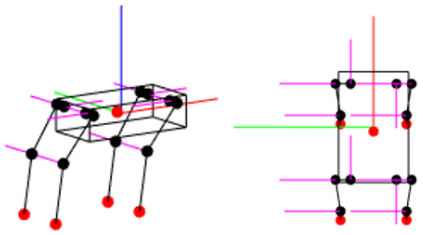
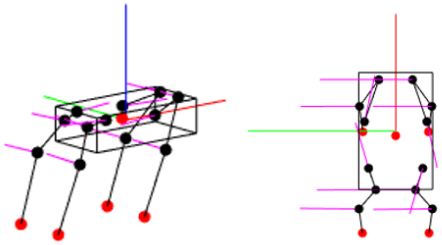


Figure 4.9: The quadruped robot optimization result: joint torques (MuJoCo).

Indeed, our framework can handle the arbitrary alteration of the joint configuration. We also optimize the quadruped robot under no design limitation except left-right symmetry. Figure 4.10 shows the resultant morphology, where the effort reduces from 4477.644 to 2910.441. Note that the axes of the actuators are also be optimized. We did not verify this resultant shape in MuJoCo since it is hard to synthesize. Using our framework without any design limitation may result in a quite weird morphology, but also permits various design changes. Many studies parametrize the design as simple as possible for the computational simplicity, *e.g.*, computing the gradients. Due to the unified method for dealing with the design parameter, our framework can handle various design possibilities and be implemented to more complex structures.



(a) Before optimization.



(b) After optimization.

Figure 4.10: Optimization result without design limitation.

5

Conclusion

We have proposed a framework simultaneously optimizing both design and motion parameters for robot systems. Rather than focusing on versatility, our optimization scheme first assumes that it is critical for the robots to be optimized for specific tasks. Therefore, our concurrent design and motion optimization method can be formulated as an expanded version of the classical optimal control problem, *i.e.,* the design parameters are added to the optimization variables. In particular, an effort has been considered as the performance criteria to guarantee the robot better dynamically performance. One of the most critical defects of the previous researches is that they had not been able to compute the analytic gradient of the equations of motion with respect to the design parameters, therefore their methods ended up laggard and inaccurate optimization result. To overcome the above issue, we have utilized recently discovered recursive differential dynamics that can compute the analytic gradient of the joint screw. Therefore, our framework can be implemented even to the complex robot structures. The constraints have been set to accomplish the specific tasks, so the optimized robot design has achieved the

given movement with the optimized motion. We have validated our framework by two numerical experiments: a 2-R planar manipulator whose end-effector trajectories are given, and a quadruped robot operating locomotion movements. Both the design and motion parameters of each robot have been adjusted to reduce the dynamic performance.

This thesis can be a cornerstone of the task-specific design optimization for the complex robot structures. However, there is still a long way to go for the superior optimization scheme. While focusing on the task-specific performance criteria, versatility may deteriorate, which is still critical to the robots. One way to mediate these two contradict performances is to limit the change of the design, thus not to spoil much the versatility measure such as workspace volume. Another drawback is that we supposed the links as the thin rods. This assumption looks fairly reasonable and practical, but combining with shape morphing theories of the links can carry better designs. We leave the above issues for future works.

In the robot kinematic design, determining the topology of the kinematic chain precedes adjusting the geometric dimensions, and two design issues are usually considered separately. Recently, attempts to solve the two problems in combination have shown great results, but they remain at a level where the kinematic performances are considered, moreover, suffer from the enormous design space. The mentioned design decision problem can be formulated as mixed-integer programming. We think our efficient optimization framework can aid to drag the combined problem down to the real application level.

A

Appendix

A.1 Local parametrization of the design

Assuming respectively the axis and the position of the joint as $\hat{z} \in S^2$ and $p \in \mathbb{R}^3$, the corresponding $T \in SE(3)$ can be represented as

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, \tag{A.1.1}$$

where the third column of T is z. The screw of the given joint is

$$A = \begin{bmatrix} \hat{z} \\ -[\hat{z}]p \end{bmatrix}. \tag{A.1.2}$$

By the update rule 3.2.7 with $\eta = (\omega, v)^\top$, the perturbed axis and position of the joint becomes $(e^{[\omega]}\hat{z}, e^{[\omega]}p + G(\omega)v)$. Therefore, the perturbed joint screw is as follows:

$$\begin{aligned}\hat{A} = \begin{bmatrix} e^{[\omega]}\hat{z} \\ -[e^{[\omega]}\hat{z}](e^{[\omega]}p + G(\omega)v) \end{bmatrix} &= \begin{bmatrix} e^{[\omega]}\hat{z} \\ -e^{[\omega]}[\hat{z}]p + [G(\omega)v]e^{[\omega]}\hat{z} \end{bmatrix} \\ &= \begin{bmatrix} e^{[\omega]} & 0 \\ [G(\omega)v]e^{[\omega]} & e^{[\omega]} \end{bmatrix} \begin{bmatrix} \hat{z} \\ -[\hat{z}]p \end{bmatrix} \\ &= [\mathrm{Ad}_{e^{[\eta]}}]A.\end{aligned} \tag{A.1.3}$$

Since the resultant equation is the same as Equation 2.3.28, we can use the recursive differential dynamics, which starts from Equation 2.3.28, by our definition of the design perturbation.

A.2 Design rule for the link

The inertial parameter of the rigid multibody with n bodies can be represented by $\Phi_b = [\phi_{b_1}^\top, \phi_{b_2}^\top, \cdots, \phi_{b_n}^\top]^\top \in \mathbb{R}^{10n}$, where

$$\phi_{bi} = [m_i, h_{b_i}, l(I_{b_i})]^\top \in \mathbb{R}^{10} \tag{A.2.4}$$

is the inertial parameters of the i^{th} rigid body with mass m_i, mass center position $p_{b_i} \in \mathbb{R}^3$, $h_{b_i} = m_i p_{b_i}$, and linearized rotational inertia $l(I_{b_i}) \in \mathbb{R}^6$. In this thesis, we set the inertia of each rigid body as the following rule: the i^{th} rigid body contains the $(i+1)^{th}$ motor with a point mass and i^{th} link whose inertia is represented by a thin rod between two motors. Assuming M_i, p_i, and ρ_L respectively denote the mass of the i^{th} motor, the position of the i^{th} motor, and the linear density of the links, the mass of the i^{th} link becomes $M_i^l = \rho_L |p_{i+1} - p_i|$. The following results represent the i^{th} rigid body inertia:

$$\begin{aligned} m_i &= M_{i+1} + M_i^l, \\ p_c &= \frac{1}{m_i}(M_{i+1}p_{i+1} + \frac{1}{2}M_i^l(p_i + p_{i+1})), \\ h_{b_i} &= m_i p_c, \\ I_{xx} &= \rho_L(p_{i+1}^\top Q_{xx} p_i + \frac{1}{3}(p_{i+1} - p_i)^\top Q_{xx}(p_{i+1} - p_i))|p_{i+1} - p_i|, \end{aligned} \tag{A.2.5}$$

where $Q_{xx} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and p_c denotes the center of mass position from the frame $\{b_i\}$. Other I can be computed in the same way with the above equation but the

different Q. The corresponding matrices are as follows:

$$\begin{aligned} Q_{yy} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, Q_{zz} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\ Q_{xy} &= \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, Q_{yz} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 \end{bmatrix} Q_{zx} = \begin{bmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}. \end{aligned} \tag{A.2.6}$$

The derivatives of ϕ_{b_i} are as follows:

$$\begin{aligned} \delta m_i &= \frac{\rho_L}{\sqrt{(p_{i+1} - p_i)^\top (p_{i+1} - p_i)}} \begin{bmatrix} p_i - p_{i+1} \\ p_{i+1} - p_i \end{bmatrix}^\top \begin{bmatrix} \delta p_i \\ \delta p_{i+1} \end{bmatrix}, \\ \delta p_c &= [\frac{\frac{1}{2} M_l^i}{M_{i+1} + M_l^i} I_{3\times 3} \quad \frac{M_{i+1} \frac{1}{2} M_l^i}{M_{i+1} + M_l^i} I_{3\times 3}] \begin{bmatrix} \delta p_i \\ \delta p_{i+1} \end{bmatrix} \\ &\quad + p_i \frac{M_{i+1} \rho_L}{2(M_{i+1} + M_l^i)^2 \sqrt{(p_{i+1-p_i})^\top ((p_{i+1} - p_i)}} \begin{bmatrix} p_i - p_{i+1} \\ p_{i+1} - p_i \end{bmatrix}^\top \begin{bmatrix} \delta p_i \\ \delta p_{i+1} \end{bmatrix} \\ &\quad + p_{i+1} \frac{-M_{i+1} \rho_L}{2(M_{i+1} + M_l^i)^2 \sqrt{(p_{i+1-p_i})^\top ((p_{i+1} - p_i)}} \begin{bmatrix} p_i - p_{i+1} \\ p_{i+1} - p_i \end{bmatrix}^\top \begin{bmatrix} \delta p_i \\ \delta p_{i+1} \end{bmatrix}, \\ \delta h_{b_i} &= \delta m_i p_c + m_i \delta p_c. \end{aligned} \tag{A.2.7}$$

Each derivative of I with respect to p can be computed by

$$\begin{aligned} \frac{\partial I}{\partial p_i} =& \rho_L (|p_{i+1} - p_i| (Q p_{i+1} - \frac{2}{3} Q(p_{i+1} - p_i) + p_{i+1}^\top Q p_i + \\ & \frac{1}{3} (p_{i+1} - p_i)^\top Q (p_{i+1} - p_i) \frac{1}{\sqrt{(p_{i+1} - p_i)^\top (p_{i+1} - p_i)}} (p_i - p_{i+1})), \end{aligned} \tag{A.2.8}$$

$$\begin{aligned}\frac{\partial I}{\partial p_{i+1}} =&\rho_L(|p_{i+1}-p_i|(Qp_i+\frac{2}{3}Q(p_{i+1}-p_i)+p_{i+1}^\top Qp_i+\\ &\frac{1}{3}(p_{i+1}-p_i)^\top Q(p_{i+1}-p_i)\frac{1}{\sqrt{(p_{i+1}-p_i)^\top(p_{i+1}-p_i)}}(p_i-p_{i+1}))+\\ &2M_{i+1}Qp_{i+1}.\end{aligned} \tag{A.2.9}$$

From Equation 3.2.7, the derivative of p becomes

$$\frac{\partial p_i}{\partial \eta_i} = \begin{bmatrix}-[p_i] & I_{3\times 3}\end{bmatrix}, \tag{A.2.10}$$

thus the matrix D in Equation 3.2.19 can be derived by the chain rule.

A.3 Derivative of the constraints

A.3.1 End-effector trajectory

Forward kinematics of an n-dof serial chain can be represented as follows:

$$T = \prod_{i=1}^{n} e^{[A_i]\theta_i} M, \tag{A.3.11}$$

where each A_i denotes the screw of joint i expressed in the base frame and M is the $SE(3)$ of the end-effector in its zero configuration. Each $e^{[A_i]\theta_i}$ can be locally parameterized by $e^{[\delta\eta_i]} e^{[A_i](\theta_i+\delta\theta_i)} e^{-[\delta\eta_i]}$ and its first-order approximation becomes

$$(I + [\delta\eta_i]) e^{[A_i]\theta_i} (I + [A_i]\delta\theta_i)(I - [\delta\eta_i]). \tag{A.3.12}$$

Therefore,

$$\begin{aligned} T^{-1}\delta T = {} & (e^{[A_2]\theta_2} \cdots M)^{-1} (e^{-[A_1]\theta_1} [\delta\eta_1] e^{[A_1]\theta_1} - [\delta\eta_1] + [A_1]\delta\theta_1)(e^{[A_2]\theta_2} \cdots M) \\ & + (e^{[A_3]\theta_3} \cdots M)^{-1} (e^{-[A_2]\theta_2} [\delta\eta_2] e^{[A_2]\theta_2} - [\delta\eta_2] + [A_2]\delta\theta_2)(e^{[A_3]\theta_3} \cdots M) \\ & + \cdots + M^{-1} (e^{-[A_n]\theta_n} [\delta\eta_n] e^{[A_n]\theta_n} - [\delta\eta_n] + [A_n]\delta\theta_n) M \\ & + M^{-1} e^{[\delta\eta_{n+1}]} M, \end{aligned} \tag{A.3.13}$$

where its vector form can be derived as follows:

$$\begin{aligned} (T^{-1}\delta T)^{\vee} = {} & \mathrm{Ad}_1(\delta\eta_1) - \mathrm{Ad}_2(\delta\eta_1) + \mathrm{Ad}_2(A_1)\delta\theta_1 + \cdots \\ & + \mathrm{Ad}_n(\delta\eta_n) - \mathrm{Ad}_{n+1}(\delta\eta_n) + \mathrm{Ad}_{n+1}(A_n)\delta\theta_n \\ & + \mathrm{Ad}_{n+1}(\delta\eta_{n+1}). \end{aligned} \tag{A.3.14}$$

Let Ad_i as $\mathrm{Ad}_{M^{-1}e^{-[A_n]\theta_n}\cdots e^{-[A_i]\theta_i}}$ for abbreviation. Thus, $(T^{-1}\delta T)^\vee = A\begin{bmatrix}\delta\eta\\ \delta\theta\end{bmatrix}$, where $A \in \mathbb{R}^{6\times(7n+6)}$ whose has elements as follows:

$$\begin{aligned}
&[A]_{(:,6(i-1)+1:6i)} = [\mathrm{Ad}_i] - [\mathrm{Ad}_{i+1}],\\
&[A]_{(:,6n+6+i)} = [\mathrm{Ad}_{i+1}]A_i,\\
&i = 1, \cdots, n,\\
&[A]_{(:,6n+1:6n+6)} = [\mathrm{Ad}_{n+1}].
\end{aligned} \tag{A.3.15}$$

A.3.2 Equations of motion of the base for quadruped robots

The equations of motion of the base can be represented as follows:

$$G_b\dot{V}_b - [\mathrm{ad}_{V_b}]^\top G_b V_b = -\sum_{j=1}^{4}([\mathrm{Ad}_{T_{b1}^{-1}}]^\top F_1)_j, \tag{A.3.16}$$

where T_{b1} denotes the $SE(3)$ from the base to the first body frame, and F_1 is the wrench acting on the first body frame with respect to this frame. The subscript j at the far right denotes the j^{th} leg. Since we fix the inertia and movement of the base, the derivative of the left-hand side becomes zero, and the right-hand side is

$$-\sum_{j=1}^{4}([\mathrm{Ad}_{T_{b1}^{-1}}]^\top R_1\delta x - [\mathrm{Ad}_{T_{b1}^{-1}}]^\top[\mathrm{ad}^*_{F_1}]([\mathrm{Ad}_{T_{b1}^{-1}}] - I)\delta\eta_1 - [\mathrm{Ad}_{T_{b1}^{-1}}]^\top[\mathrm{ad}^*_{F_1}]A_1\delta\theta_1)_j. \tag{A.3.17}$$

A.4 Laikago Specification

This chapter shows the initial design parameters of Laikago. Each frame is described with respect to the base frame of the quadruped.

Design frame	T_1	T_2	T_3	T_{ee}
Front right	$\begin{bmatrix} 0 & 0 & -1 & 0.195 \\ 0 & 1 & 0 & -0.082 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & 0.195 \\ 0 & 0 & 1 & -0.136 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & 0.053 \\ 0 & 0 & 1 & -0.118 \\ 1 & 0 & 0 & -0.161 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0.013 \\ 0 & 1 & 0 & -0.118 \\ 0 & 0 & 1 & -0.436 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Front left	$\begin{bmatrix} 0 & 0 & 1 & 0.195 \\ 0 & 1 & 0 & 0.082 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & 0.195 \\ 0 & 0 & 1 & 0.136 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & 0.053 \\ 0 & 0 & 1 & 0.118 \\ 1 & 0 & 0 & -0.161 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0.013 \\ 0 & 1 & 0 & 0.118 \\ 0 & 0 & 1 & -0.436 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rear right	$\begin{bmatrix} 0 & 0 & -1 & -0.238 \\ 0 & 1 & 0 & -0.082 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & -0.238 \\ 0 & 0 & 1 & -0.136 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & -0.380 \\ 0 & 0 & 1 & -0.118 \\ 1 & 0 & 0 & -0.161 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & -0.420 \\ 0 & 1 & 0 & -0.118 \\ 0 & 0 & 1 & -0.436 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rear left	$\begin{bmatrix} 0 & 0 & -1 & -0.238 \\ 0 & 1 & 0 & 0.082 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & -0.238 \\ 0 & 0 & 1 & 0.136 \\ 1 & 0 & 0 & 0.047 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & -0.380 \\ 0 & 0 & 1 & 0.118 \\ 1 & 0 & 0 & -0.161 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & -0.420 \\ 0 & 1 & 0 & 0.118 \\ 0 & 0 & 1 & -0.436 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table A.1: Initial design parameters of Laikago.

Bibliography

[1] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[2] Frank Chongwoo Park. The optimal kinematic design of mechanisms. 1991.

[3] Yunjiang Lou, Guanfeng Liu, Ni Chen, and Zexiang Li. Optimal design of parallel manipulators for maximum effective regular workspace. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 795–800. IEEE, 2005.

[4] Wei Li and Jorge Angeles. The design of a 3-cps parallel robot for maximum dexterity. *Mechanism and Machine Theory*, 122:279–291, 2018.

[5] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Task-based limb optimization for legged robots. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2062–2068. IEEE, 2016.

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

[7] Gabriel Bravo-Palacios, Andrea Del Prete, and Patrick M Wensing. One robot for many tasks: Versatile co-design through stochastic programming. *IEEE Robotics and Automation Letters*, 5(2):1680–1687, 2020.

[8] EJ Van Henten, DA Van't Slot, CWJ Hol, and LG Van Willigenburg. Optimal manipulator design for a cucumber harvesting robot. *Computers and electronics in agriculture*, 65(2):247–257, 2009.

[9] Yuan Yun and Yangmin Li. Optimal design of a 3-pupu parallel robot with compliant hinges for micromanipulation in a cubic workspace. *Robotics and Computer-Integrated Manufacturing*, 27(6):977–985, 2011.

[10] Sehoon Ha, Stelian Coros, Alexander Alspach, James M Bern, Joohyung Kim, and Katsu Yamane. Computational design of robotic devices from high-level motion specifications. *IEEE Transactions on Robotics*, 34(5):1240–1251, 2018.

[11] Ruta Desai, Ye Yuan, and Stelian Coros. Computational abstractions for interactive design of robotic devices. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1196–1203. IEEE, 2017.

[12] Jian Li, Sheldon Andrews, Krisztian G Birkas, and Paul G Kry. Task-based design of cable-driven articulated mechanisms. In *Proceedings of the 1st Annual ACM Symposium on Computational Fabrication*, pages 1–12, 2017.

[13] Volkert Van Der Wijk, Sébastien Krut, François Pierrot, and Just L Herder. Design and experimental evaluation of a dynamically balanced redundant planar 4-rrr parallel manipulator. *The International Journal of Robotics Research*, 32(6):744–759, 2013.

[14] Boston Dynamics. Spot. `https://www.bostondynamics.com/spot`, 2020.

[15] R McNeill Alexander. *Optima for animals*. Princeton University Press, 1996.

[16] Ewald R Weibel, Ewald R Webel, C Richard Taylor, and Liana Bolis. *Principles of animal design: the optimization and symmorphosis debate*. Cambridge University Press, 1998.

[17] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Computational co-optimization of design parameters and motion trajectories for robotic systems. *The International Journal of Robotics Research*, 37(13-14):1521–1536, 2018.

[18] Jaewoon Kwon, Keunjun Choi, and Frank C Park. Kinodynamic model identification: A unified geometric approach. *IEEE Transactions on Robotics*, 2020, under review.

[19] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[20] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[21] Frank C Park, Beobkyoon Kim, Cheongjae Jang, and Jisoo Hong. Geometric algorithms for robot dynamics: A tutorial review. *Applied Mechanics Reviews*, 70(1), 2018.

[22] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[23] Michael Nalin Mistry. *The representation, learning, and control of dexterous motor skills in humans and humanoid robots*. University of Southern California, 2009.

[24] Cheng Li, Yuanqing Wu, Harald Löwe, and Zexiang Li. Poe-based robot kinematic calibration using axis configuration space and the adjoint error model. *IEEE Transactions on Robotics*, 32(5):1264–1279, 2016.

[25] Zhijun Li, Shuzhi Sam Ge, and Sibang Liu. Contact-force distribution optimization and control for quadruped robots using both gradient and adaptive neural networks. *IEEE transactions on neural networks and learning systems*, 25(8):1460–1473, 2013.

[26] Willow Garage. Universal robot description format (urdf). *Http://Www. ros. org/urdf/, 2009*, 2009.

[27] James E Bobrow, B Martin, G Sohl, EC Wang, Frank C Park, and Junggon Kim. Optimal robot motions for physical criteria. *Journal of Robotic systems*, 18(12):785–795, 2001.

[28] C-YE Wang, Wojciech K Timoszyk, and James E Bobrow. Payload maximization for open chained manipulators: finding weightlifting motions for a puma 762 robot. *IEEE Transactions on Robotics and Automation*, 17(2):218–224, 2001.

[29] Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

[30] D Eberly. Least-squares fitting of data with b-spline curves. *url: https://www. geometrictools.com/Documentation*, 2014.

[31] Shu-jun Lian. Smoothing approximation to l1 exact penalty function for inequality constrained optimization. *Applied Mathematics and Computation*, 219(6):3113–3121, 2012.

[32] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258, 2011.

[33] Unitree Robotics. Laikago pro. `http://www.unitree.cc/e/action/ShowInfo.php?classid=6&id=1`, 2018.

[34] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. Interactive design of 3d-printable robotic creatures. *ACM Transactions on Graphics (TOG)*, 34(6):1–9, 2015.

국문초록

로봇 디자인에는 액츄에이터, 링크, 관절 등과 같은 구성요소의 수많은 조합 가능성이 존재한다. 따라서, 좋은 로봇 디자인을 찾는 과정은 전문가에게도 어려운 문제이다. 위 문제점을 극복하기 위해 로봇의 동작을 고려하여 형태를 최적화하는 방법론을 제시한다. 제시된 방법론을 통해 특정 작업을 위한 로봇 형태 및 동작의 동시 최적화가 가능하다. 위 방법론은 형태 및 동작 변수가 결합된 공간 상에서 목적함수를 가장 많이 감소시키는 구속조건 매니폴드 상에서의 방향을 찾아 최적화를 진행한다. 이전 연구들의 결점을 극복하기 위해 우리는 최근 개발된 반복 미분 동역학(recursive differential dynamics) 알고리즘을 사용한다. 이 알고리즘을 통해 관절 토크 변화와 형태 변화 사이의 해석적 관계를 계산할 수 있다. 따라서, 제시된 방법론을 사용하면 더욱 빠르고 정확한 최적화 결과를 도출할 수 있다. 총 두 가지 수치적 실험을 통해 위 최적화 방법론을 검증하였다: 엔드이펙터가 주어진 궤적을 추종하는 2축 평면 매니퓰레이터, 4족로봇의 보행작업.

주요어: 로봇 디자인 최적화, 강체 동역학, 최적 제어, 보행로봇

학번: 2018-20883