國立臺灣大學工學院機械工程學研究所

博士論文

Department of Mechanical Engineering

College of Engineering

National Taiwan University

Doctoral Dissertation

人型機器人之最佳化步態生成與即時控制

Optimized Walking Pattern Generation and Real-time Control for Humanoid Robots

嚴舉樓

Jiu-Lou Yan

指導教授：黃漢邦 博士

Advisor: Han-Pang Huang, Ph.D.

中華民國 101 年 6 月

June 2012

國立臺灣大學博士學位論文
口試委員會審定書

人型機器人之最佳化步態生成與即時控制

Optimized Walking Pattern Generation and Real-time Control for Humanoid Robots

本論文係 嚴舉樓 君（學號 F94522830）在國立臺灣大學機械工程學系完成之博士學位論文，於民國 101 年 06 月 07 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

黃漢邦 （簽名）

（指導教授）

[illegible] 羅仁權　[illegible] 蔡得民

[illegible]　[illegible]

[illegible] 李祖添

系主任 [illegible] 楊耀州 （簽名）

Department of Mechanical Engineering
National Taiwan University
Taipei, TAIWAN, R.O.C.

Date: June 7, 2012

We have carefully read the dissertation entitled

"Optimized Walking Pattern Generation and Real-time Control for Humanoid Robots"

submitted by Jiu-Lou Yan in partial fulfillment of the requirement of the degree of **DOCTOR OF PHILOSOPHY** and recommend its acceptance.

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

Advisor: [illegible]

Chairperson of
Department of Mechanical Engineering: [illegible]

致謝

首先，我要感謝長久支持我的爸爸、哥哥以及在天國享樂的媽媽，沒有你們，就沒有現在的我，接下來我要開始人生的下一段旅程了!!

感謝指導教授　黃漢邦老師多年來的教誨與指引，認識老師到現在也已經八年了，老師不只在學術方面指導我，也時常在人生與思想方面給予許多建議，讓我在這個實驗室生涯中，學到許多寶貴的知識與經驗，也度過了重重的難關，老師，謝謝您!! 在此，也要感謝論文口試委員福田敏男教授、李國民教授、羅仁權教授、李祖添教授、蔡得民教授，在論文修改上給予精闢的意見，讓本篇論文更加完整與豐富。

感謝在實驗室這些年遇到的所有人，書耘大學長從我是專題生的時候就開始指引我進入機器人這塊領域，也常常會靈光一閃提供我許多的意見；算是最大的損友子豪與聖諺在這幾年中，也和我一起度過了許多碎碎念與游泳交換八卦的時光；至於跟我同一屆打拼的軍毅、乙至與峻弘，我到現在都不時會想起我們一起度過那手忙腳亂的兩年。也感謝在我也變成學長之後，認真幫忙的學弟學妹們，尼諾機器人是由大家的努力所共同完成的!! 在這邊也要特別感謝參與這次人型機器人成果發表會的各位，畢業前能夠成功展出機器人，都是由於你們的幫忙：笑聲超級爽朗的泓逸，謝謝你總是主動幫忙，還有你真是很會找好吃的東西；認真負責的瑋軒，謝謝你都幫我們接下像是機械臉或是整線等難以處理的工作；萬能的聖翔，謝謝你幫我們負責手臂的部分；熱心的秀婷與衍文，謝謝你們一起排演邦妮與尼諾的表演；還有要特別感謝毓文，在我們最困難的時候回來加入我們，推我們一把，讓成果發表會能夠順利進行。

最後，我要謝謝我十年來的好朋友聯聖(熊)以及嘉德(Tito)，雖然我們不同實驗室，但是你們總是能在我需要的時候幫助我與陪伴我，謝謝你們。也謝謝所有在台大這十年之間遇到的所有人!!

2012/06/22 於機器人實驗室

摘要

早從二十世紀初，人類對於人型機器人的幻想就不曾停止過，隨著時代的演進，我們對於人型機器人的想像一直都是類似的，會跑、會跳、行為舉止要像人類、甚至是超越人類，都是我們對於機器人的期待，可惜由於材料與致動器的限制，機器人的重量與力量輸出的比例一直沒有辦法追上人類，導致機器人的身體能力一直都沒有辦法突破到符合我們想像的程度，直到最近，由於科技的進步以及經驗的累積，才由日本與美國的研究者開發出具有跑跳能力並且十分穩定的人型機器人，機器人的身體能力也才慢慢開始足以應付這些困難的要求。但是在運動規劃方面，目前在學界與業界的研究上，仍然是各自採用各自的機器人平台與演算法來開發機器人的控制與運動規劃，各種的方法也有各自的限制與長處。有鑑於此，本論文提出了一套可通用於人型機器人之最佳化運動控制與步態生成器，此方法兼具了即時性與機器人重心高度與地面高度可變之特點，可達成自由指定零力矩點(Zero Moment Point)軌跡與重心高度軌跡輸入之步態生成。利用此一步態生成器與牛頓—尤拉動力學(Newton-Euler dynamics)，我們可以設計一個代價函數(cost function)並且求得重心高度軌跡對於此代價函數之導數，進而求得在指定零力矩點輸入軌跡之下最佳化之重心高度軌跡，達成本論文所提出之人型機器人之最佳化 3D 重心軌跡之步態生成器。除此之外，本論文也利用多個狀態機(state machine)與 USB-to-CAN-Bus 通訊網路建立一套人型機器人之即時控制系統，利用此系統，也驗證了我們所提出的即時步態生成系統的效能。

關鍵字：人型機器人、即時步態生成、最佳控制、全身運動規劃

Abstract

Since early twentieth century, human continues to imagine the future of humanoid robots. As the time going on, we always wish humanoid robots can run, jump, act like human, and even be better than human. Because of the limitations of material technology and actuators, the ratio of weight and power output of robots cannot reach the same level as human. Until now, due to the improvement of technology and the accumulated experiences, researchers in America and Japan start to demonstrate that their new robots can run and jump smoothly and stably. And the robots nowadays start to be capable of these difficult tasks. However, in the aspect of motion planning and pattern generation, the researchers in academy and in industry use their own robot platforms and algorithms to develop their motion control and planning systems. These systems have their own advantages and limitations. In view of this, an optimized walking pattern generator for humanoid robots is proposed in this dissertation. The proposed pattern generator can solve walking patterns with arbitrary assigned COG (Center of Gravity) height trajectory and 3D ZMP (Zero Moment Point) trajectory in real-time. Thus, walking pattern generation with arbitrary assigned ground height status is achievable. Based on the proposed walking pattern generator and Newton-Euler dynamics, a cost function is designed to optimize COG height trajectory with given ZMP trajectory. An optimized 3D COG walking pattern generator can be achieved in this dissertation. In addition, state machine based distributed control system with USB-to-CAN-bus interface is used to construct a real-time robot control system. Using this system, the performance of the proposed walking pattern generator is also verified.

Keywords: Humanoid Robot, Real-time Walking Pattern Generation, Optimal Control, Whole Robot Motion Planning

Contents

List of Tables

List of Figures

Nomenclature

Notations

Robotics

Kinematics

$D_{0,i}$	position of the *i*th joint in the world coordinates
$h(\theta)$	manipulability of the mechanism
$H(\theta)$	joint limit cost in WLN method
J	Jacobian matrix of robot kinematics
J^{+}	pseudoinverse of J
J_{α}	Jacobian matrix in DLS method
$J_{W,\alpha}$	Jacobian matrix in RWLN method
$R_{0,i}$	the rotation matrix of the *i*th joint in the world coordinates
$T_{0,i}$	the homogeneous matrix of the *i*th joint
W	joint limit weighting matrix in WLN method
w_i	the *i*th diagonal element of W
z_i	unit vector along positive direction of the *i*th z-axis
X	combined position/orientation vector of all end-effectors
A	the damping factor in DLS method
θ	robot joint angle vector
$\dot{\theta}$	robot joint velocity vector
θ_i	the *i*th joint angle
Φ	null space of the solution of inverse kinematics

Dynamics

$\vec{a}_{c,i}$	acceleration of the COG of the *i*th link
$\vec{a}_i$	acceleration of the *i*th joint
$\vec{A}_M$	total angular momentum
$\vec{A}_{M,spin}$	total spin angular momentum
$\vec{A}_{M,orbit}$	total orbit angular momentum
f_i	force acts on the *i*th joint
I_i	moment of inertia of the *i*th link
$\vec{r}_i$	vector form the *i*th to the (*i*+1)th joint
$\vec{r}_{c,i}$	vector from the *i*th joint to the COG of the *i*th link
$\vec{r}_{s\to c,i}$	vector from the support foot to the *i*th joint
$\vec{L}_M$	total linear momentum
$\vec{L}_{M,i}$	linear momentum of the *i*th link
$m_i\vec{g}$	gravity force of the *i*th link
$\vec{v}_{c,i}$	velocity of the COG of the *i*th link
$\vec{v}_i$	velocity of the *i*th joint
$\vec{\alpha}_i$	angular acceleration of the *i*th joint
$\vec{\alpha}_{c,i}$	angular acceleration of the COG of the *i*th link
$\vec{\omega}_i$	angular velocity of the *i*th joint
$\vec{\omega}_{c,i}$	angular velocity of the COG of the *i*th link
$\vec{\tau}_i$	torque of the *i*th joint

Whole Body Inverse Kinematics

$\dot{d}$	input speed vector of end-effectors in each IK loop
$\dot{d}_{stance}$	input speed vector of the stance leg
$\dot{d}_{swing}$	input speed vector of the swing leg
I_a	accumulated momentum of inertia
J_{AM}	angular momentum Jacobian
J_C	COG Jacobian matrix
$J_{f\to X}$	Jacobian matrix describing the effect caused by the fixed leg
J_F	Fixed-Leg-Motion Jacobian matrix
J_{LM}	linear momentum Jacobian
J_{stance}	Jacobian matrix of the stance leg
J_{swing}	Jacobian matrix of the swing leg
m_a	mass of a link affected by the movement of a specified joint
m_i	mass of the *i*th link
m_{ua}	mass of a link unaffected by the movement of a specified joint
M	total mass of the robot
M_a	total mass of links affected by the movement of a specified joint
M_{ua}	total mass of links unaffected by the movement of a specified joint
$\vec{r}_a$	position vector of the COG of the mass m_a
$\vec{r}_{COG}$	position vector of the COG of the robot
$\vec{r}_{m,i}$	position vector of the COG of the *i*th link in the world coordinates
$\vec{r}_{i\to end}$	position vector of the *i*th joint to the end-effector
$\vec{r}_{ua}$	position vector of the COG of the mass m_{ua}
v_{end}	linear velocity an end-effector
ω_{end}	angular velocity an end-effector

Linear Quadratic State Incremental Control (LQSI Control)

Inverted Pendulum Model and State Space Model

A	continuous time state matrix
A_k	*k*th discrete time state matrix
B	continuous time state matrix
B_k	*k*th discrete time state matrix
C	continuous time state matrix
C_k	*k*th discrete time state matrix
g	gravity constant
p_k	*k*th ZMP reference value
u_k	*k*th control input of the state space system
C_X	COG position in x direction
Z_X	ZMP position in x direction
C_Y	COG position in y direction
Z_Y	ZMP position in y direction
C_Z	COG position in z direction
Z_Z	ZMP position in z direction

LQSI Controller and Preview Controller

f_p	preview gain of LQSI controller and preview controller
J	performance index of LQSI controller
$\mathrm{K}_{\mathrm{k}}^{\mathrm{v}}$	feed-forward gain of LQSI controller
K_k^{x}	state feedback gain of LQSI controller
Q	weighting of tracking error in the performance index
Q_x	weighting of state increment in the performance index
R	weighting of control input in the performance index

S_k	The *k*th iteration of *S* matrix of the Riccati equation
S_∞	solution to the Riccati equation using constant state matrices
v_k	feed-forward control input of LQSI controller
$\emptyset$	boundary condition of the performance index

3D COG optimization

H_j	cost function of joint limit
$I_{0,i}$	moment of inertia of the *i*th link in local coordinates
$J_{C_x}^{+}$	joint angle change under unit COG position change in x direction
$J_{C_y}^{+}$	joint angle change under unit COG position change in y direction
$J_{C_z}^{+}$	joint angle change under unit COG position change in z direction
P_k	totoal cost at the *k*th sampling time point
$W_{\tau,j}$	weighting of torque cost of the *j*th joint
$W_{\theta,j}$	weighting of joint limit cost of the *j*th joint
τ_j	torque of the *j*th joint
η	learning rate of COG height training

Acronyms

COG	Center Of Gravity
CAN	Controller Area Network
DH method	Denavit-Hartenberg Method
DSP	Double Support Phase
FIFO	First In First Out
FK	Forward Kinematics
IK	Inverse Kinematics
LIPM	Linear Inverted Pendulum Model
LQ	Linear Quadratic
LQI	Linear Quadratic Integral
LQSI	Linear Quadratic State Incremental
NSM	Network Scheduling Mechanism
PON	Priority Oriented Networking
RTNET	Real-time Network
RWLN	Robust Weighted Least Norm
SSP	Single Support Phase
ZMP	Zero Moment Point

Chapter 1 Introduction

The motivation, contributions, and overall framework of this dissertation are discussed in this chapter. Advantages and disadvantages of different types of robot are described in section 1.1. Section 1.2 shows several methods used for pattern generation for humanoid robots. Section 1.3 compares different humanoid robots and control systems. Section 1.4 shows the contributions of this dissertation. With this section, the main ideas and their functions can be realized easily. Finally, section 1.5 describes the overall framework of the dissertation and the relationship among all chapters.

1.1 Different Types of Robots

In order to achieve different goals and objectives, many different types of robots are built, including wheeled robots, legged robots, industrial robots, etc. Wheeled robots are most used as service robots and exploration robots. Simultaneous localization and mapping (SLAM) [35][134], exploring and mapping [11], and motion planning [64] algorithms are often verified on mobile robots. Legged robots are developed for rugged terrains. Some quadruped robots can go through rugged terrains very successful, such as the TITAN series [28][42] and the big dog (Boston Dynamics) [154]. Quadruped robots or robots with more legs can go through rugged terrains stably since the supporting polygon is much larger than biped robots. For biped robots, walking through rugged terrains is still a challenge. Some researchers in America and Japan proposed their new generation of robots and accomplish several difficult challenges, such as Petman (Boston Dynamics) [154], ASIMO [110][153], and HRP series [47][56][57][59]. Different form legged robots, industrial robots can be classified according to their objectives, such as industrial robot arms [22], grippers [116], and parallel robots [120][138].

Nowadays, there are so many types of robots and they are built for their own propose. Robots are still more expensive than other products and only industrial robots for factory and small-sized robots for education or entertainment can be commercialized. Nevertheless, we still have a dream that one day humanoid robots can be everywhere around us. However, when walking through different types of environments, there are still many problems must be solved for humanoid robots. For example, robots fall down easily in unknown environments. The linearized inverted pendulum model (LIPM) constrains humanoid robots to walk with constant COG height. Some optimizations of robot motions cannot be processed in real-time. Humanoid robots are too heavy and the endurance of batteries is not long enough for long-term operation. These problems are solved one-by-one with the improvement of technology in these years. In this dissertation, we attempt to develop a 3D optimized walking pattern generation and real-time control system. By using the proposed system, the mobility of humanoid robot can be improved and walking pattern generation in more types of environments can be achieved.

1.2 Pattern Generation System for Humanoid Robots

Starting from the algorithms for solving inverse kinematics (IK), more considerations are required for legged robots than robot arms or grippers. The balance problem of biped robots is more critical and more constrained than other types of robots. The switching between single support phase (SSP) and double support phase (DSP), and how to manipulate the kinematics relationship in these phases are also important for biped robots. In addition, some dynamics of the robots can be added to the kinematics solver, such as the control of COG, linear and angular momentum, or other physics properties of the robot. Researchers have proposed several solutions for COG Jacobian [3][119][124] and momentum Jacobian [30][50]. Based on the well-known inverted pendulum model [97][114][140], the control of COG is especially important for the balance control of biped robots. Momentum compensation is also a good method to improve the stability of robot and to generate more human-like walking patterns. However, since conventional methods needs complex computation and coordinate transformation, a generalized whole body IK solver is proposed in this dissertation to simplify and generalize the IK solver. This will be discussed in chapters 2 and 3.

Based on the basic IK solver, the walking pattern generation algorithms can be developed. The most used criterion to check the stability of biped walking is the (zero moment point) ZMP method [136]. It is a simple and powerful criterion for the stability of biped robots. By checking the position of ZMP, the status of stability can be verified. Using the concept of ZMP, there are many methods to construct a walking pattern generator. For example, methods using a controller [70][92][127], Fourier series/transform [104][108][141], neural networks [51], genetic algorithm [12][108][141], rapid-exploring random tree (RRT) and probabilistic roadmap (PRM) [7][65], and motion capture [52] can generate walking patterns for humanoid robots. Among these methods, controller based walking pattern generator is the most used one. Each method has its own advantages and disadvantages. Learning algorithms can generate optimized walking patterns but their computation/learning time is too long to be used in real-time. Controller based methods need a more precise model for good performance. RRT method is quick but it generates different results each time. Motion capture method can generate the most natural and human-like results, but since the mechanism and mass distribution of robots are different from humans, the stability of the captured trajectories must be further verified. In this dissertation, a real-time LQ (linear quadratic) control based method, the LQSI (linear quadratic state incremental) controller, is proposed to achieve real-time walking pattern generation. The robot is modeled as an inverted pendulum satisfying the COG/ZMP equations.

There are several models for modeling a humanoid robot, some are complex and some are relatively simple, as shown in Figure 1-1.

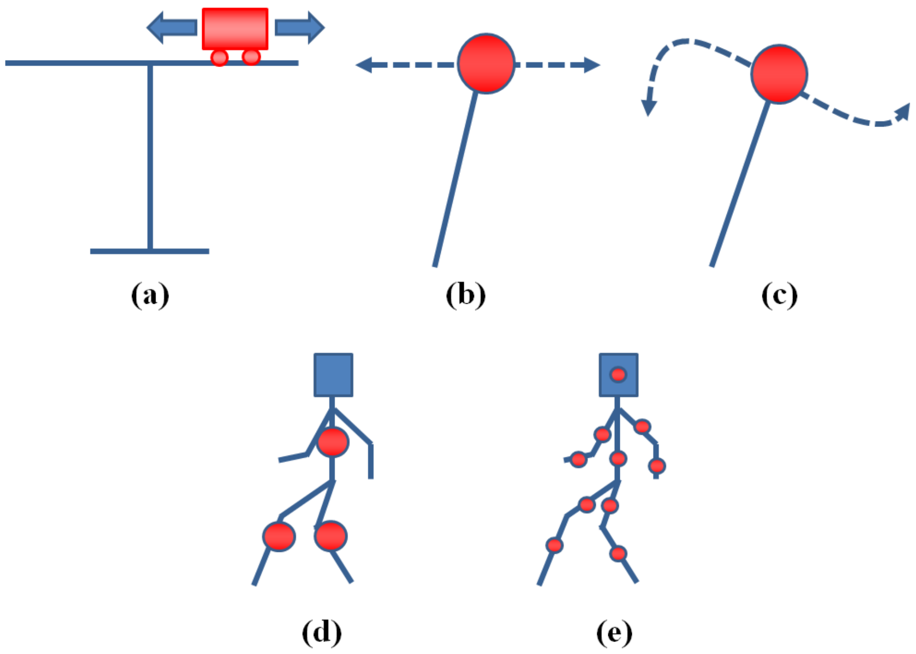


Figure 1-1. Types of model for humanoid robots

Figure 1-1 shows five types of models used to control humanoid robots. In the figure, (a) and (b) are the well-known cart-table model and the linear inverted pendulum model (LIPM) [31][71]; they have the same governing equations and are basically the same, (c) is the nonlinear inverted pendulum model [87], (d) is the three mass model [127], and (e) is the most complex one, the multi-mass model [107][128]. In these models, more complex models have less modeling errors, but they are more difficult to be used for humanoid robots because of their high nonlinearity. In this dissertation, the model of (c) is used for walking pattern generation and (e) is used for dynamic computation and momentum compensation.

For walking pattern generation methods using inverted pendulum model, there are three types of inverted pendulum models considering the relationship between COG and ZMP. Each type of model has its advantages and disadvantages. The first type of model is the most complex and nonlinear one. It considers both the motion of COG in vertical direction and the multi-link effects of robot motions. Both terms make the COG/ZMP equations nonlinear. Since the multi-link effects of robot motions cannot be handled and calculated easily, the second type of model ignores this term in the COG/ZMP equations. Fortunately, the multi-link effects of robot motions are relatively small and can be ignored in most cases which have no large and exaggerated motions. The third type linearizes the COG/ZMP equations by setting the COG height trajectory as a constant and it also ignores the multi-link effects. The computation and implementation of the third type is simple and quick. Since the third type of inverted pendulum model can be used to generate stable walking patterns easily, it is most used among the three types of model. The most well-known example is the cart-table model and the preview control method [2][36][47]. In this dissertation, the second type of model is used to extend the ability of motion when the robot needs to change its COG height in the environments with stairs, slopes, or some obstacles must be avoided. The details of the proposed method are described in chapter 4. Using the second type of model, because the multi-link effects of robot motions are ignored and this will result in un-modeled torque terms in the COG/ZMP equations. Momentum compensation can be used to reduce the multi-link effects in x and y (horizontal) directions if the robot has redundant degrees of freedom. In this dissertation, momentum compensation is used in z direction to reduce the slipping around the z-axis in the world coordinate, the derivations and discussions about momentum compensation are shown in chapter 3.

After achieving a walking pattern generator that can allow varying COG height trajectories, the input COG height trajectories can be optimized to generate more human-like and energy-saving walking patterns for humanoid robots. The LQSI controller in chapter 4 is proposed to optimize the horizontal (sagittal and lateral) COG trajectories, and the method in chapter 5 is proposed to optimize the vertical COG trajectory. In chapter 5, the pseudoinverse matrix in the proposed IK solver and the Newton-Euler dynamics are used to find the gradient of a cost function. And then the COG height trajectory is optimized by minimizing the cost function.

1.3 Control System for Humanoid Robots

The selection of the control system for humanoid robots is very important for real-time control for humanoid robots. Long time delay while transmitting and receiving data is not allowed. There are many types of communication ports and protocols, some are designed for computers and some are designed for vehicles. The communication ports and protocols for computers have higher speed but lower robustness, different from the them, the hardware and software design for communication ports for vehicles needs more robustness against noise, this also make the speed lower.

Many devices use RS232 as communication port to control the platform since it is easy and convenient. But RS232 is too slow and not robust enough for humanoid robots. USB is a quick and generalized communication port. It is fast but can only be connected back to the central controller one-by-one; serial connection is not allowed. Using Ethernet to connect the nodes on the bus can achieve a very small time delay but the circuit design is more complex. Considering the requirements of real-time and robustness, CAN-Bus is used to construct the local control bus in this dissertation for controlling the motors and receiving data from the sensors of the humanoid robot. CAN-Bus has 1Mbps speed and good robustness against noise; serial connection is also achievable. Only one set of wires is required for CAN-Bus communication, thus the whole control system can be built without using a huge amount of wires. The proposed humanoid robot contains several such CAN-Bus networks, and each local network is controlled by the main computer through USB-to-CAN-Bus adaptors. Using the proposed USB-to-CAN-Bus based control architecture, each cycle of motion control and sensor reading of the whole robot can be done in 5ms. The whole architecture and the protocol of the proposed control system are discussed in chapters 6 and 7.

1.4 Contributions

To improve the ability of motion of humanoid robots, this dissertation attempts to develop a real-time walking pattern generation and control system. There are five main contributions of this dissertation. They are listed as follows:

- The concept of Fixed-Leg-Motion Jacobian matrices.
- A whole body inverse kinematics solver for humanoid robots.
- The derivation and application of the LQSI controller for humanoid robots.
- A Newton-Euler dynamics based COG height trajectory optimization method.
- Architecture and implementation of a real-time control system for robots.

Fixed-Leg-Motion Jacobian

Using the concept of Fixed-Leg-Motion Jacobian matrices, we can build the Jacobian matrices for solving IK of legged robots with a direct and neat way. Complex computation and coordinate transformation can be avoided. In this dissertation, the concept of Fixed-Leg-Motion Jacobian is derived and used on the conventional Jacobian, the COG Jacobian, and the momentum Jacobian matrices. It can also be used to construct other kind of Jacobian matrices describing other physical quantities for legged robots.

Whole Body Inverse Kinematics Solver

The proposed whole body IK solver can be used to control all the end-effectors of the robot, including the hands, arms, legs, head, and torso. In addition, physical quantities which are directly relative to the dynamics of the robot can also be controlled, including COG, linear momentum, and angular momentum. By using the proposed Fixed-Leg-Motion Jacobian method, the proposed whole body IK solver can solve IK without complex coordinate transformation.

LQSI COG/ZMP Pattern Generator

In order to ensure and improve the stability of humanoid robots when they are walking in height-changing environments, the LQSI controller is proposed in this dissertation. LQSI controller is a LQ control based controller with state-incremental performance index. Since the proposed LQSI controller uses a linear time varying version of inverted pendulum model for COG/ZMP walking pattern generation, the input COG height trajectory can be arbitrarily assigned. Due to this property, humanoid robots can walk on height-changing environments with the LQSI controller. LQSI controller is also implemented using C++ for the proposed humanoid robot. Each re-planning for COG patterns can be processed in 1ms. Real-time 3D COG/ZMP pattern generation can be achieved using the proposed LQSI controller.

Optimized COG height Trajectory

When using the proposed LQSI controller for pattern generation, the COG height trajectory is arbitrarily assigned. The COG trajectory is optimized in horizontal (sagittal and lateral) directions. For generating more natural and energy-saving walking patterns, the derivation and discussion of the optimization of the COG height trajectory is proposed in this dissertation. A cost function including the terms of joint limit and joint torque is minimized to optimize the COG height trajectory of the robot. The gradient of the cost function is calculated by calculating the derivative of the cost function with respect to COG height. And the derivatives are derived and calculated by using the pesudoinverse matrix of the IK solver, Newton-Euler dynamics, and several kinematics techniques.

1.5 Overall Framework of the Dissertation

The overall framework of the dissertation is shown in Figure 1-2.

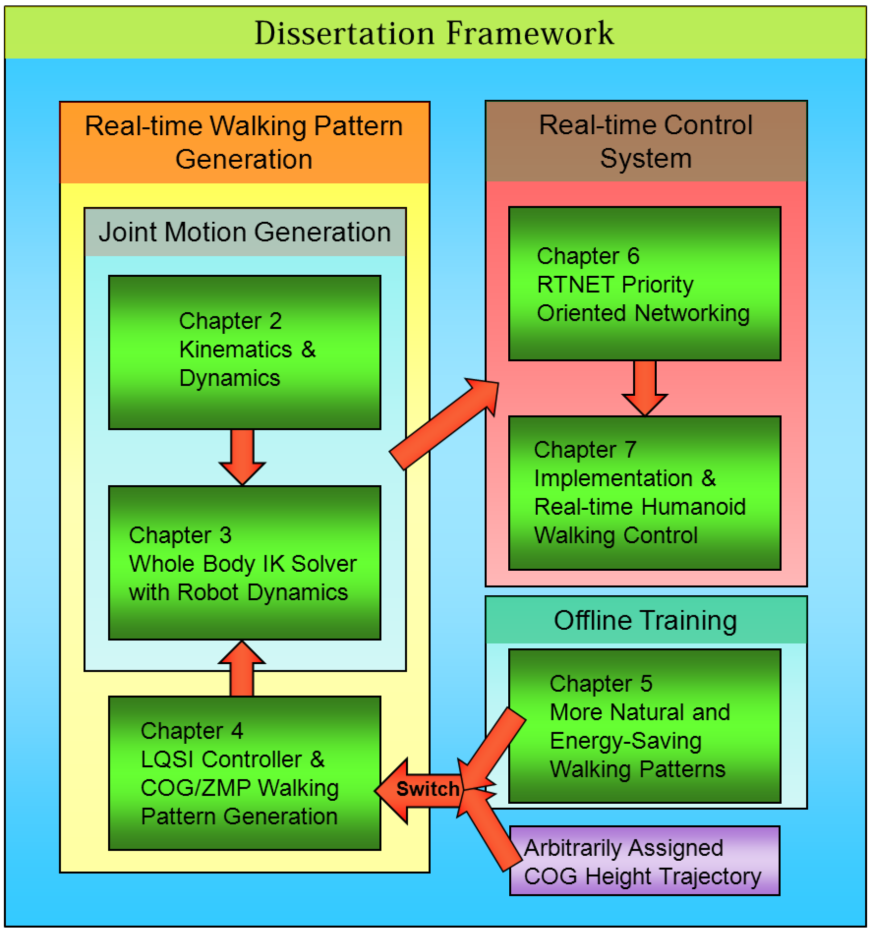

Dissertation Framework
Real-time Walking Pattern Generation
Joint Motion Generation
Chapter 2 Kinematics & Dynamics
Chapter 3 Whole Body IK Solver with Robot Dynamics
Chapter 4 LQSI Controller & COG/ZMP Walking Pattern Generation
Real-time Control System
Chapter 6 RTNET Priority Oriented Networking
Chapter 7 Implementation & Real-time Humanoid Walking Control
Offline Training
Chapter 5 More Natural and Energy-Saving Walking Patterns
Switch
Arbitrarily Assigned COG Height Trajectory

Figure 1-2. The overall framework of the dissertation

In Figure 1-2, the main topics of the chapters are listed in each block. Chapters 2 and 3 are combined as a "Joint Motion Generation" block. Using the contents in chapters 2 and 3, the trajectories of all joints of the robot can be generated. Chapter 4 focuses on the balance and stability of walking. The contents in chapters 2, 3, and 4 are used to construct the proposed real-time walking pattern generator satisfying COG/ZMP inverted pendulum equations. In this stage, the COG trajectory is optimized in horizontal (sagittal and lateral) directions. The COG height trajectory is directly calculated by considering the ground height of the environment. In chapter 5, an optimization algorithm for COG height trajectory is proposed. The time required for each optimization for different 3D ZMP input is about two minutes. As a future work, real-time implementation of the method in chapter 5 can be achieved by constructing a database of training results. Using the algorithms form chapters 2 to 5, an optimized 3D COG walking pattern generation with 3D ZMP input can be achieved. For the implementation part, chapters 6 and 7 discuss the protocol and the hardware used to build the real-time control system in this dissertation. Using the algorithms and techniques in this dissertation, optimized walking pattern generation and real-time control are achieved.

Chapter 2 Kinematics and Dynamics

Basic kinematics and dynamics of robots are developed for many years. Equations and descriptions of the behaviors of rigid body and the center of gravity (COG) of each link are very important for constructing the model of robots. This chapter shows the basic knowledge of kinematics and dynamics used in this dissertation. All equations and concepts are used in the other chapters in this dissertation.

2.1 Introduction

Many researchers have proposed the solutions to the singularity problem and the joint limit problem while solving Jacobian linearized IK. They include the damped least square method (DLS) [137] and the robust damped least square method (RDLS) [89], which are used for singularity avoidance. The weighted least-norm method (WLN) [14] is used for joint limit avoidance. The combined method of RDLS and WLN [146] is also proposed. Many other methods are proposed, such as the selectively damped least squares methods (SDLS) [1], the gradient projection method (GPM) [76] and the extended Jacobian method EJM [63][131]. In this dissertation, to improve the performance of the IK solver, RWLN method [146] is used to achieve singularity avoidance and joint limit avoidance. On the other hand, except the kinematics part, the dynamics part of the robot must be considered. Newton-Euler dynamics [4][27][132] is used to construct the dynamics part of the pattern generation algorithm, including the prediction of torques and forces of the robot joints while motion planning and the data analysis after experiments.

The rest of this chapter is organized as follows: section 2.2 describes the DH method and forward kinematics. Section 2.3 shows the IK engine used in this dissertation. Section 2.4 discusses about the Newton-Euler dynamics. Section 2.5 shows the calculation of linear and angular momentum. Section 2.6 summarizes this chapter.

2.2 Forward Kinematics

The Denavit-Hartenberg (DH) method is used to construct the forward kinematics (FK) of the robot system [133]. The homogeneous matrix describing the translation and rotation of the robot is expressed in Eq. (2-1).

$$T_{0,i} = T_{0,1}T_{1,2}T_{2,3}\cdots T_{i-1,i} \tag{2-1}$$

$T_{0,i}$ denotes the homogeneous matrix describing the ith joint in the world coordinate and $T_{i-1,i}$ denotes the homogeneous matrix from the (i-1)th to the ith joint. The homogeneous matrix is composed of two main parts, the rotational part and the translational part, shown as Eq. (2-2).

$$T_{0,i} = \begin{bmatrix} R_{0,i} & D_{0,i} \\ 0 & 1 \end{bmatrix} \tag{2-2}$$

$R_{0,i}$ denotes the rotation matrix of the ith link described in the world coordinate and $D_{0,i}$ denotes the position of the origin of the ith link in the world coordinates. The physical meaning of $R_{0,i}$ is very important for describing and deriving the equations of robot kinematics and dynamics. It is composed of the unit vectors of local x, y, and z axes described in the world coordinates, as shown in Eq. (2-3) and Figure 2-1.

$$R_{0,i} = [x_i \quad y_i \quad z_i] \tag{2-3}$$

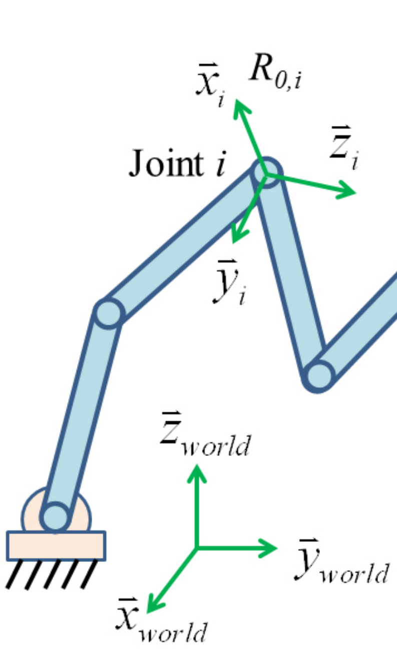


Figure 2-1. The physical meaning of rotation matrix in the world coordinates

In the equation and figure above, x_i, y_i, and z_i denote the unit vectors of local x, y, and z axes described in the world coordinates; they are all 3-by-1 vectors.

By calculating forward kinematics using DH parameters, the relationship between the end-effectors and the joint angles can be expressed as Eq. (2-4).

$$x = f(\theta) \tag{2-4}$$

In the equation, x denotes the combined vector including the position and direction of the end-effectors of the head and the limbs. The COG position is also a part of the vector x, it is found by averaging all the position multiplied by the weight of each link. On the right hand side, θ denotes all joint angles of the robot, as shown in Eq. (2-5).

$$\theta = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_j \quad \cdots \quad \theta_n]^T \tag{2-5}$$

2.3 Inverse Kinematics

Using the Jacobian linearization method, the joint speed can be mapped to the speed of x with Jacobian matrix, as shown in Eq. (2-6).

$$\dot{x} = J\dot{\theta} \tag{2-6}$$

where J denotes the Jacobian matrix. It is not a square matrix when the system has redundant degrees of freedom (DOFs). Pseudoinverse is used to solve the joint speed for the desired $\dot{x}$, as shown in Eq. (2-7).

$$\dot{\theta} = J^{+}\dot{x} \tag{2-7}$$

When solving IK, singularity configurations may cause discontinuous solved trajectory or the failure of the IK solver. This is solved with the robust damped least squares method (RDLS) in section 2.3.1. And the other problem when solving IK is the joint limit problem. When a joint of mechanism is reaching its joint limit, the mechanism may crash and broken. The weighted least-norm (WLN) method is used to solve the joint limit problem of the robot, shown in section 2.3.2.

2.3.1 Robust Damped Least Squares Method (RDLS)

If the determinant of JJ^{T} is zero or close to zero, singularity occurs. In order to avoid the singularity, robust damped least square method (RDLS) is applied. The idea of the damped least square method (DLS) is to minimize $||\dot{x} - J\dot{\theta}||^{2} + \alpha||\dot{\theta}||^{2}$, the sum of the square of the residual error and the joint velocities. Here α is a positive damping factor. Thus, the pseudoinverse matrix with DLS method is shown as Eq. (2-8).

$$J_{\alpha}^{+} = J^{T}(JJ^{T} + \alpha I_{m})^{-1} \tag{2-8}$$

where I_{m} is an identity matrix with the same dimension as JJ^{T} matrix. The damping factor α helps to avoid singularity, but it also affects the solved $\dot{\theta}$. Thus, α should not be applied at nonsingular configurations. In RDLS method, the manipulability h of the mechanism [142] is defined as Eq. (2-9).

$$h(\theta) = \sqrt{det(JJ^{T})} \tag{2-9}$$

When h approaches to zero, it is getting closer to singularity. Then α is adjusted automatically using Eq. (2-10).

$$\alpha = \begin{cases} \alpha_0(1 - h/h^s), & h < h^s \\ 0 \quad , & \text{otherwise} \end{cases} \tag{2-10}$$

where h^s denotes the threshold value, α_0 is the value of damping factor at singular configurations. With the equation above, α is effective only when the configuration is near singular configurations.

2.3.2 Weighted Least-Norm Method (WLN)

The weighted least-norm method is designed from the idea of null space. The general solution of $\dot{\theta}$ for solving IK can be written as

$$\dot{\theta} = J^+\dot{x} + (I - J^+J)\varphi \tag{2-11}$$

where φ is an arbitrary vector, $J^+\dot{x}$ is a particular solution, and $(I - J^+J)\varphi$ is the homogeneous solution. Joint limit avoidance is important for humanoid robots in order to act like human beings. A weighted least-norm (WLN) solution based scheme for avoiding joint limits is proposed by Chan & Dubey [14]. In this method, a performance criterion $H(\theta)$ is defined as

$$H(\theta) = \sum_{i=1}^{n} \frac{1}{4} \frac{\left(\theta_{i,max} - \theta_{i,min}\right)^2}{\left(\theta_{i,max} - \theta_i\right)\left(\theta_i - \theta_{i,min}\right)} \tag{2-12}$$

When any joint approaches its limit, the value of $H(\theta)$ grows very fast, and so is its partial differentiation $\partial H(\theta)/\partial\theta_i$. Thus, a weighting matrix is defined as Eqns. (2-13) and (2-14).

$$W = \begin{bmatrix} w_1 & 0 & \cdots & 0 & 0 \\ 0 & w_2 & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & w_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & w_n \end{bmatrix} \tag{2-13}$$

$$w_i = 1 + \left| \frac{\partial H(\theta)}{\partial \theta_i} \right| \tag{2-14}$$

The WLN method can be expressed as Eqns. (2-15) and (2-16).

$$J_W = JW^{-1/2} \tag{2-15}$$

$$\dot{\theta} = W^{-1/2}J_W^+\dot{x} + \left(W^{-1/2}J_W^+ - J^+\right)\dot{x} \tag{2-16}$$

2.3.3 Robust Weighted Least Norm Method (RWLN)

The RWLN method is the combination of the RDLS method and the WLN method. The equation used to solve IK with the RWLN method is shown as Eqns. (2-17) and (2-18).

$$\dot{\theta} = W^{-1/2} J_{W,\alpha}^{+} \dot{x} + \left(W^{-1/2} J_{W,\alpha}^{+} - J_{\alpha}^{+} \right) \dot{x} \qquad (2\text{-}17)$$

$$J_{W,\alpha}^{+} = J_W^T \left(J_W J_W^T + \alpha_W I_m \right)^{-1} \qquad (2\text{-}18)$$

The procedure of solving IK is shown in Figure 2-2.

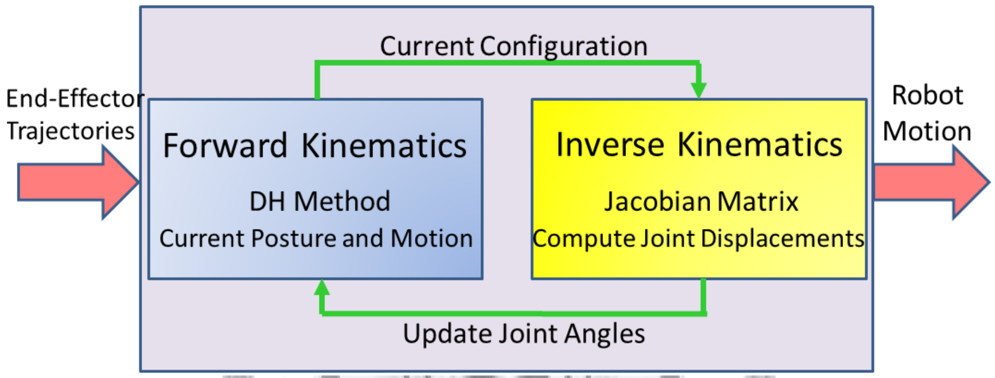


Figure 2-2. The procedure of solving inverse kinematics

In the figure, the end-effector trajectories are input to the IK solver. Firstly, the robot posture and all positions of end-effectors are calculated with FK, and then the IK solver updates the joint angles with pseudoinverse calculation. These two steps are repeated till the robot end-effectors reach the target positions and orientations. Using this procedure, the robot motions are solved one-by-one with the input trajectory.

2.4 Newton-Euler Dynamics

The joint angle, velocity and acceleration can be found using the method introduced in sections 2.2 and 2.3. In this section, Newton-Euler Dynamics is used to find the dynamics of the robot, including the velocity, acceleration, angular velocity, angular acceleration, force, and torque of all joints of the robot described in the world coordinate.

2.4.1 Forward Iteration

In the forward calculation stage of the Newton-Euler method, the velocity, angular velocity, acceleration and angular acceleration of all joints and all COGs of each link of the robot are calculated, as shown in Eqns. (2-19)–(2-22) and Figure 2-3.

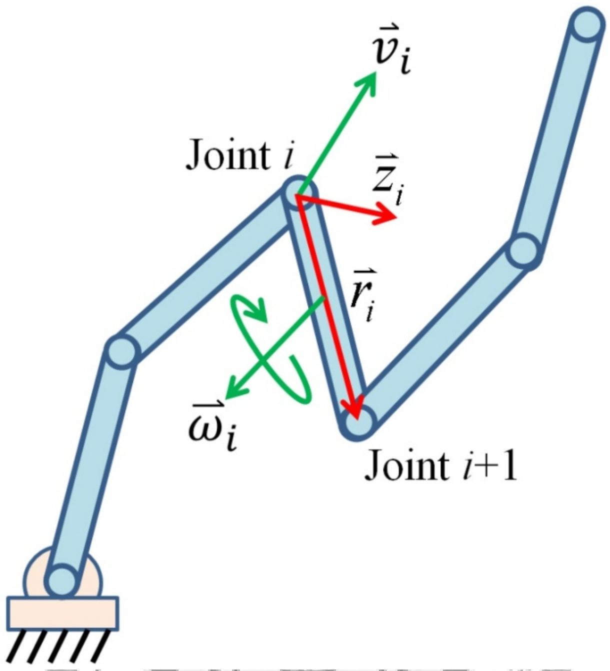


Figure 2-3. The vectors used to find the dynamics of the robot

$$\vec{\omega}_i = \vec{\omega}_{i-1} + \vec{z}_i\dot{\theta}_i \tag{2-19}$$

$$\vec{v}_{i+1} = \vec{v}_i + \vec{\omega}_i \times \vec{r}_i \tag{2-20}$$

$$\vec{\alpha}_i = \vec{\alpha}_{i-1} + \vec{z}_i\ddot{\theta}_i + \vec{\omega}_i \times \vec{z}_i\dot{\theta}_i \tag{2-21}$$

$$\vec{a}_{i+1} = \vec{a}_i + \vec{\alpha}_i \times \vec{r}_i + \vec{\omega}_i \times (\vec{\omega}_i \times \vec{r}_i) \tag{2-22}$$

In the equations, $\vec{\omega}$, $\vec{v}$, $\vec{\alpha}$, and $\vec{a}$ denote the angular velocity, the velocity, the angular acceleration, and the acceleration of each joint of the robot in the world coordinate. $\dot{\theta}$ and $\ddot{\theta}$ denote the rotational speed and acceleration of each joint of the robot. $\vec{r}_i$ is the vector from the joint *i* to the joint *i*+1. $\vec{z}_i$ is the unit vectors of the z-axis of the *i*th joint, as shown in Figure 2-4.

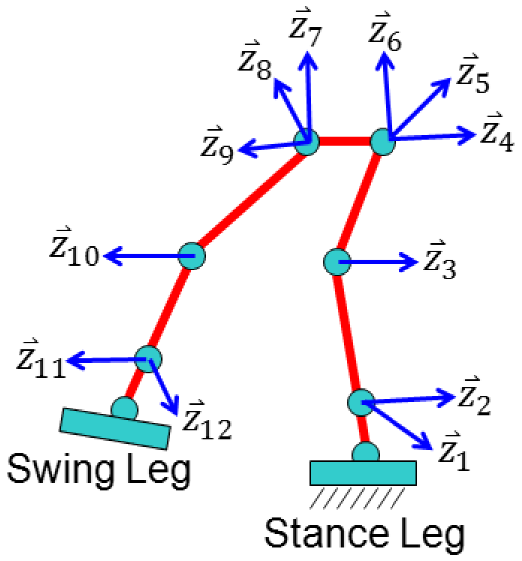


Figure 2-4. z-axis unit vectors of all joints of both robot legs

The angular velocity, velocity, angular acceleration, and acceleration of each link COG are derived as Eqns. (2-23) to (2-26) using the vectors in Figure 2-5.

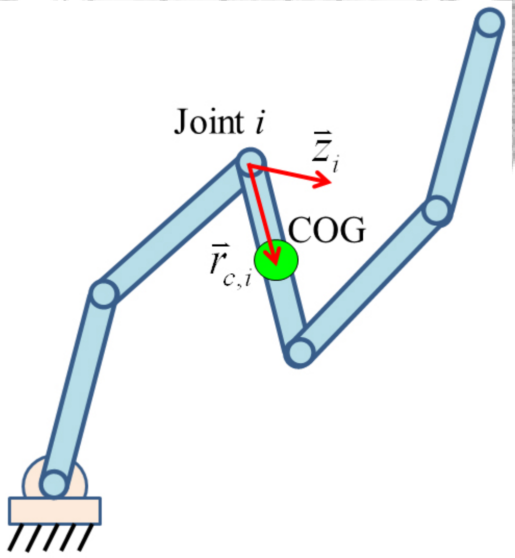


Figure 2-5. The link COG and vectors used to find its dynamics

$$\vec{\omega}_{c,i} = \vec{\omega}_i \tag{2-23}$$

$$\vec{v}_{c,i} = \vec{v}_i + \vec{\omega}_i \times \vec{r}_{c,i} \tag{2-24}$$

$$\vec{\alpha}_{c,i} = \vec{\alpha}_i \tag{2-25}$$

$$\vec{a}_{c,i} = \vec{a}_i + \vec{\alpha}_i \times \vec{r}_{c,i} + \vec{\omega}_i \times \left(\vec{\omega}_i \times \vec{r}_{c,i}\right) \tag{2-26}$$

where $\vec{r}_{c,i}$ is the vector from the ith joint to the COG of the ith link. In Eqns. (2-23) and (2-25), the ith angular velocity and angular acceleration of the link COG is the same as the ith joint angular velocity and angular acceleration. In Eqns. (2-24) and (2-26), the calculation of the velocity and the acceleration of the ith link COG is the same as Eqns. (2-20) and (2-22); in the equations, $\vec{r}_i$ is replaced with $\vec{r}_{c,i}$.

2.4.2 Backward Iteration

In the forward calculation phase, the first joint is always the ankle of the stance leg. The first joint switches between the left and right ankles during the walking process. In the backward calculation stage of the Newton-Euler method, the force and torque are calculated, as shown in Eqns. (2-27)–(2-28) and Figure 2-6.

$$\vec{f}_{i+1} = \vec{f}_i - m_i\vec{g} + m_i\vec{a}_{c,i} \tag{2-27}$$

$$\vec{\tau}_{i+1} = \vec{\tau}_i + I_i\vec{\alpha}_i + \vec{\omega}_i \times (I_i\vec{\omega}_i) - \vec{r}_{i\to i} \times \vec{f}_i + \vec{r}_{i\to i+1} \times \vec{f}_{i+1} \tag{2-28}$$

f and τ denote the force and the torque of each joint of the robot. m_i and I_i denote the mass and inertia matrix of the ith link of the robot, calculated with CAD software. $\vec{r}_{i\to i}$ and $\vec{r}_{i\to i+1}$ denote the vectors from the COG of the link to its two end-points.

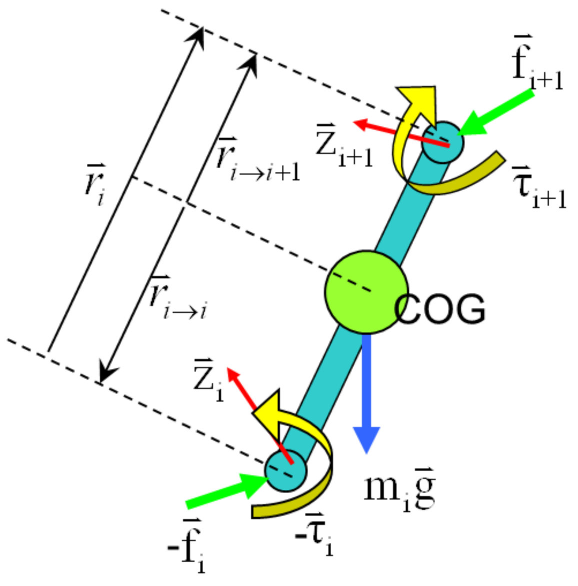


Figure 2-6. The free body diagram of the ith link

The derivative of Newton-Euler dynamics with respect to the COG height will be derived and discussed in chapter 5. It is used to optimize the input COG height trajectory of the robot when walking.

2.5 Linear Momentum and Angular Momentum

The calculation of linear momentum and angular momentum of the robot is described in this section. Calculation of the momentum of rigid bodies can be found in textbooks or papers [50][84].Considering the robot momentum can further improve the balance of the robot walking. The momentum Jacobian will be discussed in section 3.5.

2.5.1 Linear Momentum

Since the velocity of the COG of the ith link is shown in Eq. (2-24), the linear momentum of each link in the world coordinate is

$$\vec{L}_{M,i} = m_i \vec{v}_{c,i} \tag{2-29}$$

The total linear momentum of the robot is the summation of linear momentum of all links of the robot as shown in Eq. (2-30).

$$\vec{L}_M = \sum m_i \vec{v}_{c,i} \tag{2-30}$$

2.5.2 Angular Momentum

The angular momentum of the robot contains two parts: spin and orbit angular momentum, as shown in Eq. (2-31).

$$\vec{A}_M = \vec{A}_{M,spin} + \vec{A}_{M,orbit} \tag{2-31}$$

Since the inertias of all parts of the robot changes in world coordinates with the robot movements, the inertias in local coordinates should be used as references. The inertias with respect to the COG of each link in local coordinates are constant matrices and they can be calculated by using CAD software such as CATIA and SolidWorks. The local inertia matrix with respect to the COG of the ith link of the robot is defined as

$I_{0,i}$. The relationship between of the inertia matrices in local and in world coordinates is shown as Eq. (2-32) and Figure 2-7.

$$I_i = R_{0,i} I_{0,i} R_{0,i}^T \tag{2-32}$$

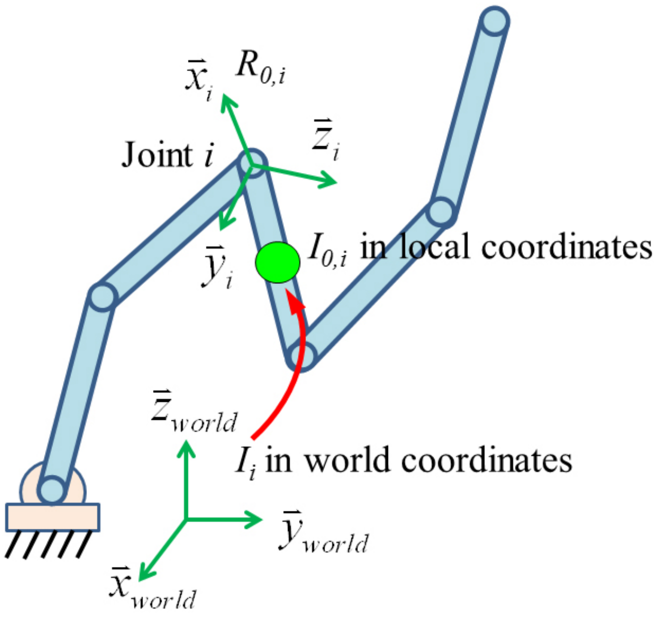


Figure 2-7. The local inertia matrix

I_i denotes the inertia matrix with respect to the COG of the *i*th link in the world coordinates, $R_{0,i}$ is the rotation matrix part of the homogeneous matrix $T_{0,i}$ shown in Eq. (2-2). The spin angular momentum can be calculated by summing all the product of the momentum of inertia and the corresponding angular velocity in the world coordinate, as Eq. (2-33).

$$\vec{A}_{M,spin} = \sum I_i \vec{\omega}_i \tag{2-33}$$

The orbit angular momentum with respect to the support point is calculated as Eq. (2-34) and shown in Figure 3-13.

$$\vec{A}_{M,orbit} = \sum \left(\vec{r}_{s \to c,i} \times m_i \vec{v}_{c,i} \right) \tag{2-34}$$

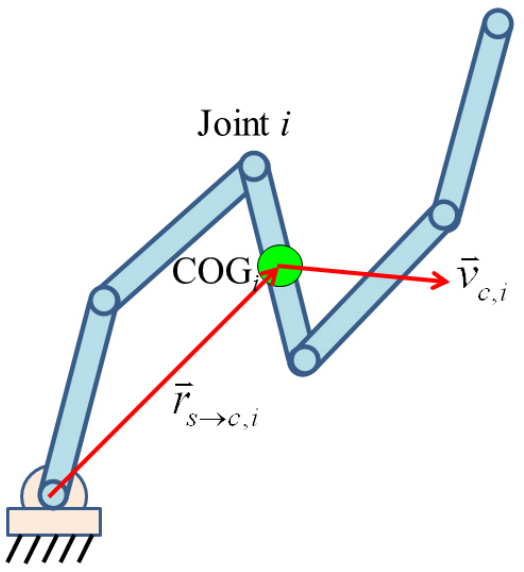


Figure 2-8. The orbit angular momentum

where $\vec{r}_{s\to c,i}$ denotes the vector from support point to the COG of the ith link. The support point of the robot is in the robot's stance foot, thus the support point of the momentum Jacobian in section 3.5.3 is chosen as the position of the ankle of the stance leg.

2.6 Summary

In this chapter, the basic kinematics and dynamics engine in the whole dissertation are introduced. The methods of forward and inverse kinematics are discussed in the first half of this chapter and then the Newton-Euler dynamics and the calculation of momentum are shown in the second half.

In the following chapters, the detailed part and constraints for solving IK, and more detailed momentum calculation are shown in chapter 3. Newton-Euler dynamics is used to optimize the COG height trajectory in chapter 5.

Chapter 3 Jacobian Based Inverse Kinematics Solver

In this dissertation, IK is solved using pseudoinverse method. The Jacobian matrix can describe the linearized relationship between the joint velocity and the end-effectors of the robot, including the conventional Jacobian matrix, the COG Jacobian matrix, momentum Jacobian matrix, and the Fixed-Leg-Motion Jacobian matrix. All the discussions and calculations in this chapter use the concept of the physical meaning of the robot Jacobian matrix: Fixing all the other joints and calculate the component caused by the movement of the *i*th joint. The concept of Fixed-Leg_Motion Jacobian is also proposed in this chapter and it is a quick and easy concept to deal with the constraint that the stance foot is fixed on the ground and has zero linear and angular velocity.

3.1 Introduction

In robot applications, IK is a common method to solve joint trajectories to follow the assigned end-effector trajectories. There are many methods to solve IK, including analytical displacement analysis method, Jacobian linearization method, searching method, etc. The analytical displacement analysis method [29][34][73] is to find the relationship between the joints and the links using sine and cosine functions directly; this is convenient for simple mechanisms to solve IK but very difficult and highly nonlinear for multi-link or redundant mechanisms. Jacobian linearization method is most used in applications of robot arms and robot legs; by linearizing the kinematics relationship of the mechanism, this method can solve IK after several iterations. There are two types of Jacobian linearization methods. One is to use the analytical solution [15][103] of the mechanism to find the Jacobian matrix and the other is to use the cross-product method [9][90][109][123][130]. Same as the analytical displacement analysis method, the Jacobian matrix found by using analytical method is also highly nonlinear and very complex for humanoid robots. Compared with the Jacobian matrix calculated using the analytical method, to calculate it using cross product method does not need to deal with the highly nonlinear equations and its calculation is also very fast. Searching methods include random search method [46][135] and motion capture and database method [102]. Random search methods can be very fast and it gives different results in each test. On the other hand, database methods need large memory for complex mechanisms and the motion capture method must have a good mapping algorithm or the robot will become unstable since the shape and mass distribution of the robot is different from human. In this dissertation, the Jacobian matrix calculated by using cross product method is used since it is fast and convenient.

This chapter is organized as follows: section 3.2 shows the conventional Jacobian matrix which describes the linearized relationship between the motion of the end-effector and joint angles, section 3.3 derives and describes the proposed Fixed-Leg-Motion Jacobian concept which can be used to all types of Jacobian matrices for legged robots, section 3.4 shows the calculation of COG and the COG Jacobian matrix, section 3.5 further describes the calculation of momentum and derives the momentum Jacobian matrix, section 3.6 combines all Jacobian matrices discussed in this chapter to a global Jacobian matrix, and finally section 3.8 summarizes this chapter.

3.2 Conventional Jacobian Matrix

After constructing the DH parameters, the Jacobian matrix can be found by the cross product method, and then the limbs and the head can be controlled independently with the Jacobian matrix. The ankles, the fingertips and the head are chosen as end-effectors. But if we solve IK for the limbs and the head of the robot independently, it is difficult to decide where the positions of the end-effectors should be in local coordinates because DH forward kinematics method constructs the joint positions and orientations in its own coordinates instead of the world coordinates, and all positions of the end-effectors in the world coordinates are influenced by the movements of the stance leg. Thus the trajectories of the end-effectors should be decided in the world coordinates directly. Eq. (3-1) describes the conventional Jacobian matrix that is used while solving IK independently.

$$\begin{bmatrix} \dot{d}_{LL} \\ \dot{d}_{RL} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \end{bmatrix} = \begin{bmatrix} J_{LL} & 0 & 0 & 0 & 0 \\ 0 & J_{RL} & 0 & 0 & 0 \\ 0 & 0 & J_{LA} & 0 & 0 \\ 0 & 0 & 0 & J_{RA} & 0 \\ 0 & 0 & 0 & 0 & J_{H} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{LL} \\ \dot{\theta}_{RL} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \qquad (3\text{-}1)$$

where $\dot{d}$ denotes the vector composed of the differences of position and orientation from the current end-effector to the desired end-effector in its own coordinates; *LL*, *RL*, *LA*, *RA*, and *H* denotes left leg, right leg, left arm, right arm, and head, respectively. If we just want to control the limbs and the head of the robot independently, it is enough to solve IK with the equation above in one iteration if $\dot{d}$ is given appropriately (not too large).

The linearized relationship of the end-effector and all joints that decides its position and orientations in the world coordinates can be described as the format in the following. Figure 3-1 shows the vectors used to construct the conventional Jacobian matrix.

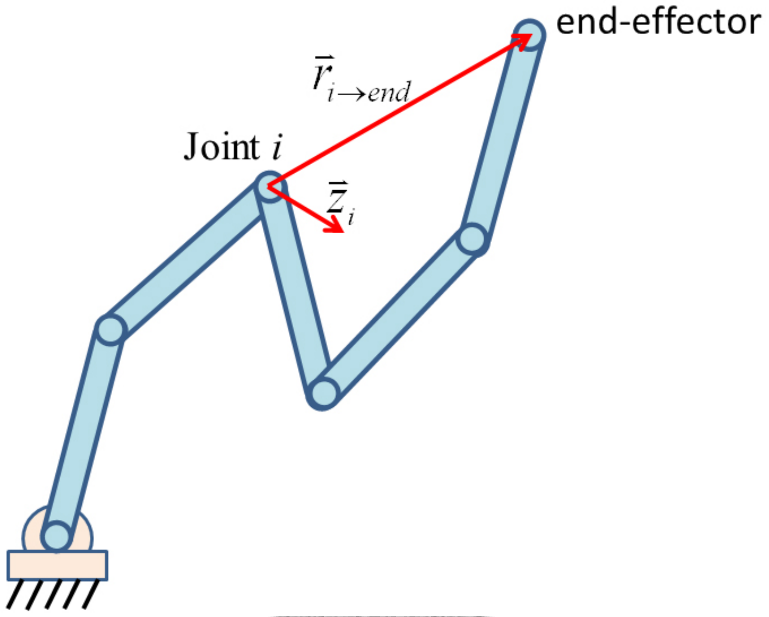


Figure 3-1. The vectors used to construct Jacobian matrix

In Figure 3-1, $\vec{r}_{i\to end}$ and $\vec{z}_i$ vectors denote the vector from the position of the *i*th joint to the end-effector and the unit vector of the rotation axis of the *i*th joint. The construction of the conventional Jacobian matrix of the left leg is used as an example. From Eq. (3-1), the relationship between the end-effector and the joints can be written as

$$\dot{d}_{LL} = J_{LL}\dot{\theta}_{LL} = \begin{bmatrix} v_{end,LL} \\ \omega_{end,LL} \end{bmatrix} \tag{3-2}$$

$\dot{d}_{LL}$ is the 6-by-1 vector composed of the linear velocity and angular velocity of the end-effector described in the world coordinates. $v_{end,LL}$ and $\omega_{end,LL}$ denote the linear velocity and angular velocity; both vectors are 3-by-1 vector. For an n-axis leg mechanism, Eq. (3-2) can be further rewritten as Eqns. (3-3) and (3-4).

$$\mathrm{v}_{\mathrm{end,LL}} = \sum_{\mathrm{i}=1}^{\mathrm{n}} (\vec{\mathrm{z}}_{\mathrm{i}} \times \vec{\mathrm{r}}_{\mathrm{i}\to\mathrm{end}})\dot{\theta}_{\mathrm{i}} \tag{3-3}$$

$$\omega_{end,LL} = \sum_{i=1}^{n} \vec{z}_i \dot{\theta}_i \tag{3-4}$$

$\vec{z}_i \times \vec{r}_{i\to end}$ is the position change under unit rotation of the ith joint and thus the $v_{end,LL}$ is the summation of the position change due to the motion of each joint. On the other hand, because the angular velocity change under unit rotation of the ith joint is $\vec{z}_i$, the $\omega_{end,LL}$ is the summation of the angular velocities due to the motion of all joints. The conventional Jacobian matrix in Eq. (3-2) can be found by rearrange the Eqns. (3-3) and (3-4) in matrix form, as shown in Eq. (3-5).

$$J_{LL}\dot{\theta}_{LL} = \begin{bmatrix} \vec{z}_1 \times \vec{r}_{1\to end} & \vec{z}_2 \times \vec{r}_{2\to end} & \cdots & \vec{z}_i \times \vec{r}_{i\to end} & \cdots & \vec{z}_n \times \vec{r}_{n\to end} \\ \vec{z}_1 & \vec{z}_2 & \cdots & \vec{z}_i & \cdots & \vec{z}_n \end{bmatrix} \begin{bmatrix} \dot{\theta}_{1,LL} \\ \dot{\theta}_{2,LL} \\ \vdots \\ \dot{\theta}_{i,LL} \\ \vdots \\ \dot{\theta}_{n,LL} \end{bmatrix} \qquad (3\text{-}5)$$

where J_{LL} is a 6-by-n matrix. Rather than using the partial derivative method in common text books to find the Jacobian matrix, it is easier and neater to use the cross-product method in Eq. (3-5).

3.3 Fixed-Leg-Motion Jacobian Matrix

Since the stance leg is "fixed" on the floor, the movements of the joints of the stance leg change the positions and orientations of other end-effectors. The concept of the "Fixed-Leg-Motion" Jacobain matrix describing the linearized relationship among the stance leg and the end-effectors is proposed in this dissertation. With the proposed Fixed-Leg-Motion Jacobian method, rather than change the origin point of all the forward kinematics trains to the stance foot or using the complex transformation formula, the original kinematics trains can be used very conveniently to construct the Jacobian matrix of the robot, as shown in Figure 3-2.

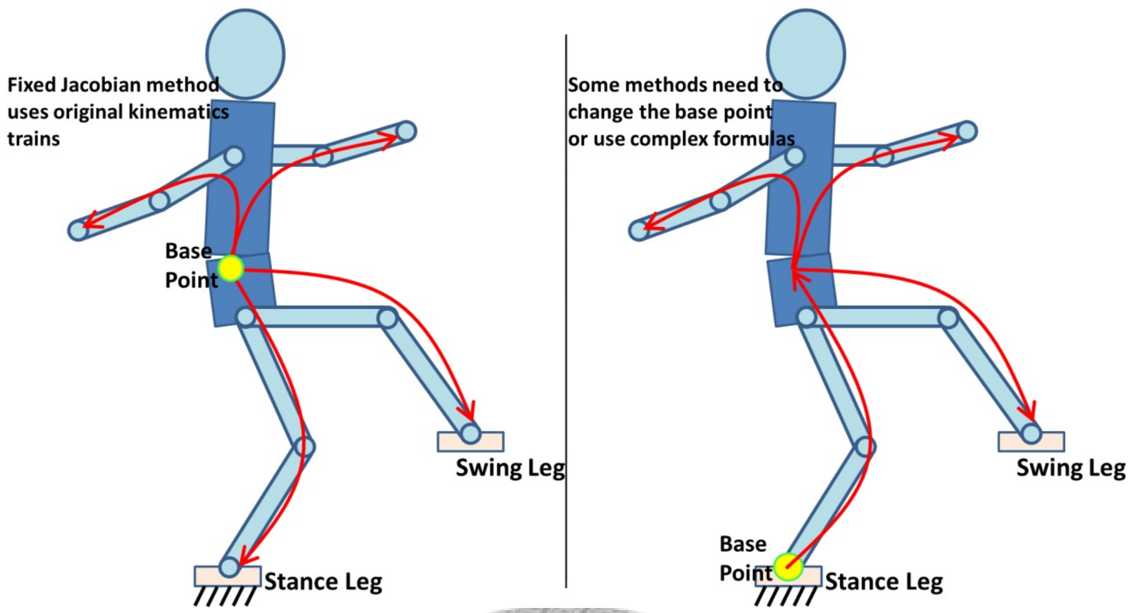

Fixed Jacobian method uses original kinematics trains
Base Point
Swing Leg
Stance Leg
Some methods need to change the base point or use complex formulas
Swing Leg
Base Point
Stance Leg

Figure 3-2. To change and not to change the origin of kinematics trains

In the figure, “base point” denotes the origin point of the kinematics trains of the robot. The concept of the Fixed-Leg-Motion Jacobian matrix is used to find how the joints in stance leg affect the positions and orientations of the end-effectors directly using the original kinematics trains. It can also be used on COG Jacobian, momentum Jacobian and other linearized relationships, as shown in the following sections.

Same as the conventional Jacobian matrix, the Fixed-Leg-Motion Jacobian matrix is also a 6-by-n matrix. It is defined as

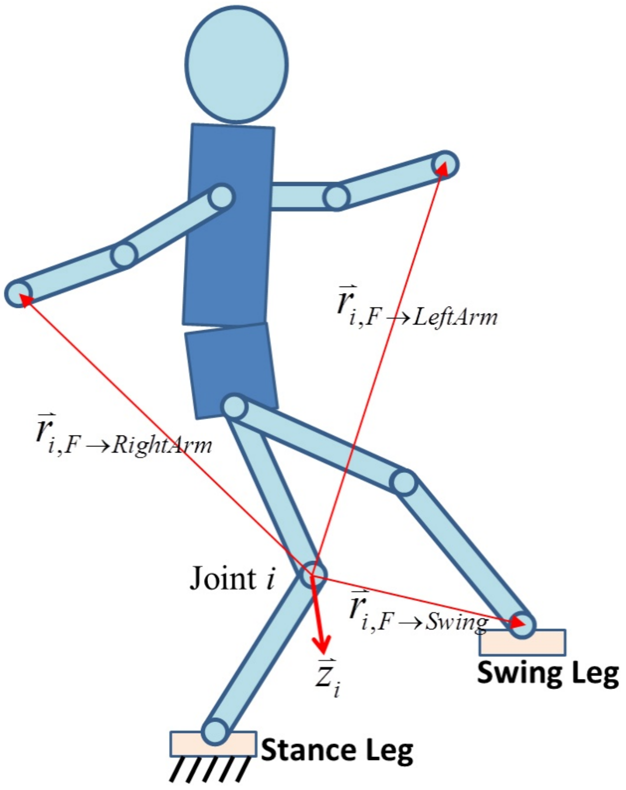
$$J_F = \begin{bmatrix} J_{F,Translational} \\ J_{F,Rotational} \end{bmatrix}, \qquad J_F \in R^{6\times n} \tag{3-6}$$

$J_{F,Translational}$ and $J_{F,Rotational}$ denote the translational and rotational parts of the Fixed-Leg-Motion Jacobian matrix. The $J_{F,Rotational}$ is defined as

$$J_{F,Rotational} = -[\vec{z}_1 \quad \vec{z}_2 \quad \cdots \quad \vec{z}_i \quad \cdots \quad \vec{z}_n] \tag{3-7}$$

where n denotes the total number of joints of the stance leg, $\vec{z}_i$ are unit normal vectors of the joints of the stance leg. The minus sign is multiplied since when a joint of the stance leg rotates clockwise in its coordinates, the body rotates counterclockwise in the world coordinates. On the other hand, the $J_{F,Translational}$ is calculated as Eq. (3-8) using the vectors in Figure 3-3.

$$J_{F,Translational}^{T} = -\begin{bmatrix} \vec{z}_1 \times \vec{r}_{1,F\rightarrow end\_eff} \\ \vec{z}_2 \times \vec{r}_{2,F\rightarrow end\_eff} \\ \cdots \\ \vec{z}_i \times \vec{r}_{i,F\rightarrow end\_eff} \\ \cdots \\ \vec{z}_n \times \vec{r}_{n,F\rightarrow end\_eff} \end{bmatrix} \tag{3-8}$$



$\vec{r}_{i,F\rightarrow LeftArm}$
$\vec{r}_{i,F\rightarrow RightArm}$
Joint i
$\vec{r}_{i,F\rightarrow Swing}$
$\vec{z}_i$
Swing Leg
Stance Leg

Figure 3-3. The vectors used to construct Fixed-Leg-Motion Jacobian matrix

In Eq. (3-8), the $\vec{r}_{i,F\rightarrow end\_eff}$ denotes the vector from the ith joint of the stance leg to each end-effector, the cross product $\vec{z}_i \times \vec{r}_{i,F\rightarrow end\_eff}$ is its vector change under unit rotation of the ith joint. The Fixed-Leg-Motion Jacobian matrix can be applied to all other end-effectors, as shown in Eq. (3-9).

$$\begin{bmatrix} \dot{d}_{stance} \\ \dot{d}_{swing} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \end{bmatrix} = \begin{bmatrix} J_{stance} & 0 & 0 & 0 & 0 \\ J_{f\to s} & J_{swing} & 0 & 0 & 0 \\ J_{f\to LA} & 0 & J_{LA} & 0 & 0 \\ J_{f\to RA} & 0 & 0 & J_{LA} & 0 \\ J_{f\to H} & 0 & 0 & 0 & J_{H} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{stance} \\ \dot{\theta}_{swing} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \tag{3-9}$$

In the equation, $J_{f\to X}$ denotes the F-Jacobian matrices; the subscript "*X*" denotes the end-effector affected by the movement of the joints of the stance leg, such as the swing leg, the fingertips, and the head.

In the world coordinates, when using the conventional Jacobian to solve IK, each kinematics train (head, arm, or leg) is considered separately. The effect from other kinematics train (i.e. the stance leg) is not considered. On the other hand, the Fixed-Leg-Motion Jacobian matrix considers the movements of the stance leg. For example, multiply the second row and the angular speed vector of Eqns. (3-1) and (3-9), we can find Eqns. (3-10) and (3-11).

$$\dot{d}_{swing} = J_{swing}\dot{\theta}_{swing} \tag{3-10}$$

$$\dot{d}_{swing} = J_{f\to s}\dot{\theta}_{stance} + J_{swing}\dot{\theta}_{swing} \tag{3-11}$$

Eq. (3-10) can solve IK using the same iterations as Eq. (3-11) if the end-effector trajectories of each kinematics train are input independently. Eq. (3-11) can solve faster when the motions of the stance leg affects the position of the other end-effectors. Eq. (3-11) has better computation efficiency in real cases, as shown in Figure 3-4.

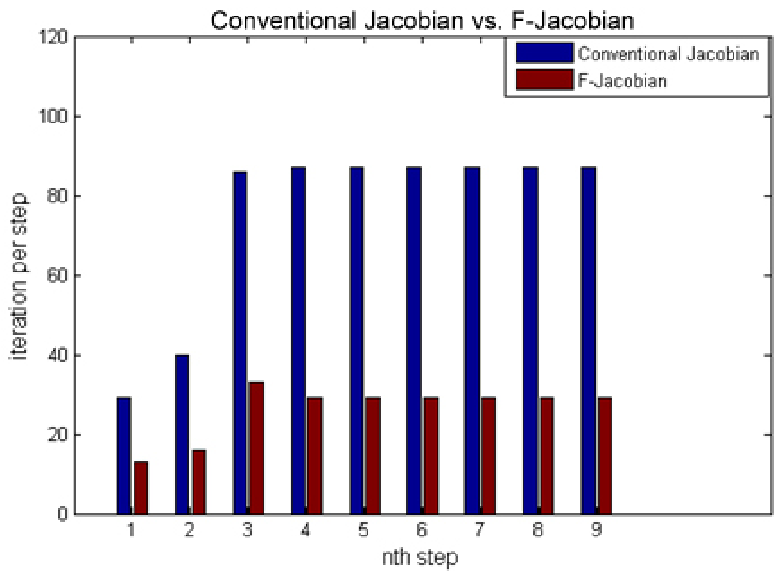


Figure 3-4. The iteration used with both methods

The graphics above show the computation time and the iterations per step of the pseudoinverse method with and without F-Jacobian method. Step 1 and step 2 are initial steps, so they are with different iterations per step. The proposed method, F-Jacobian, saved 65.73% iterations per step. Except initial steps, each step contains 27 configurations. In the 27 configurations, the first three and last three points need no iteration because they are at the same position. The proposed method, F-Jacobian, can solve each configuration in only one computation when the acceptable error is 0.2mm (0.0712% of the length of legs). "Acceptable error" means the acceptable position error value when solving IK. If the position error is smaller than the acceptable error, the next trajectory knot will be inputted to the IK solver. If the position error is still larger than the acceptable error, the same trajectory knot will be inputted to the solver again. Since the input to the IK solver are close and smooth enough, the proposed method can get smooth trajectories and solve each configuration in one iteration. But for the same input, IK without F-Jacobian makes the joints oscillate and needs about 3 iterations to solve one configuration. Figure 3-5 and Figure 3-6 show the solved trajectories of the left ankle (end-effector) with and without F-Jacobian.

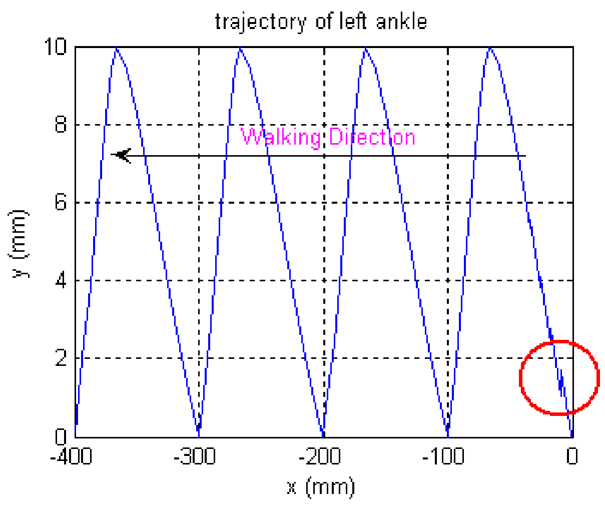


Figure 3-5. Solved trajectory with F-Jacobian

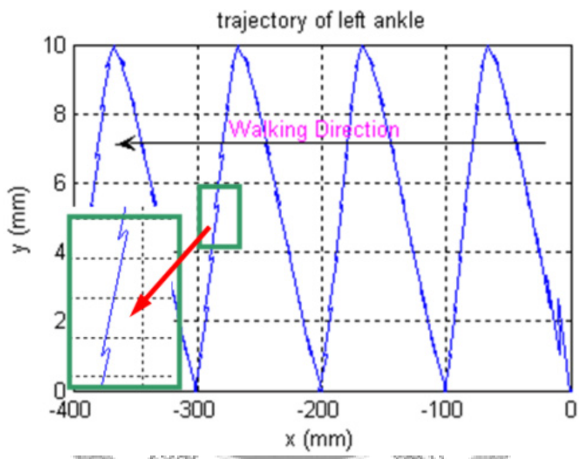


Figure 3-6. Solved trajectory without F-Jacobian

Without F-Jacobian, the solved trajectories are not all useable. Only points that are in the acceptable error range are useable. We can reduce the number of useless points with F-Jacobian method. The robot walks from 0 to -400mm in the simulation. Only some configurations in the initial steps are not solved in one computation since the home configuration of the robot is near singularity. After initial steps, the robot has bent its knees, and hence keeps the robot away from the singular configurations. All the configurations after initial steps are solved in one computation.

Because the Fixed-Leg-Motion Jacobian method can describe the relationship between end-effector and joints more accurately than conventional Jacobian method, the error decay rate when solving IK is also faster. Figure 3-7 shows the acceptable error versus the total iterations of the IK computation (for 232 trajectory knots) with and without F-Jacobian method.

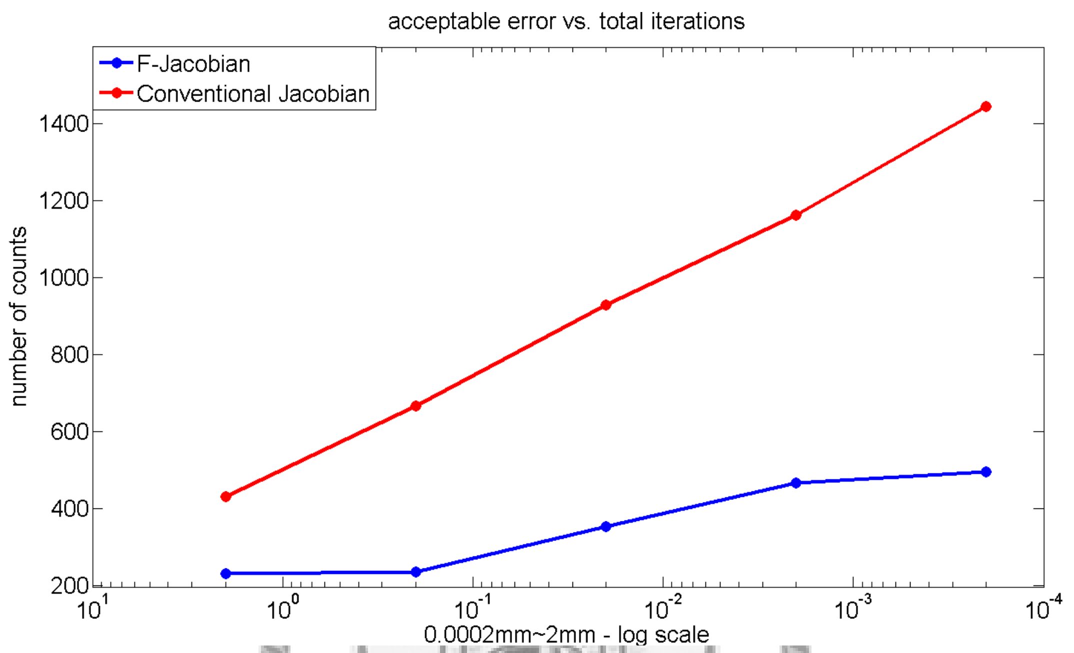


Figure 3-7. Acceptable error vs. total iterations

From Figure 3-7, we can see the number of total iterations for conventional Jacobian method grows much faster than the F-Jacobian method when we choose the acceptable error from 2mm to 0.0002mm.

3.4 COG Jacobian

The COG trajectory of walking pattern of the robot is generated with the inverted pendulum model. In the model, the mass of the robot is assumed to be a point (the COG). The position, velocity and acceleration of COG are highly related with whether the robot falls or not. Thus the trajectory of the COG is directly relative to the stability of the robot. In order to control the robot COG, the relationship between the COG and all joints of the robot must be derived. The generation and optimization of the COG trajectory will be discussed in chapters 4 and 5. In this section, the calculation of COG and the method to control the COG position using the joints of the robot is discussed.

3.4.1 Calculation of COG

The position of COG can be easily computed by averaging the sum of the product of the link masses and their position vectors, as shown in Eq. (3-12). Note that $\vec{r}_{COG}$ is a 3-by-1 vector described in world coordinates.

$$\vec{r}_{COG} = \frac{\sum_{i=1}^{n} m_i \cdot \vec{r}_{m,i}}{\sum_{i=1}^{n} m_i} \tag{3-12}$$

where $\vec{r}_{COG}$ is the position vector of the robot COG, m_i denotes the mass of the *i*th link, and $\vec{r}_{m,i}$ denotes the position vector of the COG of the *i*th link in the world coordinate.

3.4.2 COG Jacobian

With the same technique in section 3.4.1, when a joint rotates, only parts of the robot are rotated, while the others are not rotated. Separating the rotated parts and the fixed parts (without rotating), we obtain Eq. (3-13). This is used to calculate the COG Jacobian matrix.

$$M \cdot \vec{r}_{COG} = \left(\sum_{h} m_{a,h} \cdot \vec{r}_{a,h} + \sum_{k} m_{ua,k} \cdot \vec{r}_{ua,k} \right) \Bigg|_{joint=i} \tag{3-13}$$

where M denotes the total mass of the robot, subscript "*a*" denotes the parts that are affected by the rotation, subscript "*ua*" denotes the parts that are unaffected by the rotation, $m_{a,h}$ and $m_{ua,k}$ denote the mass that are affected and unaffected by the joint *i*, and the vectors $\vec{r}_{a,h}$ and $\vec{r}_{ua,k}$ denote the position of the COG of each affected part and each unaffected part. The equation can also be written as

$$M \cdot \vec{r}_{COG} = (M_a \cdot \vec{r}_a + M_{ua} \cdot \vec{r}_{ua})|_{joint=i} = M_{a,i} \cdot \vec{r}_{a,i} + M_{ua,i} \cdot \vec{r}_{ua,i} \tag{3-14}$$

where $M_{a,i}$ and $M_{ua,i}$ denote the total mass of the parts affected and unaffected by the motion of the *i*th joint , $\vec{r}_{a,i}$ and $\vec{r}_{ua,i}$ denote the position vector of the COG of the affected and unaffected parts, as shown in Figure 3-8.

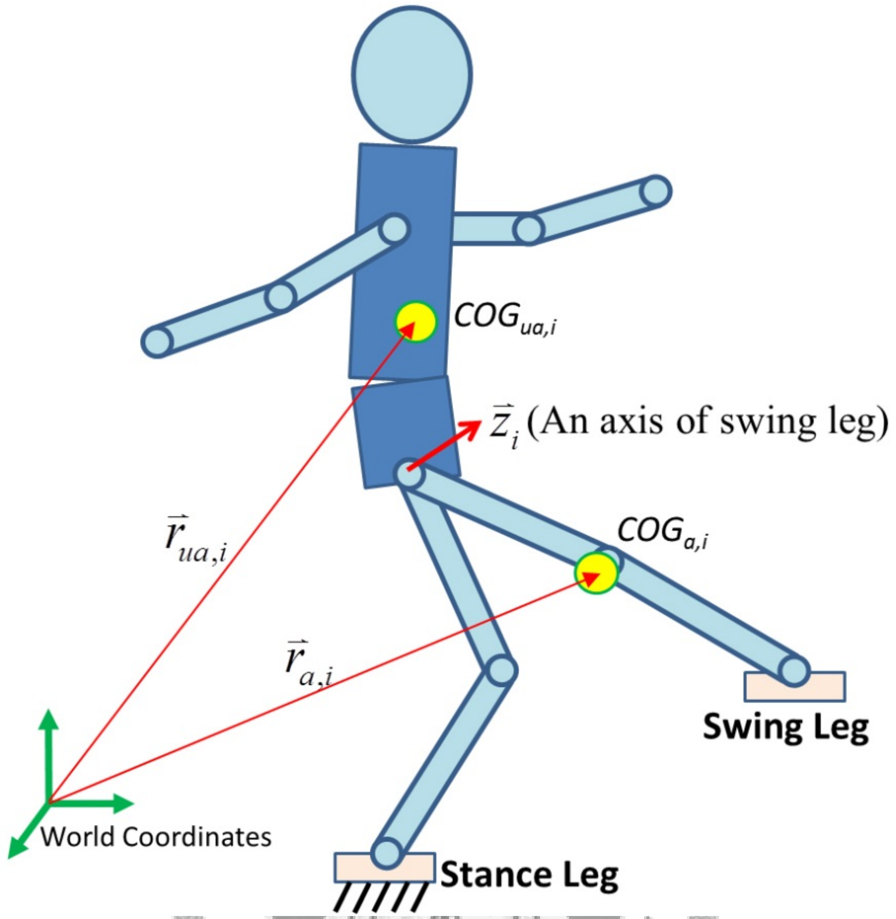


Figure 3-8. The affected and unaffected parts of the joint rotation

Note that the members of affected and unaffected parts change with different joint *i*, and they also depend upon each different mechanism. The position change of $\vec{r}_{COG}$ caused by the rotation of joint *j* can be approximated as

$$\Delta\vec{r}_{COG,i} = \frac{M_{a,i}}{M}\cdot\Delta\vec{r}_{a,i} + \frac{M_{ua,i}}{M}\cdot\Delta\vec{r}_{ua,i} \tag{3-15}$$

Since $\vec{r}_{ua,i}$ is unaffected by the rotation of the *i*th joint, $\Delta\vec{r}_{ua,i}$ is always equal to zero. $\Delta\vec{r}_{COG,i}$ denotes the displacement of the whole robot's COG caused by the rotation of the *i*th joint. Thus, the COG Jacobian can be obtained as

$$\Delta\vec{r}_{COG,i} = \frac{M_{a,i}}{M}\cdot\Delta\vec{r}_{a,i} = J_{COG,i}\cdot\dot{\theta}_i \tag{3-16}$$

where $J_{COG,i}$ and $\dot{\theta}_i$ denote the COG Jacobian and the angular speed of the *i*th joint. The COG Jacobian matrix of the joints on the limbs except the joints on the stance leg can be found as Eq. (3-17) using the vectors shown in Figure 3-9.

$$J_{COG,i} = \frac{M_a}{M}\vec{z}_i \times \vec{r}_{ac,i} \tag{3-17}$$

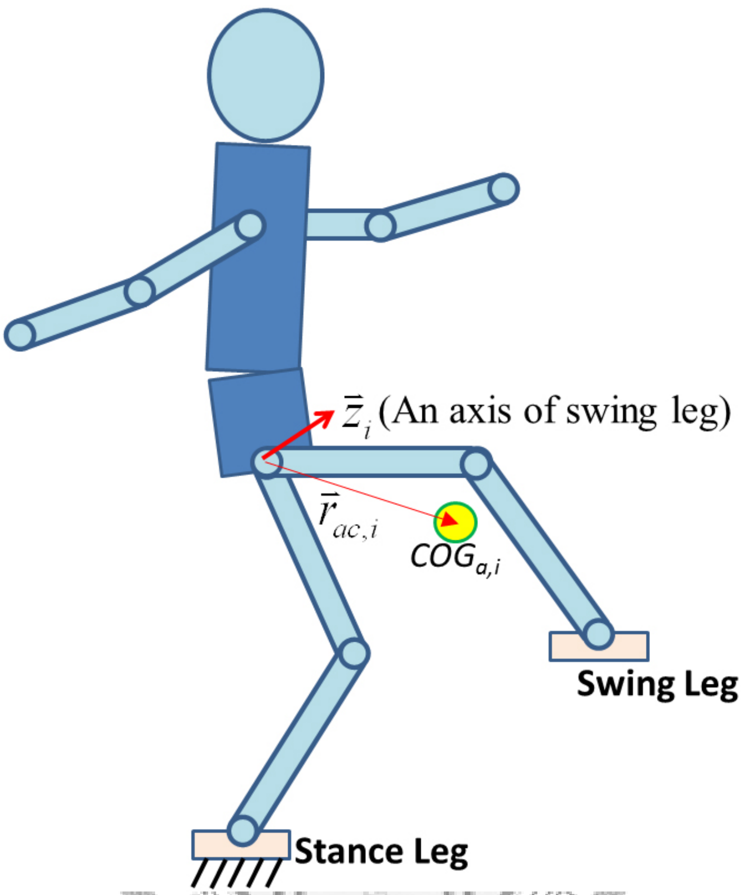


Figure 3-9. The vectors used to construct COG Jacobian

In Figure 3-9, because the ith joint is an axis of the hip joint, $COG_{a,i}$ denotes the COG of the whole swing leg; $\vec{r}_{ac,i}$ denotes the vector from the ith joint to $COG_{a,i}$. In Eq. (3-17), $\vec{z}_i \times \vec{r}_{ac,i}$ denotes the position change of $COG_{a,i}$ under unit rotation of the ith joint. Because the mass of $COG_{a,i}$ is M_a, the effect to the total COG position change is multiplied by M_a/M.

3.4.3 Fixed COG Jacobian

Using the concept of Fixed-Leg-Motion Jacobian, the COG Jacobian of the joints in stance leg is calculated as Eq. (3-18) using the vectors shown in Figure 3-10.

$$\Delta\vec{r}_{COG,i} = J_{COG,i} \cdot \dot{\theta}_i = -\frac{M_{a,i}}{M} \cdot \vec{z}_i \times \vec{r}_{ac,i}\dot{\theta}_i \tag{3-18}$$

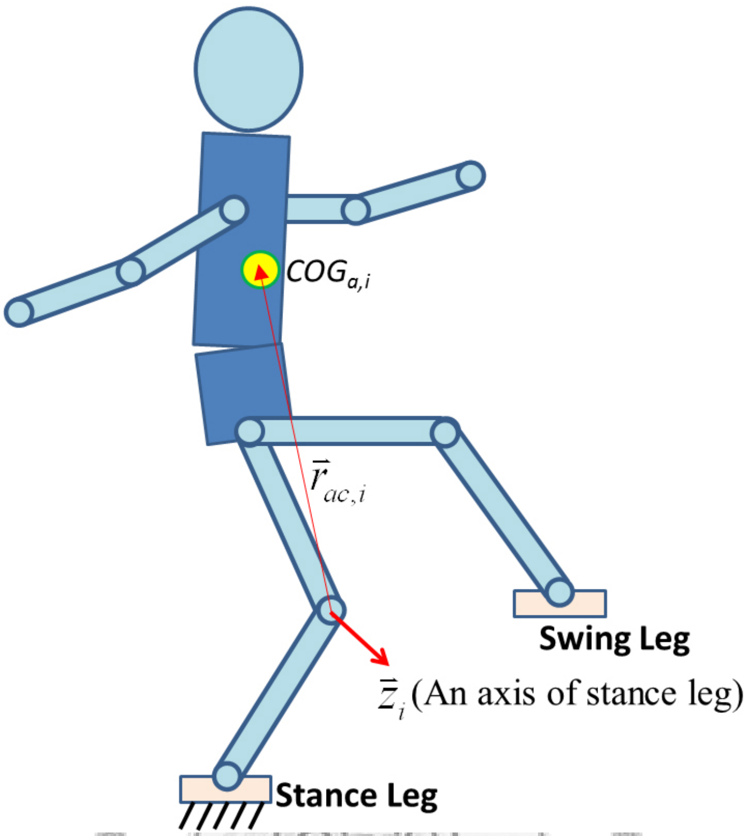


Figure 3-10. The vectors used to construct fixed COG Jacobian

Same as the Fixed-Leg-Motion Jacobian for end-effectors of the limbs, the Fixed-Leg-Motion Jacobian for COG has a minus sign in Eq. (3-18) because the rotation of the *i*th joint gives the robot body an angular velocity along $-\vec{z}_i$ direction.

With the COG Jacobian matrix, the whole Jacobian matrix including the constraint of COG position is expressed as

$$\begin{bmatrix} \dot{d}_{stance} \\ \dot{d}_{swing} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \\ \dot{d}_{COG} \end{bmatrix} = \begin{bmatrix} J_{stance} & 0 & 0 & 0 & 0 \\ J_{f\to s} & J_{swing} & 0 & 0 & 0 \\ J_{f\to LA} & 0 & J_{LA} & 0 & 0 \\ J_{f\to RA} & 0 & 0 & J_{RA} & 0 \\ J_{f\to H} & 0 & 0 & 0 & J_{H} \\ J_{f\to C} & J_{s\to C} & J_{LA\to C} & J_{RA\to C} & J_{H\to C} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{stance} \\ \dot{\theta}_{swing} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \qquad (3\text{-}19)$$

In Eq. (3-19), the COG Jacobian matrix is added to the original Jacobian matrix. In order to control the COG position of the robot, three DOFs of the robot must be used to achieve the COG position constraint (x, y, and z directions). The original Jacobian matrix described in Eq. (3-9) controls the position and direction of the stance leg and six DOFs are required (three for position and three for orientation). Because the robot does not have enough DOFs in its legs, it is impossible to control both the positions of the stance leg and COG, while keeping the orientation constraints. The constraints of the position of the stance leg is released and replaced by the COG position constraints. Thus the Jacobian matrix of the stance leg has only three rows for the three orientation constraints. The stance leg is attached on the ground, thus the positions of all joints of the robot can still be calculated by using forward kinematics.

3.5 Momentum Jacobian

In addition to tracking the COG trajectory, linear and angular momentum can also be considered to further stabilize humanoid robots while walking. Researchers proposed methods to manipulate the linear and angular momentum of the robot [50][95][121][152]. In this section, momentum Jacobian matrix using the proposed Fixed-Leg-Motion Jacobian concept is derived and described. It is also combined to the proposed global Jacobian matrix shown in section 3.6.

3.5.1 Linear Momentum Jacobian

To calculate the linear momentum Jacobian, the linear momentum caused form the rotation of each joint must be calculated first. Since the linear momentum is calculated as Eq. (2-29), the linear momentum caused form the rotation of each joint can be derived as Eq. (3-20) using the vectors shown in Figure 3-11.

$$\vec{L}_{M,i} = m_{a,i}(\vec{z}_i \times \vec{r}_{ac,i})\dot{\theta}_i \tag{3-20}$$

$$m_{a,i} = m_{a,i+1} + m_i \tag{3-21}$$

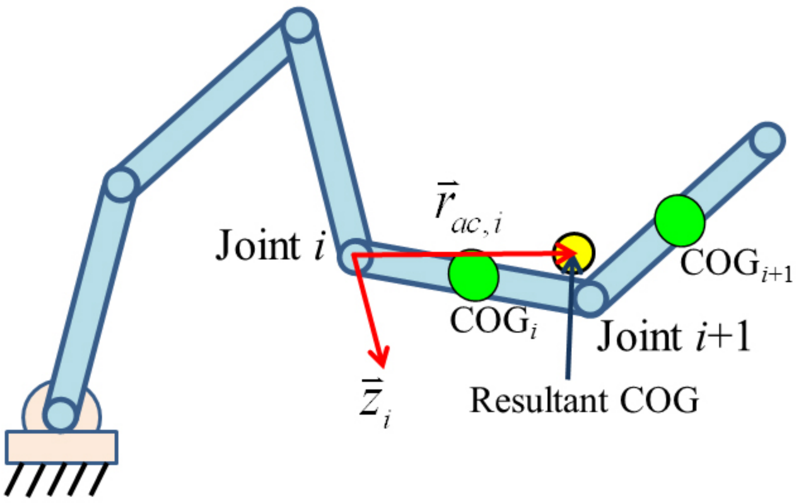


Figure 3-11. Linear momentum caused form the joint rotation

$\vec{L}_{M,i}$ denotes the linear momentum caused form the rotation of the ith joint, $m_{a,i}$ denotes the accumulated (resultant) mass from the final link back to the ith link as shown in Eq. (3-21), $\vec{r}_{ac,i}$ denotes the vector from the ith joint to the COG position of the accumulated mass. From Eq. (3-20), the linear momentum Jacobian caused from unit rotation of the ith joint is

$$J_{LM,i} = m_{a,i}(\vec{z}_i \times \vec{r}_{ac,i}) \tag{3-22}$$

where $J_{LM,i}$ is a 3-by-1 column vector; it shows the Jacobian matrix of the joints not belong to stance leg. For the joints of the stance leg, using the concept of the Fixed-Leg-Motion Jacobian, the accumulated mass contains all the mass above the joint and a minus sign must be added to the $\vec{z}_i$ vector, as shown in Eq. (3-23) and Figure 3-12.

$$J_{LM,i} = m_{a,i}(-\vec{z}_i \times \vec{r}_{ac,i}) \tag{3-23}$$

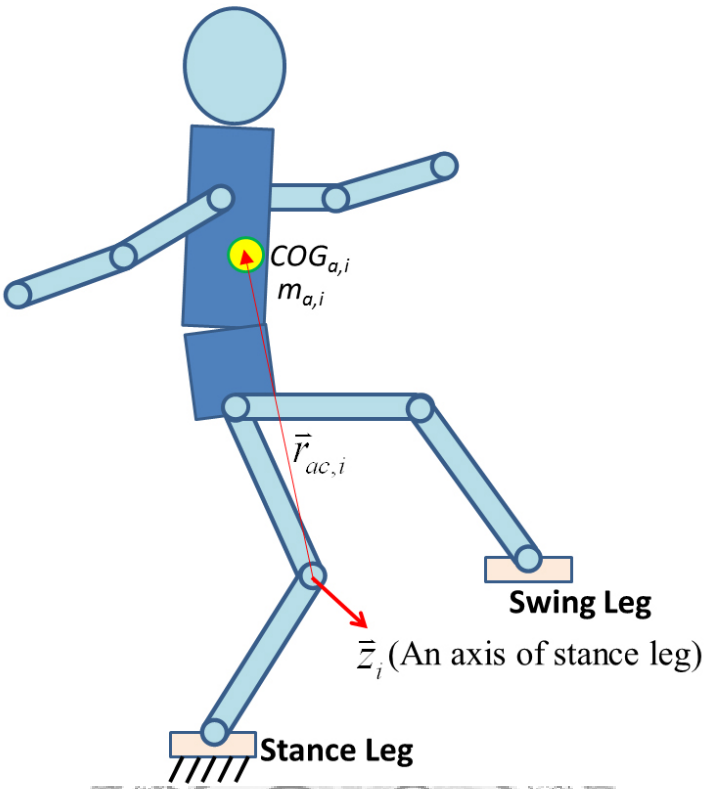


Figure 3-12. Linear momentum caused form the joint rotation of the stance leg

where $m_{a,i}$ denotes the whole mass above the ith joint of the stance leg, and $\vec{r}_{ac,i}$ denotes the vector from the ith joint to the COG of the accumulated mass. The linear momentum Jacobian matrix is almost the same as the translational part of the conventional Jacobian matrix; the only difference is the linear momentum Jacobiam matrix has the accumulated mass term in the equations.

3.5.2 Iterative Calculation of Moment of Inertia

To find the angular momentum Jacobian matrix, the spin and orbit angular momentum caused by unit rotation of robot axes must be found first. Same as the arrangement in section 2.5, the spin part of angular momentum is discussed first in this section.

To calculate the spin angular momentum caused by unit rotation of all robot joints, the iterative equations describing the accumulated inertia of the robot must be derived, as shown in Eqns. (3-24)-(3-30). The derivation starts from calculating the inertia matrix with respect to an arbitrary point. The parallel axis theorem in 3D space is used for these calculations, as shown in Eq. (3-24) and Figure 3-13. This is used to combine the inertia matrices of different links with respect to their averaged COG position.

$$I_{p,i} = I_i + \vec{r}_p^T \vec{r}_p E_3 - \vec{r}_p \vec{r}_p^T \tag{3-24}$$

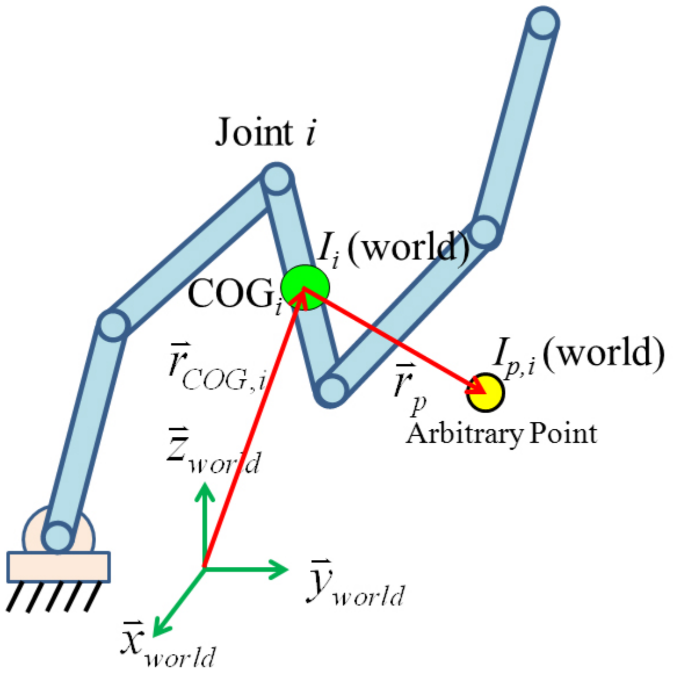


Figure 3-13. The parallel axis theorem in 3D space

where $I_{p,i}$ is the inertia matrix with respect to an arbitrary point, $\vec{r}_{COG,i}$ denotes the position vector of the ith link COG in the world coordinates, $\vec{r}_p$ denotes the vector from the COG of the ith link to an arbitrary point, $\vec{r}_p^T \vec{r}_p$ is a scalar, E_3 denotes a 3-by-3 identity matrix, and $\vec{r}_p \vec{r}_p^T$ is a 3-by-3 symmetry matrix. Define $\vec{r}_p$ as

$$\vec{r}_p = \begin{bmatrix} x_{rp} \\ y_{rp} \\ z_{rp} \end{bmatrix} \tag{3-25}$$

Thus Eq. (3-25) can be further written as

$$I_{p,i} = I_i + m_i \begin{bmatrix} y_{rp}^2 + z_{rp}^2 & -x_{rp}y_{rp} & -x_{rp}z_{rp} \\ -x_{rp}y_{rp} & x_{rp}^2 + z_{rp}^2 & -y_{rp}z_{rp} \\ -x_{rp}z_{rp} & -y_{rp}z_{rp} & x_{rp}^2 + y_{rp}^2 \end{bmatrix} \tag{3-26}$$

Using Eqns. (2-32) and (3-24) the inertia matrices of all robot links can be found with backward iterations, as shown in Eqns. (3-27)-(3-30) and Figure 3-14.

$$\vec{r}_{a,i} = \frac{m_{a,i+1}\vec{r}_{a,i+1} + m_i\vec{r}_{COG,i}}{m_{a,i}} \tag{3-27}$$

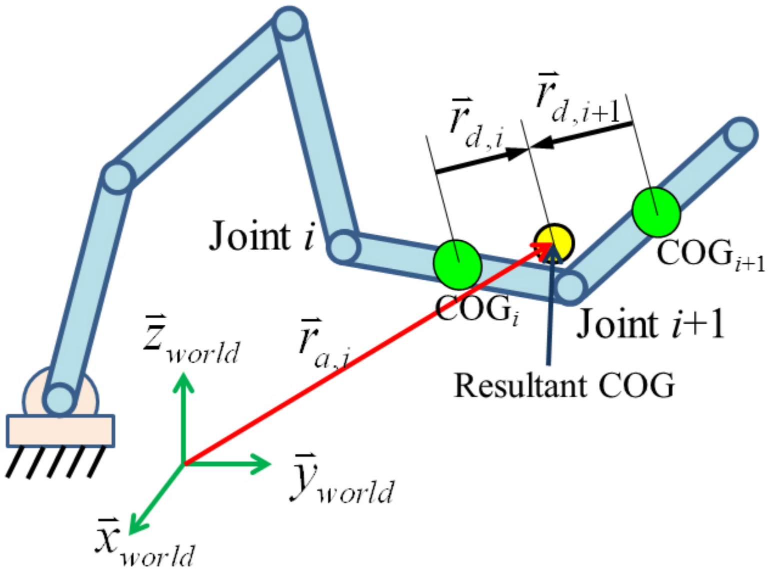


Figure 3-14. Iterative inertia calculation

where $\vec{r}_{a,i}$ denotes position vector of the COG of these links, $m_{a,i}$ is shown as Eq. (3-21). $\vec{r}_{d,i}$ and $\vec{r}_{d,i+1}$ denotes the vectors from the original accumulated COG and the ith link COG to the new accumulated COG, as

$$\vec{r}_{d,i} = \vec{r}_{a,i} - \vec{r}_{COG,i} \tag{3-28}$$

$$\begin{aligned} \vec{r}_{d,i+1} &= \vec{r}_{a,i} - \vec{r}_{COG,i+1}, \quad 1st\ iteration \\ \vec{r}_{d,i+1} &= \vec{r}_{a,i} - \vec{r}_{a,i+1}, \quad otherwise \end{aligned} \tag{3-29}$$

Using the equations above, the accumulated moment of inertia $I_{a,i}$ is calculated as

$$\begin{aligned} I_{a,i} = I_{a,i+1} + I_i + m_i\left(\vec{r}_{d,i}^{\,T}\vec{r}_{d,i}E_3 - \vec{r}_{d,i}\vec{r}_{d,i}^{\,T}\right) \\ + m_{a,i+1}\left(\vec{r}_{d,i+1}^{\,T}\vec{r}_{d,i+1}E_3 - \vec{r}_{d,i+1}\vec{r}_{d,i+1}^{\,T}\right) \end{aligned} \tag{3-30}$$

3.5.3 Angular Momentum Jacobian

Using the results in section 3.5.2, the spin angular momentum caused by the rotation of the ith joint can be found as

$$\vec{A}_{M,spin,i} = I_{a,i}\vec{z}_i\dot{\theta}_i \tag{3-31}$$

Note that the summation of Eq. (3-31) is equal to Eq. (2-33), as shown in Eq. (3-32). The summation of Eq. (3-31) considers joint rotation and Eq. (2-33) considers link rotation to calculate the total spin angular momentum; these two methods get the same results.

$$\vec{A}_{M,spin} = \sum I_i\vec{\omega}_i = \sum I_{a,i}\vec{z}_i\dot{\theta}_i \tag{3-32}$$

After calculating the spin angular momentum, the momentum caused by the rotation of the *i*th joint can be found by adding the orbit angular momentum to it. The orbit angular momentum caused by the rotation of the *i*th joint can be calculated using the vectors shown in Figure 3-15 as Eq. (3-33).

$$\vec{A}_{M,orbit,i} = \vec{r}_{s\to c,i} \times \left(m_{a,i}\vec{z}_i\dot{\theta}_i \times \vec{r}_{ac,i}\right) \tag{3-33}$$

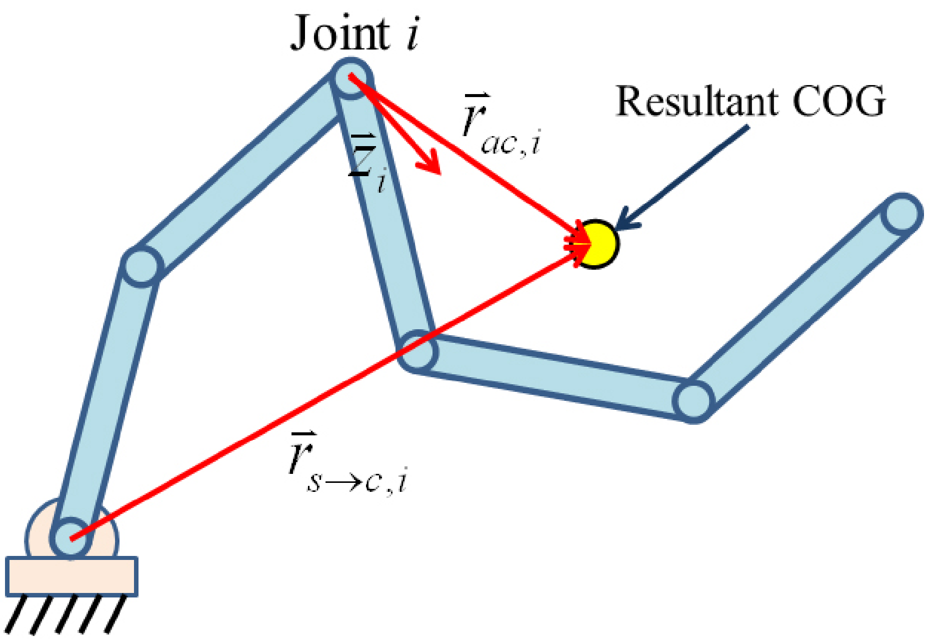


Figure 3-15. Orbit angular momentum caused by the rotation of the *i*th joint

Eq. (3-33) can be used for the limbs except the stance leg. Figure 3-15 shows the orbit angular momentum with respect to the support point. In the application of humanoid robot, the support point is the ankle of the stance foot.

Using the results above, the angular momentum caused by the rotation of the *i*th joint is calculated by summing the spin part and the orbit part of the angular momentum, as

$$\vec{A}_{M,i} = I_{a,i}\vec{z}_i\dot{\theta}_i + \vec{r}_{s\to c,i} \times \left(m_{a,i}\vec{z}_i\dot{\theta}_i \times \vec{r}_{ac,i}\right) \tag{3-34}$$

Since the joint rotation speed $\dot{\theta}_i$ is a scalar, Eq. (3-34) can be rearranged as

$$\vec{A}_{M,i} = \left(I_{a,i}\vec{z}_i + \vec{r}_{s\to c,i} \times \left(m_{a,i}\vec{z}_i \times \vec{r}_{ac,i}\right)\right)\dot{\theta}_i = J_{AM,i}\dot{\theta}_i \tag{3-35}$$

where $J_{AM,i}$ denotes the 3-by-1 angular momentum Jacobian. For the joints of the stance leg, the concept of the Fixed-Leg-Motion Jacobian is again used. Minus sign is multiplied and the accumulated mass and inertia of the parts are replaced by that of the parts above the *i*th joint, as shown in Eq. (3-36) and Figure 3-16.

$$\vec{A}_{M,i} = -\left(I_{a,i}\vec{z}_i + \vec{r}_{s\to c,i} \times \left(m_{a,i}\vec{z}_i \times \vec{r}_{ac,i}\right)\right)\dot{\theta}_i = J_{AM,i}\dot{\theta}_i \tag{3-36}$$

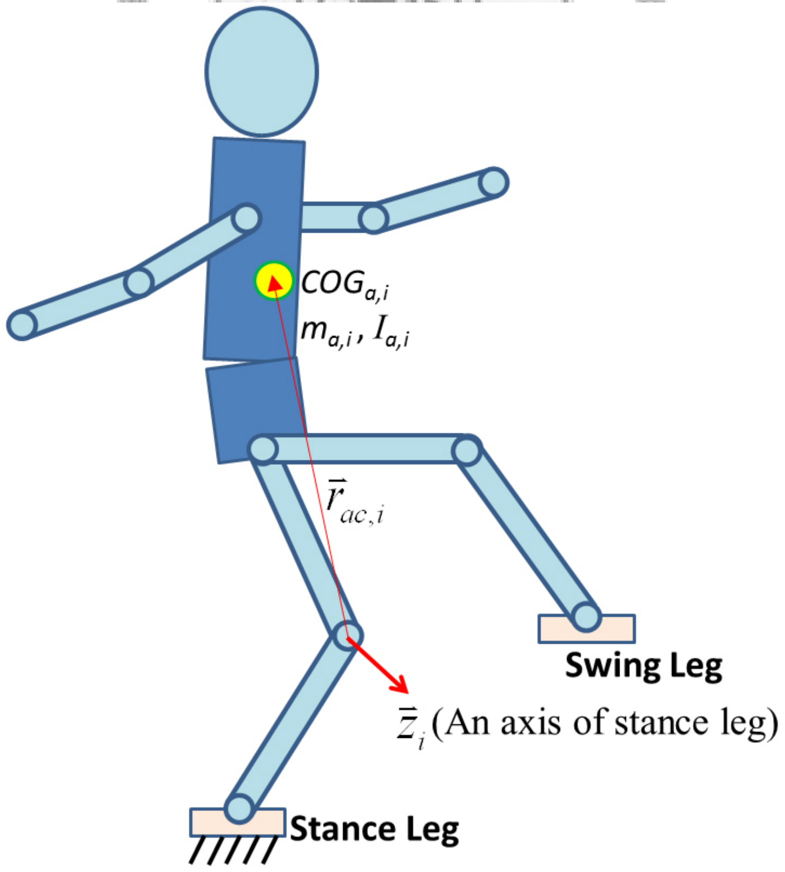


Figure 3-16. Fixed angular momentum Jacobian

Total angular momentum of the whole robot can be calculated as Eq. (3-37). It is equal to Eq. (2-33).

$$\vec{A}_M = \sum J_{AM,i}\dot{\theta}_i \tag{3-37}$$

Eq. (3-37) is more complex and it is derived for calculating the momentum Jacobian. The difference between the two equations is that Eq. (2-33) uses the concept of the rotation of each link and Eq. (3-37) uses the concept of the rotation of each joint.

3.6 Global Jacobian

Using the conventional Jacobian, COG Jacobian, momentum Jacobian, and the Fixed-Leg-Motion Jacobian, the global Jacobian matrix for controlling the whole humanoid robot can be constructed, as

$$\begin{bmatrix} \dot{d}_{stance} \\ \dot{d}_{swing} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \\ \dot{d}_{COG} \\ \dot{d}_{LM} \\ \dot{d}_{AM} \end{bmatrix} = \begin{bmatrix} J_{stance} & 0 & 0 & 0 & 0 \\ J_{f\to s} & J_{swing} & 0 & 0 & 0 \\ J_{f\to LA} & 0 & J_{LA} & 0 & 0 \\ J_{f\to RA} & 0 & 0 & J_{RA} & 0 \\ J_{f\to H} & 0 & 0 & 0 & J_{H} \\ J_{f\to C} & J_{s\to C} & J_{LA\to C} & J_{RA\to C} & J_{H\to C} \\ J_{f\to LM} & J_{s\to LM} & J_{LA\to LM} & J_{RA\to LM} & J_{H\to LM} \\ J_{f\to AM} & J_{s\to AM} & J_{LA\to AM} & J_{RA\to AM} & J_{H\to AM} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{stance} \\ \dot{\theta}_{swing} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \tag{3-38}$$

Compared with Eq. (3-19), the linear and angular momentum Jacobian matrices are added to the whole Jacobian matrix. If the robot has enough DOFs, all the motion of end-effectors, COG position and momentum can be controlled with Eq. (3-38). Since the robot has limited DOFs in the legs, arms and torso, not all of them can be controlled. The Jacobian matrix used to control the proposed humanoid robot is shown as Eq. (3-39).

$$\begin{bmatrix} \dot{d}_{stance} \\ \dot{d}_{swing} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \\ \dot{d}_{COG} \\ \dot{d}_{AM,z} \end{bmatrix} = \begin{bmatrix} J_{stance} & 0 & 0 & 0 & 0 \\ J_{f\to s} & J_{swing} & 0 & 0 & 0 \\ J_{f\to LA} & 0 & J_{LA} & 0 & 0 \\ J_{f\to RA} & 0 & 0 & J_{RA} & 0 \\ J_{f\to H} & 0 & 0 & 0 & J_{H} \\ J_{f\to C} & J_{s\to C} & J_{LA\to C} & J_{RA\to C} & J_{H\to C} \\ J_{f\to AM,z} & J_{s\to AM,z} & J_{LA\to AM,z} & J_{RA\to AM,z} & J_{H\to AM,z} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{stance} \\ \dot{\theta}_{swing} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \tag{3-39}$$

In Eq. (3-39), although the Jacobian matrices of linear and angular momentum are derived, only the angular momentum in z direction is controlled. This is because the angular momentum in z direction is directly related to the slipping rotation of the robot in z direction. When the robot turn left, it needs a counterclockwise moment in z direction to change its facing direction and vise versa; when the robot is going straight forward, the z direction moment must be as small as possible to prevent the robot from slipping and rotating about z-axis. The linear momentum Jacobian and angular momentum Jacobian in x and y directions can be used in future applications. The other parts of the Eq. (3-39) are the same as Eq. (3-19). By using Eq. (3-39), the control of arm motions, robot walking, COG trajectory tracking, and z direction angular momentum can be achieved. Versatile tasks can be achieved with the proposed IK solver.

3.7 Simulation

The physical model of the robot is constructed in SolidWorks (CAD software), and is imported to ADAMS (the physics engine), which is served as the control plant. In this section, the physical properties of the walking pattern generated with the proposed IK solver are verified. Figure 3-17 shows the scene of simulation for verifying the properties of the proposed IK solver.

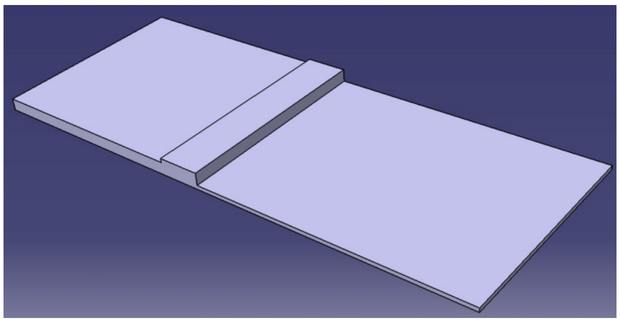


Figure 3-17. The scene of simulation for verifying the proposed IK solver

In the simulation, the robot goes onto a step and steps down to a lower plane. The COG trajectories inputted to the proposed IK solver are generated with the LQSI controller proposed in chapter 4. This simulation shows that the proposed IK solver can generate walking pattern with COG and momentum constraints.

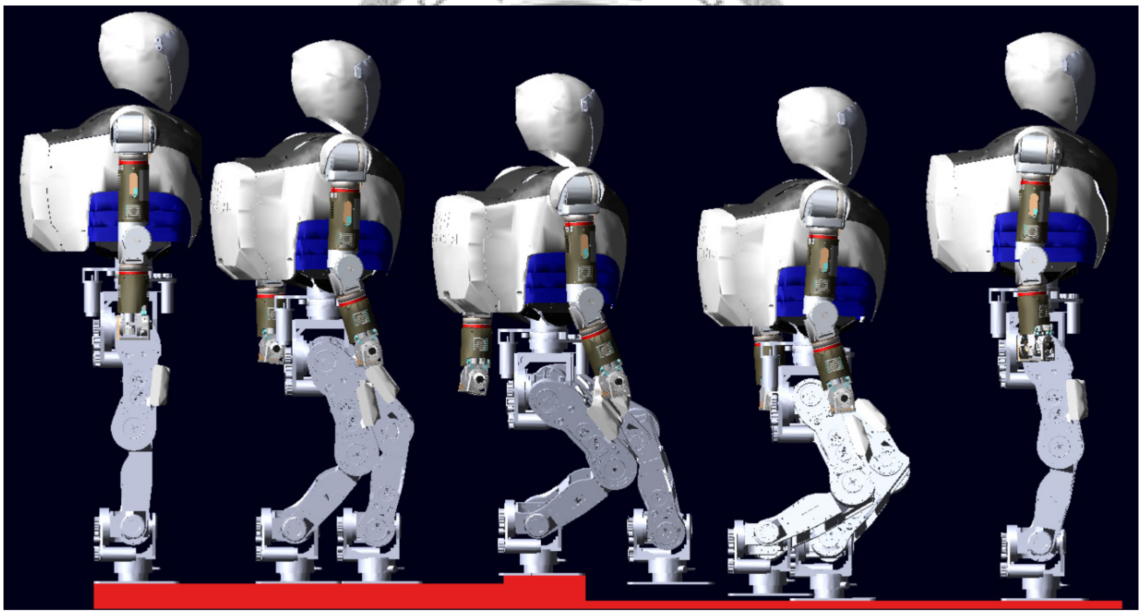


Figure 3-18. The robot motions for walking through the scene

Figure 3-18 shows the robot can walk through the scene successfully. In this scene, the slipping rotations with and without angular momentum constraint in z direction are compared, as shown in Figure 3-19. Zero angular momentum in z direction is set as the constraint in the proposed solver. This is achieved with the DOFs of waist and arms in the solver. The solved trajectory of waist and arms will compensate the angular momentum caused by the swing motion of the robot leg automatically. However, the real angular momentum will not be zero because of modeling error and impact forces, but it will be smaller than using an IK solver without momentum constraint. Since the time differentiation of angular momentum is smaller, the reaction torque acted on the stance foot is also smaller. This will reduce the slipping rotation of the robot when walking. In addition, because the robot will twist the waist and swing the arms, the solved motions are more human-like than the motions without momentum constraints. In the simulation, the robot can walk with smaller slipping rotation by enabling the angular momentum constraint in z direction. There are larger slipping rotations when the robot going onto or going down the step. A larger impact occurred when landing to the lower plane and caused a larger direction change to the robot. With zero angular momentum constraint, the robot can walk with smaller slipping rotation even when the landing impact occurred.

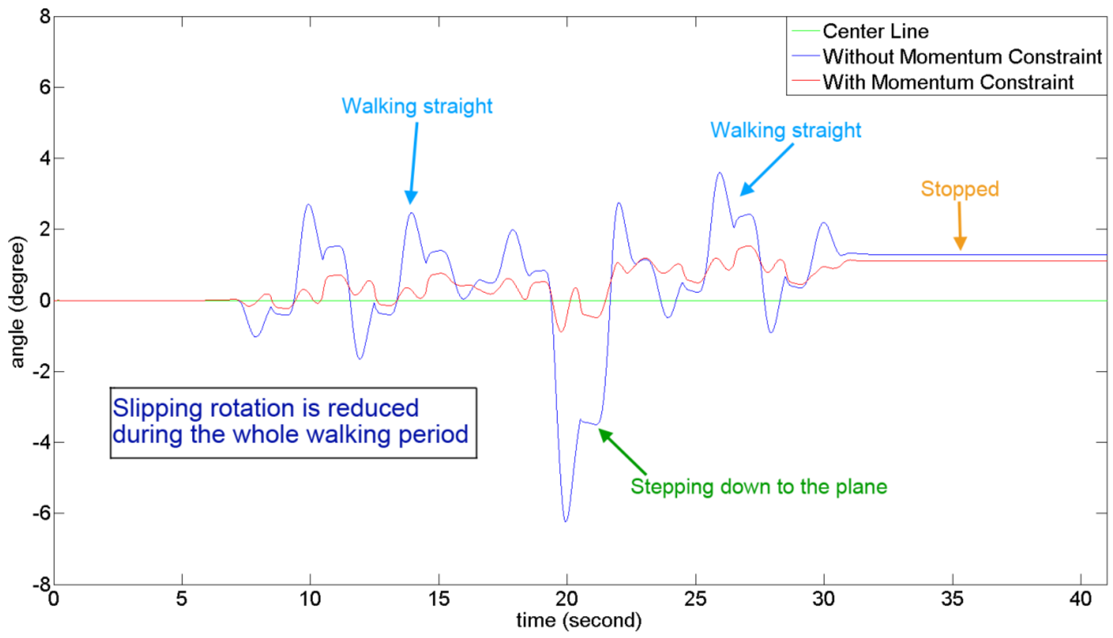


Figure 3-19. Slipping rotation of the whole robot in z direction

3.8 Summary

In this chapter, the concept of Fixed-Leg-Motion Jacobian is proposed and described. It is very convenient because no transformation of the coordinate system is required and the same kinematics chains of the robot can be used while changing the support phase during walking. Using the concept of the Fixed-Leg-Motion Jacobian method, the effects of the end-effector positions and orientations, the COG position, and the linear and angular momentum caused by the motions of the stance leg can be expressed easily. The global Jacobian matrix used for controlling humanoid robots is also proposed and derived in this chapter. Using the global Jacobian matrix, the end-effectors, COG, and the momentum of the robot can be controlled.

The IK solver used in this dissertation is derived and described in this chapter; the next point this dissertation focuses on is the planning and optimization of 3D COG/ZMP trajectories in chapters 4 and 5.

Chapter 4 Linear Quadratic State Incremental Control

Many researchers have proposed walking pattern generation methods with COG/ZMP (zero moment point / center of gravity) constraints. Some of the researchers used a neural-networks method (NN), a central pattern generator (CPG), or a genetic algorithm (GA) to solve COG/ZMP pattern generation problems. However, the parameters used in those methods are too many, and the procedure to learn or to search them cost too much computation time. Other researchers designed controllers or used analytical solution method to generate the COG trajectories. These methods can generate the COG/ZMP pattern very quickly, but the COG height is limited to a constant to linearize the inverted pendulum model of the robot. Due to this limitation, the robots cannot walk freely on surfaces that change in height. In order to solve this problem, researchers start to use the original nonlinear inverted pendulum model to make the COG height value changeable, such as using a numerical method or a feedback controller. In this chapter, a pattern generator that can allow non-constant COG height is proposed. The proposed pattern generator is based on an optimal control method with a state-incremental performance index. It can solve sagittal and lateral COG patterns with arbitrarily assigned COG height and ZMP trajectories in real-time. Thus, dynamic walking with a natural COG trajectory on height-changing surfaces can be achieved.

4.1 Introduction

In recent years, researchers have focused on many interesting aspects, such as artificial intelligence, human-machine interaction [44], trajectory/pattern generation [13], and mechanical design. In these research areas, walking pattern generation is very important for biped robots because inappropriately planned walking patterns may cause the robot to fall down and even to damage itself. Honda ASIMO [110], the AIST HRP series [47][56][57][59], Waseda University WABIAN [100], KAIST HUBO [101][105][106], and Fujitsu Hoap series [83][148] are the most successful humanoid robots in the world. Their methods for generating walking patterns are different but all consider the ZMP information.

Most walking pattern generation methods are restrained by unchangeable COG height, long computation time, or too many parameters needed to be tuned. Compared with using the original nonlinear inverted pendulum model, the computation cost will be much smaller if the COG height is set to be a constant to linearize the inverted pendulum model. Researchers used the preview control based [25][47][48][125] and the analytical solution based [37][38] walking pattern generator with the linearized model to construct their walking control systems. Real-time pattern generation can be achieved by doing this but the robot must bend its knees more while walking in order to maintain the same COG height, and this also limits the ability of the robot to walk on surfaces with height change. Researchers also discuss the effect of COG height change to walking pattern generators and provide their solutions. Nishiwaki [91] extended the preview control method to the one with changeable COG height by tuning the control gains from the pre-calculated database under different COG height. Sugihara [122] also provided an analytical solution method to generate the walking pattern in real-time. In the method, the COG height can be changed if the height change is not very large.

Morisawa et al. [87] proposed a method to construct the walking patterns in parametric surface. The method can generate walking pattern on the order of millisecond with COG height change by using numerical method. The WABIAN robot was designed with extra DOFs for its waist to keep its knee straight [98]. The walking pattern of the WABIAN robot was done with GA [100]. It is very good to keep the knee of the stance foot of the robot straight because it saves energy. The only drawback is that the GA is too slow to be used for real-time planning problems. The method in [148] used a CPG and a piecewise linear pattern generator to generate the walking pattern of the Hoap-3 robot. In that method, many parameters must be tuned with a neural network. It takes a long time to find each set of parameters for different tasks.

In other researches, the CPG [62], NN [60][61][118], and Fourier series/transform methods [18][99] were used to learn and to generate the walking patterns. The CPG method can generate a COG height changeable pattern in real-time, but it is hard to tune well and is not robust for different situations. NN methods can learn to adapt to different situations, but the learning time is not fast enough for real-time implementation. The Fourier series/transform methods can generate walking patterns fast enough, but they are difficult to adapt to various situations. Although the COG height is changeable, the searching and learning algorithms have some difficulty to be implemented in real-time.

It is always desirable to make the robot move and act smoothly and in real-time; thus a real-time algorithm to generate walking patterns is required. In this dissertation, a real-time pattern generator with changeable COG height is proposed. The proposed LQSI controller can generate walking pattern with arbitrarily assigned COG height trajectory in real-time. It is based on optimal control schemes with a specially designed performance index. Optimal control schemes are very useful for nonlinear time-varying problems. Instead of tuning numerous parameters, only a few parameters in the performance index need to be tuned to get the desired performance of the control system. Although the optimal controllers need more computation power to solve backward recursive equations and calculate all the feedback and feed-forward gains, the proposed LQSI controller can be implemented on the order of millisecond.

The rest of this chapter is organized as follows: section 4.2 describes COG/ZMP equations and inverted pendulum model. Many researches used a linear one to simplify the equation but the COG heights were constrained to constant values. In this chapter, a linear quadratic control based controller is proposed to solve this problem by using the original nonlinear inverted pendulum model. Section 4.3 introduces the well-known preview control method widely used by many robots. Preview control provides a very neat and fast solution to COG/ZMP walking pattern generation. Section 4.4 shows the control equations of the proposed LQSI controller and discusses the properties of it compared with preview controller. Detailed derivation of the LQSI controller can be found in APPENDIX A and APPENDIX B. Section 4.5 shows the simulation settings and results in MATLAB and ADAMS using the LQSI controller. Section 4.6 summarizes this chapter.

4.2 Inverted Pendulum Model and COG/ZMP Equations

As in most COG/ZMP pattern generation algorithms [17][38][45][47][69], the inverted pendulum model used to model the biped robot as shown in Figure 4-1.

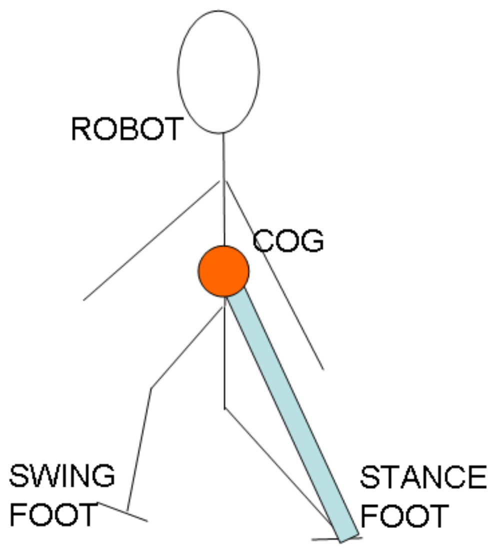


Figure 4-1. Inverted pendulum model

The governing equation in sagittal and lateral directions of the inverted pendulum model is shown in Eqns. (4-1) and (4-2).

$$Z_X = C_X - \ddot{C}_X\left(\frac{C_Z - Z_Z}{g + \ddot{C}_z - \ddot{Z}_z}\right) \tag{4-1}$$

$$Z_Y = C_Y - \ddot{C}_Y\left(\frac{C_Z - Z_Z}{g + \ddot{C}_z - \ddot{Z}_z}\right) \tag{4-2}$$

where the subscripts X, Y, and Z denote the trajectories in x, y, and, z directions in the world coordinates. Capital *C* denotes the COG, capital *Z* denotes the ZMP, and *g* denotes the gravitational acceleration, 9810 mm/s^2. In the proposed method, the $C_Z\ (t)$ and $Z_Z\ (t)$ trajectories can be arbitrarily assigned. The robot system is decoupled as two inverted pendulum systems: one is in X-Z plane and the other is in Y-Z plane. The two systems use the same $C_Z\ (t)$ and $Z_Z\ (t)$ trajectories. Since the two systems have the same characteristics when solving the pattern generation problem, the COG position in x and y directions are replaced by a state variable "x" below. Since the COG height (C_z) is changeable, the height of the ground is also changeable in the pattern generator. Thus, $C_Z = C_Z\ (t)$ and $Z_Z = Z_Z\ (t)$. $Z(t) = C_Z\ (t) - Z_Z\ (t)$ is defined to simplify the equations; it yields the height from the COG to the ground. Eq. (4-1) can be rewritten in state-space representation as

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix} = A \begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix} + Bu \tag{4-3}$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g+\ddot{Z}(t)}{Z(t)} & 0 & -\frac{g+\ddot{Z}(t)}{Z(t)} \\ 0 & 0 & 0 \end{bmatrix} \tag{4-4}$$
$$B = [0 \quad 0 \quad 1]^T$$

where *x* denotes C_X or C_Y, p denotes the corresponding ZMP position, and u is the differentiation of the ZMP position. In order to track the reference ZMP, the output y of Eq. (4-3) is chosen as

$$y = C\begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix}, \quad C = [0 \quad 0 \quad 1] \tag{4-5}$$

Eqns. (4-3) and (4-5) are the continuous time version of the control system. For simulations and real implementations, it must be discretized. After discretization, the equations become

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ p_{k+1} \end{bmatrix} = A_k \begin{bmatrix} x_k \\ \dot{x}_k \\ p_k \end{bmatrix} + B_k u_k \tag{4-6}$$

$$y_k = C_k \begin{bmatrix} x_k \\ \dot{x}_k \\ p_k \end{bmatrix} \tag{4-7}$$

Eqns. (4-3) to (4-7) represent the time-varying state-space model [55]. The ZOH (zero order hold) based method will be used to find A_k, B_k, and C_k is described as follows.

Eqns. (4-3) and (4-4) denote a linear time varying system and it is rewritten as Eq. (4-8) for simplification. State matrix A changes after each sampling time. It can be rewritten as Eqns. (4-9) and (4-10).

$$\dot{x} = Ax + Bu \tag{4-8}$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \omega_k^2 & 0 & -\omega_k^2 \\ 0 & 0 & 0 \end{bmatrix} \tag{4-9}$$

$$\omega_k^2 = \frac{g + \ddot{Z}(kT)}{Z(kT)} \tag{4-10}$$

where ω_k changes with $Z(kT)$ function. We also use ZOH on the control input u, thus the discretized relationship between x_k and x_{k+1} can be solved as

$$x_{k+1} = e^{AT} x_k + \int_{kT}^{kT+T} e^{A(kT+T-\tau)} B d\tau \, u_k = A_k x_k + B_k u_k \tag{4-11}$$

Because A_k are all the same format with different $Z(kT)$, we can find the solution of e^{AT} in advance in order to speed up the discretization of the system. Using Laplace transform, it can be obtained as

$$e^{AT} = A_k = \begin{bmatrix} cosh(\omega_k T) & \frac{1}{\omega_k} sinh(\omega_k T) & 1 - cosh(\omega_k T) \\ \omega_k sinh(\omega_k T) & cosh(\omega_k T) & -\omega_k sinh(\omega_k T) \\ 0 & 0 & 1 \end{bmatrix} \tag{4-12}$$

$$\int_{kT}^{kT+T} e^{A(kT+T-\tau)} B d\tau = B_k = \begin{bmatrix} T - \frac{1}{\omega_k} sinh(\omega_k T) \\ 1 - cosh(\omega_k T) \\ T \end{bmatrix} \tag{4-13}$$

The output of the state space system in Eqns. (4-3) to (4-5) is the ZMP position; C_k is a constant matrix, as shown in Eq. (4-14).

$$C_k = [0 \quad 0 \quad 1] \tag{4-14}$$

4.3 Preview Control

The preview controller used in humanoid robots was proposed by Kajita and et al [47][55]. It is widely used in many approaches. In the controller, the COG height is assumed to be a constant; thus, the state matrices A_k and B_k are set to be the value under the average COG height in this dissertation, as shown in Eq. (4-15). It is also called linear inverted pendulum model (LIPM), or the cart-table model [47].

$$Z(t) = \bar{Z}_C \quad A_k = A_0 \quad B_k = B_0 \tag{4-15}$$

The performance index of the preview controller is shown in equation (4-16).

$$J = \sum_{j=1}^{\infty} \left(Q\left(p_j^{ref} - p_j\right)^2 + Ru_j^2 \right) \tag{4-16}$$

where the p_j^{ref} is the reference ZMP input. The control algorithms are listed in Eqns. (4-17)–(4-19)

$$u_k = -Kx_k + [f_1, f_2, \cdots f_N] \begin{bmatrix} p_{k+1}^{ref} \\ \vdots \\ p_{k+N}^{ref} \end{bmatrix} \tag{4-17}$$

$$K = (R + B_0^T P B_0)^{-1} B_0^T P A_0 \tag{4-18}$$

$$f_i = (R + B_0^T P B_0)^{-1} B_0^T (A_0 - B_0 K)^{T(i-1)} C^T Q \tag{4-19}$$

where P is the solution of the Riccati equation in Eq. (4-20).

$$P = A_0^T P A_0 + C^T Q C - A_0^T P B_0 (R + B_0^T P B_0)^{-1} B_0^T P A_0 \tag{4-20}$$

The parameters are chosen as

$$Q = 10^6 \qquad R = 0.001 \tag{4-21}$$

In order to compare the performances in the next section, the parameters in the performance index of the LQSI controller are chosen as Eq. (4-22). The same Q and R are used in both controllers.

$$Q_x = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = Q_N = 10^6 \qquad R = 0.001 \tag{4-22}$$

4.4 Linear Quadratic State-Incremental Control (LQSI)

The linear quadratic integral (LQI) control technique is a well-known method that uses a state-incremental and input-incremental performance index [32][49][117]. The previewable reference inputs are useful to control the systems; however, it can only be used for linear time invariant systems. For a time varying inverted pendulum system, in order to track the reference input ZMP well, an optimal control method with a state-incremental performance index is proposed in this dissertation.

Before deriving and describing the controller, the control goal is stated first. The robot should track reference ZMP input with the sagittal and lateral (x and y directions) COG trajectories generated by the controller. The inputs to the controller are ZMP in x, y, and z directions, and COG in z direction. The outputs are the COG trajectories in x and y directions, as shown in Figure 4-2.

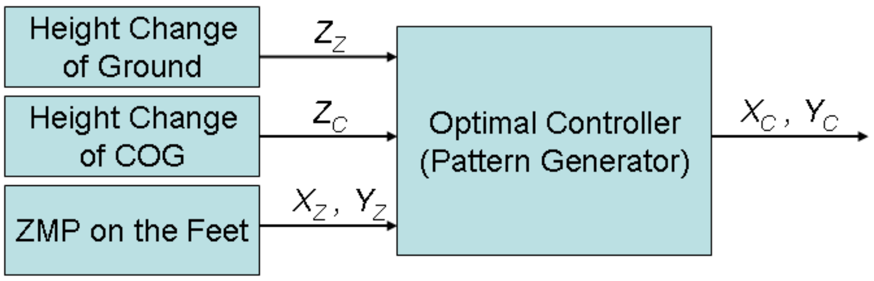


Figure 4-2. Input-output relationship of the controller

The proposed linear quadratic state-incremental control (LQSI), which is based on optimal control method, can solve the tracking problems for time-varying system. The performance index of the LQSI method is shown as

$$J = \frac{1}{2}\sum_{k=i}^{\infty}((x_{k+1} - x_k)^T Q_x (x_{k+1} - x_k) + (C_k x_k - r_k)^T Q (C_k x_k - r_k) + u_k^T R u_k) \tag{4-23}$$

where $(x_{k+1} - x_k)$ denotes the state increment, $(C_k x_k - r_k)$ denotes the tracking error, r_k is the tracking reference, and u_k denotes the control input. Q_x, Q, and R denote the weighting of each term in the performance index, respectively. Q and R are symmetric positive definite matrices and Q_x is a symmetric positive semi-definite matrix. Large Q_x reduces the state change in each control step, but it allows the accumulated state to be large. Large Q improves the tracking performance, and large R reduces the power of the control input. By solving the optimal control problem with the constraint in Eq. (4-24), the LQSI control law can be found as Eqns. (4-25)–(4-27). The detailed derivation and proof can refer to APPENDIX A.

$$f_k = x_{k+1} = A_k x_k + B_k u_k \tag{4-24}$$

$$u_k = -K_k^x x_k + K_k^v v_{k+1} \tag{4-25}$$

$$K_k^x = M_k^{-1} B_k^T \left(Q_x A_{I,k} + S_{k+1} D_k^{-1} W_k\right) \tag{4-26}$$

$$K_k^v = M_k^{-1} B_k^T \left(I - S_{k+1} D_k^{-1} B_k M_k^{-1} B_k^T\right) \tag{4-27}$$

The elements in Eqns. (4-25)–(4-27) are defined in Eqns. (4-28)–(4-31).

$$M_k = B_k^T Q_x B_k + R \tag{4-28}$$

$$A_{I,k} = A_k - I \tag{4-29}$$

$$D_k = I + B_k M_k^{-1} B_k^T S_{k+1} \tag{4-30}$$

$$W_k = A_k - B_k M_k^{-1} B_k^T Q_x A_{I,k} \tag{4-31}$$

S_k and v_k derived from the Lagrange multiplier method can be obtained as

$$S_k = P_k^T P_k + A_k^T S_{k+1} D_k^{-1} W_k \tag{4-32}$$

$$v_k = A_k^T \left(I - S_{k+1} D_k^{-1} B_k M_k^{-1} B_k^T\right) v_{k+1} + C_k^T Q r_k \tag{4-33}$$

where the $P_k^T P_k$ is a positive definite matrix and is defined as

$$P_k^T P_k = N_k^T Q_x N_k + A_{I,k}^T Q_x B_k M_k^{-1} R M_k^{-1} B_k^T Q_x A_{I,k} + C_k^T Q C_k \tag{4-34}$$

$$N_k = W_k - I \tag{4-35}$$

Eqns. (4-25)–(4-35) describe the infinite time LQSI optimal control law. It can be derived as a finite time case by changing the performance index to be Eq. (4-36) and solving the boundary conditions of the final-state-free problems in Eq. (4-37) [74].

$$J = \phi(x_N, u_N) + \frac{1}{2} \sum_{k=0}^{N-1} \left(\Delta x_k{}^T Q_x \Delta x_k + (C_k x_k - r_k)^T Q (C_k x_k - r_k) + u_k^T R u_k\right) \tag{4-36}$$

$$\frac{\partial \phi}{\partial x_N} = \lambda_N \tag{4-37}$$

More details about the boundary condition and the derivation can be found in section 4.4.1 and APPENDIX B. Figure 4-3 is the procedure by which the proposed LQSI controller works. In the “Generate Command” phase, the trajectories of the COG height, the ZMP trajectory on the ground and the height of the ground can be determined by a higher-level trajectory planner. In the “Modeling” phase, with the trajectory inputs, the A_k and B_k of the inverted pendulum system can be calculated. In the “Backward Recursion” phase, the state feedback gain K_k^x and the feed-forward gain K_k^v are calculated with Eqns. (4-25)–(4-35). Finally, in the “Optimal Control” phase, with the feedback and feed-forward gains, the control input to the system is obtained to find the COG pattern every time step that the robot should follow by forward iterations.

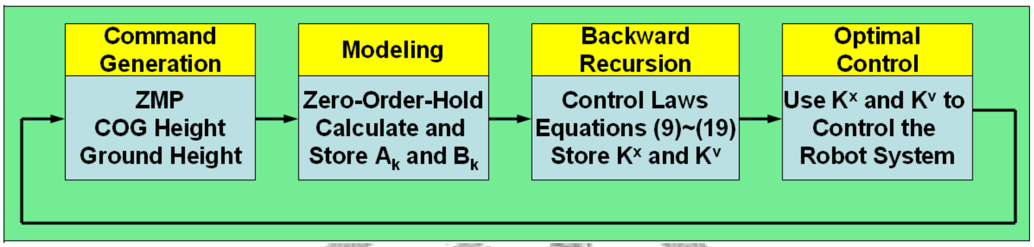

Command Generation
ZMP COG Height Ground Height
Modeling
Zero-Order-Hold Calculate and Store A_k and B_k
Backward Recursion
Control Laws Equations (9)~(19) Store K^x and K^v
Optimal Control
Use K^x and K^v to Control the Robot System
Figure 4-3. The procedure by which the LQSI controller works

Figure 4-4 shows the control block diagram of the LQSI control: the plant is modeled as an inverted pendulum system which is linear time varying. The feed-forward and feedback gains K_k^v and K_k^x are calculated by the LQSI control laws in the “Backward Recursion” phase; the input v_{k+1} is calculated in the same phase with the reference ZMP input information.

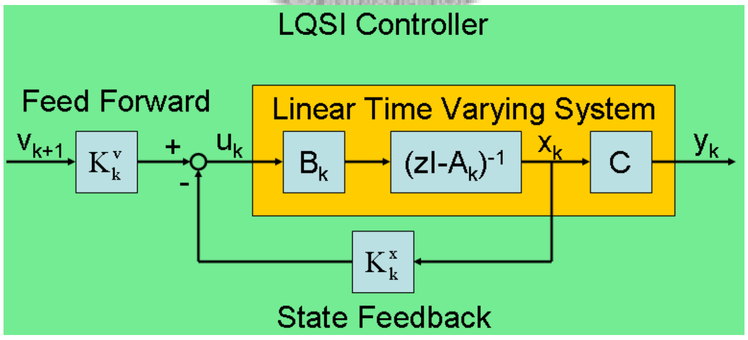


Figure 4-4. Control block diagram of the LQSI controller

Figure 4-5 shows the role that the LQSI controller plays in the entire robot system. In the figure, before deciding the commands to the robot, the robot must know information about the environment. Then the robot can do trajectory planning for walking in the environment. After trajectory planning, the LQSI controller decides the sagittal and the lateral COG positions. Then the joint angles of the robot are generated by using IK. In the feedback control loop, the LQSI controller also corrects the COG trajectories in order to track the reference ZMP input.

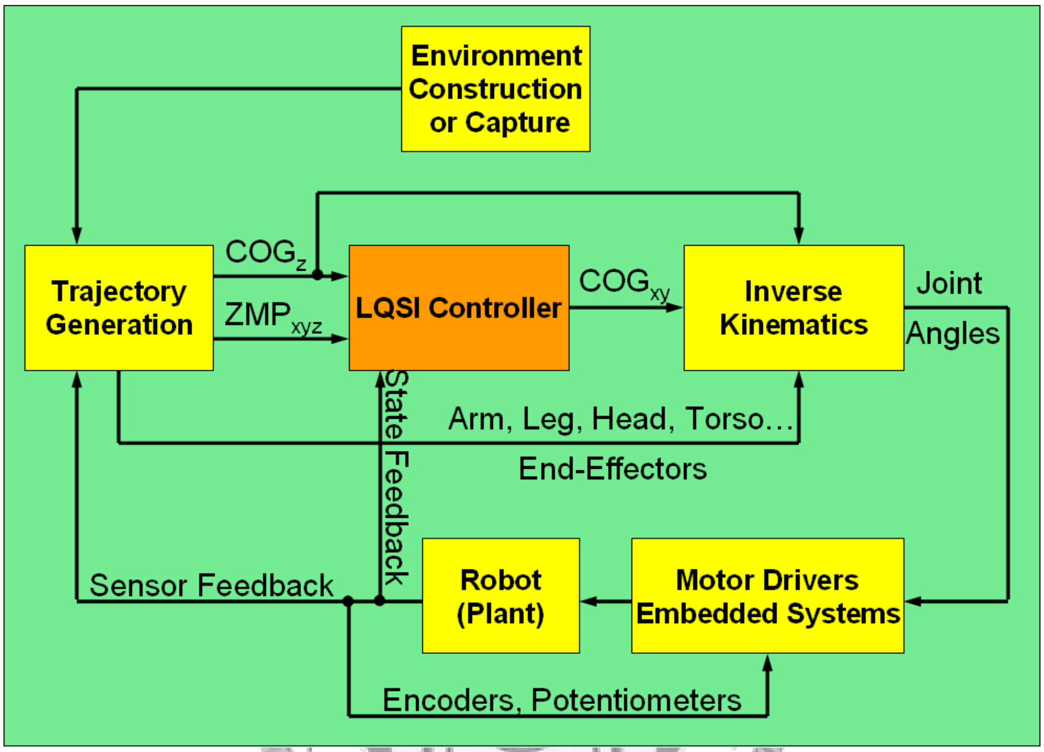


Figure 4-5. The role of the LQSI controller in the entire robot control system

4.4.1 Boundary Condition of the LQSI Controller

The boundary condition of the LQSI controller is derived in APPENDIX B. The initial values, S_N and v_N, for backward recursion can be found as classical finite-time final-state-free optimal control problems. With these initial values, S_k becomes stable after some iterations. Before S_k becomes stable, the tracking performance is not very good. This is a common problem of optimal controllers. To solve this problem, two methods can be used. The first is to extend the final state N to far future and then cut the part with transient S_k. The second choice is to treat the problem as an infinite time problem, and set the COG height change as zero in far future. In the second choice, we need to find the solution to time-invariant version of Eq. (4-32), the Riccati equation of LQSI controller, as shown in Eq. (4-38).

$$S_\infty = P^T P + A^T S_\infty (I + BM^{-1}B^T S_\infty)^{-1} W \tag{4-38}$$

The definition of the elements of Eq. (4-38) can be found in the beginning of section 4.4. Because the COG height change in far future is assumed zero, thus the state matrices become the same as the matrices used in preview control, as in Eq. (4-15). The variables, P, B, M, and W, in (4-38) are also constant matrices. By iterating Eq. (4-38) with a nonzero initial S matrix, the solution S_∞ can be found. This S_∞ can be used as the initial value for backward iteration. Simulation results show that we can get much shorter transient time if we treat the problem as an infinite time one and set S_∞ as the initial value, as shown in Figure 4-6 and Figure 4-7. In the figures, S_k is waving with the C_Z input.

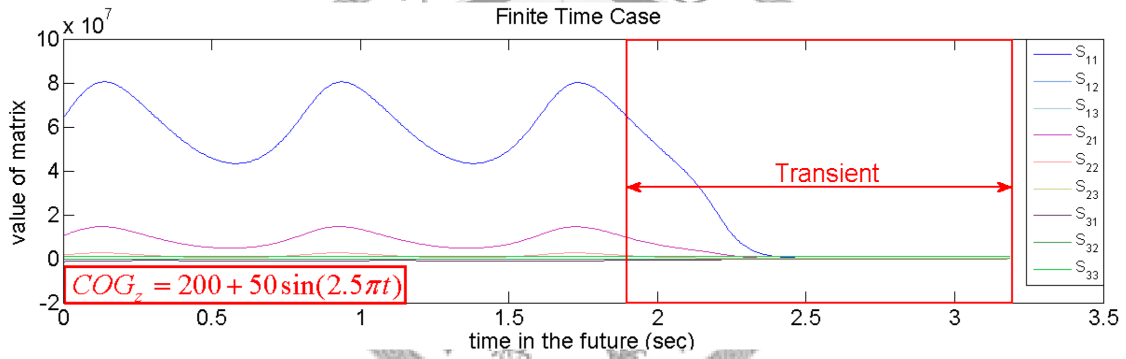


Figure 4-6. Transient time of S_k by using final-state-free boundary condition

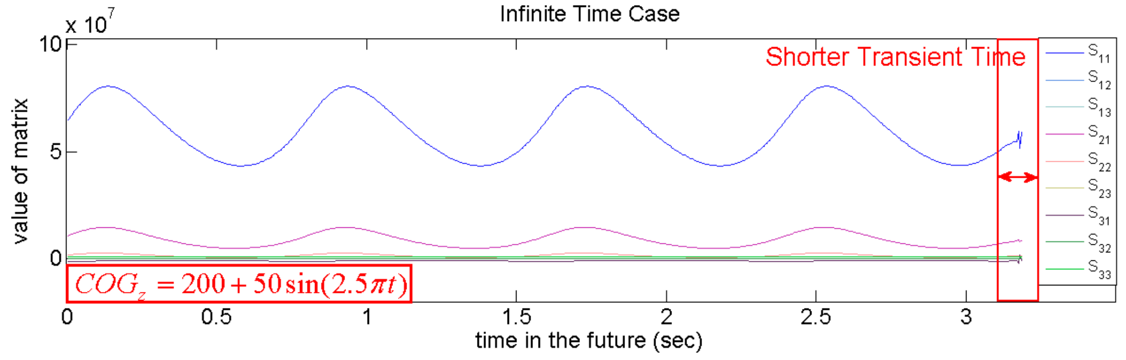


Figure 4-7. Transient time of S_k by using S_∞ as initial value

Because S_∞ is the solution of the Riccati equation of the time-invariant version of LQSI controller, the transient time is shorter. To give extra 0.3sec future command input is enough for waiting S_k to become stable.

4.4.2 Preview Gain of the LQSI Controller

Same as the preview controller, the preview gain of the LQSI controller can be found. The preview gain (it is also the feed-forward gain of the future command input.) can be found by rewriting the Eq. (4-33). Before rewriting, a variable L_k is defined to simplify Eq. (4-33), as shown in Eq. (4-39).

$$L_k = A_k^T(I - S_{k+1}D_k^{-1}B_k M_k^{-1}B_k^T) \tag{4-39}$$

Then, the Eq. (4-33) can be rewritten as

$$v_k = C_k^T Q r_k + \sum_{p=k+1}^{k+N} (L_k L_{k+1} \cdots L_{p-1}) C_k^T Q r_p + (L_k L_{k+1} \cdots L_{k+N}) v_{k+N+1} \tag{4-40}$$

When the preview time is far enough from k, $L_k L_{k+1} \cdots L_{k+N}$ becomes very small. Thus, the effect of v_{k+N+1} to v_k can be ignored and so is r_{k+N+1}. The preview gain f_k can be calculated with Eqns. (4-25), (4-27) and (4-40), as shown in Eq. (4-41).

$$f_k = M_k^{-1} B_k^T A_k^{-T} L_k L_{k+1} \cdots L_{p-1} C_k^T Q \Big|_{p=k+1}^{p=k+N} \tag{4-41}$$

The preview gain of LQSI controller under different C_Z input is shown in Figure 4-8. The frequency of the C_Z is 0, 10, 16.33 rad/s, to make the maximum acceleration of the C_Z to be 0, 3000, 8000 mm/s^2, respectively.

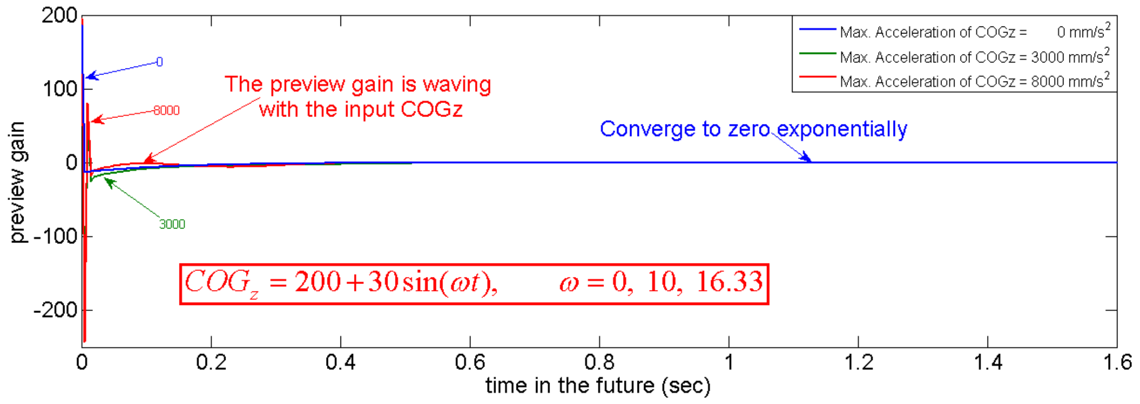


Figure 4-8. LQSI control preview gain of vs. different max. C_Z acceleration

In Figure 4-8, when ω is zero, the values of the preview gain of the LQSI controller are almost the same as the preview gain of the preview controller calculated with Eq. (4-19). Figure 4-9 shows the preview gain of the LQSI controller waves as the C_z trajectory.

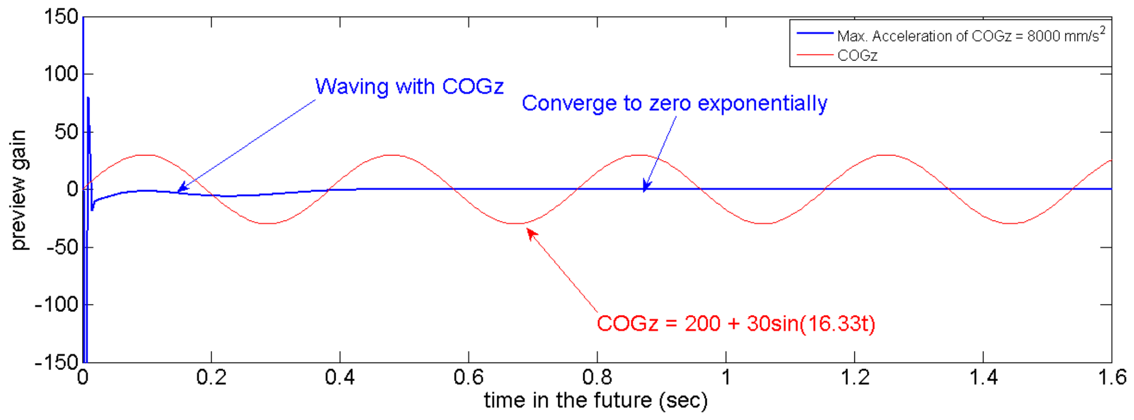


Figure 4-9. The preview gain of LQSI waving with C_Z trajectory

4.4.3 Minimum Required Future Reference Input

Since the preview gain of the LQSI controller is highly relevant to the value of S_k, the time of 0.3 seconds is required for waiting S_k to become stable from its initial value, S_∞. Then S_k will become stable within this time period, as shown in section 4.4.1. In addition, the preview gain of the LQSI controller converges to zero exponentially, as shown in Figure 4-8 and Figure 4-9. At the preview time about 1.6 seconds, the gain becomes very small and the effect of the reference input farther than 1.6 second can be ignored. It is the same as the result of preview controller shown in [47][55]. Thus, the minimum required length of future reference input for the LQSI controller is 1.9 seconds long. The LQSI controller is very similar to the preview controller. If Q and R are set the same in both the LQSI controller and the preview controller and set the Q_x in LQSI controller much smaller than Q, then the LQSI controller can be regarded as a time-varying version of preview controller. Furthermore, if the C_z input to the LQSI controller is also set as a constant, we can get the same solution to Riccati equation and the same preview gain of both controllers.

4.5 Simulation and Results

In this section, the properties and performances of the LQSI controller and preview controller are compared. Basic parameter settings of the two controllers are the same; i.e., gravity $g = 9810\text{mm}/s^2$ and the sampling time $T = 0.005s$. The system state space matrices are A_k, B_k, and C_k in Eqns. (4-12)-(4-14). The reference input to the controllers is the desired ZMP trajectory.

4.5.1 Simulation Using Inverted Pendulum Model

The simulation using inverted pendulum model is done in MATLAB. Under the COG height (C_z) reference input shown in Figure 4-10, the ZMP tracking performances of the LQSI controller, conventional optimal controller and the preview controller are compared. In this section, the humanoid robot is modeled as an inverted pendulum in MATLAB. All the mass of the robot is in one point. The dynamics of linkage motions are neglected here. The main goal of this section is to discuss the modeling error of the preview control under varying COG height trajectory and the parameter tuning in each controller. Since the characteristic in both sagittal and lateral directions are all the same in the controllers when using the inverted pendulum model. Only the results in lateral direction are shown in this section.

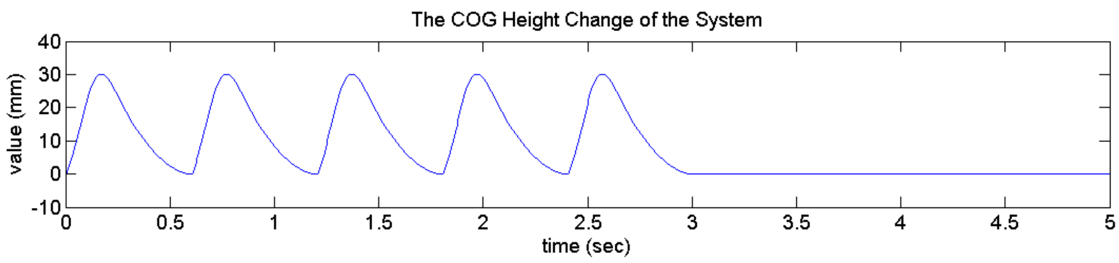


Figure 4-10. COG height (C_z) input

In Figure 4-10, the COG height varies for three seconds and then stops varying. The tracking performances are shown in Figure 4-11 and Figure 4-12.

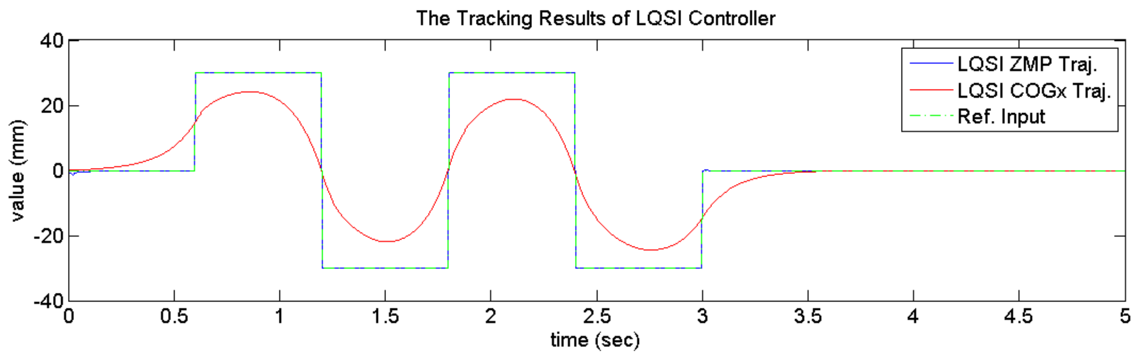


Figure 4-11. Results of LQSI controller

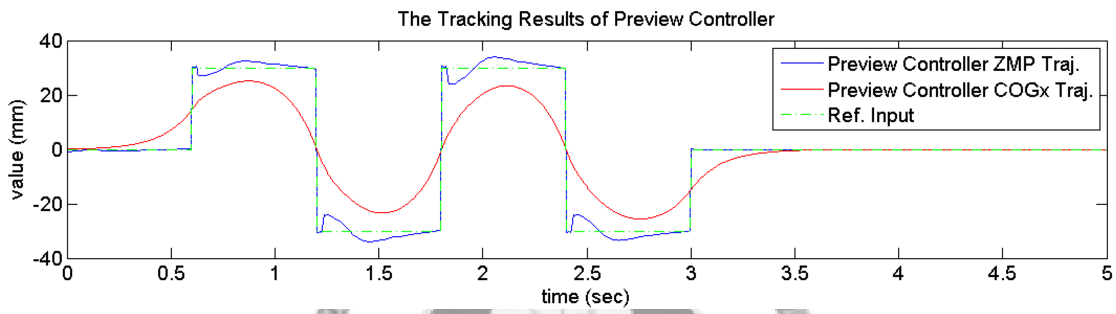


Figure 4-12. Results of preview controller

In Figure 4-11, the LQSI controller can track the ZMP input well with the COG height trajectory. In Figure 4-12, because the preview controller does not consider the COG height change, tracking error is caused by the un-modeled dynamics. Compared with the preview control, COG height change can be modeled by using conventional optimal controller. Two types of conventional optimal controller are shown as follows, the optimal tracking controller and the modified optimal tracking controller. Eqns. (4-42) and (4-43) show the performance index and its boundary condition of the conventional optimal controller.

$$J = \phi(x_N, u_N) + \frac{1}{2}\sum_{k=0}^{N-1}[(C_k x_k - r_k)^T Q (C_k x_k - r_k) + u_k^T R u_k] \qquad (4\text{-}42)$$

$$\phi(x_N, u_N) = (C_N x_N - r_N)^T Q_N (C_N x_N - r_N) \qquad (4\text{-}43)$$

In the applications of the conventional optimal controller, the weightings of the tracking performance term and the input term can be tuned to get the desired performance. The control algorithms are shown in Eqns. (4-44)–(4-48).

$$u_k = -K_k^x x_k + K_k^v v_{k+1} \tag{4-44}$$

$$K_k^x = (B_k^T S_{k+1} B_k + R)^{-1} B_k^T S_{k+1} A_k \tag{4-45}$$

$$K_k^v = (B_k^T S_{k+1} B_k + R)^{-1} B_k^T \tag{4-46}$$

$$S_k = A_k^T S_{k+1}(A_k - B_k K_k^x) + C_k^T Q C_k \qquad S_N = C_N^T Q_N C_N \tag{4-47}$$

$$v_k = (A_k - B_k K_k^x)^T v_{k+1} + C_k^T Q r_k \qquad v_N = C_N^T Q_N r_N \tag{4-48}$$

where the parameters Q and R are tuned as the preview and the LQSI controllers, as

$$Q = 10^6 \qquad R = 0.001 \tag{4-49}$$

The modified optimal tracking controller is slightly different from the conventional optimal tracking controller; the cost of the state, $x_k^T Q_x x_k$, is added to the cost function as shown in Eqns. (4-50)–(4-51).

$$J = \phi(x_N, u_N) + \frac{1}{2}\sum_{k=0}^{N-1}\left[x_k^T Q_x x_k + (C_k x_k - r_k)^T Q (C_k x_k - r_k) + u_k^T R u_k\right] \tag{4-50}$$

$$\phi(x_N, u_N) = (C_N x_N - r_N)^T Q_N (C_N x_N - r_N) + x_N^T Q_{xN} x_N \tag{4-51}$$

A small Q_x can be used to prevent the state from diverging, and the value of Q_x cannot be too large because it will affect the tracking performance.

The control algorithms of the conventional and modified optimal tracking controller are almost the same; the differences between them are the Riccati equation and the boundary condition, as shown in Eq. (4-52).

$$S_k = A_k^T S_{k+1}(A_k - B_k K_k^x) + C_k^T Q C_k + Q_x \quad S_N = C_N^T Q_N C_N + Q_x \tag{4-52}$$

The parameters are tuned as

$$Q_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Q = Q_N = 10^6 \qquad R = 0.001 \tag{4-53}$$

Using the same COG height input as shown in Figure 4-10, the tracking results of the conventional and modified optimal tracking controllers are shown in Figure 4-13 and Figure 4-14.

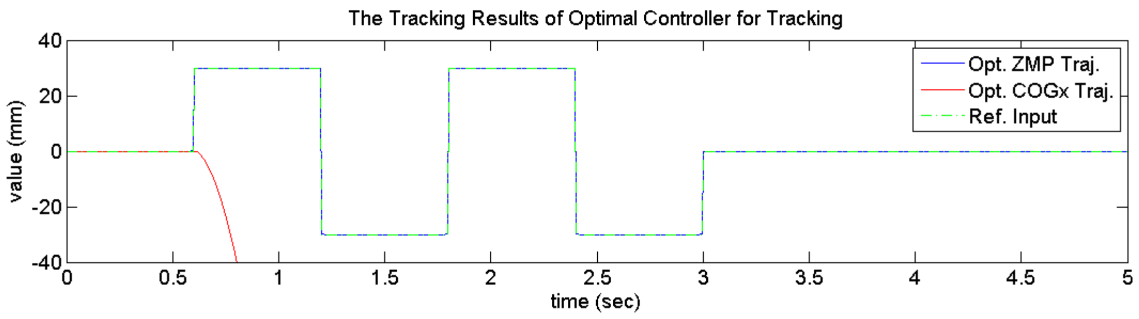


Figure 4-13. Results of optimal controller for tracking

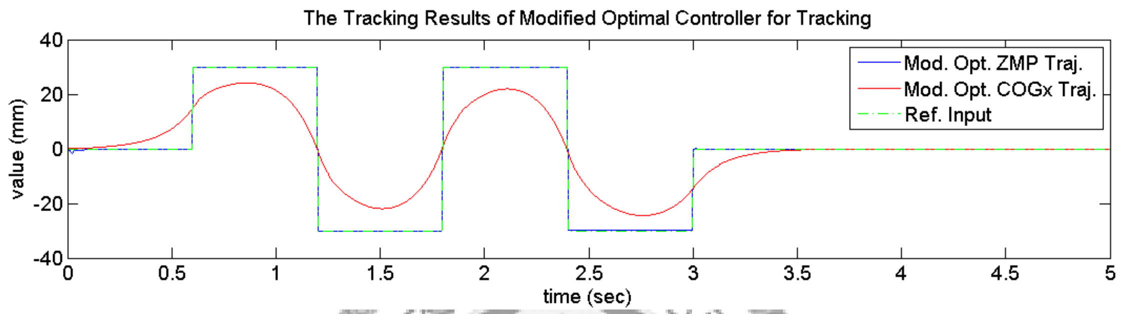


Figure 4-14. Results of modified optimal controller for tracking

In Figure 4-13, the state, the COG position, diverges to infinity because the value of the COG position does not affect the value of the performance index. Thus the term, $x_k^T Q_x x_k$, is added to the performance index in the modified optimal controller for stabilizing the COG position. The COG trajectory and tracking results are very similar to the results of the LQSI controller as shown in Figure 4-14. The difference between the LQSI controller and the modified optimal controller is discussed in the following.

In order to compare the controllers, some parameters in the performance index are defined as Eqns. (4-54) and (4-55).

$$Q_{xS} = \begin{bmatrix} Q_{S,11} & Q_{S,12} & Q_{S,13} \\ Q_{S,21} & Q_{S,22} & Q_{S,23} \\ Q_{S,31} & Q_{S,32} & Q_{S,33} \end{bmatrix} \tag{4-54}$$

$$Q_{xM} = \begin{bmatrix} Q_{M,11} & Q_{M,12} & Q_{M,13} \\ Q_{M,21} & Q_{M,22} & Q_{M,23} \\ Q_{M,31} & Q_{M,32} & Q_{M,33} \end{bmatrix} \tag{4-55}$$

where Q_{xS} is the weighting matrix of the state-incremental cost term in Eq. (4-36), Q_{xM} is the weighting matrix of the state cost term in Eq. (4-50). From the performance index in Eq. (4-36), $Q_{S,11}$ and $Q_{S,22}$ affect the first-order and the second-order difference of the COG position (i.e., the velocity and the acceleration). $Q_{S,33}$ affects the change in the first-order difference (i.e., velocity) of the ZMP position. These three terms can be seen as the regulation weightings of the COG velocity, COG acceleration, and the ZMP velocity. On the other hand, from the performance index in Eq. (4-50), $Q_{M,11}$, $Q_{M,22}$ and $Q_{M,33}$ are the regulation weightings of the COG position, COG velocity, and ZMP position, respectively. $Q_{M,11}$ and $Q_{M,33}$ pull the COG trajectory and the ZMP trajectory to zero. So, when the center of the robot is not near the origin (zero), COG and ZMP cannot fit the desired trajectory well, with non-zero $Q_{M,11}$ and $Q_{M,33}$. They must be zero for good tracking performance. The effect of $Q_{M,11}$ and $Q_{M,33}$ are shown in Figure 4-15 and Figure 4-16. In these figures, the tracking performance weighting, Q, and the input cost weighting, R, are the same as in Eq. (4-49) (i.e. $Q = 10^6$, $R = 0.001$). The robot is assumed to be one meter (1000 mm) away from the origin.

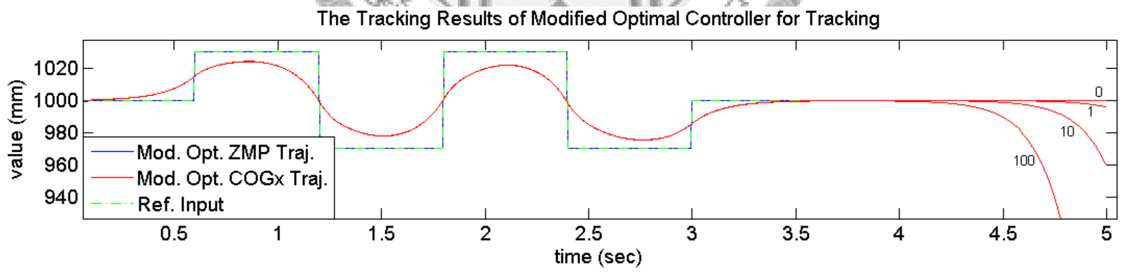


Figure 4-15. Modified optimal controller for tracking with different $Q_{M,11}$

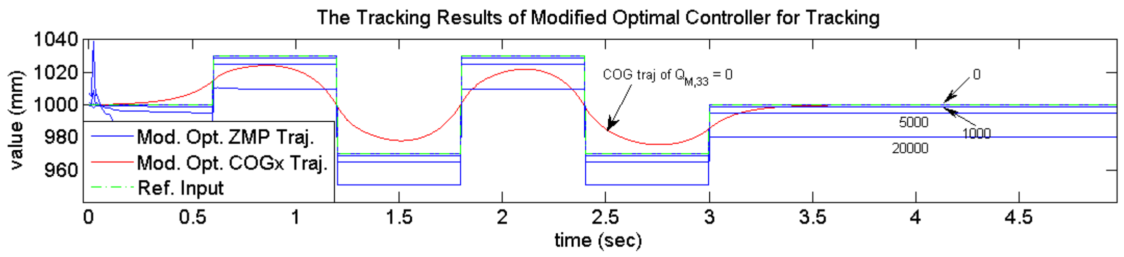


Figure 4-16. Modified optimal controller for tracking with different $Q_{M,33}$

In Figure 4-15, the parameters are set as below.

$$Q_{xM} = \begin{bmatrix} Q_{M,11} & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4-56}$$

In Eq. (4-56), $Q_{M,11}$, changes from 0 to 100. For a reasonable and nature walking pattern, the COG must converge to the desired ZMP when the robot stops; thus the only choice for the $Q_{M,11}$ is zero.

The parameters in Figure 4-16 are set as

$$Q_{xM} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & Q_{M,33} \end{bmatrix} \tag{4-57}$$

In Eq. (4-57), $Q_{M,33}$, changes from 0 to 20000. ZMP deviation becomes larger with the increasing $Q_{M,33}$. It is very important to have an accurate ZMP trajectory. For robust and stable walking, $Q_{M,33}$ must be zero. The other non-diagonal elements in Q_{xM} must also be zero, because they are the coupling terms of the COG position and the ZMP position. Thus, the only tunable parameter in Q_{xM} is $Q_{M,22}$.

Compared with Q_{xM}, more characteristics of the control system can be tuned from tuning Q_{xS} of the proposed LQSI controller. The cost of COG velocity, COG acceleration, and ZMP velocity can be tuned by $Q_{S,11}$, $Q_{S,22}$, and $Q_{S,33}$, respectively. The other coupling terms in Q_{xS} can also be tuned to change the characteristics of the control system. Tuning of $Q_{S,33}$ changes the sensitivity to the ZMP position change of the LQSI control system. Smoother ZMP trajectory provides a more stable and more human-like walking pattern. Eq. (4-58) gives the parameters of the LQSI controller in Figure 4-17. The weighting $Q_{S,33}$ is set to tune the ZMP position change rate.

$$Q_{xS} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 10^6 \end{bmatrix} \tag{4-58}$$

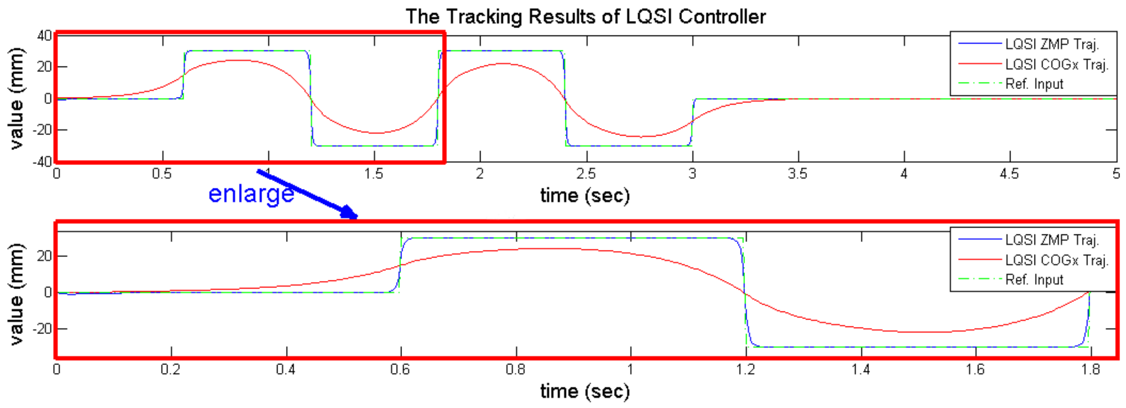


Figure 4-17. Result of tuning $Q_{S,33}$

In Figure 4-17, the output ZMP is smoothed by setting $Q_{S,33} = 10^6$ in Eq. (4-58).

4.5.2 Simulation Using Physical Model in ADAMS

Using the same IK solver and simulation engine in chapter 3, the simulation of physical properties of the LQSI controller is verified. There are two simulations represented in this section. In the first simulation, the human-sized humanoid robot lowers its COG to avoid an obstacle and then step onto another block. In the second simulation, the small-sized humanoid robot walks through a more complex environment including a slope and several blocks with different height. The environment parameters and settings in ADAMS are shown in Table 4-1. The contact model is modeled as mass-spring-damper system with a very strong spring (10^8 N/m of stiffness). Penetration depth is the penetration which ADAMS/solver turns on full damping.

Table 4-1: Environment parameters and settings in ADAMS

Gravity acceleration	9810 mm/s^2
Stiffness (contact model)	10^8 N/m
Damping (contact model)	10^4 N-sec/m
Force exponent (contact model)	2.2
Penetration depth (contact model)	0.1 mm
Static friction coefficient	0.3
Dynamic friction coefficient	0.3

During the first simulation, the robot must lower its COG to avoid its head hitting the red block, and raise its COG to step onto the green block. After passing the red block, the robot can recover the COG to the normal height. It is easy to accomplish this task with the static walking algorithm. Figure 4-18 shows the results of dynamic walking using the proposed LQSI controller. And Figure 4-19 shows the ZMP tracking results.

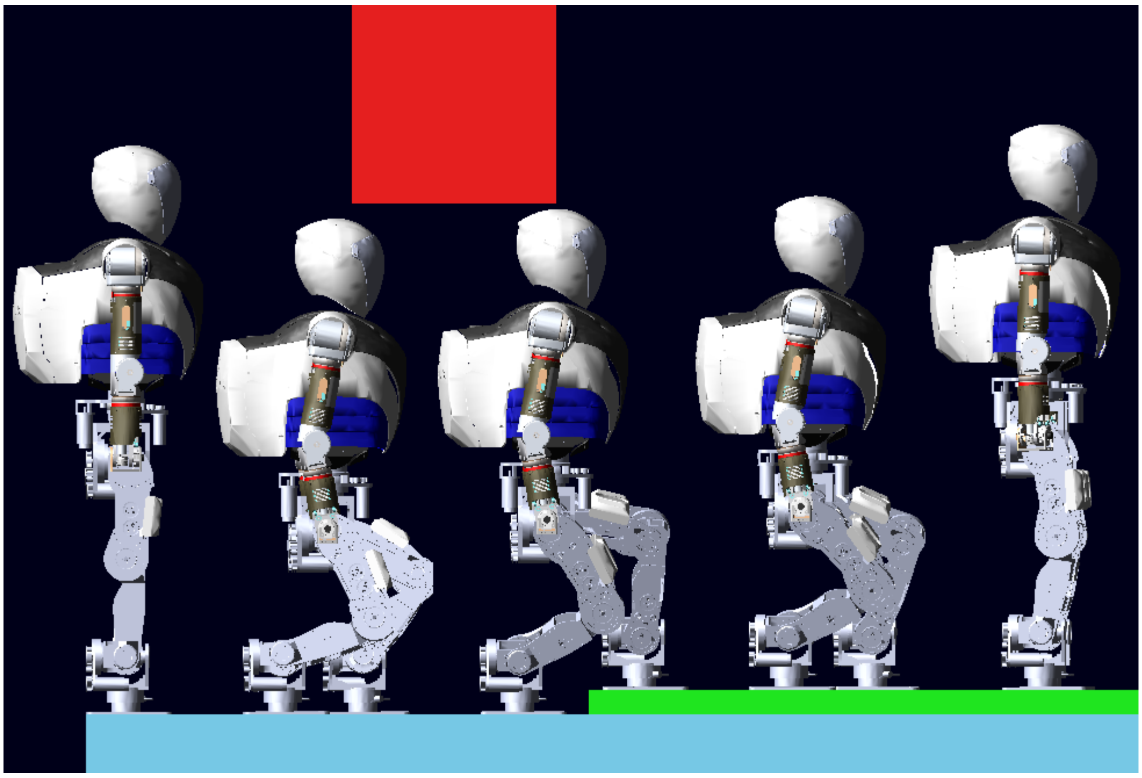


Figure 4-18. Dynamic walking with LQSI controller in the first simulation

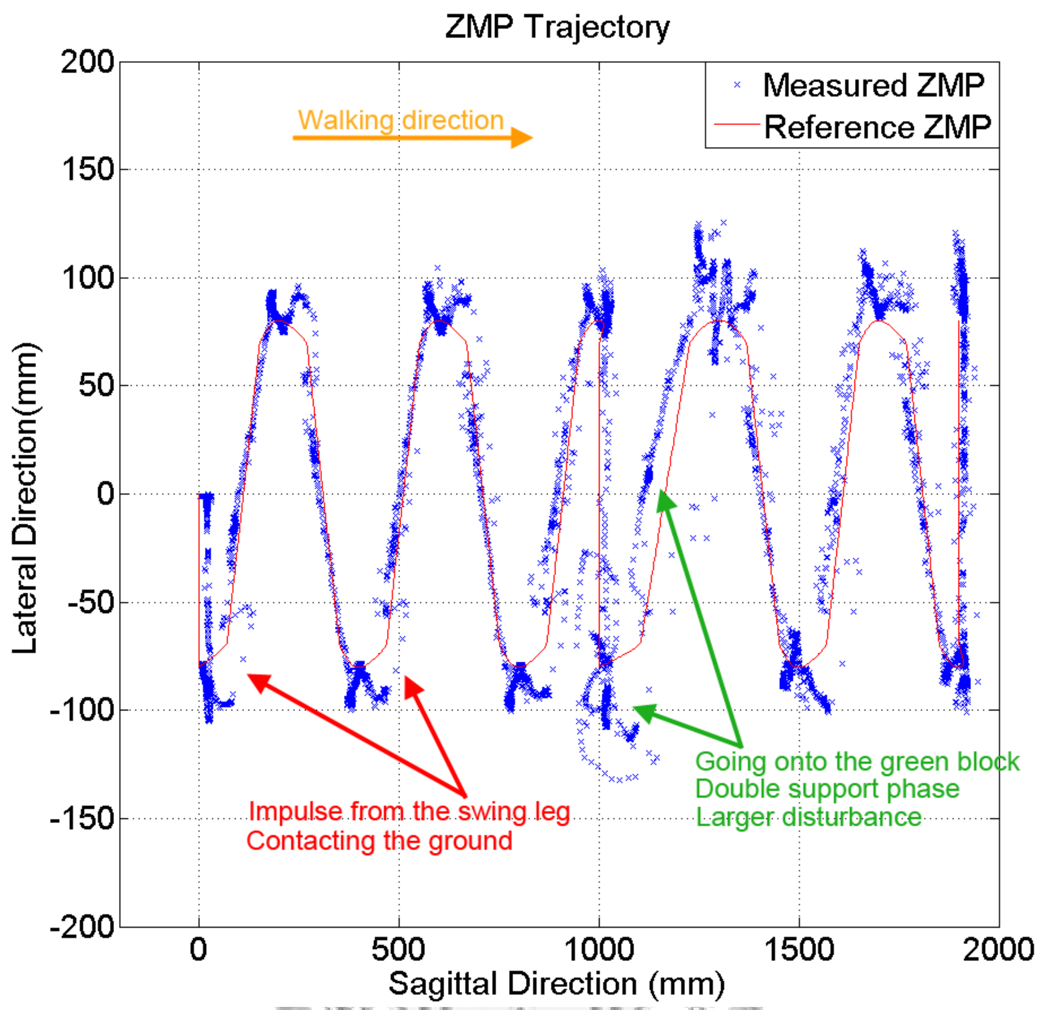


Figure 4-19. The ZMP tracking result in the first simulation

In Figure 4-19, the red line is the reference input ZMP trajectory to the LQSI controller, and the dots are the measured ZMP in ADAMS. In the figure, the deviation of ZMP occurs when the swing leg touches the ground. In each step, after the impulse from the swing leg contacting the ground, the ZMP trajectory is tracked better. In the figure, it is observed that every step has an impulse when the swing leg contacts the ground. This is caused by the penetration of the contact model in ADAMS. In ADAMS simulation, the contacting objects penetrate each other. The penetration in the simulation is unavoidable. Thus, when the swing leg contacts the ground, the stance leg is slightly lower than the ground surface. The impacts in the simulation occurred because of this error of the estimated ground position.

At the position near 500*mm* in the sagittal direction, the robot slows its pace to prepare for stepping onto the green block. And then the robot goes onto the block successfully. Finally, the robot recovers its COG height and the first simulation ends.

In the following is the second simulation. The robot must walk through a complex environment. This simulation is designed to test the performance of the proposed LQSI controller and the whole body inverse kinematics solver in ADAMS. The left part of Figure 4-20 shows the scene which the robot walks through in the second simulation. Firstly, the robot walks forward to the slope and then the robot rotates its swing foot to go onto the slope. After going onto the slope, the robot starts to prepare to walk onto the steps for going down. After doing this, the robot stops between the two square columns. The right part of Figure 4-20 shows 3D trajectories of COG and ZMP. The ZMP trajectory is generated along the footsteps that the robot follows. It swaps between left and right as the robot changes the support leg. The input COG height is also changing with the height of the surface of the scene. The horizontal COG trajectory (in x and y directions) is solved with the proposed LQSI controller.

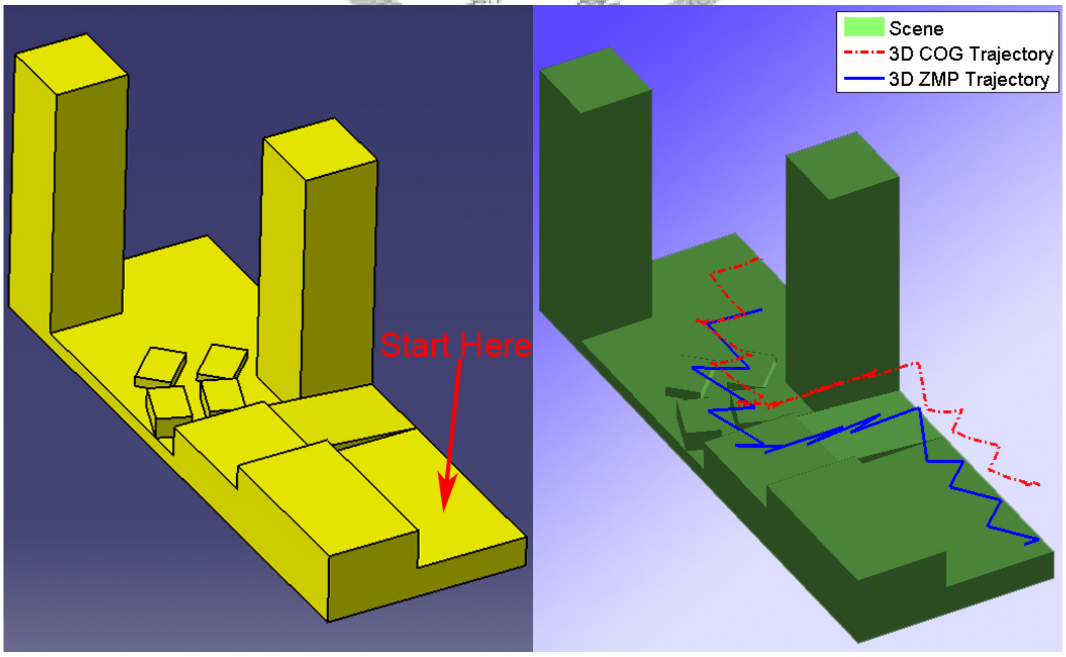


Figure 4-20. The scene and 3D COG/ZMP trajectories in the second simulation

Using the 3D COG trajectory and the trajectories of all end-effectors, all joint trajectories of the robot are generated and inputted to the robot model in ADAMS. The results shows that the robot can walk through the scene stably using the trajectory generated with the proposed LQSI controller as shown in Figure 4-21.

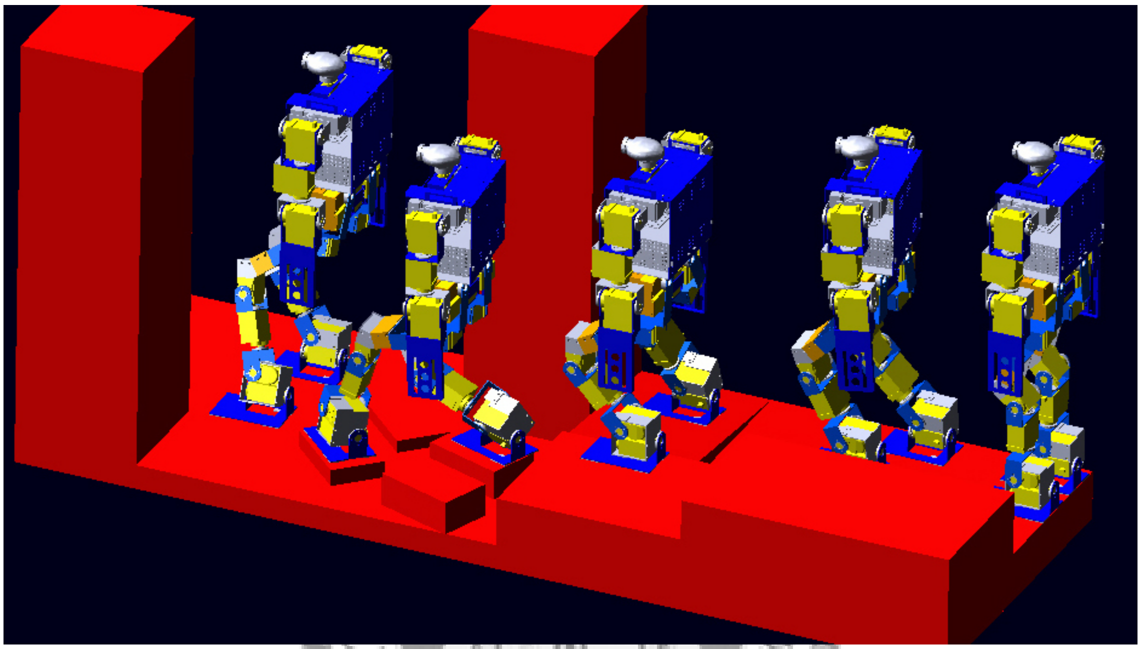


Figure 4-21. Dynamic walking with LQSI controller in the second simulation

4.5.3 Comparison of LQSI and Preview Controller Using ADAMS

In this section, the ZMP tracking performance of the LQSI controller and the preview controller are compared with different COG height (C_z) input. The COG height trajectory is designed as Eq. (4-59). And the robot is walking on a flat floor without any obstacle.

$$C_z = 200 + A_m sin(\omega t), \qquad A_m = 30, \qquad \omega = 0 \sim 16.713 \qquad (4\text{-}59)$$

ω in Eq. (4-59) is tuned to make the maximum C_z acceleration vary from 0 to $8380 mm/s^2$.

The simulation in the section is mainly used to verify the performance of the controllers under varying COG height trajectory. Due to the speed limitation of real robot and in order to protect the robots from damage, large ω cannot be used in real experiments. Thus in this section, only simulations are used to show the control performance of the LQSI controller under rapid COG height changing. The estimated maximum achievable ω of the robot in real experiment is about 2 rad/s.

There are two reasons why the angular velocity is chosen to be the factor to compare the ZMP tracking performance. The first one is when we fix the ω to be 2π/0.8 (the robot walks 0.8 seconds per step), the robot reaches its joint limit at $A_m = 55$. At this time, the maximum C_z acceleration is just 2171.3mm/s^2. This cannot change the dynamics of Eqns. (4-3) and (4-4) too much from the average one. The second reason is that the acceleration term in Eqns. (4-3) and (4-4) plays an important role. If we set the C_z acceleration larger, the variation of the dynamics of the equation will also become larger. In ADAMS, the robot walks as shown in Figure 4-22, and the results are shown in Figure 4-23 to Figure 4-25.

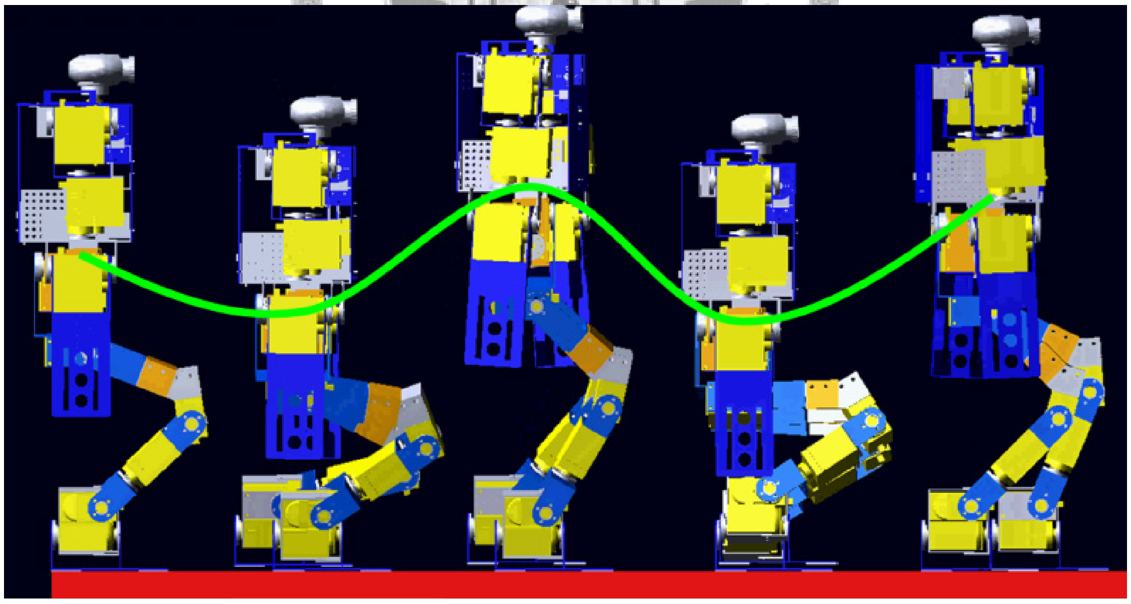


Figure 4-22. Walking on a plane with varying C_z

Figure 4-23 shows the results that the robot walks without COG height change. In the figure, the LQSI controller and the preview controller have almost the same ZMP tracking performance. Figure 4-24 shows that the robot walks with COG height change, the maximum C_z acceleration is $6000mm/s^2$. Because of the modeling error of the preview control, the error of ZMP tracking is larger. Figure 4-25 shows the relationship between the average ZMP tracking error and the maximum C_z acceleration.

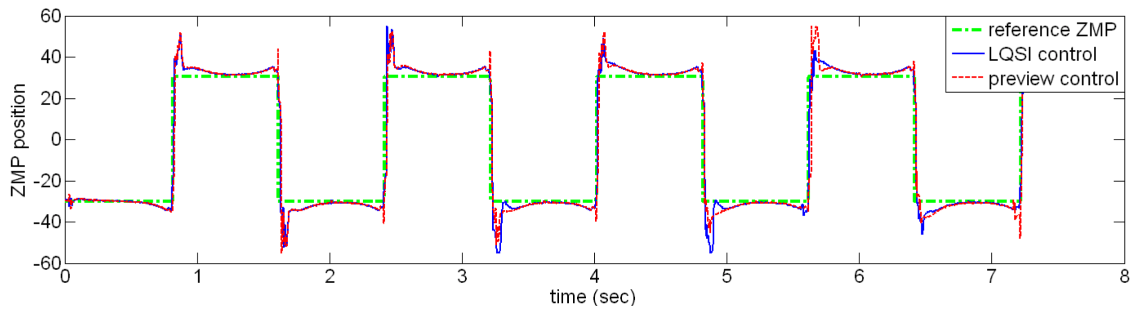


Figure 4-23. Angular velocity $\omega = 0$

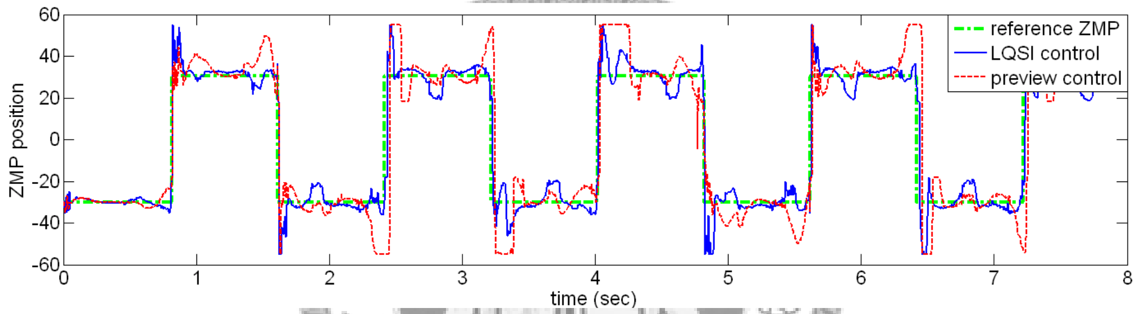


Figure 4-24. Angular velocity $\omega = 4.5\pi$

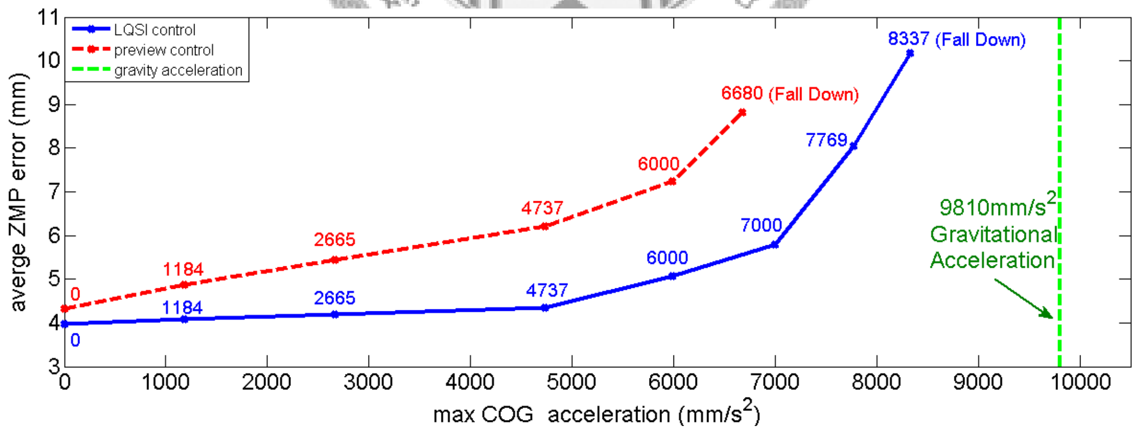


Figure 4-25. Average ZMP error under different maximum C_z acceleration

In the figure, the average ZMP tracking error grows as the max C_z acceleration grows. The value grows faster as the C_z acceleration approaches to the gravitational acceleration. With the LQSI controller, the robot falls down when the maximum C_z acceleration is larger than 8337 mm/s^2. On the other hand, with the preview controller, the robot falls down when the maximum C_z acceleration is larger than 6680 mm/s^2. By considering the C_z change, the LQSI controller can sustain larger maximum C_z acceleration and has smaller average ZMP tracking error than the preview controller.

4.5.4 Computation Complexity of the LQSI Controller

In this section, the LQSI controller is implemented on a PC with Intel Core2 Duo E8400 processor, 32bit Windows XP. The C++ compiler is Microsoft Visual Studio 2005. Optimal controllers have a common problem that the feed-forward gain and the feedback gain in the future should be recalculated if any state matrix changes. Since the calculation is a recursive process, the computation load is too heavy for online computation on low cost micro-controllers or embedded systems. However, due to the improvement of technology, implementation of online optimal control becomes possible on PC-based solutions. In Table 4-2, the average computation time of the LQSI controller and the preview controller are compared. We use inline assembly to construct a matrix operation library to optimize add, minus and multiply operations of both controllers. And the matrix inversion is done by using open source library CLAPACK [155].

Table 4-2: Average computation time of the controllers using C++

	LQSI	Preview
Initialization/ memory allocation	Only once	Only once
	0.68ms	0.06ms
Calculation of feedback and feed-forward gains	Re-plan for change of state matrix	Only once
	0.86ms	0.038ms

In Table 4-2, the minimum required future input lengths are used, 1.9 seconds for the LQSI controller and 1.6 seconds for the preview controller. We choose the sampling time of the control system as 5ms to test the computation speed of the control algorithms. The computation of LQSI controller is much heavier than the preview controller. All the control gains of the preview controller need to be calculated only once because it is a time invariant controller. On the other hand, from Eqns. (4-32) and (4-33), both S_k and v_k are relevant to the state matrices and only v_k is relevant to the reference ZMP input. If the reference ZMP input is changed in lateral or sagittal directions, v_k will be changed. In this case, no re-plan of control gains is needed because the state matrices and the preview gain are calculated before. But if we change the COG height or the ZMP height trajectory, all the state matrices and the control gains should be recalculated and the time required is 0.86ms. Table 4-3 shows the comparison of computation time of re-planning for different situations among learning algorithm, the LQSI controller, and the preview controller.

Table 4-3: Comparison of computation time of re-planning for different situations

	Learning	LQSI	Preview
Need of Training/database	Yes	No	No
Real-time on micro-controller or embedded systems	No	No	Yes
Real-time on PC/notebook	No	Yes	Yes
Memory usage (double precision)	100~10000Kbytes or more	220Kbytes	0.4Kbytes
Order of time for initialization	Offline training	< 1ms	< 0.1ms
Order of time for re-planning	Offline training	< 1ms	Doesn't need re-planning
C_z changeable	Yes	Yes	Slightly changeable

Although the C_z is changeable when using learning algorithm to generate walking patterns, the computation is too heavy to be real-time. The memory usage is also huge. The LQSI controller can be regarded as the time varying version of the preview controller. Its computation load and memory usage is much larger than the preview control but can still be processed within 1ms for each re-planning. From the implementation result using C++, real-time control with the LQSI controller can be achieved on PC or notebook. Although the preview control requires less memory and computation power, its capability to handle variable C_z is limited.

4.6 Summary

In this chapter, LQSI control is proposed to provide a more versatile walking pattern generator. It solves the sagittal and lateral COG trajectories with the input trajectories, vertical COG and ZMP in a 3D space. With the arbitrarily assigned C_z trajectory, the robot does not have to stop walking in order to change its COG height. Real-time dynamic walking with arbitrary COG height trajectory is thereby achieved.

While using the calculated feedback gains in the LQSI algorithm, we need to know the values of the states (COG position, COG velocity, and ZMP position). The present ZMP position can be calculated by force sensors on the robot's legs, but the COG states must be estimated. In this chapter, we have discussed the COG state values obtained directly from the iterations of the state-space model, where the only feedback is the ZMP position. In future investigations, observers or estimators should be designed for more accurate COG states in order to improve the performance of ZMP tracking.

The LQSI controller can also be used for reducing the torques on the robot's legs, especially on knee joints because the robot does not have to bend its knees to keep the COG height constant. In chapter 5, a method that can optimize the COG height trajectory is proposed and verified.

Chapter 5 Optimized 3D COG trajectory Generation

Many recent methods have achieved stable walking patterns for biped robots, but discussions about their optimization are relatively scarce. In this chapter, a COG trajectory optimization method is proposed to minimize the weighted cost of torque and joint limit (performance index) of the robot. The COG height trajectory is updated and optimized by calculating the derivative of the performance index with respect to COG height. To achieve this, the derivative of the Newton-Euler dynamics is also derived. The proposed LQSI controller in chapter 4 can optimize the COG trajectories in sagittal and lateral directions with the updated COG height trajectory in each COG height training iteration. The COG height training results show that the proposed method can reduce the joint torque of the robot when walking, and generate a more human-like walking pattern.

5.1 Introduction

In recent years, the humanoid robot has become a popular research topic. Such robots use more complex multi-axis control systems than mobile robots do. They also have to overcome the problem of stability while walking; they are, however, adaptable to many more terrain types. Many different approaches have been proposed to achieve stabile walking. Some are successful and well known, such as ASIMO [26][110][126], the HRP series [56][57][59], WABIAN [40], PETMAN, and HUBO [16]. Their research topics cover a very broad area, including motion control and trajectory generation [39], path and motion planning [1][63], human-robot interaction [6][143], and mechanical design [1][98] among many interesting topics.

Many researchers have devised walking pattern generation algorithms. The most used methods employ the COG/ZMP (Center of Gravity/Zero Moment Point) equations and inverted pendulum model. Early researchers focused on constant C_z (linear inverted pendulum model or cart-table model) for walking pattern generation, and the preview control method [47][48] is widely used in many studies to solve linear inverted pendulum problems.

More and more methods have recently been proposed that involve the use of changeable C_z to generate the walking pattern, as mentioned in section 4.1. The non-constant C_z trajectory means that the inverted pendulum problem is nonlinear. The solution to the varying C_z problem is solved, but the optimization of the C_z trajectory remains relatively little discussed. For the same robot model, if the C_z trajectory is well defined, the gait of the robot will be more human-like and energy-saving than the walking gaits with arbitrarily assigned C_z trajectories. In this dissertation, LQSI controller in chapter 4 is used to solve the nonlinear inverted pendulum problem, and the C_z trajectory is optimized by minimizing the performance index with the derivative of the Newton-Euler dynamics in this chapter. Note that, there are many aspects to make the walking pattern more natural and more energy-saving. The proposed method is used for the walking patterns for non-passive robots since the energy consumptions of each active-controlled robot joints are directly relative to the currents passing through the motors. And the currents passing through the motors are directly relative to the joint torques.

The rest of this chapter is organized as follows. Section 5.2 introduces the performance index and optimization procedure. In section 5.3, the derivative of the basic vectors and the Newton-Euler dynamics with respect to C_z is derived. All the required parameters used to optimize the performance index in section 5.2 can be found in section 5.3. Section 5.4, presents and discusses the parameter setting and training. Finally, section 5.5 summarizes this chapter and shows the conclusions and future works.

5.2 Goal and Procedure of Optimization

The optimization of COG trajectory in sagittal and lateral (x and y) directions in the world coordinates is described in chapter 4. In order to achieve optimized 3D COG trajectory generation for more natural and energy-saving walking pattern, the optimization of COG trajectory in z direction is discussed in the following.

5.2.1 Performance Index

The performance index for COG trajectory optimization in z direction is chosen as Eq. (5-1). It is minimized by updating the C_z trajectory using the method proposed in this chapter.

$$P_k = \frac{1}{2}\sum_{j=1}^{n} W_{\tau,j}\tau_j^2 + \frac{1}{2}\sum_{j=1}^{n} W_{\theta,j}H_j^2 \tag{5-1}$$

The performance index is non-dimensionalized with the weightings $W_{\tau,j}$ and $W_{\theta,j}$, where $W_{\tau,j}$ denotes the weighting of the joint torque cost of the jth joint, $W_{\theta,j}$ denotes the weighting of the joint limit cost of the jth joint, τ denotes the joint torque and the H^2 denotes the joint limit cost, as Eq. (5-2). The suffix k denotes the kth sampling point of the C_z trajectory.

$$H_j^2 = \frac{1}{\left(\theta_{Max,j} - \theta_j\right)^2\left(\theta_j - \theta_{min,j}\right)^2} \tag{5-2}$$

$\theta_{Max,j}$ and $\theta_{min,j}$ denote the upper and lower joint limits of the jth joint. When the joint angle approaches the joint limit, H^2 will become larger and larger to prevent the robot from reaching its joint limit.

5.2.2 Optimization Procedure

To minimize the performance index by updating C_z trajectory, its derivative with respect to C_z must be found, as shown in Eq. (5-3).

$$\frac{dP_k}{dC_z} = \sum_{j=1}^{n} W_{\tau,j}\frac{d\tau_j}{dC_z}\tau_j + \frac{1}{2}\sum_{j=1}^{n} W_{\theta,j}\frac{dH_j}{dC_z}H_j \tag{5-3}$$

To calculate Eq. (5-3), the Newton-Euler dynamics and its derivative with respect to C_z are required, as shown in section 2.4 and section 5.3. The COG-height-updating equation is shown as

$$C_{z,next} = C_{z,current} - \eta\frac{dP}{dC_z} \tag{5-4}$$

where $C_{z,next}$ and $C_{z,current}$ denote the COG height trajectory in the next and current iteration, η denotes the learning rate, and dP/dC_z denotes the trajectory of dP_k/dC_z for all sampling points. The COG height optimization procedure is shown in Figure 5-1.

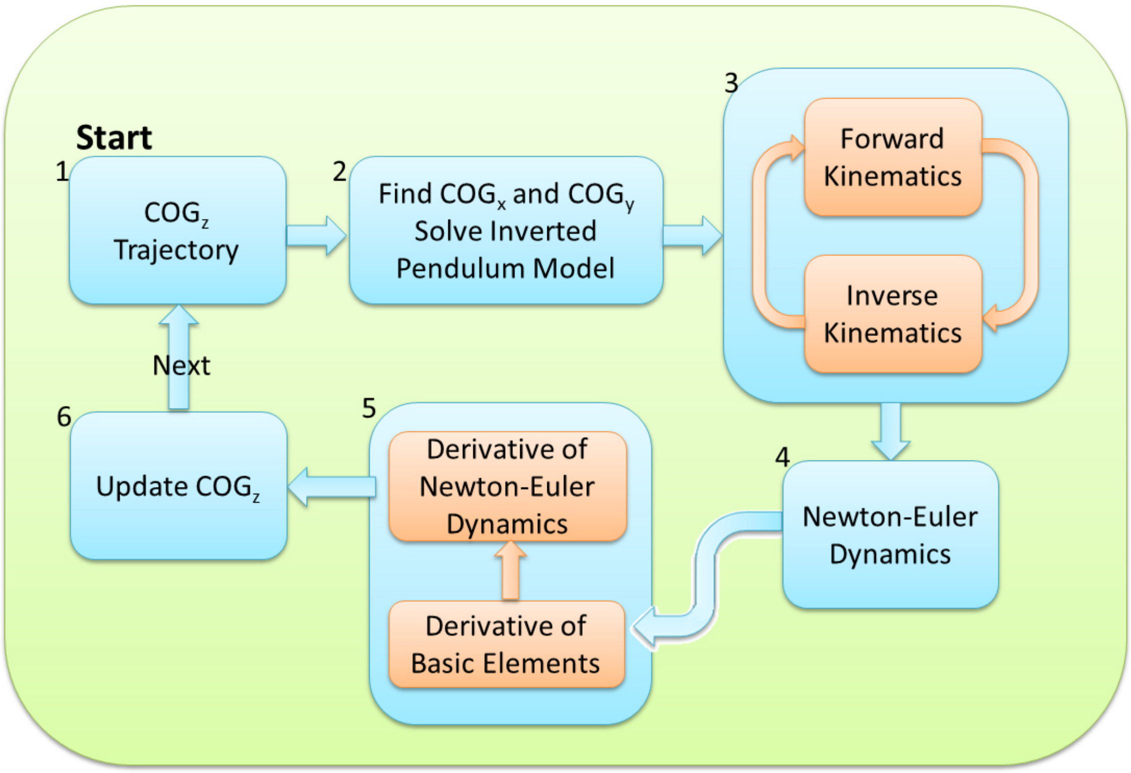

Start
1
COG$_z$ Trajectory
2
Find COG$_x$ and COG$_y$ Solve Inverted Pendulum Model
3
Forward Kinematics
Inverse Kinematics
4
Newton-Euler Dynamics
5
Derivative of Newton-Euler Dynamics
Derivative of Basic Elements
6
Update COG$_z$
Next

Figure 5-1. Procedure for optimizing COG height trajectory

The COG-height-optimization procedure starts with an original C_z trajectory. It is simply assigned as a constant. The second step is to solve the COG trajectory in the sagittal (x) and lateral (y) directions, using the proposed LQSI controller. At this step, the COG trajectories of all three axes (x, y, and z) are all available. In the third step, the 3D COG trajectory and the trajectories of the end-effectors (such as that of the swing leg) are used to solve IK and thus find all the joint-angle trajectories for walking. In the fourth step, with the joint-angle trajectories known, Newton-Euler dynamics [21] can be found as Eqns. (2-19)-(2-28). After calculating the Newton-Euler dynamics, the concepts of Jacobian matrix and pseudoinverse are used to find the derivatives of the basic variables and Newton-Euler dynamics with respect to C_z. In the final step, the derivative of the performance index can be calculated with the variables calculated in steps 2-5 above, and C_z trajectory is updated as Eq. (5-4). Because numerical differentiation is used in step 5, the input trajectories of the end-effectors of the robot for solving IK in step 3 must be smooth and differentiable. To achieve this, all interpolations of the end-effector trajectories are performed using polynomial functions.

5.3 The Derivatives with Respect to COG Height

To find the derivatives, it is first necessary to find the derivatives of the basic elements. Chapter 4 describes the use of the proposed LQSI solver and inverted pendulum model, of the COG optimized trajectories in the sagittal and lateral directions; these need to be known first as they will change with the C_z input to the solver. Section 5.3.1 describes the change of the horizontal COG (sagittal and lateral) trajectory generated by LQSI controller with respect to the change of the vertical COG trajectory (C_z trajectory). The derivatives of the joint z-axis vector z and the link vector r are derived in section 5.3.2. The Newton-Euler dynamics is also described in chapter 2.

These results allow the calculation of the derivatives of Newton-Euler dynamics with respect to C_z change, as described in this section.

5.3.1 Horizontal COG Change with Respect to Vertical COG Change

As shown in chapter 4, Eqns. (4-24)–(4-35) enable the generation of the COG walking pattern in both sagittal and lateral directions. Based on the results from chapter 4, in order to find the deviation under unit C_z change, a small constant deviation is added to the original C_z trajectory, as shown in Eq. (5-5).

$$C_{z,perturbed}(t) = C_{z,original}(t) + \Delta C_z(t) \tag{5-5}$$

Given the perturbed and original C_z trajectories, two sets of sagittal and lateral COG trajectories can be generated. The deviation under unit C_z change of the COG trajectories in sagittal and lateral direction can be found by comparing the perturbed and the original trajectories, as shown in Eqns. (5-6) and (5-7).

$$\frac{dC_x}{dC_z} = \frac{1}{\Delta C_z}\left(C_{x,perturbed} - C_{x,original}\right) \tag{5-6}$$

$$\frac{dC_y}{dC_z} = \frac{1}{\Delta C_z}\left(C_{y,perturbed} - C_{y,original}\right) \tag{5-7}$$

To find the all the derivatives with respect to COG position, it is important to find the joint angle change of all joints with respect to the COG position change, and this is accomplished by using the pseudoinverse matrix, as shown in Eqns. (5-8)–(5-10).

$$J_{Cx}^{+} = J^{+}[0 \quad 0 \quad \cdots \quad 0 \quad 1 \quad 0 \quad 0]^T \tag{5-8}$$

$$J_{Cy}^{+} = J^{+}[0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 1 \quad 0]^T \tag{5-9}$$

$$J_{Cz}^{+} = J^{+}[0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 0 \quad 1]^T \tag{5-10}$$

Eqns. (5-8)–(5-10) show the joint angle change of all joints for unit COG position change in each direction. These intermediate variables are important for the derivations described in the following sections in this chapter.

The deviation of joint angles for unit C_z change is the resultant of the effects of the COG position change in all three directions, as shown in Eq. (5-11).

$$\frac{d\theta}{dC_z} = J^+ \begin{bmatrix} 0 & 0 & \cdots & 0 & \frac{dC_x}{dC_z} & \frac{dC_y}{dC_z} & 1 \end{bmatrix}^T \tag{5-11}$$

Using Eqns. (5-8)–(5-10), (5-11) can be rewritten as Eq. (5-12).

$$\frac{d\theta}{dC_z} = \frac{dC_x}{dC_z} J_{C_x}^+ + \frac{dC_y}{dC_z} J_{C_y}^+ + J_{C_z}^+ \tag{5-12}$$

Eq. (5-12) shows that the derivative of the joint angles is the summation of the COG deviation in each direction multiplied by the corresponding column of the pseudoinverse matrix.

5.3.2 Derivatives of Basic Vectors

The time series of joint angles can be obtained by using the IK solver. To find the joint speeds and joint accelerations, fourth order numerical differentiation is used, as shown in Eq. (5-13).

$$\frac{d}{dt} f_p = \frac{f_{p-2} - 8f_{p-1} + 8f_{p+1} - f_{p+2}}{12h} + O(h^4) \tag{5-13}$$

f denotes an arbitrary variable, h denotes the time interval and p denotes the pth sample in the series. Eq. (5-13) is the numerical method used to find the velocity and the acceleration of the joint angle, as shown in Eqns. (5-14) and (5-15), where Δt denotes the sampling time of the joint trajectory.

$$\dot{\theta}_p \approx \frac{\theta_{p-2} - 8\theta_{p-1} + 8\theta_{p+1} - \theta_{p+2}}{12\Delta t} = V(\theta_p) \tag{5-14}$$

$$\ddot{\theta}_p \approx \frac{\dot{\theta}_{p-2} - 8\dot{\theta}_{p-1} + 8\dot{\theta}_{p+1} - \dot{\theta}_{p+2}}{12\Delta t} = A(\theta_p) \tag{5-15}$$

Eq. (5-12) is an n-by-1 column vector and is rewritten as Eq. (5-16).

$$\frac{d\theta}{dC_z} = \begin{bmatrix} \frac{d\theta_1}{dC_z} & \frac{d\theta_2}{dC_z} & \cdots & \frac{d\theta_i}{dC_z} & \cdots & \frac{d\theta_n}{dC_z} \end{bmatrix}^T \tag{5-16}$$

$d\theta_i/dC_z$ denotes the *i*th element of $d\theta/dC_z$, it is the *i*th joint angle change under unit C_z change. Because the joint velocity is approximated as a linear combination of the joint angle, the derivative of joint velocity with respect to the C_z can be written as Eq. (5-17). It can also be rewritten as Eq. (5-18).

$$\frac{d\dot{\theta}_p}{dC_z} \approx \frac{1}{12\Delta t}\left(\frac{d\theta_{p-2}}{dC_z} - 8\frac{d\theta_{p-1}}{dC_z} + 8\frac{d\theta_{p+1}}{dC_z} - \frac{d\theta_{p+2}}{dC_z}\right) \tag{5-17}$$

$$\frac{d\dot{\theta}}{dC_z} = \begin{bmatrix} \frac{d\dot{\theta}_1}{dC_z} & \frac{d\dot{\theta}_2}{dC_z} & \cdots & \frac{d\dot{\theta}_n}{dC_z} \end{bmatrix}^T \tag{5-18}$$

From Eq. (5-16), the total derivatives of z and r vectors can be expressed as Eqns. (5-19) and (5-20).

$$\begin{gathered} \frac{d\vec{z}_i}{dC_z} = \sum_{k=1}^{i-1}\left(\frac{d\vec{z}_i}{d\theta_k}\frac{d\theta_k}{dC_z}\right) = \sum_{k=1}^{i-1}\left((\vec{z}_k \times \vec{z}_i)\frac{d\theta_k}{dC_z}\right) \\ \frac{d\vec{z}_i}{d\theta_k} = \vec{0}, \quad k \in \{i, i+1, \cdots, n\} \end{gathered} \tag{5-19}$$

$$\begin{gathered} \frac{d\vec{r}_i}{dC_z} = \sum_{k=1}^{i}\left(\frac{d\vec{r}_i}{d\theta_k}\frac{d\theta_k}{dC_z}\right) = \sum_{k=1}^{i}\left((\vec{z}_k \times \vec{r}_i)\frac{d\theta_k}{dC_z}\right) \\ \frac{d\vec{r}_i}{d\theta_k} = \vec{0}, \quad k \in \{i+1, i+2, \cdots, n\} \end{gathered} \tag{5-20}$$

As shown in Figure 2-4, the rotation of joint i changes only the position and orientation of the joint i+1 to joint n. All joints movements from joint 1 to joint i–1 change the status of joint i. Thus in Eq. (5-19), the summation starts at joint 1 and ends at joint i–1. This is why the total, rather than the partial, derivative is used here. The cross-product with $\vec{z}_k$ means that the vector changes under unit rotation along the $\vec{z}_k$ vector. In Eq. (5-20), the summation stops at joint i because the vector $\vec{r}_i$ is the position vector from joint i to joint i+1. It will be affected by the rotation of the ith joint.

5.3.3 Derivatives of Newton-Euler Dynamics

With the derivations in section 2.4, all the variables of in both forward and backward calculations can be found. The derivatives of angular velocity, velocity, angular acceleration, and acceleration with respect to the C_z change can be found using chain rule, as shown in Eqns. (5-21)–(5-24).

$$\frac{d\vec{\omega}_i}{dC_z} = \frac{d\vec{\omega}_{i-1}}{dC_z} + \frac{d\vec{z}_i}{dC_z}\dot{\theta}_i + \vec{z}_i\frac{d\dot{\theta}_i}{dC_z} \tag{5-21}$$

$$\frac{d\vec{v}_{i+1}}{dC_z} = \frac{d\vec{v}_i}{dC_z} + \frac{d\vec{\omega}_i}{dC_z} \times \vec{r}_i + \vec{\omega}_i \times \frac{d\vec{r}_i}{dC_z} \tag{5-22}$$

$$\begin{aligned}\frac{d\vec{\alpha}_i}{dC_z} = \frac{d\vec{\alpha}_{i-1}}{dC_z} + \frac{d\vec{z}_i}{dC_z}\ddot{\theta}_i + \vec{z}_i\frac{d\ddot{\theta}_i}{dC_z} + \\ \frac{d\vec{\omega}_i}{dC_z} \times \vec{z}_i\dot{\theta}_i + \vec{\omega}_i \times \frac{d\vec{z}_i}{dC_z}\dot{\theta}_i + \vec{\omega}_i \times \vec{z}_i\frac{d\dot{\theta}_i}{dC_z}\end{aligned} \tag{5-23}$$

$$\begin{aligned}\frac{d\vec{a}_{i+1}}{dC_z} = \frac{d\vec{a}_i}{dC_z} + \frac{d\vec{\alpha}_i}{dC_z} \times \vec{r}_i + \vec{\alpha}_i \times \frac{d\vec{r}_i}{dC_z} + \frac{d\vec{\omega}_i}{dC_z} \times (\vec{\omega}_i \times \vec{r}_i) + \\ \vec{\omega}_i \times \left(\frac{d\vec{\omega}_i}{dC_z} \times \vec{r}_i\right) + \vec{\omega}_i \times \left(\vec{\omega}_i \times \frac{d\vec{r}_i}{dC_z}\right)\end{aligned} \tag{5-24}$$

From the equations above, the derivatives of force and torque can be found as Eqns. (5-25) and (5-26).

$$\frac{d\vec{f}_{i+1}}{dC_z} = \frac{d\vec{f}_i}{dC_z} + m_i\frac{d\vec{a}_i}{dC_z} \tag{5-25}$$

$$\begin{aligned}\frac{d\vec{\tau}_{i+1}}{dC_z} = \frac{d\vec{\tau}_i}{dC_z} + \frac{dI_i}{dC_z}\vec{\alpha}_i + I_i\frac{d\vec{\alpha}_i}{dC_z} + \frac{d\vec{\omega}_i}{dC_z} \times (I_i\vec{\omega}_i) + \\ \vec{\omega}_i \times \left(\frac{dI_i}{dC_z}\vec{\omega}_i\right) + \vec{\omega}_i \times \left(I_i\frac{d\vec{\omega}_i}{dC_z}\right) - \frac{d\vec{r}_{i\to i}}{dC_z} \times \vec{f}_i - \\ \vec{r}_{i\to i} \times \frac{d\vec{f}_i}{dC_z} + \frac{d\vec{r}_{i\to i+1}}{dC_z} \times \vec{f}_{i+1} + \vec{r}_{i\to i+1} \times \frac{d\vec{f}_{i+1}}{dC_z}\end{aligned} \tag{5-26}$$

Recall Eq. (2-2), to find the derivative of the inertia matrix in world coordinates, the rotation matrix $R_{0,i}$ in the DH homogeneous matrix is rewritten as Eq. (5-27).

$$T_{0,i} = \begin{bmatrix} R_{0,i} & D_{0,i} \\ 0 & 1 \end{bmatrix} \tag{2-2}$$

$$R_{0,i} = R_{0,1}R_{1,2}R_{2,3}\cdots R_{i-1,i} \tag{5-27}$$

where $R_{i-1,i}$ denotes the rotation part in each homogeneous matrix; it represents the rotation from the (i-1)th to the ith joint. Using Eq. (5-27) and recall Eq. (2-32), the inertia matrix represented in the world coordinate can be written as Eq. (5-28).

$$I_i = R_{0,i}I_{0,i}R_{0,i}^T \tag{2-32}$$

$$I_i = R_{0,1}R_{1,2}R_{2,3}\cdots R_{i-1,i}I_{0,i}R_{i-1,i}^T\cdots R_{2,3}^TR_{1,2}^TR_{0,1}^T \tag{5-28}$$

Eq. (5-28) enables the derivative of the inertia matrix in world coordinates to be found with Eqns. (5-29)–(5-31).

$$\frac{dI_i}{dC_z} = \frac{dR_{0,i}}{dC_z} I_{0,i} R_{0,i}^T + R_{0,i} I_{0,i} \frac{dR_{0,i}^T}{dC_z}$$
$$= \sum_{k=1}^{i} \left(\frac{dR_{0,i}}{d\theta_k} \frac{d\theta_k}{dC_z} \right) I_0 R_{0,i}^T + R_{0,i} I_0 \sum_{k=1}^{i} \left(\frac{dR_{0,i}^T}{d\theta_k} \frac{d\theta_k}{dC_z} \right) \tag{5-29}$$

$$\frac{dR_{0,i}}{dC_z} = R_{0,1} R_{1,2} \cdots \frac{dR_{k-1,k}}{d\theta_k} \cdots R_{i-2,i-1} R_{i-1,i} \tag{5-30}$$

$$\frac{dR_{0,i}^T}{dC_z} = R_{i-1,i}^T R_{i-2,i-1}^T \cdots \frac{dR_{k-1,k}^T}{d\theta_k} \cdots R_{1,2}^T R_{0,1}^T \tag{5-31}$$

5.4 Training Results

With the procedure of Figure 5-1, the training starts from a constant C_z trajectory and the parameters of walking are set as Table 5-1.

Table 5-1. Parameters of Walking

Stride length	100 mm
Step height	30 mm
Step time (each stride)	2.0 s
Double support phase (DSP)	0.4 s
Single support phase (SSP)	1.6 s
Sampling time	0.005 s

When the robot is standing on both feet, it is said to be in the *double support phase* (DSP) of its stride, whereas it has only one foot on the ground during the *single support phase* (SSP). Sampling time is the interval between each time point.

The weightings of joint torque and joint limit in DSP and SSP are set as shown in Table 5-2. The directions of pitch, roll, and yaw are defined in Figure 2-4. The proposed robot has three joints in each hip, one in each knee, and two in each ankle. The hip can move in all pitch, roll, and yaw directions, the knee joint only in pitch direction, and the ankle joint can move in both pitch and roll directions.

To achieve a more natural walking pattern, the robot knee joint must not bend too much, so the weightings of the knee joint are larger than those of the other joints under all conditions. In SSP, the torque weighting of the knee joint of stance leg is $0.9w_a$ ($w_a = 0.05$). It is larger and more important than the weightings of the other joints. None of the swing leg joints takes much torque, so their weightings are all $0.1w_a$. In DSP, both legs support the robot, so the torque weighting of each knee, the most important of all the joints, is $0.5w_a$. Because the all the joint limits are the same during both SSP and DSP, the joint-limit weightings remain the same throughout. To allow for the human-like knee-stretching motion, the knee must approach its joint limit, and, in order to prevent a singularity when solving IK, its joint-limit weighting is larger than the weightings for the other joints, at $1.0w_b$ ($w_b = 22.5$).

Table 5-2. Torque and joint-limit weightings for all joints

Axis	DSP (torque)	$w_a = 0.05$	SSP (left support) (torque)	$w_a = 0.05$	SSP (right support) (torque)	$w_a = 0.05$	DSP & SSP (joint limit)	$w_b = 22.5$
Left Hip Yaw	$W_{\tau,1}$	$0.10w_a$	$W_{\tau,1}$	0.10wa	$W_{\tau,1}$	0.10wa	$W_{\theta,1}$	$0.10w_b$
Left Hip Roll	$W_{\tau,2}$	$0.30w_a$	$W_{\tau,2}$	0.40wa	$W_{\tau,2}$	0.10wa	$W_{\theta,2}$	$0.15w_b$
Left Hip Pitch	$W_{\tau,3}$	$0.30w_a$	$W_{\tau,3}$	0.70wa	$W_{\tau,3}$	0.10wa	$W_{\theta,3}$	$0.20w_b$
Left Knee Pitch	$W_{\tau,4}$	$0.50w_a$	$W_{\tau,4}$	0.90wa	$W_{\tau,4}$	0.10wa	$W_{\theta,4}$	$1.00w_b$
Left Ankle Pitch	$W_{\tau,5}$	$0.30w_a$	$W_{\tau,5}$	0.70wa	$W_{\tau,5}$	0.10wa	$W_{\theta,5}$	$0.20w_b$
Left Ankle Roll	$W_{\tau,6}$	$0.30w_a$	$W_{\tau,6}$	0.40wa	$W_{\tau,6}$	0.10wa	$W_{\theta,6}$	$0.15w_b$
Right Hip Yaw	$W_{\tau,7}$	$0.10w_a$	$W_{\tau,7}$	0.10wa	$W_{\tau,7}$	0.10wa	$W_{\theta,7}$	$0.10w_b$
Right Hip Roll	$W_{\tau,8}$	$0.30w_a$	$W_{\tau,8}$	0.10wa	$W_{\tau,8}$	0.40wa	$W_{\theta,8}$	$0.15w_b$
Right Hip Pitch	$W_{\tau,9}$	$0.30w_a$	$W_{\tau,9}$	0.10wa	$W_{\tau,9}$	0.70wa	$W_{\theta,9}$	$0.20w_b$
Right Knee Pitch	$W_{\tau,10}$	$0.50w_a$	$W_{\tau,10}$	0.10wa	$W_{\tau,10}$	0.90wa	$W_{\theta,10}$	$1.00w_b$
Right Ankle Pitch	$W_{\tau,11}$	$0.30w_a$	$W_{\tau,11}$	$0.10w_a$	$W_{\tau,11}$	$0.70w_a$	$W_{\theta,11}$	$0.20w_b$
Right Ankle Roll	$W_{\tau,12}$	$0.30w_a$	$W_{\tau,12}$	$0.10w_a$	$W_{\tau,12}$	$0.40w_a$	$W_{\theta,12}$	$0.15w_b$

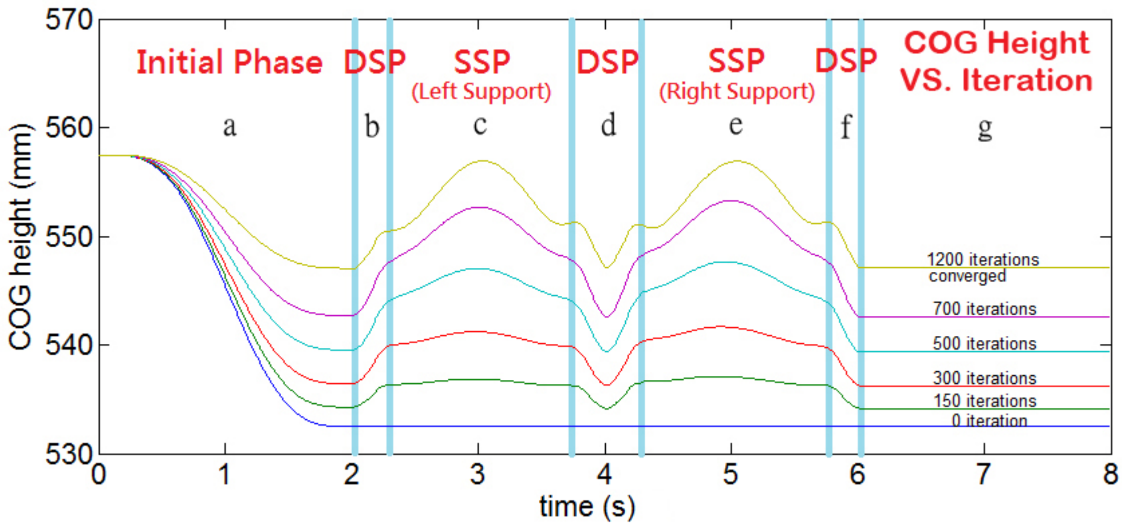


Figure 5-2. The COG height training results

The training result is shown in Figure 5-2. Training occurs over 2 ~ 6 s, during which two strides are taken (one with each leg). The C_z curve converges at the 1200th iteration (cost change rate is less than 0.00001%). The plot of the total cost to the number of iterations is shown in Figure 5-3.

The C_z trajectory can be divided into several regions, described as follows:

a. Initial phase: the robot moves all joints from home position to the initial configuration.
b. DSP: the robot switches from initial configuration to left support phase.
c. SSP: the left leg supports the robot and the right leg is the swing leg.
d. DSP: the robot switches from left support to right support.
e. SSP: the right leg supports the robot and the left leg is the swing leg.
f. DSP: the robot switches from right support to double support, preparing to stop walking.
g. Ending phase: the robot stops walking and the iteration ends.

In Figure 5-2, the C_z trajectory rises as the robot raises its leg in SSP. In DSP, the robot is shifting from left to right (or right to left) support. In these two strides, although the robot lacks a toe mechanism, the training result shows a COG height trajectory similar to a human's [33][67][149]. The proposed algorithm can be used on humanoid robots with toe and heel mechanisms in the future. With toe and heel mechanisms, the joint-limit cost will be smaller during the DSP of the walking period. This will make the trained C_z trajectory smoother and higher during DSP.

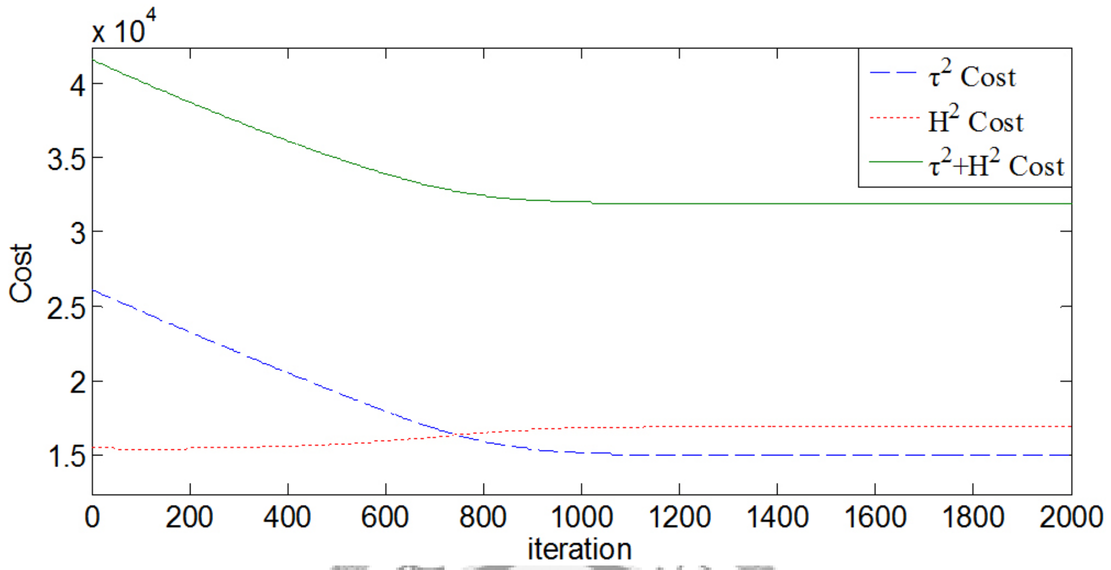


Figure 5-3. Cost against iterations

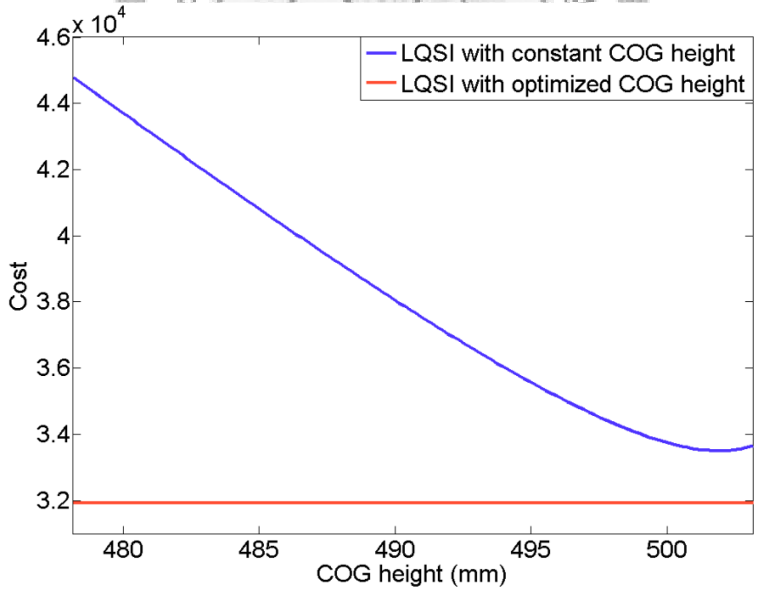


Figure 5-4. non-dimensionalized cost with different constant COG height

In Figure 5-3, τ^2 cost and H^2 cost denote the summation of the torque cost and the joint-limit cost for each iteration. $\tau^2 + H^2$ cost is the total cost. It converges at the 1200th iteration. To reduce torque cost, the robot must stretch its leg during walking, but as this means it must approach its joint limit, the cost of joint limit increases. The torque cost of the joints is directly relative to the C_z trajectory and the joint angles of the robot. The straighter the legs when walking, the smaller the torque cost will be. Figure 5-4 shows the non-dimensionalized cost of the walking pattern with different constant input C_z trajectory.

In Figure 5-4, the red line shows the cost when the robot walks with the optimized C_z trajectory. The blue line shows the total cost using constant C_z trajectory. The total cost reduces as the constant C_z value increases in the beginning. When the constant C_z value exceeds 500mm, some joints of the robot reach their limits. This causes the rapid increment of the joint limit cost, thus the total cost become higher although the torque cost becomes lower, as shown in Figure 5-5. Compared with the constant C_z trajectory, the optimized C_z trajectory can keep higher position and have lower cost. In Figure 5-4, the average height of the optimized COG trajectory is 507.3mm and results in the cost value of the red line. The blue curve stops at 503mm constant C_z since the robot reaches the joint limit.

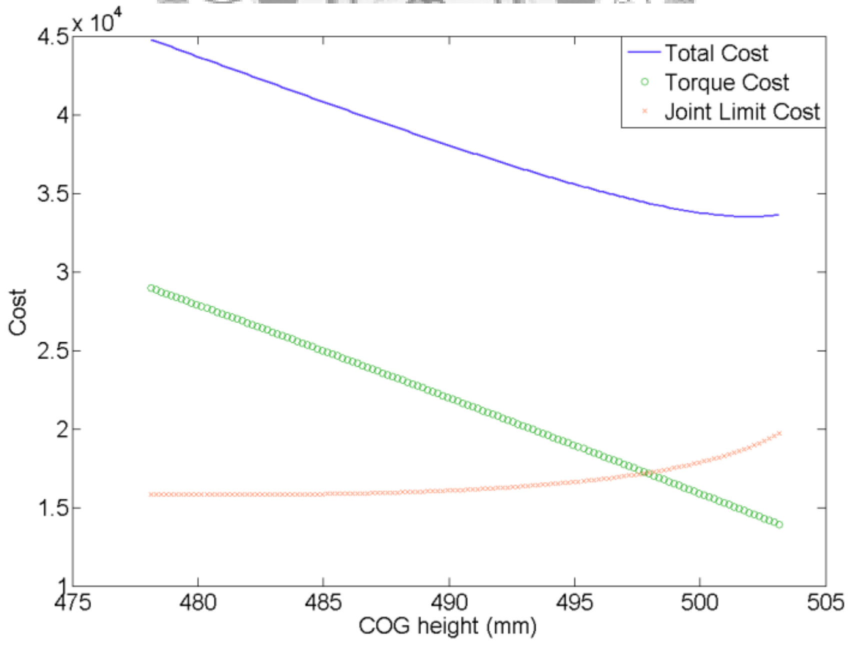


Figure 5-5. Torque, joint limit, and total cost with different constant C_z

In the following, the joint angle and joint torque trajectories of the robot are discussed. The trajectories are shown in Figure 5-6–Figure 5-17.

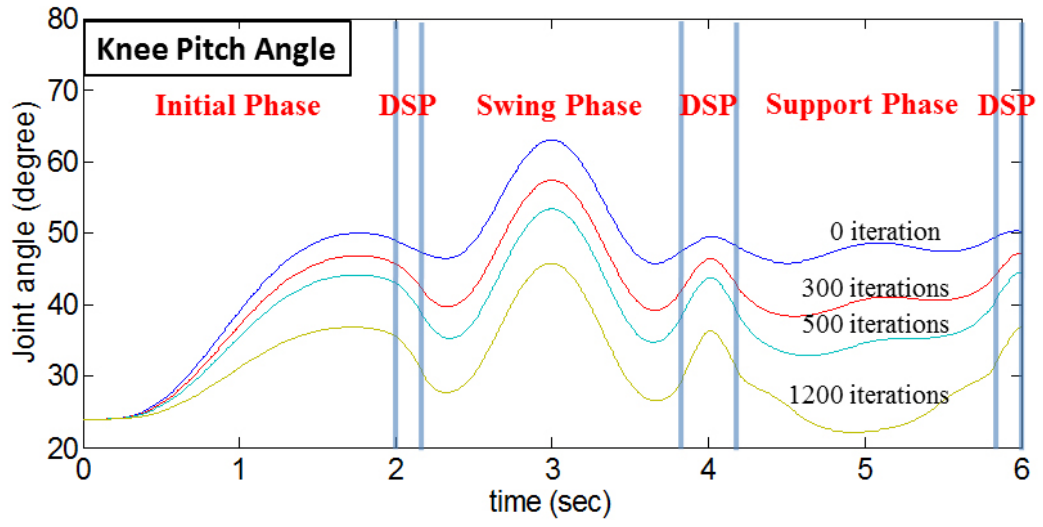


Figure 5-6. Angle trajectory of the knee pitch joint

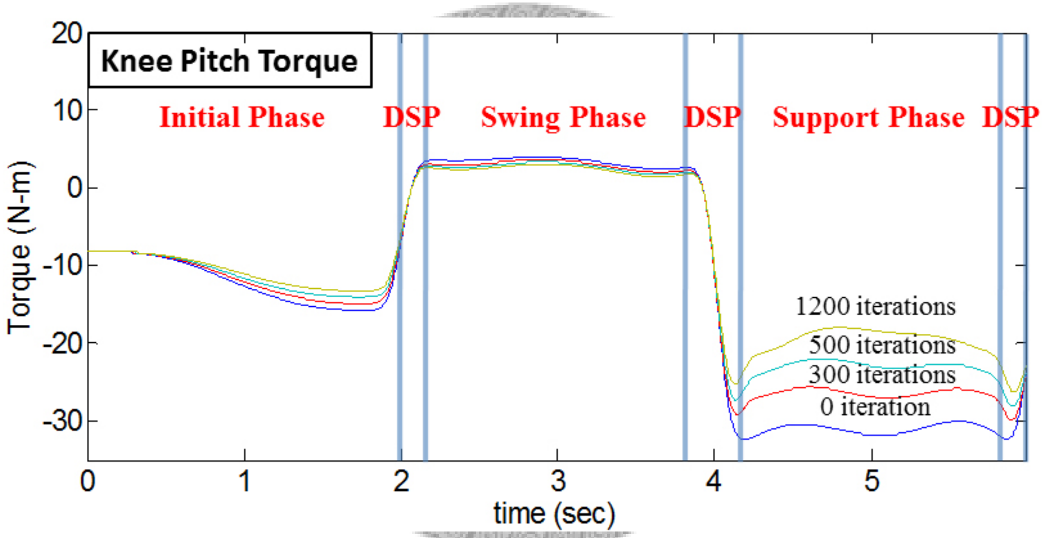


Figure 5-7. Torque trajectory of the knee pitch joint

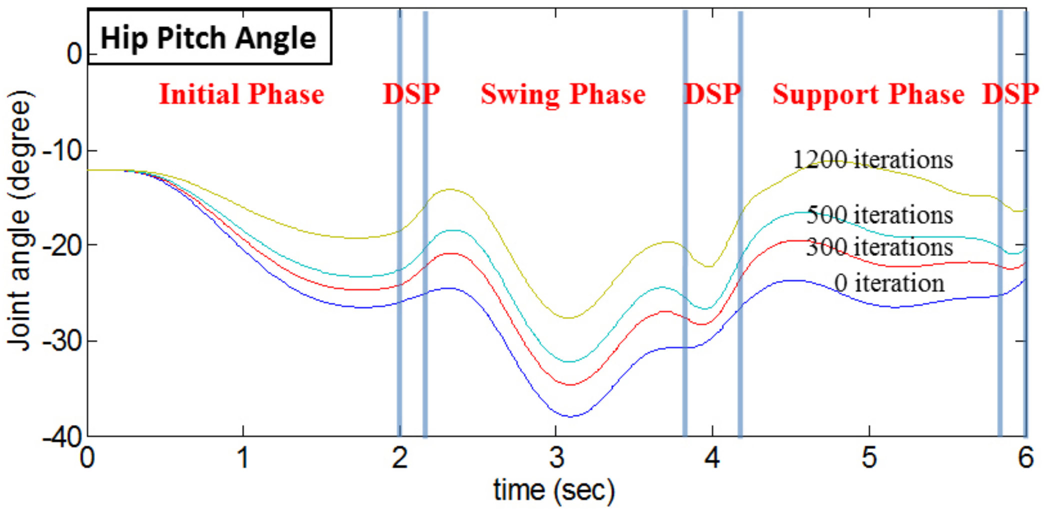


Figure 5-8. Angle trajectory of the hip pitch joint

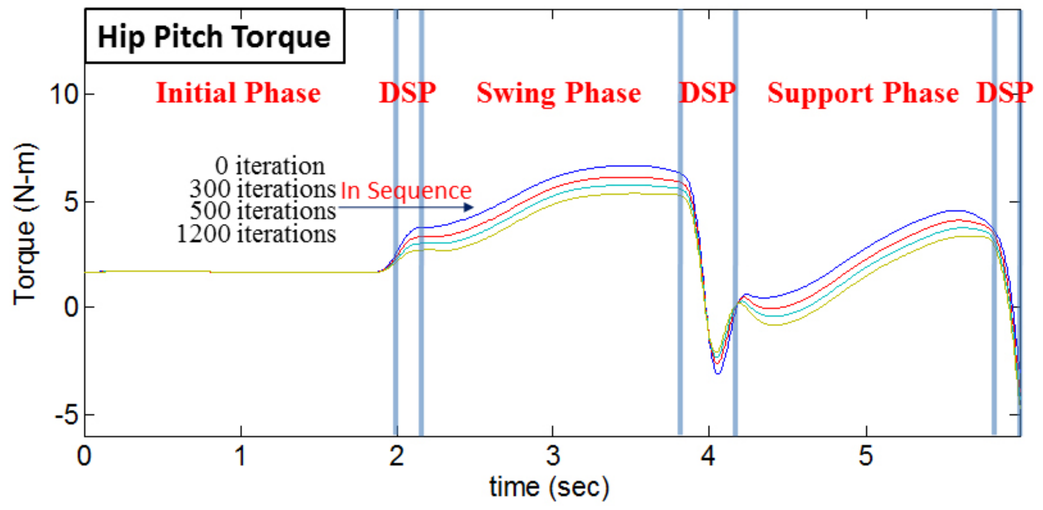


Figure 5-9. Torque trajectory of the hip pitch joint

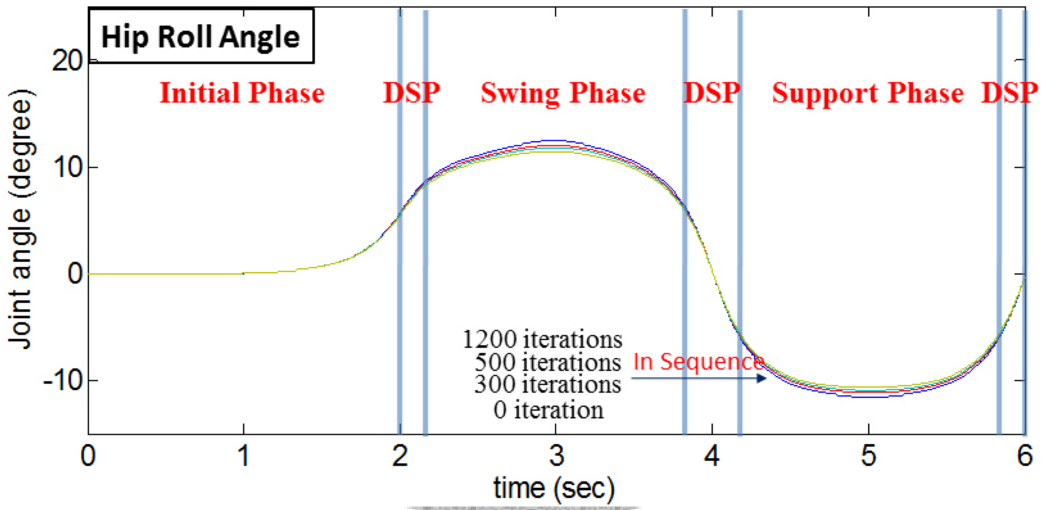


Figure 5-10. Angle trajectory of the hip roll joint

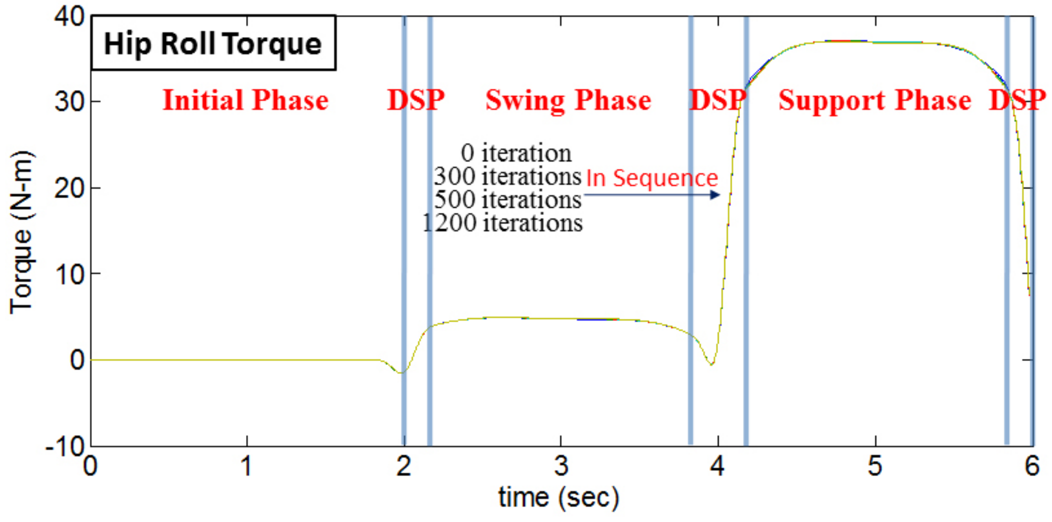


Figure 5-11. Torque trajectory of the hip roll joint

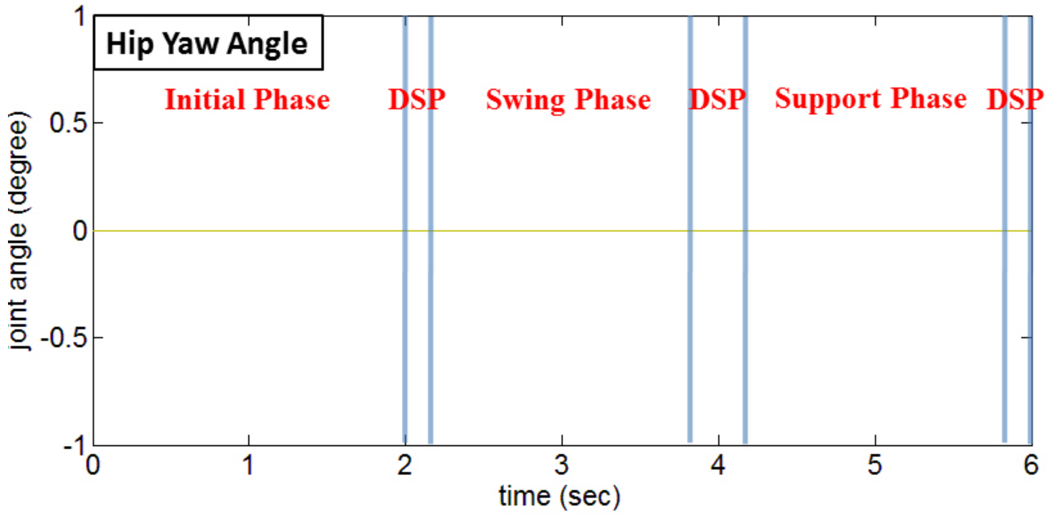


Figure 5-12. Angle trajectory of the hip yaw joint

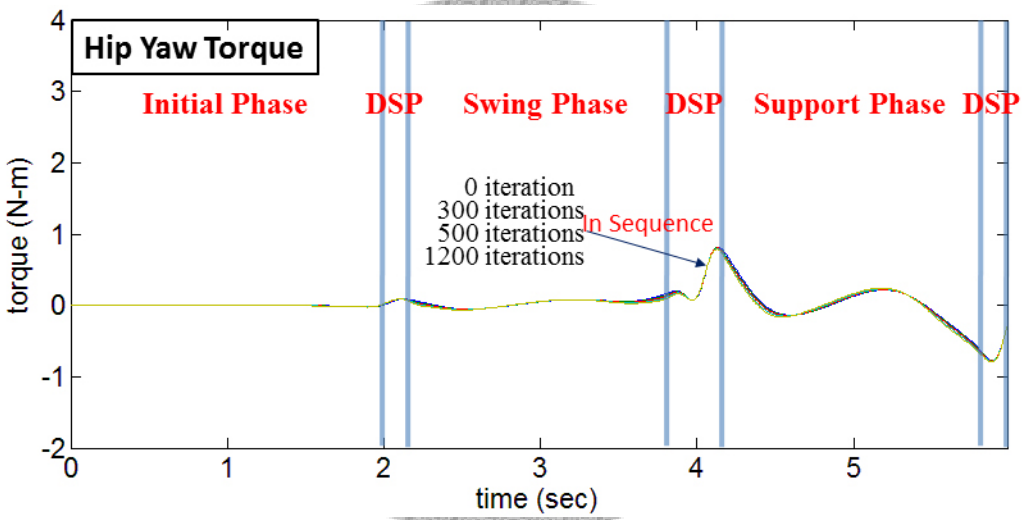


Figure 5-13. Torque trajectory of the hip yaw joint

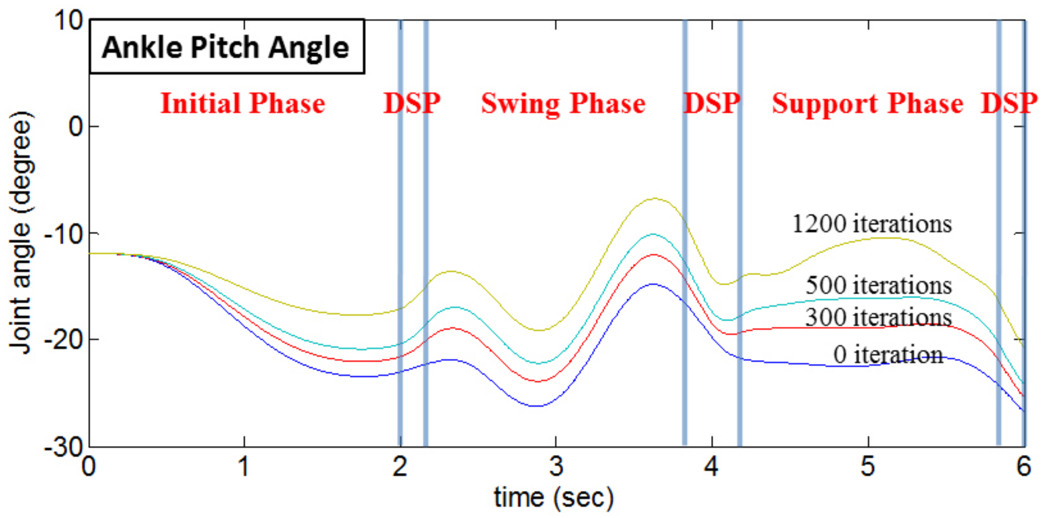


Figure 5-14. Angle trajectory of the ankle pitch joint

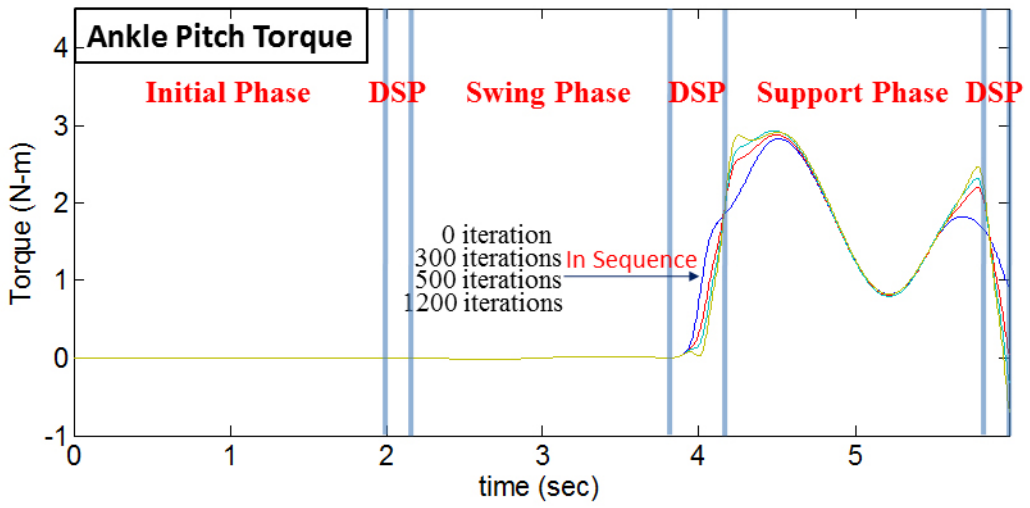


Figure 5-15. Torque trajectory of the ankle pitch joint

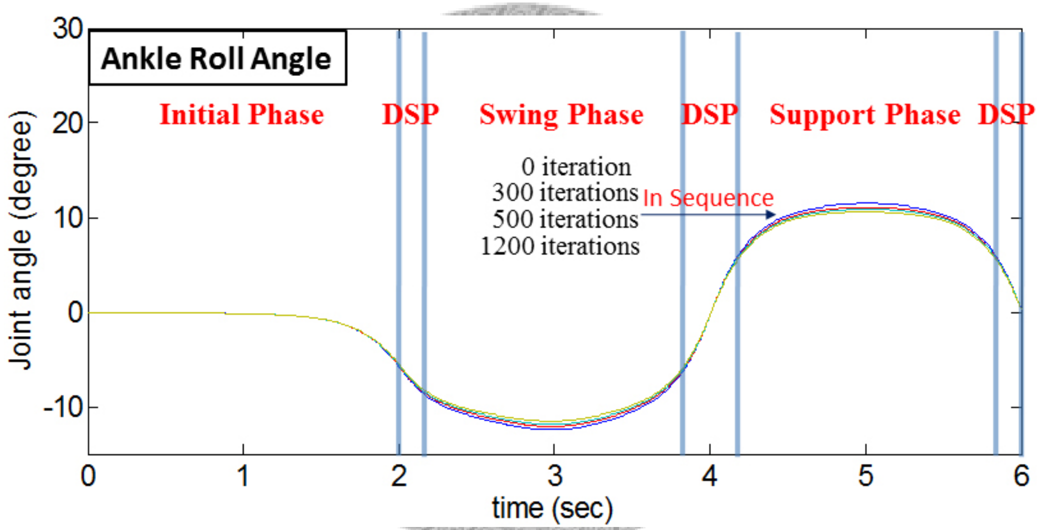


Figure 5-16. Angle trajectory of the ankle roll joint

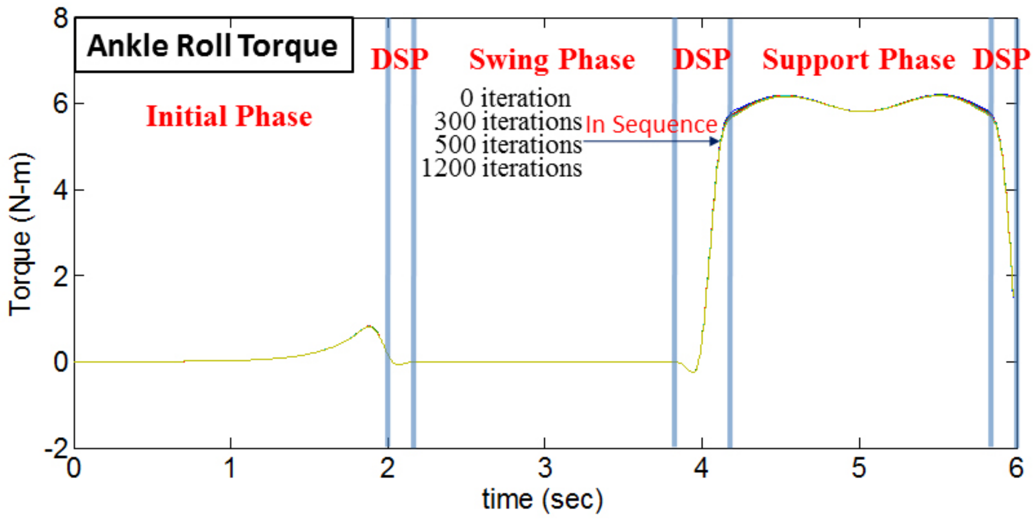


Figure 5-17. Torque trajectory of the ankle roll joint

Because the trajectories of angle and torque are very similar for each leg, only those for the left leg are shown in Figure 5-6–Figure 5-17, covering the initial phase, double support phase (DSP), swing phase and support phase (both SSP).

The discussion below looks at two aspects depicted in Figure 5-6–Figure 5-17: the movement directions of the joints (pitch, roll, and yaw), and the different support phases (swing, support and double support).

The three pitch axes in the hip, knee, and ankle change significantly with the COG height training iterations because they are directly related to the COG height. With the training iterations, the joint angle becomes smaller and smaller, so that the robot leg becomes straighter and straighter when walking. The joint torque values of these axes also become smaller. In the hip and the ankle joints, however, the joint torque values of the roll axes do not change significantly with the training iterations, as they are used to match the constraints of the lateral COG trajectory and the roll angle of the end-effector. The trajectories of the roll axes do change slightly with training iterations, because the C_z trajectory becomes higher as the training process continues. The longer the inverted pendulum (COG height is also higher) gets, the smaller the amplitude of the pendulum swing (lateral and sagittal COG motions) becomes. The amplitudes of the roll angles thus become smaller with training iterations. Because the walking trajectory for training is a straight line, the yaw angle trajectory is zero, and the yaw torque is much smaller than pitch and roll torques, and the yaw-angle trajectories do not change during COG height training.

In the swing phase, the shape of the joint-angle trajectories change very little because the swing motion is constrained by the desired motion of the end-effector (three translational and three rotational). The same applies to the joint-torque trajectories because the motors of the swing leg need only to drive the swing motion and do not have to support the body weight.

In the support phase, the joints of the leg support the whole body weight and sustain a larger joint torque than in the swing phase. During the COG height training iterations, compared with other joints, the torque trajectories of the hip and knee pitch joints undergo more significant changes than other joints. After optimization, the minimum torque of the knee joint becomes much smaller, from 29.98 N-m to 14.59 N-m.

During the DSP, one leg is switching roles from swing to support (and vice versa for the other leg). In this phase, none of the joint-torque trajectories changes very much with training iterations, and the robot bends its knee slightly because the six-axis robot leg has no toe mechanism to facilitate landing or leaving the ground.

With the proposed optimization method, the loads on the joints (especially the knee) become much smaller, and a more human-like COG height trajectory and walking pattern can be generated.

5.5 Summary

In this chapter, a COG trajectory optimization method is proposed. By minimizing the performance index, the proposed pattern generator can generate a more human-like walking pattern with smaller joint torque. The same COG optimization method can be used on different robot models. The robot model for training in this chapter has no toe mechanisms, thus a COG height optimization algorithm that does include toe use can be achieved in the future to generate an even more natural walking pattern.

With the proposed COG trajectory optimization method, the COG patterns under different circumstances and conditions can be generated in advance as a walking pattern database, which can then be used to achieve online control with optimized COG height trajectory.

Chapter 6 Real-time Control Architecture of Humanoid Robots

In this chapter, a network communication approach named real-time network (RTNET) is designed and implemented for humanoid robots. The proposed five network objects – alarm, condition, message, mail, and file are used to represent the task and priority of the communication data. Compared to the existing protocols, the network scheduling mechanism of RTNET more efficiently arranges the priority and flow control of the five network communication objects to meet real-time requirements for the limited bandwidth of the local area network (LAN). RTNET can be further integrated with controller area networks (CAN-Bus) for local control systems, such as mobile robots or humanoid robots, to improve the communication mechanism. The RTNET can also be used over Ethernet to connect each subsystem and to exchange information among those systems. The RTNET has been implemented on the NTU humanoid robot control system with CAN-Bus.

6.1 Introduction

The development of the microprocessor, microelectronics, communication method, and computer are very rapid. Robotics systems are often composed of many units such as computers, controllers, actuators, and sensors. Commands and data must be sent among those units in order to gain the desired performance. As the number of the units grows, the commands and data format become more and more complex. Humanoid robots often have more than one hundred units and they are distributed in multiple locations. In such a large system, a real-time and well-scheduled communication system becomes very important.

Some robots were designed with the centralized control architecture [41][57][58][68][86][93][111][112] including IEEE-1394, local ISA, VME, RS-232 and PCI bus. Their communication interfaces have high communication speed but the wire systems are too heavy and complicated. The other robots or platforms used distributed control/computation architecture such as CAN-Bus [75][82][94][115][144], Ethernet [24][79][81][85][145], Ethernet-CAN [77][78][139][147][151] to achieve their real-time control and communication systems. Communication systems using CAN-Bus are reliable and can achieve real-time control, but the bandwidth may not be enough when the nodes in the bus are too many. Communication systems using Ethernet has high bandwidth, but the requirements for using in local microprocessors are higher and the wire system and protocol for Ethernet is more complex. The universal serial bus (USB) is a good solution for communication devices. It has high bandwidth and can be implemented in real-time. But it is designed as a master-slave structure rather than a distributed system. Thus, if we control several nodes, it needs the same number of cables to control the nodes. If there are many nodes, the wires may be a big problem if we just use USB as the communication interface to construct the system. Another problem is how to achieve peer to peer communication by using USB devices and reduce the number of wires. Each communication device and protocol has their advantages and disadvantages. In order to find a balance between them, Ethernet, USB and CAN-Bus are integrated. To merge those different communication systems, it is necessary to define how to transmit the data among those systems. There are many methods to solve this problem. It is usual to add a microcontroller as the buffer, to connect different types of data structures between different communication networks.

The easiest methods to coordinate different communication networks are to implement them in the application layer.

The above protocols using Ethernet, CAN-Bus, and USB do not provide the mechanism of priority scheduling. The reason we need the priority scheduling is in that the data in distributed computation and control are all stored in a queue. If we use a first-in-first-out (FIFO) mechanism to send them, important data may be jammed and delayed. There are several types of communications that should be sent with a higher priority, for example, the alarm caused from the failure in control systems or the error signals from computers and microcontrollers on the robots.

RTNET is developed as a communication tool among robots, equipment, devices, personal computers and workstations in versatile platforms. It is designed and implemented to satisfy the following goals:

1. Real-time communication with network priority scheduling.
2. Unified approach for small to large scale systems.
3. Satisfaction for control, computation, manufacturing, and general applications.
4. Ease of use.

In this chapter, the humanoid robot networking system will be shown in section 6.2 and how does RTNET work is shown in section 6.3. In section 6.3, five network communication objects which are used to represent the tasks and properties of communications are also proposed. The network scheduling mechanism used to deal with the five network objects will be described in section 6.4. Section 6.5 shows the simulation and implementation using RTNET on humanoid robots and distributed computation and control systems. Finally, section 6.6 summarizes this chapter.

RTNET is implemented on both Ethernet and CAN-Bus for command/data transmission. The Ethernet based RTNET is used to connect the PCs, laptop computers and workstations, and the CAN-Bus based RTNET is used to control each sub-systems. The whole system architecture is shown in Figure 6-1.

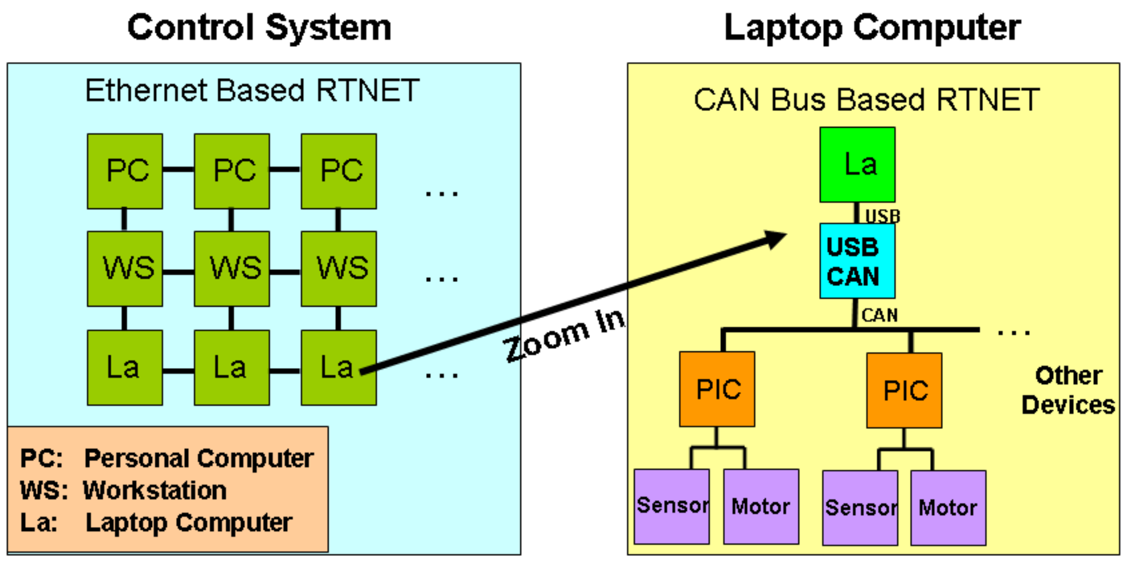


Figure 6-1. System architecture of the whole control system

Clearly, personal computers, workstations and laptop computers can be connected using Ethernet based RTNET. Each node in the Ethernet based RTNET can contain one or more CAN-Bus based RTNETs as sub control systems. For example, in a humanoid robot control system, a laptop computer is used as one node in the Ethernet based RTNET. It is also the central control computer of the robot. In addition, we can use several CAN-Bus based RTNETs to control the arms, the hands and the legs. The CAN-Bus based RTNETs are connected to the laptop computer with USB-to-CAN-Bus adaptors.

6.2 Networking for Humanoid Robot Control System

Humanoid robots often have more than one hundred units and they are distributed in different parts of the robot. A neat, real-time and well-scheduled communication system is very important for such complex systems.

6.2.1 Control Bus of the Humanoid Robot

The proposed human-sized humanoid robot system has 50 motors and 102 sensors. In such a large system, reducing the number of wires is very important for the setup and maintenance of the robot. To find an optimal solution for real-time control and the wire system, the limbs and sensors on the robot are connected to the central control laptop computer with several USB-to-CAN-Bus adaptors, and the limbs and sensors are connected in several CAN-Buses.

In each CAN-Bus, all nodes can be connected with four wires (Vcc, Gnd, CANH and CANL). Two wires are digital power lines and two wires are for CAN-Bus communication. This can reduce a large number of wires compared to directly connecting the devices using RS232, I2C and USB or other methods.

6.2.2 Joint Controllers and Nodes of the Robot

Micro controller units, PIC (dsPIC30F4011), are used as the digital signal processing (DSP) nodes in the CAN-Bus. It has 9-Channel 10-bit Analog to Digital Converters (ADC), 3 pulse width modulation (PWM) modules, 1 encoder module and DSP functions. The hardware of each node is composed of one joint controller and motor unit. The architecture is shown in Figure 6-2.

In the DSP unit shown in Figure 6-2, the PWM modules and encoder module are used for motor control, and the ADCs can be used to acquire the signals from the sensors near the node. The DSP unit has one CAN-Bus module. With this module, it can connect to the local CAN-Bus based RTNET to transmit the data of the sensors and receive the commands form higher-lever controllers.

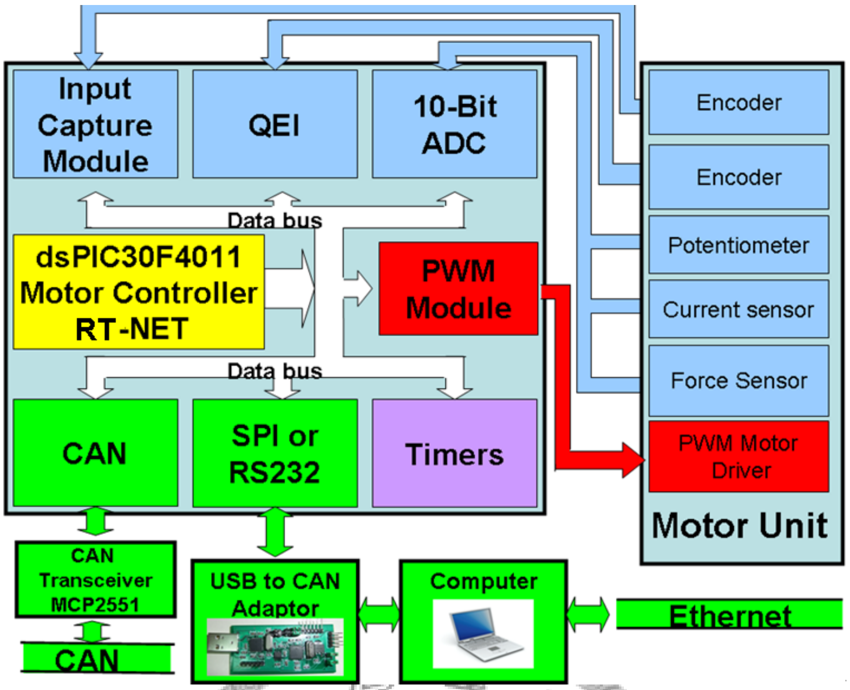


Figure 6-2. The architecture of joint controllers

The local control node communicates with other nodes through CAN-Bus. However, the computer communicates with other computers through Ethernet. The communication between these two networks, the adaptor of USB-to-CAN-Bus is designed, as shown in Figure 6-3. With this adaptor module, laptop computer can send control commands to each sub CAN-Bus to achieve real-time multi-axis motor control and multi-sensor reading. To transfer the data in the network effectively, RTNET is implemented in the DSP to manage data transmission of each node in the bus.

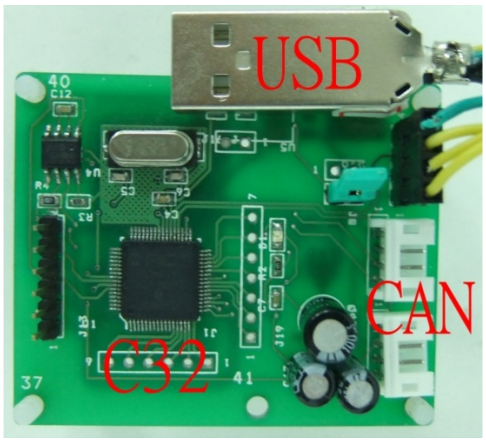


Figure 6-3. The USB-to-CAN-bus adaptor module

6.2.3 Multi-Node Control Structure for the Humanoid Robot

Because the maximum baud rate of CAN-Bus is 1Mbps, the command update rate will be too slow if there are too many nodes in one local CAN-Bus. To solve this problem, several USB-to-CAN adaptors are used to share the data flow from the laptop computer to CAN-Bus. In the C++ program in the laptop computer, multi-thread program is used to send control commands through the USBs in parallel.

The number of nodes in each CAN-Bus should be determined in terms of the requirement of command update rate. For example, for walking robots, high speed control and sensor feedback are required in order to improve the walking stability. High speed control and sensor feedback are also required for robot arms because the robot arms will vibrate under non-smooth (slow) position commands. On the other hand, for the facial expression control on the robot head, the command update rate can be slower because the requirement of fast and accuracy motion control for facial expressions is less than robot arms and robot legs. The configuration of CAN-Bus in the proposed humanoid robot is shown in Figure 6-4.

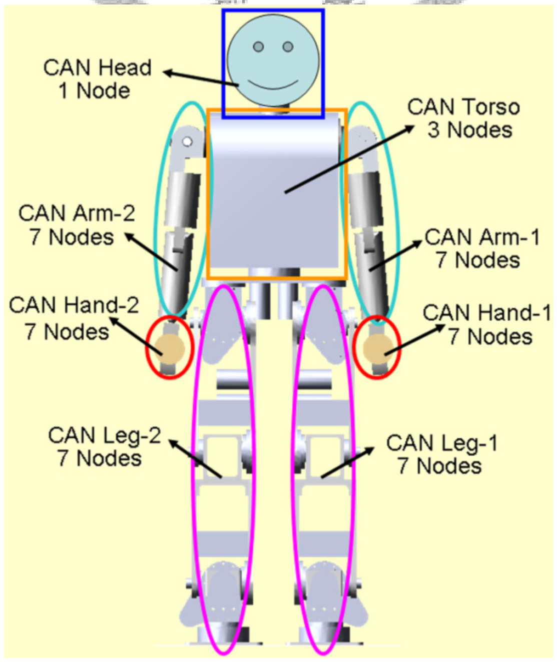


Figure 6-4. CAN-bus structure for the proposed humanoid robot

As shown in Figure 6-4, eight USB-to-CAN-Bus adaptors are used to connect each part of the robot and the laptop computer, two for the arms, two for hands, two for the legs, one for the torso and one for the head. In the head, body and limbs of the robot, each CAN node is used for controlling motors and collecting the data from the sensors near the node.

Although the control system is divided into many parts to improve the command update rate and reduce the data flow in each bus, a good scheduling and data transmitting mechanism are still needed for large and multiple data transmission. A reliable, real-time and well-scheduled networking algorithm will be described in section 6.3.

6.2.4 Multi-Robot Control and Communication System

RTNET is used to construct a communication system for one robot, but how about the communication among many robots? When executing multi-robot works, a communication protocol must be defined among the robots, the laptop computers, and the workstations. The interface among them is constructed by using Ethernet with the priority oriented networking protocol, the Ethernet based RTNET, as described in the following.

Note that the CAN-Bus based RTNET is used for data flow and control under one laptop computer or one workstation. The Ethernet based RTNET can provide the communication and dataflow among robots, laptop computers, and workstations.

6.3 Priority Oriented Networking (PON)

Network communications are various and flexible. Properties of the network communication objects and their priorities should be well-defined in order to have good performance on data transmission and can send emergency alarms in real-time. For example, the communication to carry emergency alarms should be treated as the first priority, and the communication to perform the handshaking between processes should be treated as another level of priority. There are many other types of communications which must be dealt with, such as mail and file transmissions.

6.3.1 Objects of Network Communications

RTNET has five basic network communication objects called alarm, condition, message, mail and file. Each communication object is assigned a corresponding priority. They also represent the classification of information flow on the control systems. The five network objects of RTNET are defined as follows.

Alarm communication

The alarm of network communication is used to indicate that the system is damaged, malfunctioning, or there is some emergency. For example, the programmed machine needs to send an alarm to indicate that the problem is serious and maintenance is needed. A computer may send an alarm to notify and ask the remote operator to reset or repair the robot, control system, or other units.

Condition communication

Many network communications are used to facilitate cooperation between the system state and the process units. For example, two or more computers can use the communication to perform handshaking. One robot/computer can use the communication to know the condition of the other robot/computer, and then perform the corresponding action. This type of network object is very desirable in networking applications. Such network communications are called “the condition object.”

Message communication

Data transfer can be done in network communications. In other words, the data can be simultaneously shared by different applications in the networks. The message objects are the objects that are not very large in file size and they are readable by human directly.

<u>Mail communication</u>

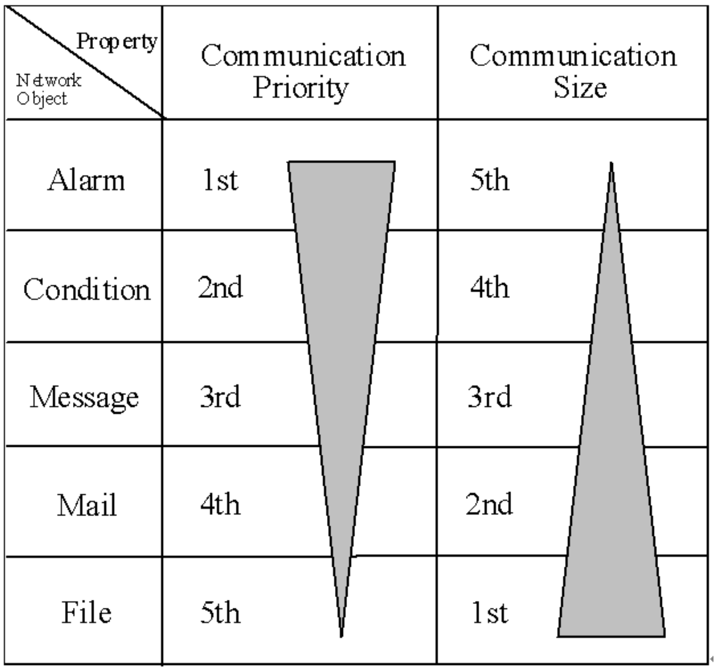
The mail communication is used to transfer mail text in network. Its function is the same as E-mail. Its file size is larger than message and can contain some text files. It can be used to transfer text data for human reading.

<u>File communication</u>

The file communication is used to transfer files which can be in any format, such as robot configuration/command file, video, audio, picture, data, and so on. The function of the file communication is the same as the FTP (file transfer protocol).

6.3.2 Priority and Size of Network Objects

Since the “Alarm” object is absolutely essential, it has the highest priority. For the sake of network interaction, the communication object “Condition” has the second high priority.



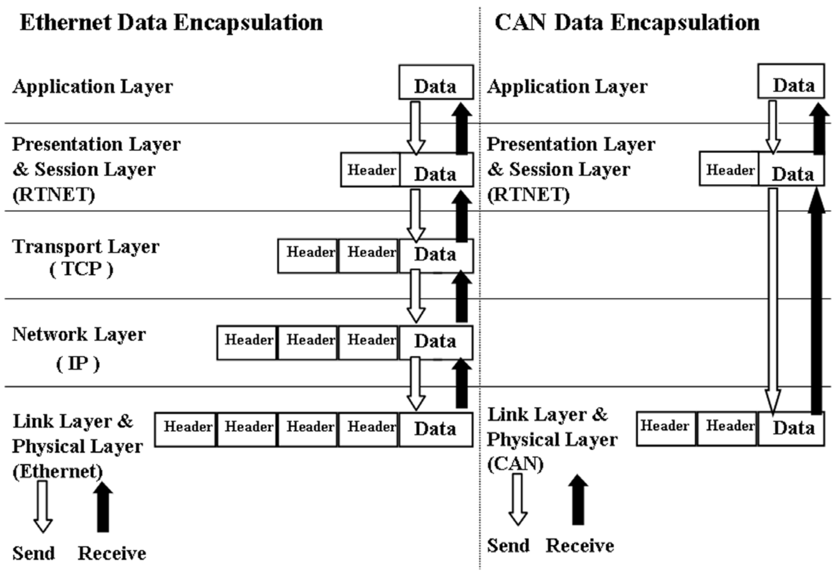
Property / Network Object	Communication Priority	Communication Size
Alarm	1st	5th
Condition	2nd	4th
Message	3rd	3rd
Mail	4th	2nd
File	5th	1st

Figure 6-5. The priority and size of the communication object

The communication object “Message” which transfer messages among the devices has the third high priority. The communication object “Mail” has the fourth high priority. The communication object “File” has the lowest priority since it may be huge and time consuming. In general, the above priority arrangement is opposite to the size of the communication data, as shown in Figure 6-5. This fact also shows the fitness of the five network objects that have been classified.

6.3.3 Common Properties of the Network Objects

Each network object has some properties which represent the network object itself. However, some properties are common for the network objects, i.e., the destination of communication, which type of network object, and its size.


Ethernet Data Encapsulation
CAN Data Encapsulation
Application Layer
Data
Presentation Layer & Session Layer (RTNET)
Header
Transport Layer (TCP)
Network Layer (IP)
Link Layer & Physical Layer (Ethernet)
Link Layer & Physical Layer (CAN)
Send
Receive
Figure 6-6. The data encapsulation and header presentation

Our method to deliver the information through the network objects is to add a network header before each communication. Since the RTNET is stacked on the TCP/IP and CAN-Bus, the data encapsulation and header presentation are also stacked on it, as shown in Figure 6-6.

Although the stacked data and commands in TCP/IP based and CAN-Bus based RTNET can be transmitted through a first-in-first-out (FIFO) data transmission mechanism, the network priority scheduling mechanism of RTNET will achieve a better performance. This will be presented and discussed in section 6.4.

6.4 Network Scheduling

One of the special features of RTNET is the queuing mechanism which stores the request for transmissions and it can be performed in background. Once there are five network objects requested to be sent and received in the background of the application, how does so many communications be processed in background? A scheduling mechanism is provided to arrange the network objects to be sent in RTNET. In this section, the scheduling method and how it works are presented.

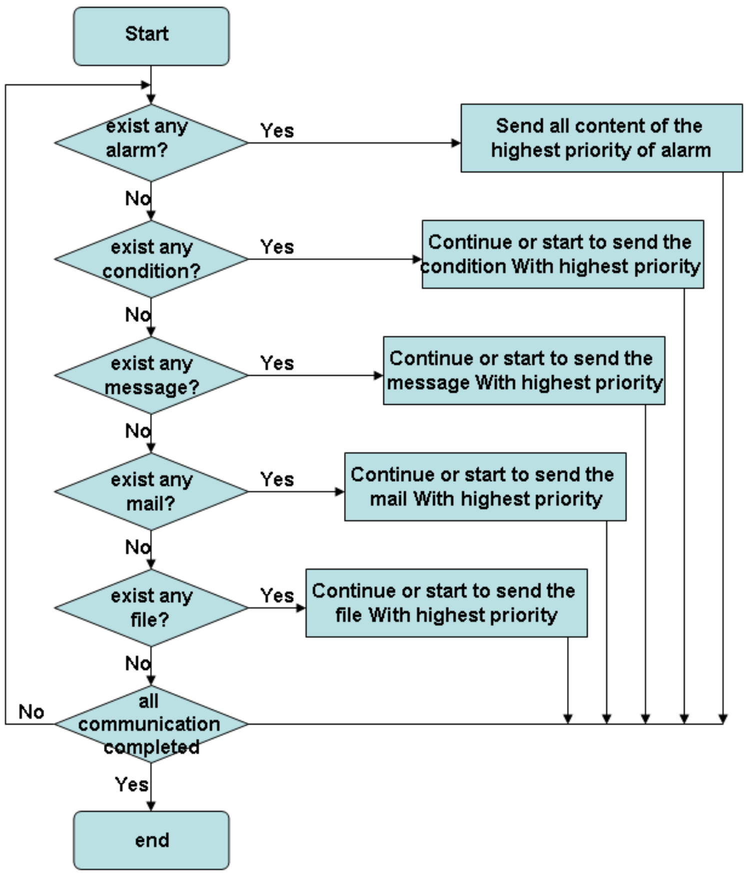
6.4.1 Network Scheduling Mechanism

When transferring data in the network, a scheduling mechanism is required for preventing the important data from being jammed in the queue of communication objects. The network scheduling mechanism (NSM) of RTNET can find the communication objects in queue with higher priority and send them first. For example, if an alarm is triggered when a large file is being transmitted through RTNET, the RTNET will interrupt the file transmission and then send the alarm first.

6.4.2 Flow Control of the Scheduling Mechanism

The flow control of the NSM is performed by setting the scanning time of the algorithm. In our system, the scanning time is set as 0.005 second. Within this time interval, RTNET scans if there are any communication objects with higher priority than the object which is being transmitted. The scanning procedure is shown in Figure 6-7. When the interval is set smaller, the communication objects with higher priority can be sent more quickly. At the same time, more interruptions will occur when transmitting large files.

The efficiency of sending larger files will be reduced by the time delay caused by the interruptions. On the other hand, if we choose a larger scanning time, the alarm will be delayed but the efficiency of transmitting files will be higher. With these considerations, a suitable scanning rate should be chosen for gaining an acceptable delay of alarms and the best file transmitting efficiency. It is set as 0.005 seconds in the proposed humanoid robot.



Start
exist any alarm?
Yes
Send all content of the highest priority of alarm
No
exist any condition?
Yes
Continue or start to send the condition With highest priority
No
exist any message?
Yes
Continue or start to send the message With highest priority
No
exist any mail?
Yes
Continue or start to send the mail With highest priority
No
exist any file?
Yes
Continue or start to send the file With highest priority
No
No
all communication completed
Yes
end

Figure 6-7. The flow chart of NSM for performing communication

6.5 Simulation and Implementation

The RTNET structure is implemented in the proposed humanoid robot, as shown in Figure 6-4. In the humanoid robot, CAN-Bus based RTNET is used to construct the control/computation system for locomotion control. Robots, computers, and workstations can be connected with the Ethernet based RTNET networking, as shown in Figure 6-8.

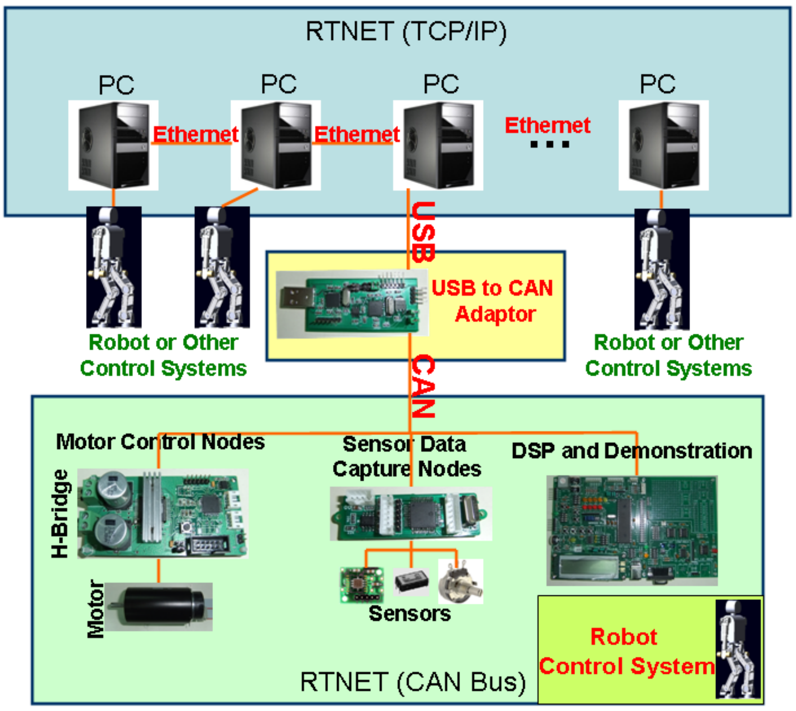


Figure 6-8. Multi-robot control system with RTNET

6.5.1 Ethernet Based RTNET

RTNET can also be implemented on a multi-robot control system. The upper-end RTNET is Ethernet-based, and the PCs, laptop computers or workstations can be connected to this RTNET. Each node can control robots, machine tools or control systems, as shown in Figure 6-8.

In the multi-robot control system, CAN-Bus based RTNET is also used as the lower-ends, as shown in the figure. The upper and lower RTNETs are connected through USB-to-CAN-Bus adaptors which can buffer and adapt the upload/download dataflow. By using RTNET, better scheduling, data transfer, and control performance are achieved.

6.5.2 CAN-Bus Based RTNET for Local Networks

Following the concept of the Ethernet based RTNET, the CAN-Bus Based RTNET is designed. Similar to the Ethernet based RTNET, the CAN-Bus based RTNET has the same objects of network communications: alarm, condition, message, mail and file.

Because the CAN controllers can only store the message, mail and file in their RAM (random access memory), large data cannot be sent over the CAN-Bus based RTNET. Except this limitation, the CAN-Bus based RTNET can have the same function as the Ethernet-based RTNET in its local network.

6.5.3 Performance on Data Transmission through RTNET

The performance between a FIFO communication mechanism and the proposed RTNET is compared in the following. In both the FIFO communication and the RTNET, the size and the communication rate of the communication objects are listed in Table 6-1. The bandwidth of the LAN is set as the bandwidth of the IEEE 802.11g wireless LAN, 54Mbps (in limited distance).

Table 6-1 Size and Rate of the Objects

Communication Object	Size	Rate
File	about 100MB	5 files, fixed
Mail	about 8MB	0.2% in each sampling interval
Message	about 500KB	0.9% in each sampling interval
Condition	about 10KB	every 5 seconds
Alarm	about 1KB	0.7% in each sampling interval

Because the speed of the wireless LAN are the same in both tests, the total time for FIFO transmission and RTNET transmission are almost the same. Also, the number and probability of all communication objects are set the same in both tests. Five "files" which might exhaust the bandwidth of the wireless LAN are set. "mails" and "messages" are sent randomly with fixed probabilities. The "condition" is used to check the condition of the nodes in the network. It will be triggered every 5 seconds. Finally, the "alarm" indicates the alarms in the network. It will be triggered with a fixed probability. The results of using FIFO transmission and RTNET are shown in Figure 6-9 and Figure

6-10. In the figures, the x-axis shows the time in second, and the y-axis shows the accumulated amount or size of each communication objects.

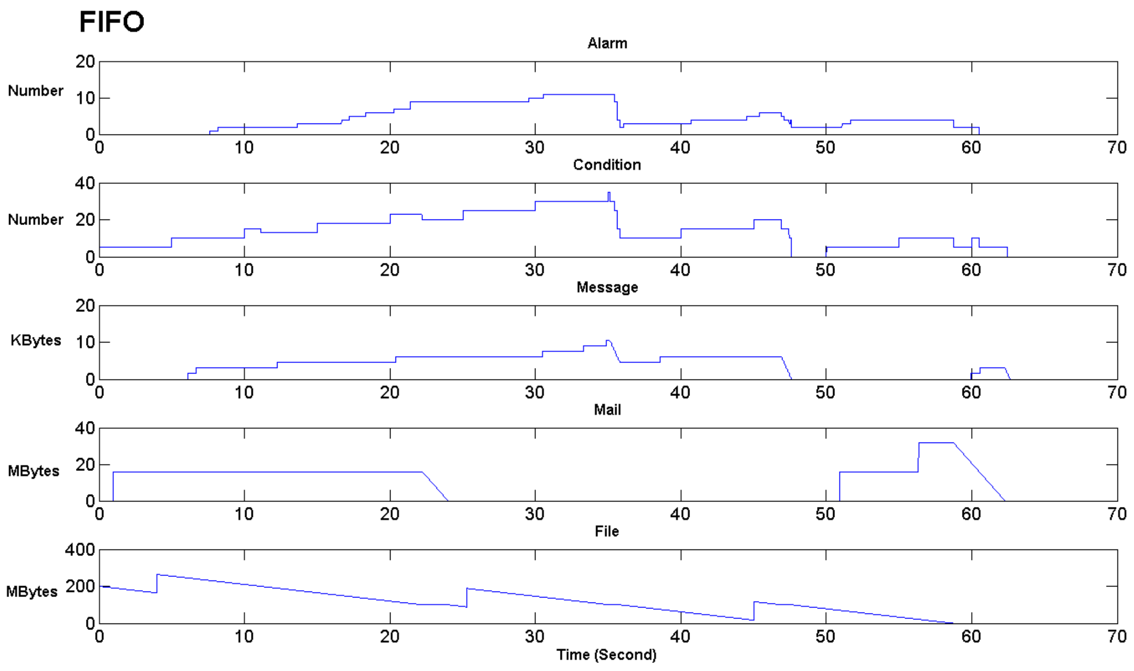


Figure 6-9. Transmit the communication objects with a FIFO stack

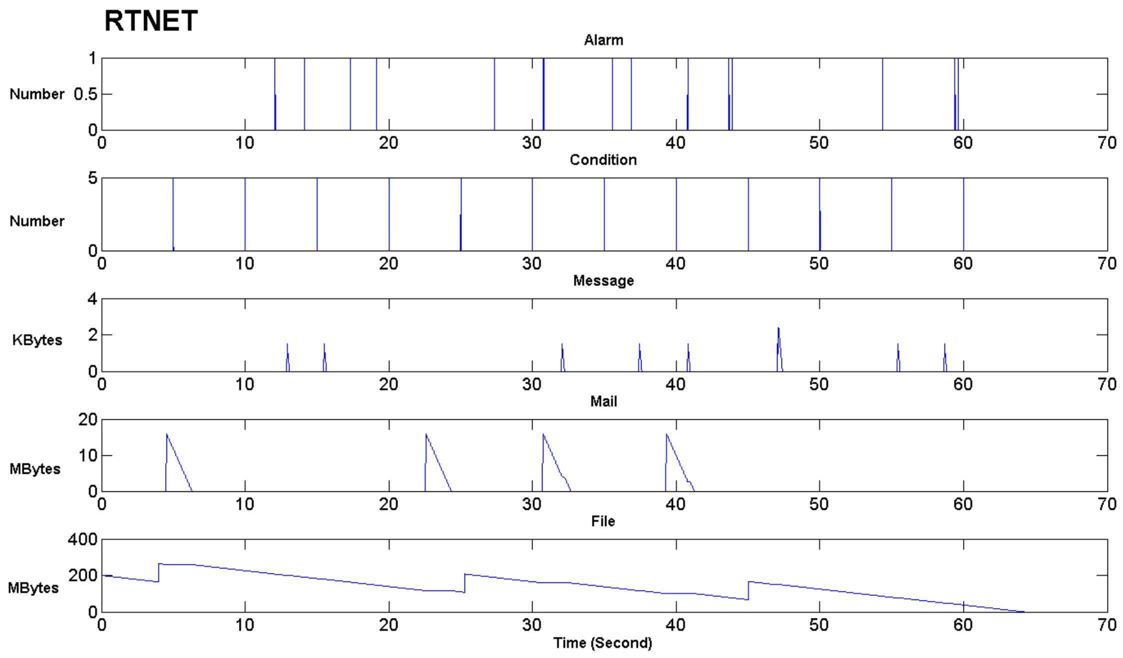


Figure 6-10. Transmit the communication objects with RTNET

The communication objects transmitted with a FIFO stack and RTNET are illustrated in Figure 6-9 and Figure 6-10. Since the size of the communication object “file” is too larger to be sent in a short time with the bandwidth of the wireless LAN, the communication objects will be stacked in a queue. When using the FIFO, the communication object which is in the first position in the queue will be transmitted first. When a “file” object is transmitted, the “alarm” signals might be blocked. In Figure 6-9, the number of the alarm is accumulated to 10 because the alarms are queued later than other large objects. Time delay of alarms might cause serious damage to humans, robots or machines. The “condition” is important because it indicates the status of each node in the network. In Figure 6-9, the condition objects are also delayed for waiting for larger objects. In Figure 6-10, because the RTNET considers the priority of each object, the important objects will be transmitted earlier. Although the waveforms of the “file” are almost the same in Figure 6-9 and Figure 6-10, the waveforms of the other objects are quite different. In Figure 6-10, the alarm objects and condition objects are transmitted immediately after they are queued. This will help the administrator or operator to judge the status of the network or the system can stop the robots or machines immediately after the emergency alarms. Thus, with the RTNET, communication objects with high priority will be sent first, and will not be jammed in the queue, as shown in Figure 6-9.

6.6 Summary

In this chapter, the RTNET is designed and implemented for distributed control and computation for the proposed humanoid robot and other control systems. After considering the requirements of networking, the network communications are categorized into five objects, and a unified communication approach that works efficiently with embedded applications is provided. These objects also make RTNET suitable for robotic systems or for network based equipment.

The scheduling mechanism is provided to send and receive communications with priority in background and to achieve the goal of real-time communication. A corresponding network management is also provided to monitor the performance and traffic of the network communication and to resolve traffic jams in the networks. RTNET has been verified and implemented on the proposed humanoid robot. Its performance is quite satisfactory.

Chapter 7 Implementation

In this dissertation, a generalized method for COG trajectory optimization is proposed. Two humanoid robots are used as simulation and experiment platform; one is a human-sized humanoid robot and the other is a small-sized humanoid robot. Small-sized robots are constrained by its limited space and the control accuracy of RC servo motors (radio control motors), but they can be manufactured and assembled quickly to test control algorithms. On the other hand, human-sized humanoid robot have enough space to install DC motors, driver boards, sensors, and embedded computer, but they are very expensive and cost a long time to manufacture and assemble. With these two robots, the proposed optimized walking pattern generator can be implemented and tested.

In this chapter, the specifications of the robots used in this dissertation are shown in section 7.1, the real-time planning/control architecture are described in section 7.2, and section 7.3 shows and discusses settings and results of the experiments using the proposed methods in this dissertation. The performances of the methods in this dissertation are verified.

7.1 Specifications of the Proposed Humanoid Robots

Specifications of the two robots are described in this section. The photos of the two robots are shown in Figure 7-1 and Figure 7-2. The physical specifications of the proposed human-sized and small-sized humanoid robots are shown as Table 7-1.

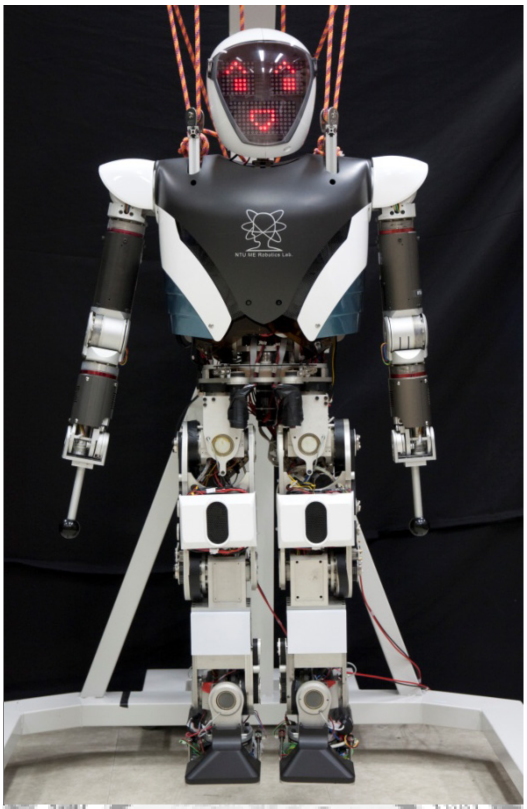


Figure 7-1. Human-sized humanoid robot

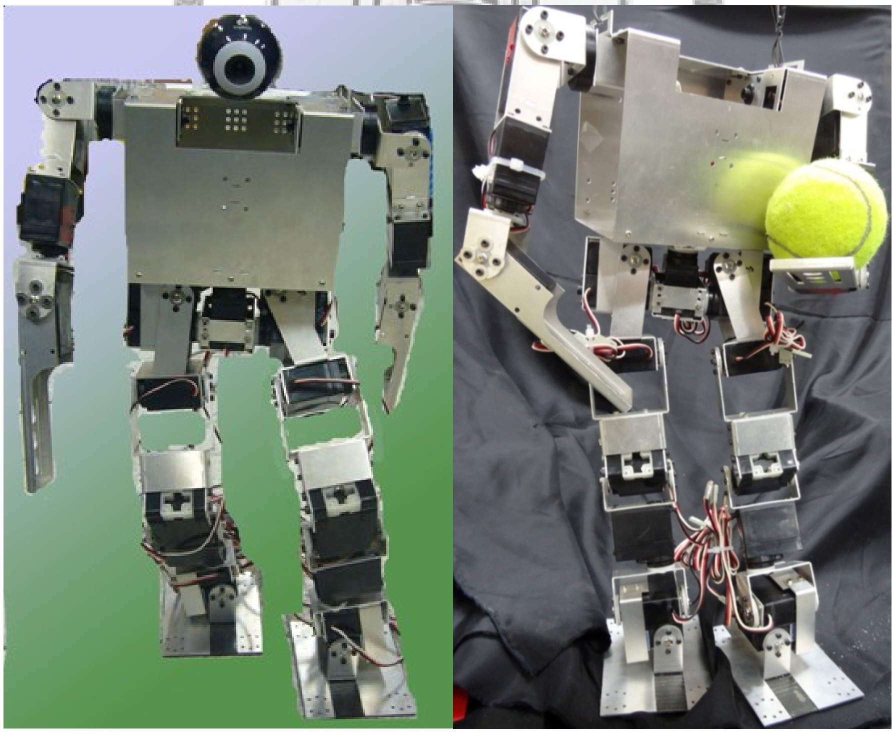


Figure 7-2. Small-sized humanoid robot

Table 7-1 Physical specifications of the proposed robots

	Human-sized Robot	Small-sized Robot
Height	1450 (mm)	430 (mm)
Weight	68.0 (Kg)	1.8 (Kg)
Mechanism material	7075 aluminum alloy	5052 aluminum alloy
Motor type	DC brushed servo motors	RC servo motors
Reducer	Harmonic drives, belts and pulleys	Gears in the RC motors
Controller interface	USB and CAN-Bus	RS232

Because of the limited bandwidth of the RS232 interface, algorithms can be tested quickly by using the small-sized robot, but for real-time control, human-sized robot must be used. The human-sized robot has more DOFs than the small-sized robot; it is more complicated and can achieve more tasks. Table 7-2 shows the arrangement of DOFs of the two robots.

Table 7-2 Arrangement of degrees of freedom

	Human-sized Robot	Small-sized Robot
Head	0 (LED Array)	2
Arms	12 (6×2)	8 (4×2)
Hands	24 (12×2)	0
Torso	2	2
Legs	12 (6×2)	12 (6×2)
Total	50 DOFs	24 DOFs

Compared with the small-sized robot, in addition to walking and arm motions, the human-sized robot has DOFs in its head and hands to achieve facial expressions and grasping motions.

7.2 Real-time Planning/Control System of Humanoid Robots

Chapter 6 describes the networking system of humanoid robots; high level communications and protocols are proposed. On the other hand, how the robot generates the walking patterns and how it communicates with local controllers are described and proposed in this section. The real-time planning and control system of humanoid robots are proposed and implemented on the robot.

7.2.1 Real-time Planning and Control Architecture

Figure 7-3 Shows the real-time planning and control system.

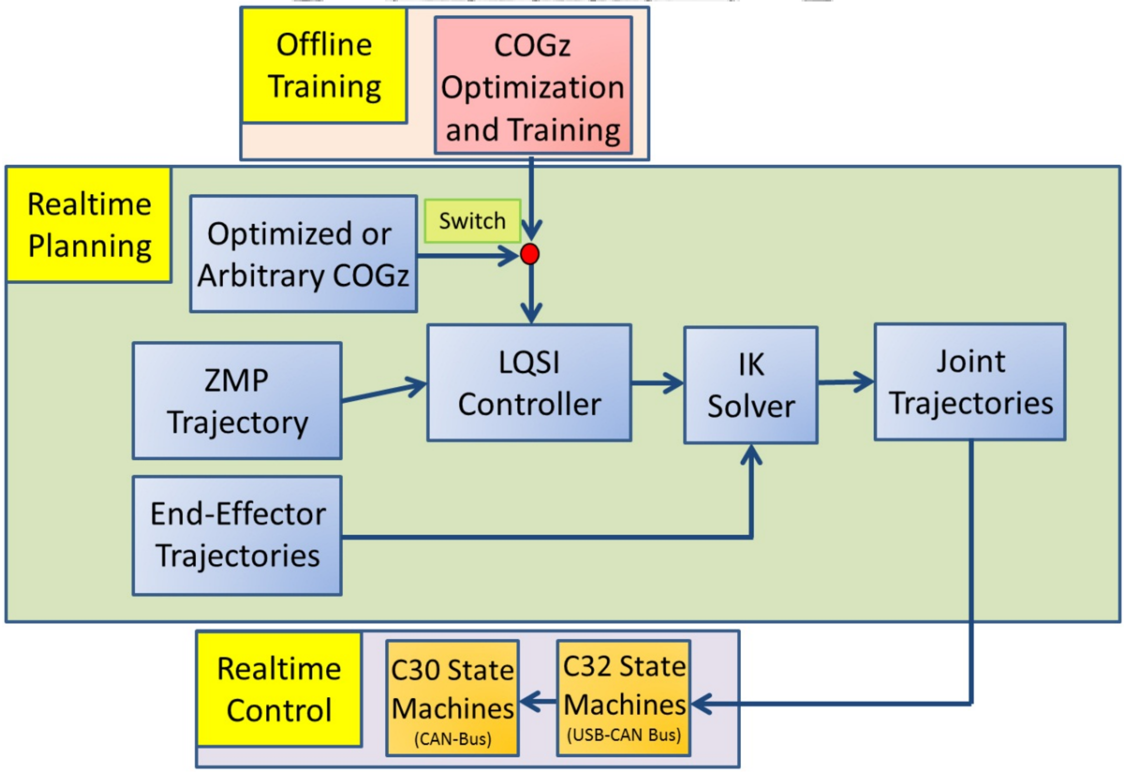

Offline Training
COGz Optimization and Training
Realtime Planning
Optimized or Arbitrary COGz
Switch
ZMP Trajectory
LQSI Controller
IK Solver
Joint Trajectories
End-Effector Trajectories
Realtime Control
C30 State Machines (CAN-Bus)
C32 State Machines (USB-CAN Bus)

Figure 7-3. Real-time planning and control system

In Figure 7-3, the IK solver, LQSI controller, and the C_z optimization and training are proposed and described in chapters 3, 4, and 5. The C_z optimization and training needs some iteration to converge; it is an offline training procedure. After the training, the optimized C_z trajectory can be saved and sent to the LQSI controller. The ZMP trajectory and the C_z trajectory are sent to the LQSI controller and the COG trajectories in sagittal and lateral directions are solved. The 3D COG trajectory and the trajectories of the end-effectors of the robot are the inputs to the IK solver. Finally, the IK solver solves the joint trajectories of the robot and the real-time planning phase ends here. The joint trajectories are sent to the C32 controllers through USB interface and be stored in a FIFO (first-in-first-out) queue. C32 controllers send the joint trajectories and receive sensor feedback signals with C30 controllers every 5ms. The real-time control phase ends here. Except the training phase, the whole robot planning and control system is real-time and can be processed in the order of millisecond.

The proposed robot control system is a combined centralized and distributed control system. A mini-ITX (17cm×17cm) personal computer with Intel Core^{TM} I7 870 CPU is used as the centralized part to execute walking pattern generation and motion planning. It is faster than the computer used in chapter 4 and provides more computation power for real-time computation. On the other hand, the distributed 16-bit dsPIC30F4011 controllers and their 32-bit master PIC32MX795F512H are all running state-machines when the robot is powered on. They are always checking their state and executing commands. As shown in chapter 6, dsPIC30F4011 is the local controller and PIC32MX795F512H is the USB-to-CAN-Bus adaptor. They will be described in the follow sections.

7.2.2 State Machine Architecture of C30 Controllers

Figure 7-4 shows the architecture and how the state machine of C30 local controllers works.

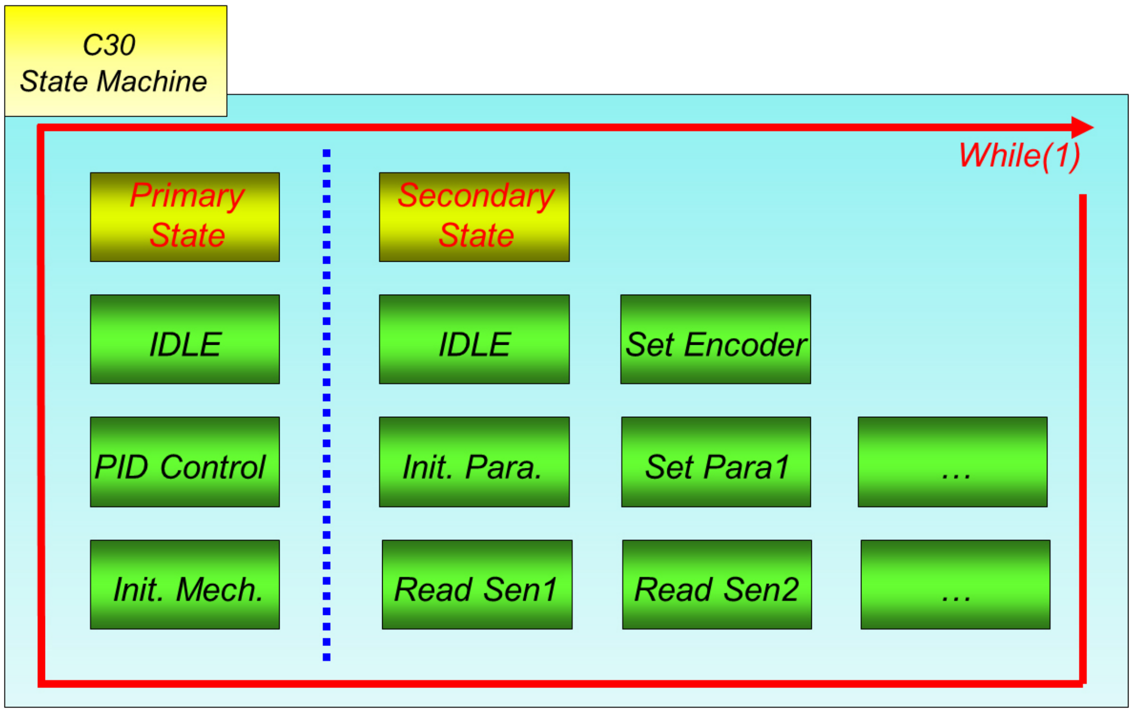


Figure 7-4. C30 state machine

In the state machine architecture, two types of states are defined. One is primary state and the other is secondary state. Computation of primary states is more time consuming and the secondary states are lighter and faster. One primary and one secondary state in each C30 controllers can be set at the same time; this helps the C30 controller to process two tasks alternately. Small-sized or emergent signals will not be jammed by time consuming tasks. The primary states contains the IDLE, PID Control, and Mechanism Initialization states and the secondary states contains the IDLE, Set Encoder, Initialize Parameters, Set Parameter, and Reading Sensors. Each C30 controller controls a motor with the onboard SA57 H-bridge power amplifier using PID control. SA57 H-bridge power amplifier can sustain 60V voltage and 8A continuous current; it is quite enough to be used to drive the motors of the robot legs. The sampling rate of PID update is set as 5 KHz and it is fast enough to control the motor without oscillation. The C30-SA57 motor control module is shown in Figure 7-5. C30 controllers also read the sensors near to them, such as the encoders of the motors, limit switches, and temperature sensors.

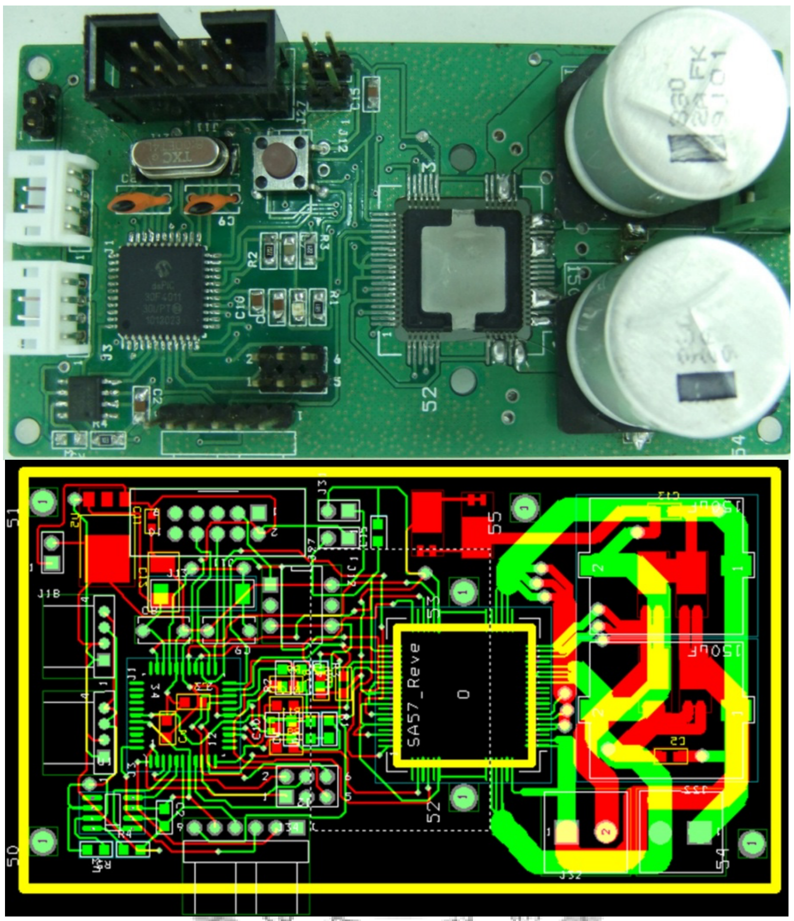


Figure 7-5. C30-SA57 motor control module

The baud rate of the CAN-Bus in each local network is 1Mbps. Since the standard package size of the CAN-Bus is 107bits, the theoretical value of transmission per second is 9345 times/second. The tested value of transmission per second is about 5200 times/second. For a six-node local CAN-Bus such as robot leg or robot arm, the maximum command update and data receive rate for each node is 5200/6 = 867 times/second. The command update and data receive from and back to C32 boards are both set as 200 times/second (5ms interval). Because the bandwidth required is 400 times/second and the CAN-Bus bandwidth is capable of 867 times/second, the bandwidth of the CAN-Bus is fast enough for controlling the motor and receiving sensor data of each node.

7.2.3 State Machine Architecture of C32 Controllers

Figure 7-6 shows the architecture and how the state machine of C32 controllers works. The photograph of the C32 controller is shown in Figure 6-3.

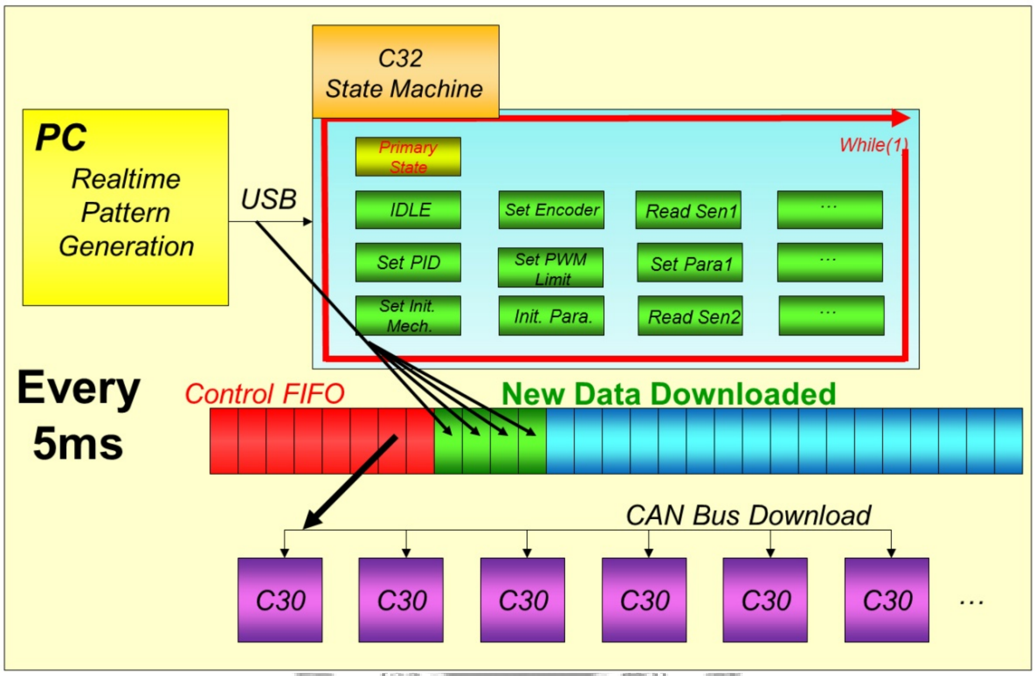


Figure 7-6. C32 state machine and the FIFO queue

The state machine architecture of the C32 controllers is similar to that of the C30 controllers. Different from the C30 controllers, C32 controllers focus on the coordination between the main personal computer and the local controllers. C32 controllers do not have to control motors and initialize the mechanisms, thus only the primary state are designed in its state machine architecture. The states of C32 include IDLE, Set PID, Set Initialize Mechanism, Set Encoder, Set PWM Limit, Initial Parameters, Set Parameters, Read Sensors, etc. These states help the computer to set the command and the states of C30 local controllers and read local sensors.

If the motors in local network are controlled by the main personal computer directly, asynchronous and time shifting problem will occur. The asynchronous problem occurs because the personal computer can only control the local motors through RS232 or USB if no other devices are installed to the computer. Without specially defined protocol and circuit, RS232 and USB must send the command one-by-one to the motors. The time shift problem is since the timing control precision of a personal computer is about 1~2ms, the sampling period will be longer or shorter. To solve these problems, a control FIFO queue is allocated in each C32 controller. It is designed that the personal computer processes motion planning and solves IK for four time steps in advance and then sends the four future joint trajectories to the C32 FIFO queue. This procedure triggers every 20ms and the C32 controller sends joint trajectories to all local nodes every 5ms. With the control FIFO queue, the encoder commands of local controllers can be updated with precise time interval. The C32 USB-to-CAN-Bus adaptor uses 12Mbps baud rate in the USB side and 1Mbps in the CAN-Bus side. The test results show the USB side can execute the transmission of 512Bytes package for about 500 times per second. It is also enough for achieving the uploading/downloading requirement for controlling a humanoid robot. Because the CAN-Bus transmits signal through broadcast, all nodes in the local network can receive the trigger signals to change their target encoder position at the same time. Simultaneous motion control can be achieved using CAN-Bus. Thus, with the control FIFO queue in C32 controllers, the asynchronous and time shifting problem of controlling local nodes using personal computer directly is solved.

7.3 Experiments

In this section, the experiment results using the proposed walking pattern generator and control system are discussed. The performances of LQSI controller using optimized and constant C_z trajectories are also compared. The experiment settings is shown as

Table 7-3. Settings of the experiments

	Experiment 1	Experiment 2	Experiment 3
Scenario	Straight walking	Straight walking	Robot hung on a shaft
Period per step	6 seconds per step	6 seconds per step	6 seconds per step
Double support phase	1.8 seconds	1.8 seconds	1.8 seconds
Single support phase	4.2 seconds	4.2 seconds	4.2 seconds
C_z trajectory	Constant	Optimized	Robot hung on a shaft
Average C_z per step	497.56mm	507.34mm	Robot hung on a shaft
Pattern generator	LQSI	LQSI	LQSI

Table 7-3 shows the settings of the experiments, experiments 1 and 2 are used to compare the performance between the walking patterns with constant and optimized C_z trajectory. Experiment 3 is used to see if the robot is hung on a shaft, how the performance of joint angle tracking changes.

In each experiment, all segments of the input ZMP trajectory are connected smoothly using 9-degree polynomials to ensure the trajectories are totally differentiable. By doing this the robot can move smoother than just setting the ZMP position directly under the center position of the stance foot.

7.3.1 Tracking Performance of Joint Angles

The tracking performances of joint angles are compared using the sensor feedback data in experiments 2 and 3. In the experiments, all settings are the same except the walking status of the robot. In experiment 2, the robot walks on the ground; in experiment 3, the robot is hung on a shaft. Since the motors of robot legs do not need to support the weight of the robot, joint load in experiment 3 is smaller than experiment 2. Therefore the joint angle tracking performance is better in experiment 3. Since the results of each leg are very similar, only the joint angle tracking results of left leg is shown in the following figures.

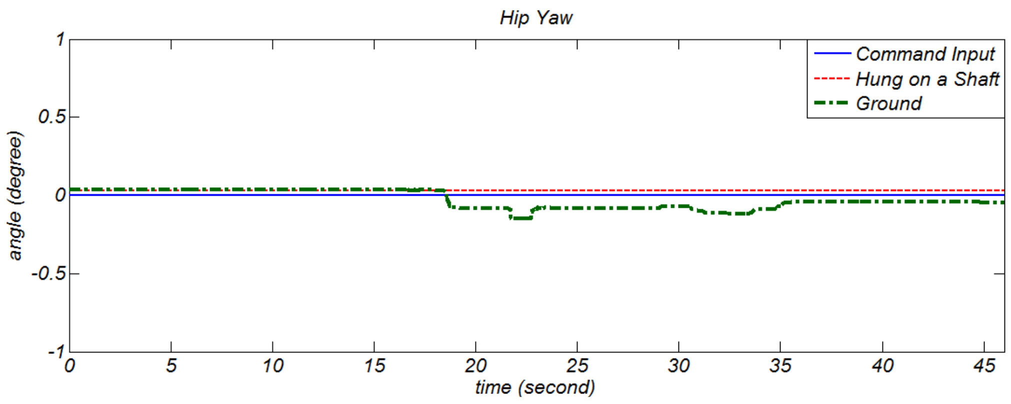


Figure 7-7. Tracking results of hip yaw axis

In Figure 7-7, since the robot is walking straightly, the command input of yaw axis is zero. The local controllers use PID control to track the command reference. Thus the encoder feedback in experiments 2 and 3 has steady state errors. In addition, because the motors must support the body weight of the whole robot, the tracking error is larger in experiment 2.

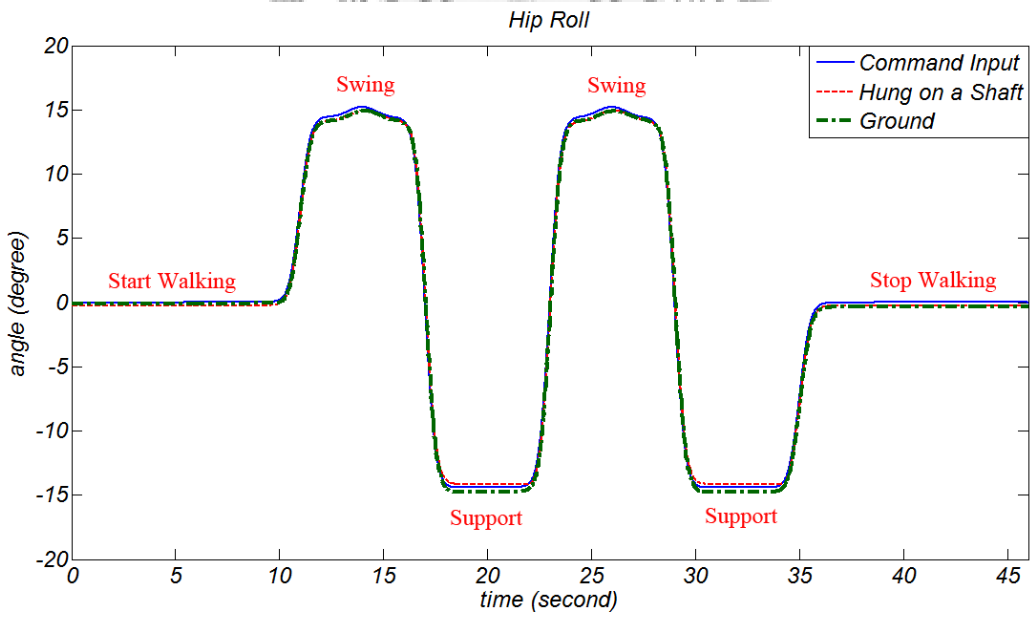


Figure 7-8. Tracking results of hip roll axis

Figure 7-8 shows the tracking results of hip roll axis. As we can see, in the swing phase, the tracking errors of experiments 2 and 3 are similar since the loading of the hip roll axis in both experiments are similar. Different from the swing phase, since the loading of the axis in single support phase is larger in experiment 2, the tracking error is also larger. The tracking performances of the other axes have the same characteristics as the hip roll axis. They are shown in Figure 7-9 to Figure 7-12. In the figures, since the values of errors are much smaller than the values of joint angles, they cannot be seen very clearly. To show the magnitude of the errors, their mean absolute errors will be listed in Table 7-4.

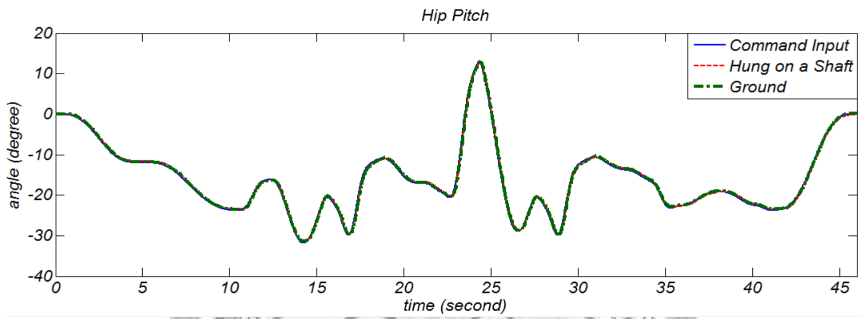


Figure 7-9. Tracking results of hip pitch axis

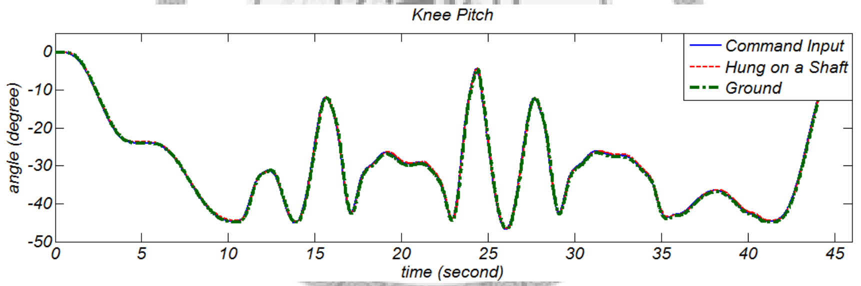


Figure 7-10. Tracking results of knee pitch axis

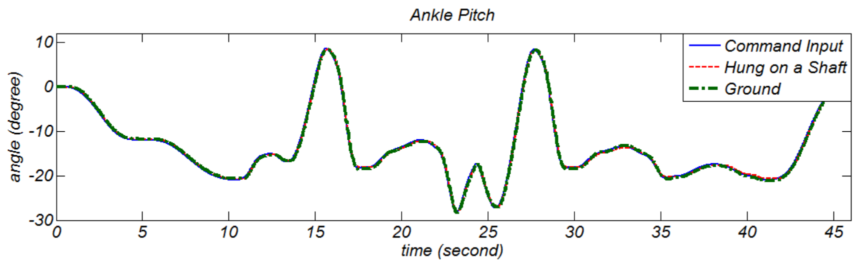


Figure 7-11. Tracking results of ankle pitch axis

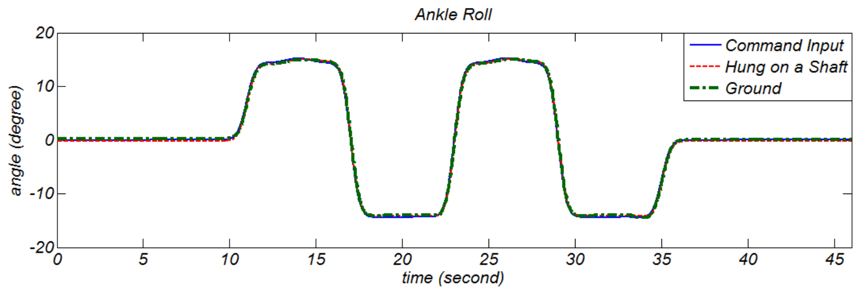


Figure 7-12. Tracking results of ankle roll axis

The torques on the axes can be seen as the disturbances to local PID controllers. Thus the tracking errors in experiment 2 are larger than that in experiment 3. The same result can be observed by calculating the mean absolute error in experiments 2 and 3, as shown in Table 7-4.

Table 7-4. Mean absolute joint trajectory tracking error in experiments 2 and 3

	Experiment 2 (Ground)	Experiment 3 (Hung on shaft)
Axis 01–Left Hip Yaw	0.073 degrees	0.030 degrees
Axis 02–Left Hip Roll	0.299 degrees	0.234 degrees
Axis 03–Left Hip Pitch	0.258 degrees	0.239 degrees
Axis 04–Left Knee Pitch	0.342 degrees	0.213 degrees
Axis 05–Left Ankle Pitch	0.244 degrees	0.214 degrees
Axis 06–Left Ankle Roll	0.254 degrees	0.232 degrees
Axis 07–Right Hip Yaw	0.043 degrees	0.044 degrees
Axis 08–Right Hip Roll	0.308 degrees	0.231 degrees
Axis 09–Right Hip Pitch	0.250 degrees	0.235 degrees
Axis 10–Right Knee Pitch	0.321 degrees	0.204 degrees
Axis 11–Right Ankle Pitch	0.274 degrees	0.201 degrees
Axis 12–Right Ankle Roll	0.420 degrees	0.355 degrees

In Table 7-4, only the hip yaw axis has larger mean absolute error in experiment 3. It is because the yaw axes use smaller motors than other axes. The mechanism design of the yaw axes also causes larger friction forces. Due to these two reasons, the PID controller can only give voltage commands that can overcome the friction forces when the error is large enough. The position errors of yaw axis are too small to be eliminated by a PID controller in experiments 2 and 3. Except this, the mean absolute errors are larger in experiment 2 because the loading is larger.

Generally, the tracking performances in the experiments can achieve stable robot walking. PID control algorithm is capable of local joint trajectory tracking, but for even better performance, more complex local joint controllers must be used in the future to improve the tracking performance, such as impedance control, current control, and torque control.

7.3.2 Tracking Performance of COG trajectory

The tracking performances of COG trajectories are compared using the sensor feedback data in experiments 1 and 2. Using the encoder feedback joint trajectories, the COG trajectory of the robot can be calculated and estimated. Figure 7-13 and Figure 7-14 show the COG trajectories in 3D directions in experiments 1 and 2.

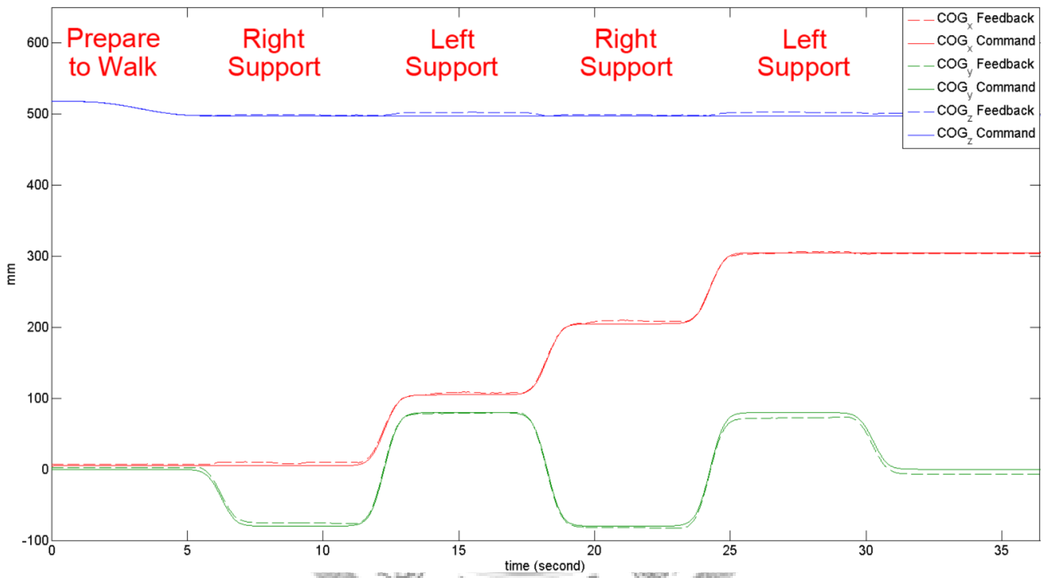


Figure 7-13. COG trajectory in experiment 1 (LQSI with constant C_z)

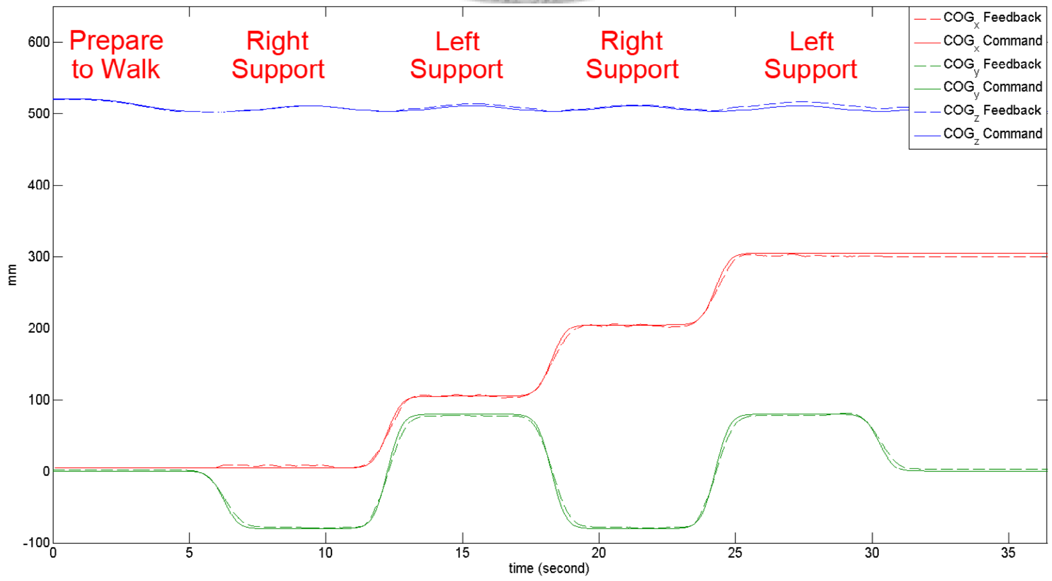


Figure 7-14. COG trajectory in experiment 2 (LQSI with optimized C_z)

The COG tracking performances of the proposed real-time control system has been tested many times. The trends of the errors are slightly different in each test, but they are all bounded and the robot can complete the whole walking motion in these tests. The main reason that the errors have different trend is the initial placement of the robot. Each time the robot is released and placed on the ground from the shaft, the landing status is slightly different. This makes the COG and ZMP trajectories shift slightly left or right. Figure 7-15 shows the robot and the shaft.

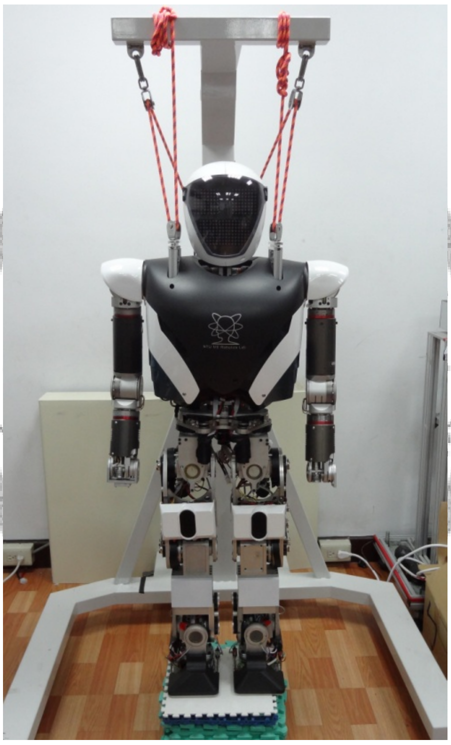


Figure 7-15. The robot and the shaft

In Figure 7-13 and Figure 7-14, the COG trajectory tracking errors are the combined effect of joint tracking errors. The tracking performance of joint controller directly affects the tracking performance of COG trajectory. Because the tracking performances of the LQSI controller with optimized or constant C_z trajectories are similar, only the COG errors of experiment 2 are discussed in the following. By discussing the tracking performance of COG trajectory, several future works that can improve the performance are found.

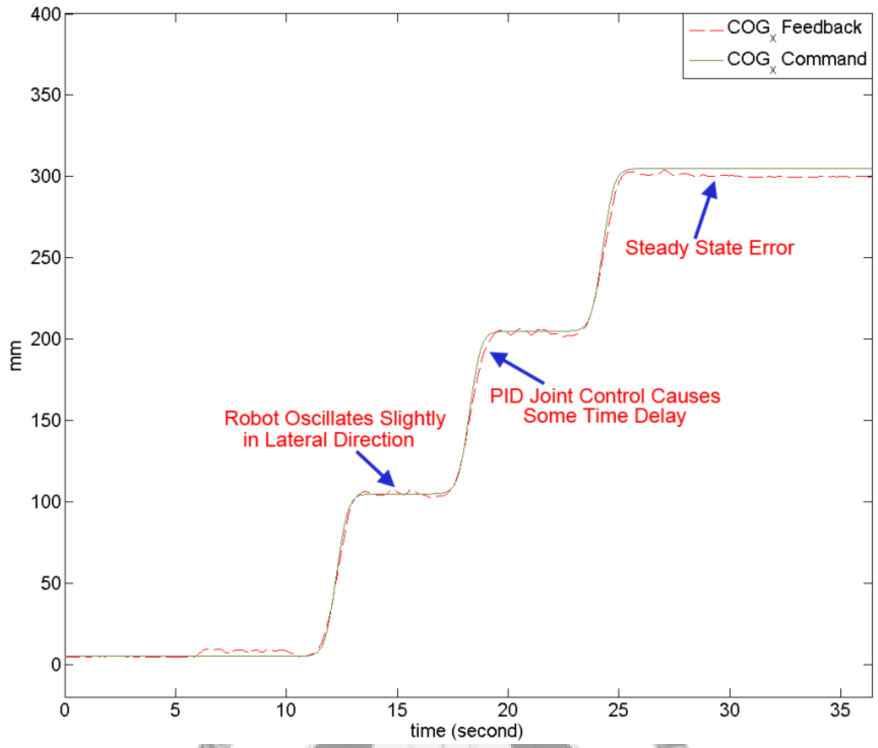


Figure 7-16. Sagittal COG trajectory tracking in experiment 2

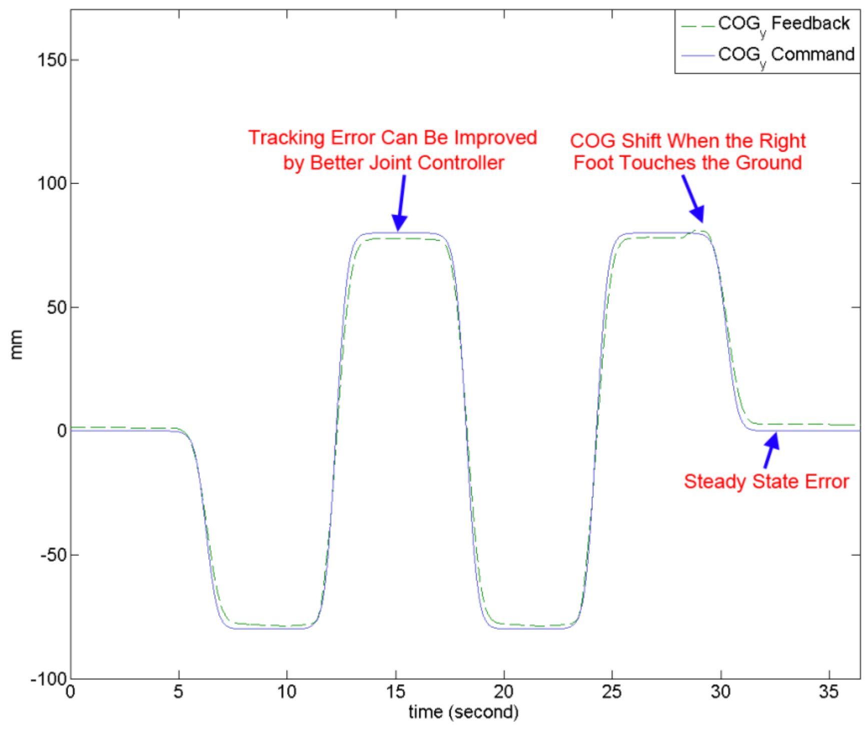


Figure 7-17. Lateral COG trajectory tracking in experiment 2

Figure 7-16 shows the sagittal COG trajectory tracking results in experiment 2. The COG trajectory follows the desired COG trajectory with some oscillation. In each single support phase, the desired COG trajectory is almost constant, and the feedback COG trajectory oscillates slightly around the desired value. Since PID controller is used as local controller to track the motor position, feed-forward compensation must be used to eliminate the effect of time delay. The steady state error is caused from the friction of the robot joints. This can be improved by designing an online COG feedback observer to achieve closed loop COG feedback control.

In Figure 7-17, the same problems occur as in Figure 7-16, such as the steady state error and tracking error. Excepting these problems, it is observed that a small COG shift occurred when the robot entering double support phase and prepare to stop walking. The robot place the right foot on the ground and the COG shifts left about 2.5mm. This is because that when the right foot is landing, it also pushes the robot left. An undesired disturbance occurs when switching to double support phase. To solve this, a landing controller or mechanism dealing with the landing problem must be used in the future.

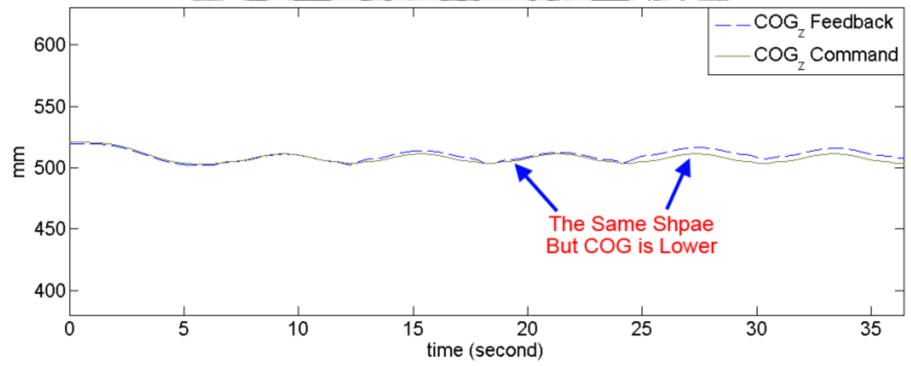


Figure 7-18. Vertical COG trajectory in experiment 2

In Figure 7-18, another point that can be improved for the COG tracking performance is observed. In all experiments, the vertical COG trajectories calculated from encoder feedback are all lower than the command reference. The reason is quickly found: the gravity force. The gravity force pulls the robot lower. For PID controllers, it is an unknown disturbance. Thus the gravity force causes a steady state error in vertical direction. This problem can be fixed by using gravity compensation control. To achieve this, the torque/current control must be implemented as joint controller and the sensor used for measuring torque/current must be installed.

7.3.3 Tracking Performance of ZMP trajectory

The ZMP tracking performance is discussed in the following this section. The feedback ZMP trajectory can be calculated using the six-axis force/torque sensor installed in the ankle of the robot legs. Figure 7-19 to Figure 7-22 show the force and torque data collected form the six-axis force/torque sensor in experiments 1 and 2. Since the original data from the sensor are very noisy, the data shown in the figures are filtered using Kalman filter.

In Figure 7-19 and Figure 7-20, left support phase, right support phase and double support can be observed clearly. The loading in z direction of the sensor installed in each leg becomes larger when the corresponding leg is the support leg. The forces in x and y directions cause the COG shifting in horizontal directions; they are relatively smaller than the force in z direction.

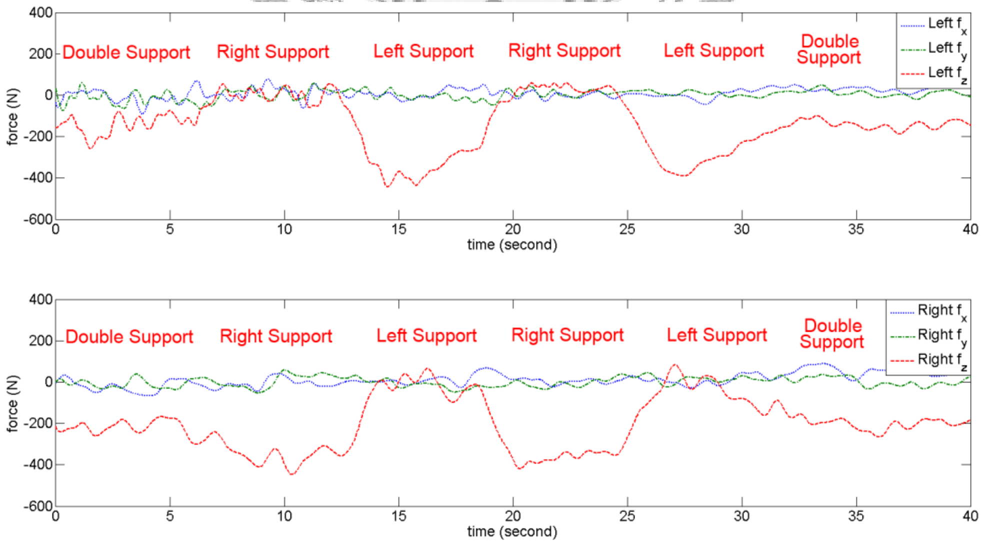


Figure 7-19. Force data (LQSI with constant C_z trajectory)

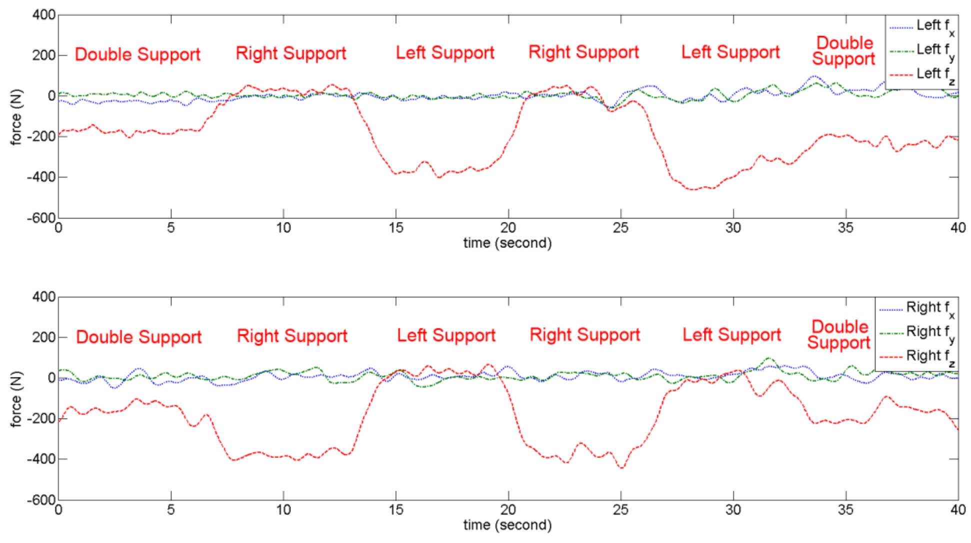


Figure 7-20. Force data (LQSI with optimized C_z trajectory)

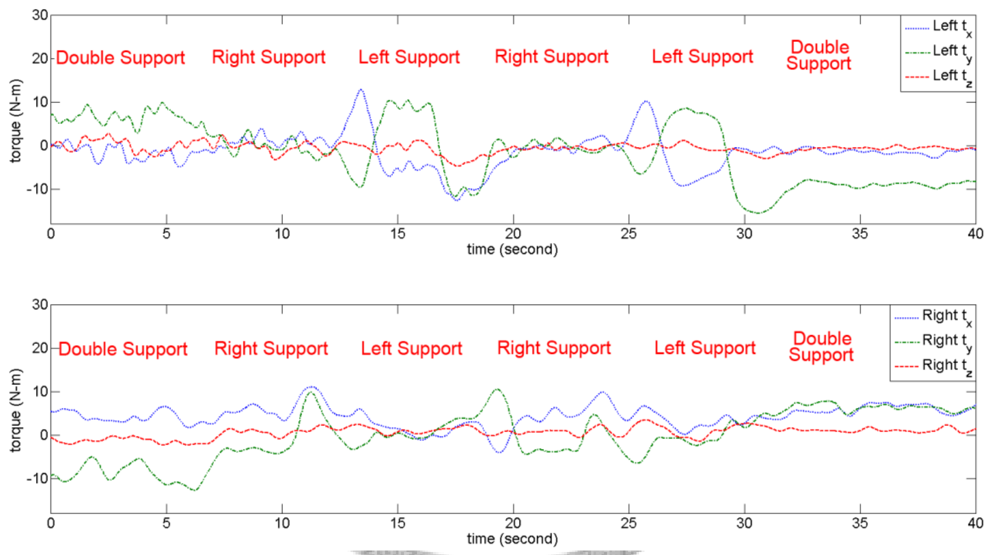


Figure 7-21. Torque data (LQSI with constant C_z trajectory)

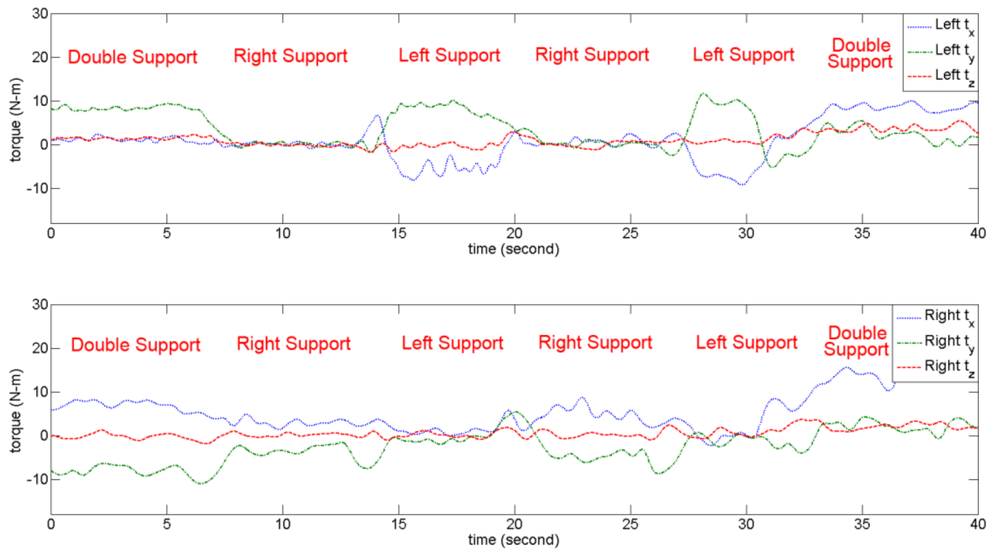


Figure 7-22. Torque data (LQSI with optimized C_z trajectory)

In the single support phases in Figure 7-21 and Figure 7-22, the value change of torque trajectories on the stance ankle in x and y directions can be observed in the figure. They are generated by the gravity force and the desired COG motion. In double support phases, since the legs are both contacting with the ground, the torques are distributed on both feet. The relationship between the torque trajectories cannot be observed by our eyes directly. The combined results of all forces and torques will be discussed using the ZMP trajectory in the following.

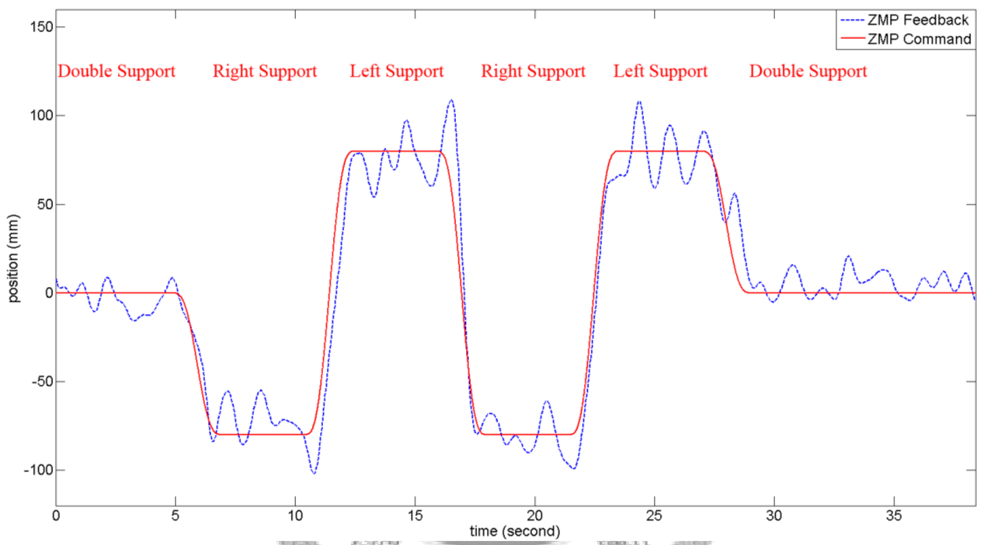


Figure 7-23. Lateral ZMP in experiment 1 (LQSI with constant C_z)

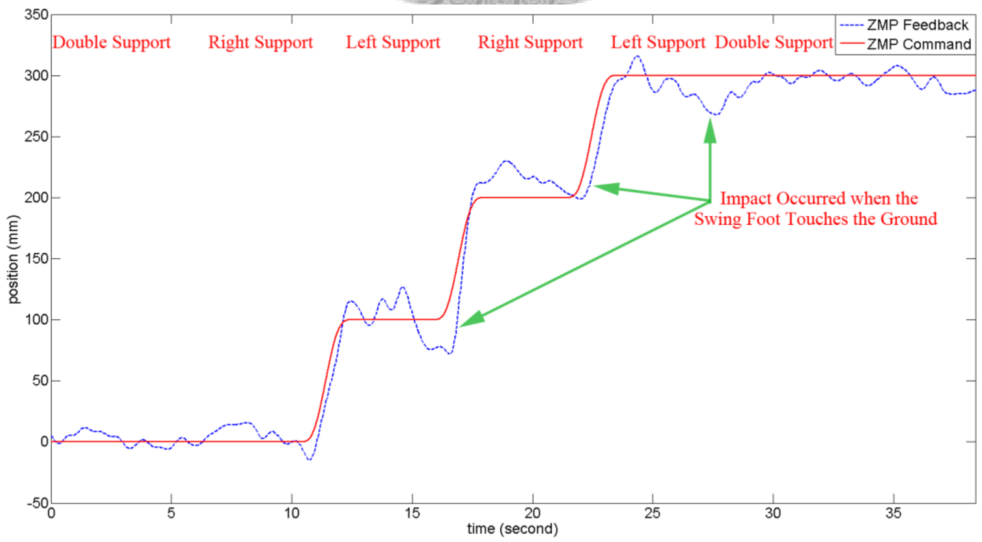


Figure 7-24. Sagittal ZMP in experiment 1 (LQSI with constant C_z)

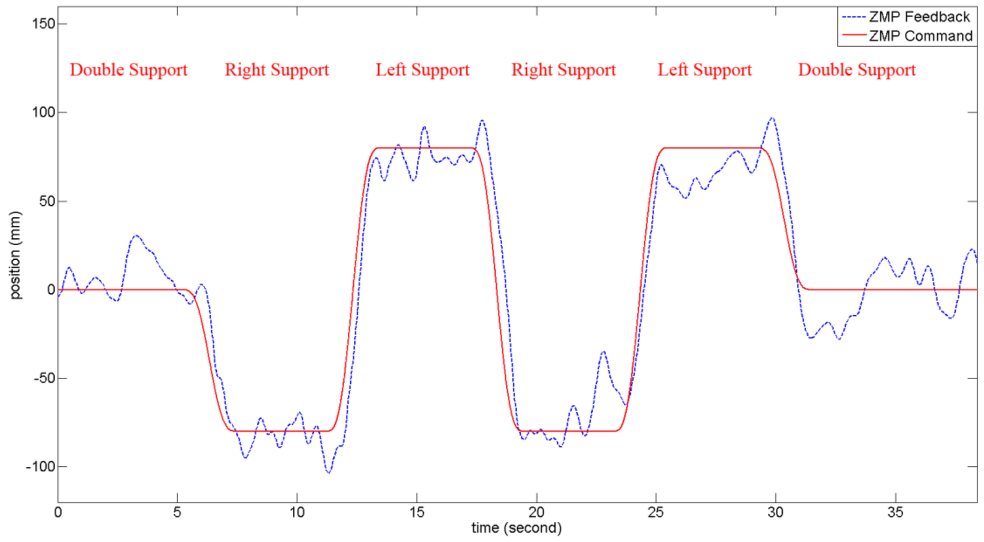


Figure 7-25. Lateral ZMP in experiment 2 (LQSI with optimized C_z)

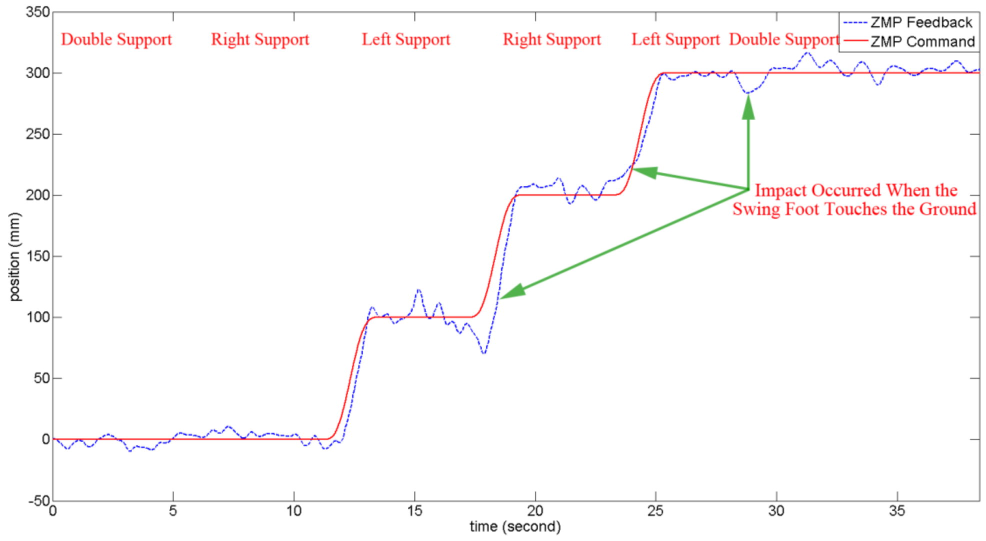


Figure 7-26. Sagittal ZMP in experiment 2 (LQSI with optimized C_z)

Figure 7-23 to Figure 7-26 show the sagittal and lateral ZMP trajectories using LQSI controller with constant and optimized C_z trajectory. The input ZMP trajectories in the experiments are moving under the robot foot. They are not always located in the center of the robot foot. This will reduce the walking stability slightly but get smoother walking patterns. Also, since the ZMP trajectories are always located in the support polygon of the robot, the robot can still walk stably. In the figures, the feedback ZMP trajectories can track the ZMP reference with some oscillations and small time delay. The oscillations are caused by the noise of the six-axis force/torque sensor and the impact when each time the swing foot touches the ground. Better data processing of the sensor signals and a six-axis force/torque sensor with lower noise can be used to improve the ZMP feedback signal. And force/impedance control can be used to reduce the impact when the swing foot is landing in the future. With the ZMP tracking results, the LQSI controller is capable to generate COG/ZMP walking patterns with both constant and varying C_z trajectories. The stability of the walking pattern generated with LQSI controller is verified.

7.3.4 Calculated Knee Joint Torque

The knee joint torque can be estimated by using the sensor feedback of six-axis force/torque sensor and the equations of Newton-Euler dynamics. The knee torque trajectories with constant and optimized C_z trajectory are compared in this section. Using the Newton-Euler dynamics Eqns. (2-19)-(2-28) and the encoder trajectories, the knee torque trajectories in experiments 1 and 2 are calculated as shown in Figure 7-27 and Figure 7-28.

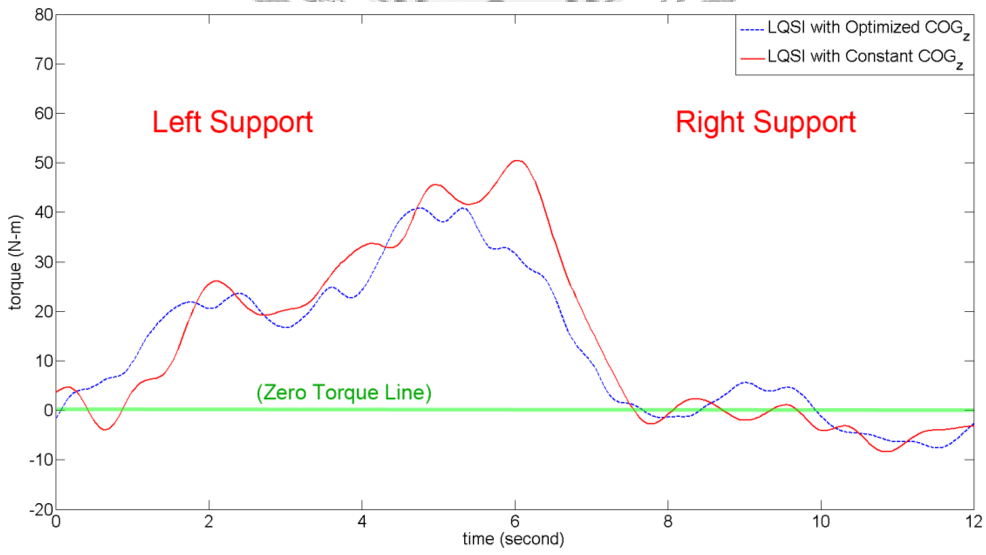


Figure 7-27. Left knee torque in experiments 1 and 2

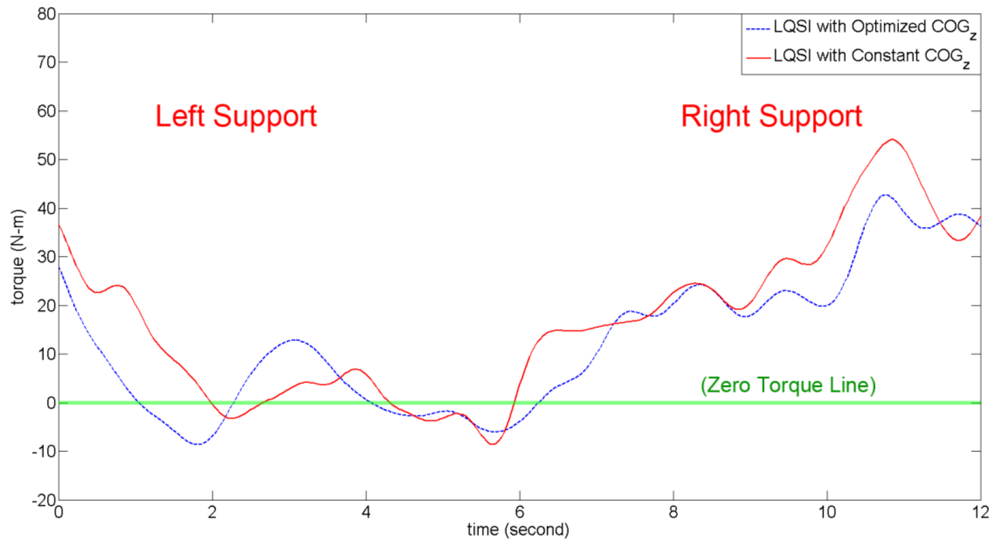


Figure 7-28. Right knee torque in experiments 1 and 2

The figures show the torque trajectories for two steps during the robot walking period. The total time is 12 seconds. Left leg is the stance lag in the beginning and the right leg becomes the stance leg since the 6th second. Because the walking pattern with optimized C_z trajectory has smaller knee joint rotation. Thus the knee joint is straighter and the joint torque is also smaller. The comparison of torque trajectories in experiments 1 and 2 are shown in Table 7-5.

Table 7-5. Comparison of torque performance

	Experiment 1 (Constant C_z)	Experiment 2 (Optimized C_z)	Improvement
Average C_z per step	497.56mm	507.34mm	
Mean absolute torque (whole period)	17.535N-m	14.901N-m	15.02%
Mean absolute torque (swing phase)	7.110N-m	5.623N-m	20.91%
Mean absolute torque (support phase)	27.960N-m	24.180N-m	13.52%

In Table 7-5, the mean absolute torque of the knee joints of both legs are calculated and compared. In the table, the values are the summations of absolute torque value of both knee joints in the whole period, swing phase, and support phase. For example, the mean absolute torque for support phase is the mean absolute torque of all the torque values of both left and right leg in their own support phase. The walking pattern with optimized C_z trajectory has better performance because of the straighter knee joint motion. With the experiment results, the improvement by using the optimized C_z trajectory is verified.

7.4 Summary

In this chapter, the specification, control architecture, and the experiment results are presented and discussed. The two robots shown in this chapter are used as control plant in the simulations and experiments in the whole dissertation. The real-time control architecture can update control commands and read sensors in every 5 millisecond. Using the proposed humanoid robot and the real-time control system, the performances of the LQSI controller and the optimized C_z trajectory are verified by analyzing the results of the experiments in this chapter.

Chapter 8 Conclusions and Future Works

The whole robot control and planning system are described and discussed in previous chapters. This chapter summarizes the whole dissertation in section 8.1. The main ideas and structures of all methods proposed in this dissertation are shown in this section. Section 8.3 shows the future works. This dissertation mainly discusses the kinematics, dynamics, and control of humanoids. Many other interesting topics such as artificial intelligence, machine vision, human-robot interaction, and the control of more difficult motion such as jumping and running can be implemented based on the concepts and theories proposed and discussed in this dissertation. Finally, section 8.2 is the conclusion of this dissertation. This dissertation ends here. It is exciting and delightful if the concepts and theories in this dissertation can help more researchers to implement or to compare their control systems.

8.1 Summary

In this dissertation, the walking pattern generation and its optimization are most focused. After the introduction in chapter 1, chapter 2 shows the basic knowledge used in the whole robot kinematics and dynamics control system. The following three chapters are all based on chapter 2. In chapter 3, the concept of the Fixed-Leg-Motion Jacobian is proposed and the global Jacobian matrix used to control the whole robot is designed. In chapter 4, the LQSI controller is proposed to generate optimized COG trajectories in sagittal and lateral directions. In addition to chapter 4, chapter 5 shows a method based on Newton-Euler method to optimize the COG height trajectory. Using the techniques and theories in chapters 4 and 5, 3D COG trajectory optimization is achieved. The next two chapters are the implementation parts of the dissertation. Chapter 6 describes the networking system can be used among different robot, personal computers, and workstations. The networking system is also used in local networks for robot control such as robot arms, hands, and legs. Chapter 7 describes the two robots used as simulation and implementation platforms in the dissertation, and also shows the detailed implementation of the proposed real-time planning and control system. The main theories and topics in this dissertation are listed below.

Fixed-Leg-Motion Jacobian and Global Jacobian

Fixed-Leg-Motion Jacobian matrix describes the relationship between each joint of the stance leg and each component of the kinematics and dynamics of the end-effectors. In chapter 3, the Fixed-Leg-Motion Jacobian is firstly proposed for the relationship between joint and end-effector position and orientation. After deriving this, the COG Jacobian and momentum Jacobian are also derived. The Fixed-Leg-Motion Jacobian is also applied to these Jacobian methods.

The concept of Fixed-Leg-Motion Jacobian is extended form the level of kinematics to dynamics. Compared with other methods to find the Jacobian describing the linearized relationship between the stance leg and other end-effectors, Fixed-Leg-Motion Jacobian is easy and fast since it uses the physical meaning of the equations to find the partial derivatives. After constructing all Jacobian matrices of the head, arms, legs, COG, and momentum, the global Jacobian matrix can be built to solve IK for the whole robot. With the global Jacobian matrix, whole body motion control can be achieved. The positions of the end-effector of the arms, head, swing leg, and COG, the orientation of the end-effector of the arms, head, and both legs, and finally the angular momentum in z direction of the proposed robots are controlled in this dissertation.

LQSI Controller

The LQSI controller is a linear quadratic control based controller with state incremental performance index. Since the humanoid robot is modeled as an inverted pendulum model, the robot model can be written in state-space form. The state matrices are constant matrices if the COG height is constrained to a constant value. Based on the view of potential energy, to constrain the COG height seems energy-saving. However, robot must cost energy to maintain its posture. The power consumption of the robot directly related to the joint torque and the current pass the motors. On the other hand, when using a walking pattern with constant COG height, the robot must bend the knee of stance leg while the swing leg rises.

Humans will not do this when normal walking because the knee sustain larger torque in order to keep the same COG height. Thus the changeable COG height trajectory is needed for more energy-saving and more human-like walking patterns. When the input COG height trajectory is not a constant, the state matrices of the inverted pendulum also become time varying. The inverted pendulum model also becomes a nonlinear time varying model. Optimal control can deal with nonlinear tracking problems well. By using ZOH method, the proposed LQSI controller can generate walking patterns with changeable COG height trajectory, as shown in chapter 4. At this stage, walking pattern generation with arbitrary assigned COG height trajectory is achieved. COG trajectories in sagittal and lateral directions are optimized using the LQSI controller. COG height trajectory is optimized with the method proposed in chapter 5.

3D COG Trajectory Optimization

For smooth walking, the arbitrarily assigned trajectories to the LQSI controller must be smooth and continuous, so the optimized trajectory should be also smooth and continuous. In fact, smooth and continuous COG height trajectories can be generated according to the status of the ground directly. Robot can raise their COG when stepping onto stair or walk over a small obstacle. To decide COG height trajectory directly has several advantages: fast, easy, and modifications can be done with tuning some simple parameters. On the other hand, optimization of COG height trajectory is required when humanoid robots need to repeat the same trajectory many times. For example, walking, it is the most repeated function for humanoid robots. As humanoid robots become more common in the future, optimization of walking or other motions are required in order to reduce the power consumption for longer operation of the batteries like the notebooks nowadays. The optimization of COG height trajectory in this dissertation is to minimize a cost function including the square of torque and the cost function of joint limit. The differentiation of the cost function with respect to the COG height is the index to update the COG height trajectory but it cannot be found directly. To find the differentiation, the derivative of the joint angles with respect to the COG height is calculated first. This is done by using the physical meaning of the pseudoinverse of the global Jacobian matrix of the robot. The second step is to calculate the derivative of the joint torques and the cost of joint limit with respect to the joint angles, by using the equations of Newton-Euler dynamics. Using the results of these two steps, the derivative of the cost function can be found, and the optimization can be processed by updating the COG height trajectory until it converges. Using a personal computer with Intel Core^{TM} i5 CPU, the procedure of the COG height training costs about two minutes until the COG height trajectory converges. The trained COG height trajectory can be input to the proposed planning and control system to achieve walking pattern generation and control with optimized 3D COG trajectory.

Networking and Implementation

The networking and implementation of the proposed humanoid robot are described in chapters 6 and 7. The networking system of the robot is based on a priority oriented networking algorithm, RTNET, to achieve real-time control and communication. RTNET checks the priority of each communication and data to decide whether the interrupts when transmitting data should be triggered or not in order to let the data with higher priority can be transmitted earlier. RTNET can be used to achieve communication among personal computers, workstations, and robots through Ethernet and it can also be used to control local nodes and read local sensors through CAN-Bus.

Detailed descriptions of RTNET are shown in chapter 6 and detailed descriptions of implementation using PIC C32 and PIC C30 are discussed in chapter 7. Chapter 7 also shows the specifications of robot platforms and how the state machines of each controller works in the proposed robot system. The state machines always wait for commands from higher level controllers and execute their assigned missions periodically. In the proposed control system, FIFOs are used as control buffer to improve the time accuracy of motion control and sensor reading. Using the algorithm and methods shown in these two chapters, real-time control and communication for robots can be achieved.

8.2 Conclusions

In this dissertation, several algorithms and methods for walking pattern generator are proposed. A global Jacobian matrix is proposed for solving IK with whole body motion constraints. The concept of Fixed-Leg-Motion Jacobian is proposed for simplifying the construction of the global Jacobian matrix; no complex computation and coordinate transformation is required with the Fixed-Leg-Motion Jacobian method. Based on the proposed IK solver, the proposed LQSI controller serves as the COG/ZMP walking pattern generator of the humanoid robots to generate COG patterns satisfying the COG/ZMP equations. In addition, since the proposed LQSI controller can generate horizontal COG patterns in real-time with arbitrarily assigned COG height and 3D ZMP trajectories, real-time walking pattern generation can be achieved. Based on the LQSI controller and the whole robot motion solver, the proposed pattern generator can generate walking patterns with varying COG height and ground status. This enables the optimization of the COG patterns including vertical direction. Using the idea of cost function and Newton-Euler dynamics, a COG height optimization method is proposed in this dissertation.

Besides of the walking pattern generation algorithm proposed in this dissertation, a real-time control system is also built. By designing and implementing the architecture of local control networks and state machines, real-time robot walking pattern generation and control can be achieved. Simulation and experiment parts in this dissertation also show the performance of the proposed methods. By using the algorithms and methods in this dissertation as motion generation and control engines, many further researches can be carried out. The future works parts shows the researches can be developed directly using the results in this dissertation. There are still many interesting and exciting researches waiting to be developed. It is gratifying if the algorithms and methods in this dissertation can contribute to the development of robot technology, and we hope the robot technology can become better by continuing our works in the future.

8.3 Future Works

America and Japan are the two main pioneer countries of robotics research. Many new and novel robot technologies are from these two countries. By comparing these researches as benchmarks, the quality of our research and the parts must be improved can also be found. Another advantage to compare these researches is that we can find and use mature and well-known technologies to save time of research and discover new techniques and topics based on these technologies, as Newton said: "*If I have seen a little further it is by standing on the shoulders of Giants*." Based on the comparison of existing researches and our current research results and the view of future applications, there are several future works to extend and improve the proposed algorithm and methods, including momentum planning and control, jumping and running control using LQSI controller, optimized COG height trajectory database, sensor fusion and stability control, force/impedance control and joint control, and autonomous navigation.

Momentum Planning and Control

In this dissertation, the only constraint of angular momentum in z direction is applied in the IK solver to reduce the slipping of robot in z direction. If the trajectory of angular momentum in z direction can be planned well, the robot can turn smoother and more natural. However, to further improve the motion behaviors in other directions of translation and rotation, a good planning mechanism is required or the motion behaviors solved by the IK solver will become even worse than the IK solver without momentum constraints. Thus a good planner for linear and angular momentum trajectories must be built before more momentum constraints are added and applied to the IK solver.

Jumping and Running Control Using LQSI Controller

Since the proposed LQSI controller can solve COG patterns with ground and COG height change, the control architecture can be used for jumping and running control. To achieve this, modifications of the proposed LQSI controller are required. The first idea for modification is to change the method of discretization. In the proposed LQSI controller, ZOH discretization is used to discretize the continuous nonlinear state-space model of the inverted pendulum model. It is quite enough for robot walking because the COG height does not varying very severely. As shown in chapter 4, the proposed LQSI controller failed when the maximum acceleration is 8337mm/s^2 (the gravity acceleration is 9810mm/s^2); it is expected better to use a triangular hold to discretize the state-space model because it can fit the original curve better, as shown in Figure 8-1.

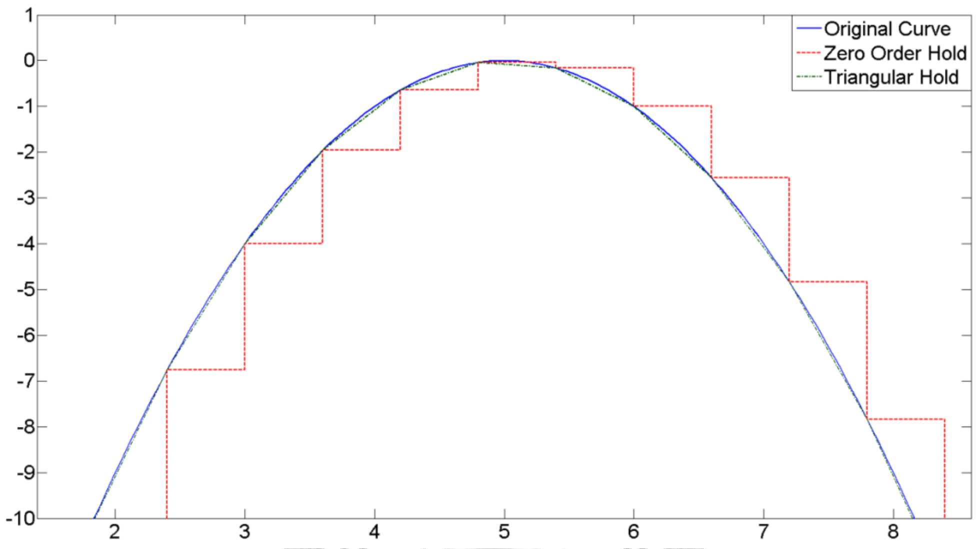


Figure 8-1. Zero order hold and triangular hold for COG height

In Figure 8-1, the drawback of using ZOH method is exaggerated by choosing a large sampling time. Although the performance of ZOH method can be improved by choosing a small sampling time, triangular hold method can always fit the original curve better.

The LQSI controller can generate walking patterns except flying phase when running and jumping. In the flying phase, since the robot cannot use any reaction force from ground to change the state of COG, the only thing the robot can do is to prepare landing. The LQSI controller does not work in this phase. New control algorithms must be used in flying phase in the future to achieve running and jumping control. When the robot is preparing to take off, the vertical acceleration will approach the magnitude of gravity acceleration and then exceed it. The robot will take off when the COG speed is larger than the speed of feet stretching in vertical direction. If the feet of the robot reach the joint limit before the COG is accelerated fast enough, the robot cannot jump and run well. On the other hand, if the robot takes off too early (if the motors of the robot can accelerate the COG in a short time); the robot also cannot jump well. To find a balance between the COG speed and the feet stretching speed, a good planner of COG and feet speeds for running and jumping must be designed and proposed in the future.

Optimized COG Height Trajectory Database

Since the proposed COG height trajectory optimization method can optimize the COG height trajectory with given 3D ZMP trajectory. Natural walking with optimized joint torques can be achieved. Due to large computational cost, real-time COG height optimization cannot be achieved. Each execution of optimization costs around two minutes on a personal computer with Intel Core^{TM} i5 CPU.

An idea to achieve real-time optimized 3D COG trajectory walking is to construct an optimized COG database. By training the optimized 3D COG trajectory under different 3D ZMP inputs, a database can be collected. This is like the technique of human motion capture, but can be used on different robots because it is a model based method. Different models of different robots can be input to the optimization engine, thus the proposed algorithm can be used on a legged robot which is not human-like. When using a database, searching and interpolation algorithms become important. How to search and interpolate the trajectories in the optimized 3D COG trajectory database will be the key to achieve real-time 3D optimized COG trajectory and walking pattern generation in the future.

Sensor Fusion and Stability Control

For walking control, the most used sensors used to improve walking stability are force sensors and IMUs (inertial measurement unit). Force sensors are used to measure the ground reaction forces and the ZMP position for feedback control. IMUs are used to measure the rotation and acceleration of the robot. Many researchers [16][53][54] proposed their methods to stabilize their robot using these two sensors. Different form the view of stabilizing the robot, the localization and obstacle avoidance are also important for robots. Laser range finders and cameras are used to achieve localization and obstacle avoidance of humanoid robots [72][80][96]. Although humanoid robots can walk stably with offline generated walking patterns and known environments, they are still far from walking stably in unknown environments. A humanoid robot must know the information and status of the environment around it and a powerful artificial intelligence is required for walking fully autonomously. To know the information of the environment, many sensors are required. Humans use their eyes and feet to see and to feel the shape and condition of the environment. Similar sensors such as cameras and force sensors can be used to achieve the same thing. On the other hand, the cooperation of eyes and semicircular canals in the ears can further stabilize human walking. For a humanoid robot, this can be achieved by the cooperation of cameras and IMUs. There are many other types of sensors can be used on humanoid robots, such as low-cost infrared sensors (photoelectric distance sensors), ultrasonic sensors, or Microsoft Kinet.

Most researches nowadays use just one kind of sensor to stabilize the robot; the improvements of robot walking are also limited by the sensors. Multi-sensor fusion algorithms and techniques can be discussed in the future in order to further improve the robustness of robot walking for different environments.

Force/Impedance Control and Joint Control

In the experiment parts in chapter 7, several future works are found by observing the experiment results in order to improve the tracking performance and the walking stability of the robot.

The first point must be improved is the joint tracking error and the COG tracking error. Since the local joint controllers are PID controllers, when the joint angle command is sent to each joint controller, the joint will move when the error is large enough to generate a large enough voltage input for overcoming the joint friction. This causes a time delay when controlling the robot. Tuning the P gain of the controller larger can reduce the time delay and have better tracking performance, but the oscillation of the robot also become larger. In order to solve this problem, feed-forward control and control algorithms considering the friction in the model can be used. The joint tracking performance directly affects the COG tracking performance. COG tracking performance can also be improved by improving the joint tracking performance.

The second point must be improved is the ZMP feedback and tracking performance. Since the force and torque feedback signals are very noisy, the captured data must be filtered before being used. To use the data for feedback control, a better data processing and filtering algorithm or a sensor with higher signal/noise ratio can be used to solve this problem. The landing impact problem of swing foot is also observed. It affects the stability of walking more when the robot is walking faster. This problem can also be improved by improving the joint tracking performance. We can further improve this problem by applying force/impedance control when the swing foot is landing.

In some recent researches, the topics of COG/ZMP feedback control [8][88] and force/impedance control [113][129] are discussed for many purposes. We can also improve the performance of the proposed walking control system based on existing methods. By improving the joint tracking performance and applying force/impedance control methods, the robot can have better walking stability. Better stability and more robustness for long-term operation of humanoid robot can be achieved in the future.

Autonomous Navigation

Autonomous navigation algorithms are very mature and complete. In the last two decades, SLAM (simultaneous localization and mapping) and motion planning [19][20][43] are widely discussed on mobile robot platforms. It is rarely discussed on humanoid robot since the development time is faster and maintenance cost is lower for mobile robots. It is much faster and easier for developing core algorithms and techniques on a mobile robot. On the other side, researches of humanoid robots focus on the mobility in these years. The mobility of humanoid robots becomes more capable and even better than mobile robot in some environments. In addition, the weight and price of laser range finders become lighter and cheaper. Because of these reasons, autonomous navigation becomes more important and more possible for humanoid robots if the robots need to go farther. The only gap of autonomous navigation algorithms must be overcome from mobile robots to humanoid robots is the difference of their motion patterns. The position and orientation of the cameras or laser sensors installed on the humanoid robot will wave when the robot is walking or even running. This is different form the motion of mobile robots. To achieve autonomous navigation of humanoid robots, SLAM or visual SLAM and motion planning algorithms can be used and implemented in the future.

REFERENCES

[1] A. Albers, S. Brudniok, J. Ottnad, C. Sauter, K. Sedchaicham, "Upper Body of a New Humanoid Robot - The Design of ARMAR III," *IEEE/RAS Int. Conf. on Humanoid Robots,* Genova, Italy, pp. 308-313, 2006.

[2] M. Arbulu and C. Balaguer, Real-Time Gait Planning for Rh-1 Humanoid Robot, Using Local Axis Gait Algorithm, *IEEE/RAS Int. Conf. on Humanoid Robots*, PA, USA, pp. 563-568, 2007.

[3] M. Arbulu, C. Balaguer, C. Monge, S. Martinez, and A. Jardon, "Aiming for Multibody Dynamics on Stable Humanoid Motion with Special Euclideans Groups," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan, pp. 691-697, 2010.

[4] M. D. Ardema, "Kinetics of the Mass Center of a Rigid Body," in *Newton-Euler Dynamics*, 1st ed., NY, USA, Springer Science+Bisness LLC, 2005, ch. 6, pp. 135-164.

[5] Y. Ayaz, T. Owa, T. Tsujita, A. Konno, K. Munawar and M. Uchiyama, "Footstep Planning for Humanoid Robots Among Obstacles of Various Types," *IEEE/RAS Int. Conf. on Humanoid Robots*, Paris, France, pp. 361-366, 2009.

[6] A. Austermann, S. Yamada, K. Funakoshi, and M. Nakano, "How Do Users Interact with a Pet-Robot and a Humanoid?" *CHI 2010: ACM Conf. on Human Factors in Computing Systems*, Atlanta, GA, USA, pp. 3727-3732, 2010.

[7] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida, "Real-time Replanning Using 3D Environment for Humanoid Robot," *IEEE/RAS Int. Conf. on Humanoid Robots*, Bled, Slovenia, pp. 584-589, 2011.

[8] R. Beranek, H. Fung, and M. Ahmadi, "A Walking Stability Controller with Disturbance Rejection Based on CMP Criterion and Ground Reaction Force Feedback," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, CA, USA, pp. 2261-2266, 2011.

[9] S. R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods," Typeset manuscript, available from http://math.ucsd.edu/~sbuss/ResearchWeb , pp. 1-19, 2009.

[10] S. R. Buss, and J. S. Kim, "Selectively Damped Least Squares for Inverse Kinematics," *Journal of Graphics Tools*, Vol. 10, No. 3, pp. 37-49, 2004.

[11] J. Butzke and M. Likhachev, "Planning for Multi-robot Exploration with Multiple Objective Utility Functions," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, San Francisco, USA, pp. 3254-3259, 2011.

[12] S. L. Cardenas-Maciel, O. Castillo, L. T. Aguilar, “Generation of Walking Periodic Motions for a Biped Robot via Genetic Algorithms,” *Applied Soft Computing*, Vol. 11, Issue 8, pp. 5306–5314, 2011.

[13] R. Chalodhorn, K. F. MacDorman and M. Asada, “Humanoid Robot Motion Recognition and Reproduction,” *Advanced Robotics*, Vol. 23, No. 3, pp. 349-366, 2009.

[14] T. F. Chan, and R. V. Dubey, “A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators,” *IEEE Trans. On Robotics and Automation*, Vol. 11, No. 2, pp. 286-292, 1995.

[15] S. L. Chang, J. J. Lee, and H. C. Yen, “Kinematic and Compliance Analysis for Tendon-Driven Robotic Mechanisms with Flexible Tendons,” *Mechanism and Machine Theory*, Vol. 40-6, pp. 728-739, 2005.

[16] B. K. Cho, J. H. Kim, and J. H. Oh, “Online balance controllers for a hopping and running humanoid robot,” *Advanced Robotics*, Vol. 25, No. 9-10, pp. 1209-1225, 2011.

[17] Y. Choi, D. Kim and B. J. You, “On the Walking Control for Humanoid Robot based on the Kinematic Resolution of CoM Jacobian with Embedded Motion,” *IEEE Int. Conf. on Robotics and Automation*, Orlando, Florida, pp. 2655-2660, 2006.

[18] Y. Choi, B. J. You and S. R. Oh, “On the Stability of Indirect ZMP Controller for Biped Robot Systems,” *IEEE/RSJ Int. Conf. on intelligent Robots and Systems*, Sendai, Japan, pp. 1966-1971, 2004.

[19] S. Y. Chung and H. P. Huang, “Predictive Navigation by Understanding Human Motion Patterns,” *Int. Journal of Advanced Robotic Systems*, Vol.8, No. 1, pp. 52-64, 2011.

[20] S. Y. Chung and H. P. Huang, “SLAMMOT-SP: Simultaneous SLAMMOT and Scene Prediction,” *Advanced Robotics*, Vol. 24, No. 7, pp. 979-1002, 2010.

[21] John, J. Craig, “Manipulator dynamics,” in *Introduction to Robotics Mechanics and Control*, 3rd ed. NJ, USA, Pearson Education, Inc., 2005, ch. 6, pp. 165-192.

[22] C. Connolly, “Motoman Markets Co-operative and Humanoid Industrial Robots,” *Industrial Robot: An Int. Journal*, Vol. 36, Issue 5, pp.417-420, 2009.

[23] S. Coros, P. Beaudoin, and M. van de Panne, “Generalized Biped Walking Control,” *proceeding of ACM SIGGRAPH 2010 conf.*, pp. 1-9, 2010.

[24] A. Cuenca, J. Salt, A. Sala, and R. Piza, “A Delay-Dependent Dual-Rate PID Controller Over an Ethernet Network,” *IEEE Trans. on Industrial Infomatics*, Vol. 7, No. 1, pp. 18-29, 2011.

[25] S. Czarnetzki, S. Kerner, O. Urbann, "Observer-based Dynamic Walking Control for Biped Robots," *Robotics and Autonomous Systems*, Vol. 57, Issue 8, pp. 839-845, 2009.

[26] B. Dariush, M. Gienger, A. Arumbakkam, Y. Zhu, B. Jian, K. Fujimaru, and C. Goerick, "Online Transfer of Human Motion to Humanoids," *Int. Journal of Humanoid Robotics*, Vol. 6, No. 2, pp. 265-289, 2009.

[27] B. Dasgupta and T. S. Mruthyunjaya, "A Newton-Euler Formulation for the Inverse Dynamics of the Stewart Platform Manipulator," *Mechanism and Machine Theory*, Vol. 33, No. 8, pp. 1135-1152, 1998.

[28] T. Doi, R. Hodoshima, S. Hirose, Y. Fukuda, T. Okamoto, and J. Mori, "Development of a Quadruped Walking Robot to Work on Steep Slopes, TITAN XI (Walking Motion with Compensation for Compliance)," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Aichi, Japan, pp. 3413-3418, 2005.

[29] A. G. Erdman, G. N. Sandor, and S. Kota, "Displacement and Velocity Analysis," in *Mechanism Design Analysis and Synthesis Volumn I*, 4th ed., NJ, USA, Pearson Education, Inc., 2001, ch.3, pp. 119-187.

[30] R. Featherstone and D. Orin, "Robot Dynamics: Equations and Algorithms," *IEEE Int. Conf. Robotics Automation*, CA, USA, pp. 826-834, 2000.

[31] J. P. Ferreira, M. Crisostomo, A. P. Coimbra, "ZMP Trajectory Reference for the Sagittal Plane Control of a Biped Robot Based on a Human CoP and Gait," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, St. Louis, USA, pp. 1588-1593, 2009.

[32] E. Furutani, T. Hagiwara and M. Araki, "Two-degree-of-freedom Design Method of State-predictive LQI Servo Systems," *IEE Proc. Control Theory & Application*, Vol. 149, No. 5, pp.365-378, 2002.

[33] S. A. Gard, S. C. Miff, and A. D. Kuo, "Comparison of Kinematic and Kinetic Methods for Computing the Vertical Motion of the Body Center of Mass During Walking," *Human Movement Science*, Vol. 22, No. 6, pp. 597-610, 2004.

[34] M. Griffis and J. Duffy, "A Forward Displacement Analysis of a Class of Stewart Platforms," *Journal of Robotic Systems*, Vol. 6, Issue 6, pp. 703-720, 1989.

[35] J. S. Gutmann, E. Eade, P. Fong, and M. E. Munic, "Vector Field SLAM---Localization by Learning the Spatial Variation of Continuous Signals," *IEEE Trans. on Robotics*, Vol. PP, Issue 99, pp. 1-18, 2012.

[36] K. Harada, H. Hirukawa, F. Kanehiro,K. Fujiwara, K. Kaneko, S. Kajita, and M. Nakamura, "Dynamical Balance of a Humanoid Robot Grasping an Environment," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, Japan, pp. 1167-1173, 2004.

[37] K. Harada, S. Kajita, F. Kanehiro, K. Fujiwara, K. Kaneko, K. Yokoi and H. Hirukawa, "Real-time Planning of Humanoid Robot's Gait for Force Controlled Manipulation," *IEEE Int. Conf. on Robotics & Automation New Orleans*, USA, pp. 616-622, 2004.

[38] K. Harada, S. Kajita, K. Kaneko and H. Hirukawa, "An Analytical Method on Real-time Gait Planning for a Humanoid Robot," *IEEE-RAS/RSJ Int. Conf. on Humanoid Robots (Humanoids 2004)*, Los Angeles, USA, pp. 640-655, 2004.

[39] K. Harada, M. Morisawa, K. Miura, S. Nakaoka, K. Fujiwara, K. Kaneko, and S. Kajita, "Kinodynamic Gait Planning for Full-Body Humanoid Robots," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, pp. 1544-1550, 2008.

[40] K. Hashimoto, Y. Takezaki, K. Hattori, H. Kondo, T. Takashima, H. O. Lim, and A. Takanish, "A Study of Function of Foot's Medial Longitudinal Arch Using Biped Humanoid Robot," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan, pp. 2206-2211, 2010.

[41] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The Development of Honda Humanoid Robot," *IEEE Trans. on Robotics and Automation*, Leuven, Belgium, pp. 1321-1326, 1998.

[42] R. Hodoshima, Y. Fukumura, H. Amano and S. Hirose, "Development of Track-changeable Quadruped Walking Robot TITAN X: Design of Leg Driving Mechanism and Basic Experiment," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan, pp. 3340-3345, 2010.

[43] Y. F. Huang and K. Gupta, "RRT-SLAM for Motion Planning with Motion and Map Uncertainty for Robot Exploration," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, pp. 1077-1082, 2008

[44] K. H. Hyun, E. H. Kim and Y. K. Kwak, "Emotional Feature Extraction Method Based on the Concentration of Phoneme Influence for Human–Robot Interaction," *Advanced Robotics*, Vol. 24, No. 1-2, pp. 47-67, 2010.

[45] S. Kagami, T. Kitagawa, K. Nishiwaki, T. Sugihara, M. Inaba and H. Inoue, "A Fast Dynamically Equilibrated Walking Trajectory Generation Method of Humanoid Robot," *Autonomous Robots*, Vol. 12, Issue 1, 71–82, 2002.

[46] S. Kagami, J. Kuffner, K. Nishiwaki, K. Okada, M. Inaba, and H. Inoue, "Humanoid Arm Motion Planning using Stereo Vision and RRT Search," *IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 2167-2172, 2003.

[47] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi and H. Hirukawa, "Biped Walking Pattern Generation by using Preview Control of

Zero-Moment Point," *Proc. of the 2003 IEEE Int. Conf. on Robotics & Automation*, Taipei, Taiwan, pp. 1620-1626, 2003.

[48] S. Kajita, M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara and H. Hirukawa, "Biped Walking Pattern Generator Allowing Auxiliary ZMP Control," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, pp. 2993-2999, 2006.

[49] T. Katayama, T. Ohki, T. Inoue and T. Kato, "Design of an Optimal Controller for a Discrete-time System Subject to Previewable Demand," *Int. Journal of Control*, Vol. 41, No. 3, pp. 677-699, 1985.

[50] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi and H. Hirukawa, "Resolved Momentum Control: Humanoid Motion Planning based on the Linear and Angular Momentum," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Las Vegas, Nevada, pp. 1644-1650, 2003.

[51] D. W. Kim, N. H. Kim, and G. T. Park, "ZMP Based Neural Network Inspired Humanoid Robot Control," *Nonlinear Dynmics*, Vol. 67, No. 1, pp. 793–806, 2012.

[52] J. Y. Kim and Y. S. Kim, "Walking Pattern Mapping from Imperfect Motion Capture Data onto Biped Humanoid Robots," *Int. Journal of Humanoid Robotics*, Vol. 7, Issue 1, pp. 127–156, 2010.

[53] M. S. Kim and J. H. Oh, "Posture Control of a Humanoid Robot with a Compliant Ankle Joint," *Int. Journal of Humanoid Robotics*, Vol. 7, No. 1, pp. 5-29, 2010.

[54] K. Matsumoto and A. Kawamura, "The Direction Control of a Biped Robot Using Gyro Sensor Feedback," *IEEE Int. Workshop on Advanced Motion Control*, Nagaoka, Japan, pp. 137-142, 2010.

[55] S. Kajita, M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara and H. Hirukawa, "Biped Walking Pattern Generator Allowing Auxiliary ZMP Control," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, pp. 2993-2999, 2006.

[56] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori and K. Akachi, "Humanoid Robot HRP-3," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Acropolis Convention Center, Nice, France, pp. 2471-2478, 2008.

[57] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi and T. Isozumi, "Humanoid Robot HRP-2," *IEEE Int. Conf. on Robotics & Automation*, New Orieans, LA, pp. 1083-1090, 2004.

[58] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokohama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi,"Design of prototype humanoid robotics platform for HRP," *IEEE/RSJ Int. Conf. on Intelligent Robots and System*, 2002, pp. 2431-2436.

[59] K. Kaneko, K. Miura, F. Kanehiro, M. Morisawa, S. Nakaoka, and S. Kajita, "Cybernetic Human HRP-4C," *IEEE/RAS Int. Conf. on Humanoid Robots*, Paris, France, pp. 7-14, 2009.

[60] D. W. Kim, N. H. Kim and G. T. Park, "ZMP Based Neural Network Inspired Humanoid Robot Control," *Nonlinear Dynamics*, Vol. 67, Issue 1, pp. 1-14, 2011.

[61] D. Kim, S. J. Seo and G. T. Park, "Zero-moment Point Trajectory Modeling of a Biped Walking Robot using an Adaptive Neuro-fuzzy System," *IEE Proc. Control Theory Appl.*, Vol. 152, No. 4, pp. 441-426, 2005.

[62] H. Kimura, Y. Fukuoka, Y. Hada and K. Takase, "Three-dimensional Adaptive Dynamic Walking of a Quadruped-rolling Motion Feedback to CPGs Controlling Pitching Motion -," *IEEE Int. Conf. on Robotics Automation*, Washington, DC, pp. 2228-2233, 2002.

[63] C. A. Klein, C. J. Caroline, and S. Ahmed, "A New Formulation of the Extended Jacobian Method and its Use in Mapping Algorithmic Singularities for Kinematically Redundant Manipulators," *IEEE Trans. on Robotics and Automation*, Vol. 11, No. 1, pp. 50-55, 1995.

[64] L. Krammer, W. Granzer, W. Kastner, "A New Approach for Robot Motion Planning using Rapidly-exploring Randomized Trees," *IEEE Int. Conf. on Industrial Informatics*, Caparica, Lisbon, pp. 263-268, 2011.

[65] P. Kulkarni, D. Goswami, P. Guha, and A. Dutta, "Path Planning for a Statically Stable Biped Robot Using PRM and Reinforcement Learning," *Journal of Intelligent & Robotic Systems*, Vol. 47, Issue 3, pp. 197-214, 2006.

[66] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, H. Inoue, "Online Footstep Planning for Humanoid Robots," *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, pp. 932-937, 2003.

[67] A. D. Kuo, "The Six Determinants of Gait and the Inverted Pendulum Analogy: A Dynamic Walking Perspective," *Human Movement Science*, Vol. 26, Issue 4, pp. 617-656, 2007.

[68] Y. Kuroki, M. Fujita, T. Ishida, K. Nagasaka, and J. Yamaguchi, "A Small Biped Entertainment Robot Exploring Attractive Applications," *IEEE Int. Conf. on Robotics & Automation*, pp. 471-476, 2003.

[69] O. Kurt and K. Erbatur, "Biped Robot Reference Generation with Natural ZMP Trajectories," *Int. Workshop on Advanced Motion Control*, pp. 403-410, 2006.

[70] V. Lebastard, Y. Aoustin, F. Plestan, and L. Fridman, "An Alternative to the Measurement of Five-links Biped Robot Absolute Orientation: Estimation Based on High Order Sliding Mode," *Modern Sliding Mode Control Theory: New Perspectives and Applications*, Vol. 375, pp. 363-380, 2008.

[71] B. J. Lee, D. Stonier, Y. D. Kim, J. K. Yoo, and J. H. Kim, "Modifiable Walking Pattern of a Humanoid Robot by Using Allowable ZMP Variation," *IEEE Trans. on Robotics*, Vol. 24, No. 4, pp. 917-925, 2008.

[72] J. H. Lee, K. Abe, T. Tsubouchi, R. Ichinose, Y. Hosoda, and K. Ohba, "Collision-Free Navigation Based on People Tracking Algorithm with Biped Walking Model," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, pp. 2983-2989, 2008.

[73] H. Y. Lee and C. G. Liang, "Displacement Analysis of the General Spatial 7-Link 7R Mechanism," *Mechanism and Machine Theory*, Vol. 23, No. 3, pp. 219-226, 1988.

[74] F. L. Lewis and V. L. Syrmos, "Optimal Control of Discrete-time Systems," and "The Tracking Problem and Other LQR Extensions," in *Optimal Control*, 2nd ed. NJ, USA, John Wiley & Sons, Inc., ch.2, pp. 27-55 and 229-235, 1995.

[75] Y. Li, C. Li, P. Chen, "Research and Design of Control System for a Tracked SAR Robot Under Coal Mine," *IEEE Int. Conf. on Automation and Logistics*, pp. 1957-1961, 2009.

[76] A. Liegeois, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Trans. systems, Man, and Cybernetics,* Vol. 7, No. 12, 1977.

[77] C. L. Lai, P.L. Hsu, and B. C. Wang, "Design of the Adaptive Smith Predictor for the Time-varying Network Control System," *SICE Annual Conf.* pp. 2933-2938, 2008.

[78] X. Liu, X. Gong, and Y. Liu, "Research on Salinity Detecting Based on Embedded CAN-Ethernet Gateway," *Int. Conf. on Measuring Technology and Mechatronics Automation*, pp. 257-260, 2009.

[79] Q. Liu, X. Tang, and J. Zhou, "Delay and Stability Analysis of Networked Robot System," *IEEE Int. Conf. on Control and Automation*, pp. 2903-2906, 2007.

[80] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. F. Seara, W. Gerth, and G. Schmidt, "Experiments in Vision-guided Biped Walking," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, pp. 2484-2490, 2002.

[81] R. Marin, G. Leon, R. Wirz, J. Sales, J.M. Claver, P.J. Sanz, and J. Fernandez, "Remote Programming of Network Robots Within the UJI Industrial Robotics

Telelaboratory: FPGA Vision and SNRP Network Protocol," *IEEE Trans. on Industrial Electronics*, Vol. 56, No. 12, pp. 4806-4816, 2009.

[82] P. Marti, A. Camacho, M. Velasco, and M. El Mongi Ben Gaid, "Runtime Allocation of Optional Control Jobs to a Set of CAN-based Networked Control Systems," *IEEE Trans. on Industrial Infomatics*, Vol. 6, No. 4, pp. 503-520, 2010.

[83] T. Matsubara, J. Morimoto, J. Nakanishi, S. H. Hyon, J. G. Hale, and G. Cheng, "Learning to Acquire Whole-body Humanoid CoM Movements to Achieve Dynamic Tasks," *IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, pp. 2688-2693, 2007.

[84] J. L. Meriam and L. G. Kraige, "Introduction to Three-dimensional Dynamics of rigid Bodies," in *Endineering Mechanics, Dynamics*, 5th ed. NJ, USA, John Wiley & Sons, Inc., ch.7, pp. 515-590.

[85] A. Mifdaoui, F. Frances, and C. Fraboul, "Performance Analysis of a Master/Slave Switched Ethernet for Military Embedded Applications," *IEEE Trans. on Industrial Infomatics*, Vol. 6, No. 4, pp. 534-547, 2010.

[86] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in Industrial Control Applications," *IEEE Trans. on Industrial Infomatics*, Vol. 7, No. 2, pp. 224-243, 2011.

[87] M. Morisawa, S. Kajita, K. Kaneko, K. Harada, F. Kanehiro, K. Fujiwara, H. Hirukawa, "Pattern Generation of Biped Walking Constrained on Parametric Surface," *IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, pp. 2405-2410, 2005.

[88] M. Morisawa, K. Kaneko, F. Kanehiro, S. Kajita, K. Fujiwara, K. Harada, and H. Hirukawa, "Motion Planning of Emergency Stop for Humanoid Robot by State Space Approach," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, pp. 2986-2992, 2006.

[89] Y. Nakamura and H. Hanafusa, "Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control," *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 108, pp. 163-171, 1986.

[90] Y. Nishida, T. Sonoda, and K. Ishii, "Jacobian Matrix Derived from Cross Product and its Application into High Power Joint Mechanism Analysis," *Journal of Bionic Engineering*, Vol. 7, pp. S218–S223, 2010.

[91] K. Nishiwaki and S. Kagami, "Online Walking Control System for Humanoids with Short Cycle Pattern Generation," *The Int. Journal of Robotics Research*, pp. 729-742, 2009.

[92] K. Nishiwaki and S. Kagami, “Simultaneous Planning of CoM and ZMP based on the Preview Control Method for Online Walking Control,” *IEEE/RAS Int. Conf. on Humanoid Robots*, Bled, Slovenia, pp. 745-751, 2011.

[93] K. Nishiwaki, T. Sugihara, S, Kagami, F. Kanehiro, M. Inaba, and H. Inoue, “Design and Development of Research Platform for Perception-action Integration in Humanoid Robot: H6,” *IEEE/RSJ Int. Conf. on Intelligent Robots and System*, pp.1559-1564, 2000.

[94] K. D. Nguyen, I. M. Chen, Z. Luo, S. H. Yeo and H.B.L. Duh , “A Body Sensor Network for Tracking and Monitoring of Functional Arm Motion,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,* pp. 3862-3867, 2009.

[95] M. Nohmi, D. N. Nenchev and M. Uchiyama, “Momentum Control of a Tethered Space Robot Through Tether Tension Control,” *IEEE Int. Conf. on Robotics & Automation*, Leuven, Belgium, pp. 920-925, 1998

[96] N. Oda, J. Yoneda and T. Abe, “Visual Feedback Control of ZMP for Biped Walking Robot,” *IEEE Int. Conf. on Industrial Electronics Society*, Melbourne, Australia, pp. 4543-4548, 2011.

[97] K. Ogata, K. Terada, and Y. Kuniyoshi, “Falling Motion Control for Humanoid Robots while Walking,” *IEEE/RAS Int. Conf. on Humanoid Robots*, PA, USA, pp. 306-311, 2007.

[98] Y. Ogura, H. Aikawa, K. Shimomura, H. Kondo, A. Morishima, H. O. Lim and A. Takanishi, “Development of a New Humanoid Robot WABIAN-2,” *IEEE Int. Conf. on Robotics and Automation*, Florida, pp. 77-81, 2006.

[99] Y. Ogura, T. Kataoka, H. Aikawa, K. Shimomura, H.O. Lim and A. Takanishi, “Evaluation of Various Walking Patterns of Biped Humanoid Robot,” *IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, pp. 603-608, 2005.

[100] Y. Ogura, K. Shimomura, H. Kondo, A. Morishima, T. Okubo, and S. Momoki, H.O. Lim and A. Takanishi, “Human-like Walking with Knee Stretched, Heel-contact and Toe-off Motion by a Humanoid Robot,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, pp. 3976-3981, 2006.

[101] J. H. Oh, D. Hanson, W. S. Kim, I. Y. Han, J. Y. Kim and I. W. Park, “Design of Android type Humanoid Robot Albert HUBO,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, pp. 1428-1433, 2006.

[102] E. J. Ong and A. Hilton, “Learnt Inverse Kinematics for Animation Synthesis,” *Graphical Models*, Vol. 68, Issue 5-6, pp. 472-483, 2006.

[103] S. Parasuraman and L. S. Pe, “Bio-mechanical Analysis of Human Joints and Extension of the Study to Robot,” *World Academy of Science, Engineering and Technology*, Issue 39, pp. 1-6, 2008.

[104] I. W. Park and J. Y. Kim, "Fourier Series-Based Walking Pattern Generation for a Biped Humanoid Robot," *IEEE/RAS Int. Conf. on Humanoid Robots*, TN, USA, pp. 461-467, 2010.

[105] I. W. Park, J. Y. Kim and J. H. Oh, "Online Biped Walking Pattern Generation for Humanoid Robot KHR-3(KAIST Humanoid Robot - 3: HUBO)," *IEEE/RAS Int. Conf. on Humanoid Robots*, Italy, pp. 398-403, 2006.

[106] I. W. Park, J. Y. Kim and J. H. Oh, "Online Walking Pattern Generation and Its Application to a Biped Humanoid Robot —KHR-3 (HUBO)," *Advanced Robotics*, Vol. 22, No. 2-3, pp. 159-190, 2008.

[107] F. Plestan, J. W. Grizzle, E. R. Westervelt, and G. Abba, "Stable Walking of a 7-DOF Biped Robot," *IEEE Trans. on Robotics and Automation*, Vol. 19, No. 4, pp. 653-668, 2003.

[108] N. Shafii, M. H. S. Javadi, B. Kimiaghalam, "A Truncated Fourier Series with Genetic Algorithm for the control of Biped Locomotion," *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Singapore, pp. 1781-1785, 2009.

[109] S. K. Saha, "A Unified Approach to Space Robot Kinematics," *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 3, pp. 401-405, 1996.

[110] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki and K. Fujimura, "The Intelligent ASIMO: System Overview and Integration," *IEEE/RAS Int. Conf. on Intelligent Robots and Systems*, Switzerland, pp. 2478-2483, 2002.

[111] U. Saranl, A. Avc, and M. C. Ozturk, "A Modular Real-time Fieldbus Architecture for Mobile Robotic Platforms," *IEEE Trans. on Instrumentation and Measurement*, Vol. 60, No. 3, pp. 916-927, 2011.

[112] M.O.F. Sarker, C. H. Kim, S. Back, and B. J. You, "An IEEE-1394 Based Real-time Robot Control System for Efficient Controlling of Humanoids," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1416-1421, 2006.

[113] T. Sato, S. Sakaino, and K. Ohnishi, "Trajectory Planning and Control for Biped Robot with Toe and Heel Joints," *IEEE Int. Workshop on Advanced Motion Control,* Nagaoka, Japan, pp. 129-136, 2010.

[114] T. Sato, S. Sakaino, and K. Ohnishi, "Walking Stabilization Control using Virtual Plane Method for Biped Robots," *IEEJ Trans. on Industry Applications*, Vol. 130, Issue 5, pp. 685-691, 2010.

[115] J. J. Scarlett, and R. W. Brennan, "Evaluating a New Communication Protocol for Real-time Distributed Control," *Robotics and Computer-Integrated Manufacturing*, Vol. 27, Issue 3, pp. 627–635, 2011.

[116] J. Schmitt, H. Haupt, M. Kurrat, A. Raatz, "Disassembly Automation for Lithium-ion Battery Systems using a Flexible Gripper," *IEEE Int. Conf. on Advanced Robotics*, Tallinn, Estonia, pp. 291-297, 2011.

[117] K. Seto, D. Fuji, H. Hiramathu and Tonu Watanabe, "Motion and Vibration Control of Three Dimensional Flexible Shaking Table using LQI Control Approach," *Proc. of the American Control Conf.*, Anchorage, pp. 3040-3045, 2002.

[118] M. Y. Shieh, K. H. Chang, C. Y. Chuang and Y. S. Lia, "Development and Implementation of an Artificial Neural Network based Controller for Gait Balance of a Biped Robot," *IEEE Industrial Electronics Society*, Taipei, Taiwan, pp. 2778-2782, 2007.

[119] T. Sonoda, K. Ishii, and D. Isobe, "Dynamics Computation of Link Mechanisms Employing COG Jacobian," *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Xian, China, pp. 482-487, 2008.

[120] S. Staicu, "Recursive modelling in dynamics of Delta parallel robot," *Robotica*, Vol. 27, Issue 2, pp. 199-207, 2009.

[121] B. J. Stephens and C. G. Atkeson, "Push Recovery by Stepping for Humanoid Robots with Force Controlled Joints," *IEEE/RAS Int. Conf. on Humanoid Robots Nashville*, TN, USA, pp. 52-59, 2010.

[122] T. Sugihara and Y. Nakamura, "Boundary Condition Relaxation Method for Stepwise Pedipulation Planning of Biped Robots," *IEEE Trans. on robotics*, Vol. 25, No. 3, pp. 658-669, 2009.

[123] T. Sugihara and Y. Nakamura, "Enhancement of Boundary Condition Relaxation Method for 3D Hopping Motion Planning of Biped Robots," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 444-449, 2007.

[124] T. Sugihara and Y. Nakamura, "Whole-body cooperative balancing of humanoid robot using COG Jacobian," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2575-2580, 2002.

[125] W. Suleiman, F. Kanehiro, K. Miura and E. Yoshida, "Enhancing Zero Moment Point-Based Control Model: System Identification Approach," *Advanced Robotics*, Vol. 25, No 3, pp. 427-446, 2011.

[126] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real Time Motion Generation and Control for Biped Robot -1st Report: Walking Gait Pattern Generation-," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1084-1091, St. Louis, USA, 2009.

[127] T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura, "Real Time Motion Generation and Control for Biped Robot -2nd Report: Running Gait Pattern

Generation-,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, St. Louis, USA, pp. 1594-1600, 2009.

[128] T. Takenaka, T. Matsumoto, and T. Yoshiike, “Real Time Motion Generation and Control for Biped Robot -3rd Report: Dynamics Error Compensation-, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, St. Louis, USA, pp. 1092-1099, 2009.

[129] T. Takubo, K. Inoue, and T. Arai, “Pushing an Object Considering the Hand Reflect Forces by Humanoid Robot in Dynamic Walking,” *IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, pp. 1706-1711, 2005.

[130] K. Tchon, “Optimal Extended Jacobian Inverse Kinematics Algorithms for Robotic Manipulators,” *IEEE Trans. on Robotics*, Vol. 24, No. 6, pp. 1440-1445, 2008.

[131] G. Tevatia, and S. Schaal, “Inverse Kinematics for Humanoid Robots,” *IEEE Int. Conf. on Robotics and Automation*, pp. 294-299, 2000.

[132] S. Thomas, “Dynamics of Spacecraft and Manipulators,” *Simulation*, ISSN 0037-5497, July, 1991.

[133] L. W. Tsai, “Position Analysis of Serial Manipulators,” in *Robot Analysis*: *the mechanics of serial and parallel manipulators*, 1st ed. NJ, USA, John Wiley & Sons, Inc., 1999, ch. 2, pp. 55-85.

[134] F. Tungadi and L. Kleeman, “Autonomous Loop Exploration and SLAM with Fusion of Advanced Sonar and Laser Polar Scan Matching,” *Robotica*, Vol. 30, Issue 1, pp. 91-105, 2012.

[135] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, “Humanoid Motion Planning for Dual-Arm Manipulation and Re-Grasping Tasks,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, St. Louis, USA, pp. 2464-2470, 2009.

[136] M. Vukobratovic and B. Borovac, “Zero-Moment Point - Thirty Five Years of Its Life,” *Int. Journal of Humanoid Robotics*, Vol. 1, Issue 1, pp. 157-173, 2004.

[137] C. W. Wampler, “Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least Squares Methods,” *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 1, pp. 93-101, 1986.

[138] X. Wang and J. K. Mills, “Dynamic Modeling of a Flexible-link Planar Parallel Platform using a Substructuring Approach” *Mechanism and Machine Theory*, Vol. 41, Issue 6, pp. 671-687, 2006.

[139] M. Xu, and W. Zhu, “A Research and Design of Ethernet Real-Time Application Bus Based on FPGA.” *Int. Conf. on Embedded Computing Scalable Computing and Communications*, pp. 42-46, 2009.

[140] K. Yamamoto and Y. Nakamura, “Switching Control and Quick Stepping Motion Generation Based on the Maximal CPI Sets for Falling Avoidance of Humanoid Robots,” *IEEE Int. Conf. on Robotics and Automation*, Anchorage, Alaska, USA, pp. 3292-3297, 2010.

[141] L. Yang, C. M. Chew, A. N. Poo, and T. Zielinska, “Adjustable Bipedal Gait Generation using Genetic Algorithm Optimized Fourier Series Formulation,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing , China, pp. 4435-4440, 2006.

[142] T. Yoshikawa, “Manipulability of Robotic Mechanisms,” *The Int. Journal of Robotics Research*, Vol. 4, No. 2, pp. 3-9, 1985.

[143] T. Yoshikawa and O. Khati, “Compliant Motion Control for a Humanoid Robot in Contact with the Environment and Human,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, pp. 1544-1550, 2008.

[144] B. J. You, M. Hwangbo, S. O. Lee, S. R. Oh, Y. D. Kwon, and S. Lim, “Development of a Home Service Robot 'ISSAC',” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2630-2635, 2003.

[145] W. You, M. X. Kong, L. N. Sun, and C.C. Guo, “Control System Design for High Payload Industrial Robot via High Speed Communication Bus and Real-time System,” *Key Engineering Materials*, Vol. 464, 2011 pp. 272-278, 2011.

[146] S. W. Yu, “Walking Pattern Analysis and Control of a Humanoid Robot,” *Master Thesis*, Department of Mechanical Engineering, National Taiwan University, pp. 28-30, 2006.

[147] Z. Yu, Q. Huang, J. Li, X. Chen, and K. Li, “Computer Control System and Walking Pattern Control for a Humanoid Robot,” *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, pp. 1018-1023, 2008.

[148] R. Zaier and S. Kanda, “Adaptive Locomotion Controller and Reflex System for Humanoid Robots,” *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Acropolis Convention Center, Nice, France, pp. 2492-2497, 2008.

[149] W. Zijlstra and A.L. Hof, “Displacement of the Pelvis During Human Walking: Experimental Data and Model Predictions,” *Gait & Posture*, Vol. 6, Issue 3, pp. 249-267, 1997.

[150] W. Zhang, W. Ren, Y. Qin, S. Chen, and B. Yang, “The Implementation of CAN—Ethernet Communication System on the Missile Simulation and Detection Platform,” *Int. Conf. on Computer-Aided Industrial Design and Conceptual Design*, pp. 642-645.

[151] W. Zhang, Y. Yang, and J. Wang, “A dsPIC-based Excitation Control System for Synchronous Generator,” *Int. Conf. on Mechatronics and Automation*, pp. 3844-3848, 2007.

[152] V. Zordan, “Angular Momentum Control in Coordinated Behaviors,” *MIG'10 Proceedings of the Third int. conf. on Motion in games*, Springer-Verlag Berlin, Heidelberg, pp. 1-12, 2010.

[153] http://world.honda.com/ASIMO/

[154] http://www.bostondynamics.com/

[155] http://www.netlib.org/clapack/

APPENDIX A

PROOF AND DERIVATION OF LQSI CONTROLLER

The proof and derivation that the LQSI controller can minimize the performance index is described here. More detailed and basic derivation methods can be referred to [74]. Recall the performance index of LQSI control in Eq. (4-23) and the constraint in Eq. (4-24).

$$J=\frac{1}{2}\sum_{k=i}^{\infty}((x_{k+1}-x_k)^TQ_x(x_{k+1}-x_k) + (C_kx_k-r_k)^TQ(C_kx_k-r_k)+u_k^TRu_k) \tag{4-23}$$

$$f_k=x_{k+1}=A_kx_k+B_ku_k \tag{4-24}$$

Thus the Hamiltonian can be designed as Eq. (A-1).

$$H_k=\frac{1}{2}((x_{k+1}-x_k)^TQ_x(x_{k+1}-x_k) + (C_kx_k-r_k)^TQ(C_kx_k-r_k)+u_k^TRu_k)+\lambda_{k+1}^Tf_k \tag{A-1}$$

where λ_k denotes the Lagrange multiplier. And $x_{k+1}-x_k$ in the equation can be rewritten as Eq. (A-2).

$$x_{k+1}-x_k=A_kx_k-x_k+B_ku_k=A_{I,k}x_k+B_ku_k \tag{A-2}$$

From reference [74], to minimize the performance index, the following Eqns. must be satisfied.

$$\frac{\partial H_k}{\partial u_k}=0 \tag{A-3}$$

$$\frac{\partial H_k}{\partial x_k}=\lambda_k \tag{A-4}$$

With Eq. (A-3), we can find Eq. (A-5).

$$\frac{\partial H_k}{\partial u_k}=0=B_k^TQ_x\left(A_{I,k}x_k+B_ku_k\right)+Ru_k+B_k^T\lambda_{k+1} \tag{A-5}$$

Rearranging the Eq. (A-5), we have

$$\begin{aligned} u_k &= -(R + B_k^T Q_x B_k)^{-1} B_k^T Q_x A_{I,k} x_k - (R + B_k^T Q_x B_k)^{-1} B_k^T \lambda_{k+1} \\ &= -M_k^{-1} B_k^T Q_x A_{I,k} x_k - M_k^{-1} B_k^T \lambda_{k+1} \end{aligned} \tag{A-6}$$

With thisu_k, the partial derivative of u_k in the direction x_k can be found as Eq. (A-7).

$$\frac{du_k}{dx_k} = -M_k^{-1} B_k^T Q_x A_{I,k} \tag{A-7}$$

From Eqns. (A-4) and (A-7), Eq. (A-8) can be obtained as follows.

$$\begin{aligned} \frac{dH_k}{dx_k} = \lambda_k = \left(A_{I,k} - B_k M_k^{-1} B_k^T Q_x A_{I,k}\right)^T Q_x \left(A_{I,k} x_k + B_k u_k\right) \\ - \left(M_k^{-1} B_k^T Q_x A_{I,k}\right)^T R u_k + C_k^T Q (C_k x_k - r_k) + A_k^T \lambda_{k+1} \end{aligned} \tag{A-8}$$

Define a variable N_k to simplify the Eq. (A-8) as Eq. (A-9).

$$N_k = A_{I,k} - B_k M_k^{-1} B_k^T Q_x A_{I,k} \tag{A-9}$$

Replacing u_k in Eq. (A-8) with Eq. (A-6) and after lines of work, we can find Eq. (A-10).

$$\begin{aligned} \lambda_k = N_k^T Q_x N_k x_k + A_{I,k}^T Q_x B_k M_k^{-1} R M_k^{-1} B_k^T Q_x A_{I,k} x_k \\ + C_k^T Q (C_k x_k - r_k) + A_k^T \lambda_{k+1} \end{aligned} \tag{A-10}$$

The Lagrange multiplier is assumed to be decoupled as follows.

$$\lambda_k = S_k x_k - v_k \tag{A-11}$$

The u_k in Eq. (4-24) is also replaced with Eq. (A-6), thus Eq. (A-12) can be found.

$$x_{k+1} = A_k x_k - B_k M_k^{-1} B_k^T Q_x A_{I,k} x_k - B_k M_k^{-1} B_k^T \lambda_{k+1} \tag{A-12}$$

Replacing the Lagrange multiplier with Eq. (A-11), we can solve the x_{k+1} as Eq. (A-13).

$$x_{k+1} = D_k^{-1} W_k x_k + D_k^{-1} B_k M_k^{-1} B_k^T v_{k+1} \tag{A-13}$$

where D_k and W_k are shown in Eqns. (4-30) and (4-31).

Using Eqns. (4-34) and (A-13), Eq. (A-10) can be rewritten as Eq. (A-14).

$$\begin{aligned}\lambda_k = {} & P_k^T P_k x_k + C_k^T Q (C_k x_k - r_k) + A_k^T S_{k+1} D_k^{-1} W_k x_k \\ & + A_k^T S_{k+1} D_k^{-1} B_k M_k^{-1} B_k^T v_{k+1} - A_k^T v_{k+1}\end{aligned} \tag{A-14}$$

Thus, we can find the Eqns. (4-32) and (4-33) by separating the parts with and without x_k.

APPENDIX B

THE FINITE TIME CASE OF LQSI CONTROLLER

The $\phi(x_N, u_N)$ in the performance index in Eq. (4-36) is defined as Eq. (B-1).

$$\begin{aligned}\phi &= \frac{1}{2}(C_N x_N - r_N)^T Q_N (C_N x_N - r_N)\\ &\quad + \frac{1}{2}(x_{N+1} - x_N)^T Q_{xN} (x_{N+1} - x_N)\end{aligned} \tag{B-1}$$

Using the boundary condition in Eqns. (4-37), (A-6), and (A-7), we can find Eq. (B-2).

$$\begin{aligned}\frac{\partial \phi}{\partial x_N} = \lambda_N = C_N^T Q_N (C_N x_N - r_N) + \left(A_{I,N} - B_N M_N^{-1} B_N^T A_{I,N}\right)^T Q_{xN}\\ \left(\left[A_{I,N} - B_N M_N^{-1} B_N^T A_{I,N}\right] x_N - B_N M_N^{-1} B_N^T \lambda_{N+1}\right)\end{aligned} \tag{B-2}$$

In Eq. (B-2), λ_{N+1} is indeterminable, so we must choose Q_{xN} as zero. Thus, ϕ is defined again as follows.

$$\phi = \frac{1}{2}(C_N x_N - r_N)^T Q_N (C_N x_N - r_N) \tag{B-3}$$

$$\frac{\partial \phi}{\partial x_N} = \lambda_N = S_N x_N - v_N = C_N^T Q_N (C_N x_N - r_N) \tag{B-4}$$

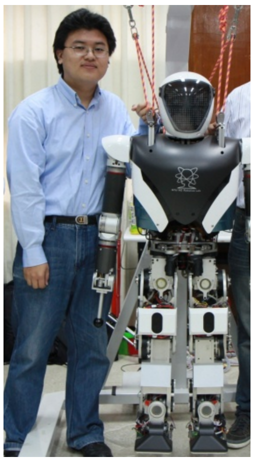
$$S_N = C_N^T Q_N C_N \tag{B-5}$$

$$v_N = C_N^T Q_N r_N \tag{B-6}$$

Eqns. (B-3)-(B-6) are just the same as the boundary condition of the classical final-state-free tracking control problems. It means only the tracking error affects the value of the cost function at the final step. Thus, the initial values, S_N and v_N, for backward recursion can be found. We can choose a large Q_N for smaller tracking error at the final step.

BIOGRAPHY

Jiu-Lou Yan



Robotics Laboratory,
Department of Mechanical Engineering,
National Taiwan University.

Education

2007~2012

Ph.D. Student in Department of Mechanical Engineering, National Taiwan University

2006~2007

M.S. course in Department of Mechanical Engineering, National Taiwan University

2002~2006

B.S. course in Department of Mechanical Engineering, National Taiwan University

Publications

Journal paper

[1] Han-Pang Huang, Jiu-Lou Yan, and Teng-Hu Cheng, "Development and Fuzzy Control of a Pipe Inspection Robot," *IEEE Trans. on Industrial Electronics*, Vol. 57, No. 3, March, 2010.

[2] Han-Pang Huang, Jiu-Lou Yan, and Teng-Hu Cheng,, "State-Incremental Optimal Control of 3D COG Pattern Generation for Humanoid Robots", *Advanced Robotics*, accepted.

[3] Han-Pang Huang, Jiu-Lou Yan, and, Kenneth Yi-Wen Chao, "A Whole Body Inverse Kinematics Solver for Humanoid Robots," *Journal of the Chinese Society of Mechanical Engineers*, accepted.

[4] Han-Pang Huang, Jiu-Lou Yan, Tz-How Huang, "Priority Based Networking for Humanoid Robots," *IEEE Trans. on Industrial Informatics*, submitted.

[5] Han-Pang Huang, Jiu-Lou Yan, and Shin-Wei Lin, "Cerebellar Model Based Controller with Adaptive Learning Rate for Robotic Manipulators," *IEEE Journal*, in preparation.

[6] Han-Pang Huang, Jiu-Lou Yan, Yao-Joe Yang, Wen-Pin Shih, Chia-Chen Lin and Gun-Hwa Liu, Design of a Face Robot with Facial Expressions, *IEEE Journal*, in preparation.

Conference paper

[1] Jiu-Lou Yan and Han-Pang Huang, "A fast and smooth walking pattern generator of biped robot using Jacobian inverse kinematics," *IEEE Workshop on Advanced Robotics and Its Social Impacts*, pp. 25-30, 2007.

[2] Han-Pang Huang, Shu-Wen Yu and Jiu-Lou Yan, "Walking Pattern Analysis and Control of a Humanoid Robot," *The 13th Int. Conf. on Advanced Robotics, Jeju*, Korea, pp. 862-867, 2007.

[3] Han-Pang Huang, Jia-Yang Wu and Jiu-Lou Yan, "A Stable Walking Biped Robot based on Combined CPG/ZMP," *Int. Forum on Systems and Mechatronics*, Tainan, Taiwan, pp. 305-310, 2007

[4] Chao-Pei Lu, Han-Pang Huang, Jiu-Lou Yan, Ting-Hu Cheng, "Development of a Pipe Inspection Robot," *IEEE Industrial Electronics Society*, Taipei, Taiwan, pp. 626-631, 2007.

[5] S.Y. Liu, T.H. Cheng, J.L. Yan and H.P. Huang, "Development of a Dexterous Cable Driven Spine Mechanism for Humanoid Robots," *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Singapore, July 14-17, pp.1588-1593, 2009.

[6] Han-Pang Huang, Shin-Wei Lin, Jiu-Lou Yan, Tzu-Hsin Kuo and Yang-Lun Liu, "Learning-based Controller for Robotic Manipulators with Grey Learning Rate," *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Budapest, Hungary July 3-7, pp. 701-706, 2011.

[7] Kenneth Yi-Wen Chao, Jiu-Lou Yan, Meng-Ku Chi, and Han-Pang Huang, "Natural Walking Pattern Generation for Humanoid Robots with Toe and Heel Mechanism", *The 43rd Int. Symposium on Robotics*, Taipei, Taiwan, Aug. 29-31, 2012, accepted.

[8] Han-Pang Huang and Jiu-Lou Yan, "An Inverted Pendulum Model Based Optimized Walking Pattern Generator," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, submitted.

Book Chapter

[1] Han-Pang Huang and Jiu-Lou Yan, "A Fast and Smooth Walking Pattern Generator for Biped Robots," *Biped Robots*, INTECH open science, ISBN: 978-953-307-216-6, pp. 283-298, 2011.