Walking Algorithm for Small Humanoid

2002 Master Project

By Patrick Lam

Supervisor: Dr. Jacky Baltes

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Computer Science Center for Image Technology and Robotics

Department of Computer Science

University of Auckland

New Zealand

ABSTRACT

This thesis describes the design and implementation of a Walking Algorithm for the humanoid robot. The humanoid is intended to be an autonomous robot, and has a total of 6 degrees of freedom coming from the hip joint, knee joint and ankle joint for each leg. The aim of this thesis project is to develop a walking pattern to control and walk in a stable manner.

Exploit methods for developing a natural dynamic motion for small humanoid robots. The frontal swing of the gait or bipedal walking algorithms can be modeled by simple inverted pendulum models. With a combination of Bezier Curves, experimental robot had implemented a smooth gait, which looks quite natural.

ACKNOWLEGEMENTS

Firstly, I would like to thank my supervisor, Dr Jacky Baltes, for his time and effort in guiding me throughout the entire course of this thesis project. His invaluable advises and suggestions had been the beacon at times.

I would like to thank Dr. Nadir Ould Kheddal and his group at Temasek Polytechnic, Singapore. For their friendly helps building the mechanical structure, it brings me the success and victories in the Fira2002 and Robocup2002 competitions.

I would like to thank associate professor Thomas Braunl and his group at University of Western Australia for sharing the knowledge of humanoid projects and using the Mobile Robot Lab.

I am grateful to my friends for being a fountain of inspiration all the time.

Lastly and mostly, I would like to thank my parents and my sister Fanny for their unconditional love and care. This thesis is dedicated to them.

CONTENTS

LIST OF FIGURES & TABLES

Figures

Tables

CHAPTER 1

INTRODUCTION

Machine robots perhaps were the first autonomous walking machines in the world. It was the obstacles of their design that led to the first real advances in walking automation – biped robots, or often termed as humanoid robots.

Legs were first seen as the prerequisite for propulsion through the blueprint of the design of the first legged vehicle from the 18th century. This is because two-legged walking is one of the most sophisticated movements in nature, which influenced the development of bipedal walking robots.

This chapter discusses the motivations for studying bipedal walking and the objectives of this project followed by an outline of the thesis.

1.1 Motivation

In recent years, a large proportion of attention has been placed on machine robot development rather than humanoid robot. There are two obvious reasons that lead to this outcome. Firstly, the results from humanoid research provide less benefit than those from flight and wheeled travel research. The second reason is that walking and running systems are extremely complex to control, difficult to build and sometimes involve a lot of capital investment to setup.

Nevertheless the benefits of biped robots cannot be excluded from robot research and development process. With a wider operating region in terms of a variety of terrain types, it drives into the study of humanoids, which aimed to remove the obstacles of machine robots.

In general, wheeled robots require an environment that is tailored to their needs. For example: They must be placed onto a flat surface with no objects in the way in order to perform to their expectations.

The major motivation for this research project is to design a robot that can stimulate the activities of human beings in the area of leg movements. With humanoid robots that have the mobility and ability similar to that of humans, it opens an extensive territory in future technology development that currently requires human to perform the tasks.

1.3 Objectives

Cost was an important design criterion in humanoid development. Previous experience has shown us that the use of commonly available cheap components does not only help to keep the cost of a project down, but it also has led to the development of novel, versatile, and robust approaches to problems in robotics.

The main aim of this project is to develop a walking gait for small humanoid that can walk naturally, and build up library functions as the stepping stone for Intelligent Learning.

1.4 Outline

This section outlines the overall structure of the thesis, and a brief description for each chapter.

Chapter 2 discusses the differences between Powered walk and Passive walk. Define static balance and dynamic balance. Give an evaluation on the latest humanoids.

Chapter 3 presents several simple models for bipedal walking. These model help exploit the behavior of humanoid. Which aids the algorithm development.

Chapter 4 describes the mechanical design, include the microcontroller, sensors, actuators, and the structure. The effect of the hardware is also discuss.

Chapter 5 presents a walking algorithm, which combine Bezier Curve. It explains the systems that had contribution to the algorithm. Review the theory of phases, behaviors and control points.

Chapter 6 evaluates the implemented algorithm in four area, the used of Bezier Curve to produce different walking pattern, the Pattern Generation System, the Vision System and the performance in competitions

Chapter 7 summarizes the outcome of the project and the result that was obtained, and discuss the future works

CHAPTER 2

BACKGROUND

This chapter we review the latest humanoid around the world, from academic research through commercial research supported by two large international companies, SONY and Honda. This is a very important aspect, since not many researches had been done in the field of Robotics and AI for humanoid until recent years. By interpreting the mechanical design for all different humanoid around the world, this section has influenced the thesis, the experimental research and mechanical design of our humanoid.

2.1 Powered walk and Passive walk

The aim of our project is to develop a walking gait for small humanoid. The followings are the area of research people had done around the world for generating a walking gait. There are a number of powered bipedal walking robots that have been built by various groups throughout the world. A handful exploits the inherent robustness of walking by using simple control methods [7]. A few of them benefit from the natural dynamics of the mechanism, though none explicitly exploits the natural dynamics. The robots fall into two broad categories. Many play back pre-recorded joint trajectories. Others use heuristic control approaches. Some of them combine both playback and heuristic control. Passive dynamic walkers are a class of mechanisms, which can walk smoothly and stably down a shallow incline with no sensing, control, or actuation. These passive dynamic walkers exhibit a steady

walking cycle that is smooth and efficient and appears incredibly natural, Mariano, Andy, and Michael had describe some result [17].

2.2 Static Balance and Dynamic Balance

To begin with, there is already number of classifications distinguishing between different types of walking and control schemes in the texts examined. One distinction that can be made is the type of balance that is maintained.

The two main classifications are static and dynamic balance. Static balance means that the center of mass (COM) of the robot is kept within the support area of the feet at all times. W.Thomas Miller [20] gives a definition of walking with dynamic balance.

"When walking with dynamic balance, the projected center of mass is allowed outside of the area inscribed by the feet, and the walker may essentially be falling during some parts of the gait cycle. A foot must be moved so as to catch the biped at the proper instant, breaking the fall and achieving the desired net translational acceleration or deceleration."

2.3 Humanoids

2.3.1 Elvis

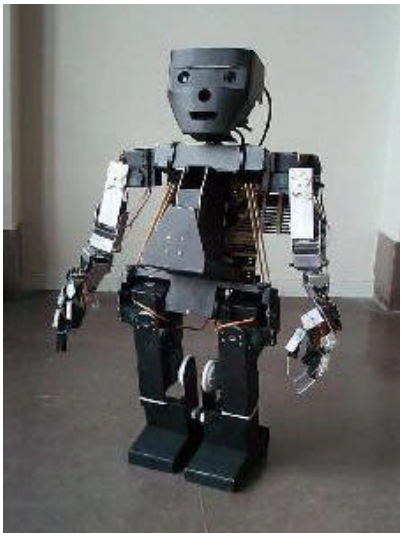

Figure 2.1 – Humanoid: Elvis

Elvis is the oldest of the humanoids. Prototype 1 was assembled in the summer of 1998 and the initial projects included adaptive balance evolution and extensive hearing and vision experiments. Elvis took his first steps in July 1998 and learned to toast in October the same year. In august 1999 Elvis was equipped with a new head, torso and legs. He took his first autonomous steps in April 2000 and participated in Expo2000 in Hannover, Germany, the following summer.

Elvis is a low weight humanoid robot. Developed at Chalmers University of Technology, Department of Physical Resource Theory, Complex Systems Group [5]. He has a 23 degrees of freedom, 6 in each leg, 4 in each arm, one in the torso and 2 in the neck. Three independent sensory systems provide visual information as well as a sense of pressure and acceleration. He is being developed and constructed using easily available and cheap components.

As the main processing unit ‘Elvis’ uses a 3.5” single onboard computer, the operating system and the software are stored on Compact Flash. Distributed hardware constitutes the low-level interface to sensors and servos.

2.3.2 Elvina

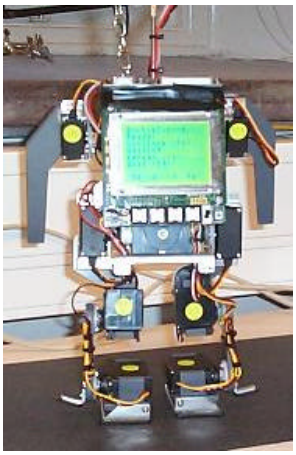

Figure 2.2 – Humanoid: Elvina

Elvina is the second generation developed at Chalmers University of Technology, Department of Physical Resource Theory, Complex Systems Group [5]. It was designed to be a low weight platform, serve as a full-scale prototype. The mechanical design aims to be as less energy consuming as possible and to make it easy to keep the stability while the robot is walking or standing. The ‘Elvina’ robot has three independent sensory systems, vision, pressure sensors and accelerometers.

2.3.3 Priscilla

Figure 2.3 – Humanoid: Priscilla

Priscilla is a different class humanoid, it is modify based on a plastic full-scale replica of a human skeleton reinforced with metal joints and hydraulic pistons. The project was initiated in the summer of 2000.

The Priscilla robot consists of a plastic skeleton with titanium reinforcements and linear electric actuators. The philosophy behind all our robots is that the software architecture should mainly build on evolutionary algorithms and specifically genetic programming.

2.3.4 Murphy

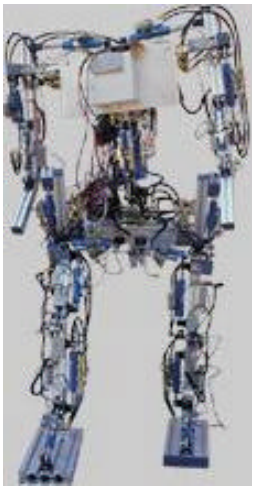

Figure 2.4 – Humanoid: Murphy

The Murphy project at Uppsala University is part of the ongoing RFC-Uppsala effort. The aim of the project for the spring 2002 is to make the humanoid robot Murphy balance, walk and ultimately kick a football past a robot goalkeeper into the goal.

The name of the project is “developing the autonomous humanoid robot Murphy”. The main challenge is building a brain that will control Murphy's movements. This is what eventually will allow him to walk - and even though they tend to use the word brain, it is really more of a central nervous system. They are trying to use some learning algorithm approach - Murphy has too many degrees of freedom to be controlled by a mathematical system (although that might be possible in theory). To allow the brain to learn the right moves for walking without hurting the expensive hardware, They also intend to use a simulator that simulates the movements of Murphy and the world around him.

There are already ways to control his movements using commands directly, but these are just commands. This totally relies on the user. With our improvements, it should be possible to tell Murphy to move to a certain location and he'll walk right there.

The Murphy is a humanoid robot based on an aluminum skeleton. He is powered by hydraulics, which has some advantages and some disadvantages compared to electrical motors or linear actuators. More power and more flexibility are the main advantages. Recently Murphy has installed Moog valves together with high-pressure hydraulic couplings from Cejn, this will allow Murphy to hit and kick with raw muscular power of 600kg. Murphy has 21 degrees of freedom, and a neural net that in turn is being trained on a simulator control.

2.3.5 GuRoo

Figure 2.5 – Humanoid: GuRoo

GuRoo is a humanoid robot platform, developed at the University of Queensland, Department of Information Technology and Electrical Engineering [6]. It stands 1.2m high with limb propotions based heavily on anthropomorphic data. It consists of 23 Degrees of Freedom, 6 in each leg, 3 in each arm and an articulated spine and head. Actuation is achieved by a combination of brushed DC and RC servomotors. A

distributed multi-master system controls the robot along a Controller Area Network. A system of 6 motor controller board all running DSP micro-controllers, provide local control and monitoring of each DoF. A Compaq IPAQ handheld PC provides the gait generation pattern and user interface. Vision processing is provided via a CMOS camera connected to a dedicated Hitachi SH4 board.

SolidEdge, a solid modeling package, was used to design the mechanical structure of the GuRoo. By allocating material properties to each component, it was possible to achieve a mathematical representation of the mass distribution through inertia tensors. These tensors, along with link lengths, motor characteristics and environmental constriants, were modeled in DynaMechs, a high fidelity dynamic simulator. Through this simulator various humanoid gaits and movements can be analysed before operation on the actual robot.

2.3.6 Senchans

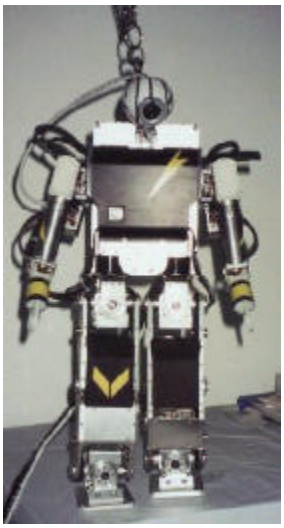

Figure 2.6 – Humanoid: Senchans

Senchans is a project from Osaka University, Department of Adaptive Machine Systems, Graduate School of Engineering. The humanoid robot platform Fujitsu HOAP-1 (Humanoid for Open Architecture Platform) that is 48 cm in height and 6kg in weight with totally 20 Degree of Freedoms, 6 for each leg and 4 for each arm operated by RT-Linux. USB interface is used for communication with a host machine. The basic control is done in terms of position and its cycle time is 1ms. The acceleration and angular velocity sensors, joint encoders and FSR (force sensing register) at foot surface are equipped.

2.3.7 Viki

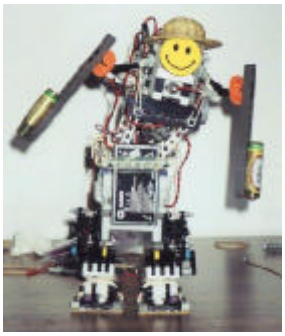

Figure 2.7 – Humanoid: Viki

Viki was developed at University of Southern Denmark. The Maersk Mc-Kinney Moller Institute for Production Technology. The principle behind Viki is minimalistic design, the minimalistic approach to the construction of Viki combines the bottom-up approach with the new AI focus on interaction between physical properties and control (i.e. between body and brain). The design of Viki is to minimal the complexity that is necessary to achieve the mechanical properties and control for specific behaviors.

Viki consists of 5 motors. Two motors are used for leg turning, one motor for hip movement, one motor for body balance, and one for arm swinging. The height of the humanoid is approximately 25 cm. And during the walking gait, all body postural are static balanced.

The control hardware is a custom-made printed circuit board with a 186-type controller (an AMD 186 running at 20Mhz) with 1.7MB flash, an IC bus, and another board with motor controller. It is powered with two lithium polymer batteries (3.6V 920mAh) from Danionics. We have developed the possibility to integrate Bluetooth (Ericsson ROK101) with the humanoid, in order to facilitate wireless communication.

The humanoid has a number of IR sensors, and can, for instance, detect a small IR emitting ball over the distance of 2m-3m.

2.3.8 Robot DB

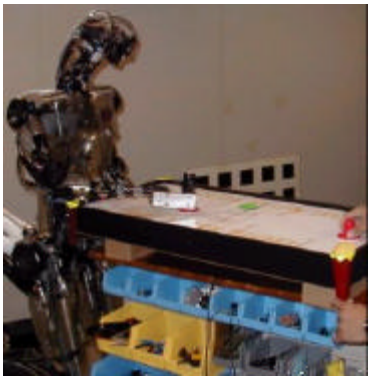

Figure 2.8 – Humanoid: Robot DB

Robot DB (Dynamic Brain) was developed from the Kawato Dynamic Brain Project of the Japan Science and Technology Corp [21]. The purpose is to build an experimental tool to explore issues of biological motor control, theories of neuroscience, machine learning, and general-purpose biomimetic robotics. Robot DB is185 cm high, weight 80 kg, has hydraulic actuators. For the vision system, it got two 2 degree of freedom eyes (pan and tilt), each with two cameras (foveal and wide angle, corresponding to human foveal and peripheral vision). The Sensors system consists of four cameras, an artificial vestibular system (three-axis angular velocity, three- axis translational acceleration), load sensors in 26 degrees of freedom, position sensors in all degrees of freedom. The cost is around $1 million US.

Robot DB learns from demonstration by imitating the movements of a person. Learning is preferred to programming movements, because the robot can, in principle, learn any new movement much faster than it can be programmed. The following are

some things Robot DB had trained so far; learned to strike the tennis forehand and backhand moves, it can also paddle a ball, it can also play air hockey against human, and emulate a human dance.

2.3.9 ASIMO

Figure 2.9 – Humanoid: Asimo

Asimo is developed from Honda Motor Co., Ltd. (HMC) [12]. ASIMO stand for Advance, Step in, Innovation, and Mobility. It was conceived to function in an actual human living environment in the near future. The design is to achieve compact and lightweight, more advanced walking technology, wider arm operating parameters, easy to operate and friendly design.

ASIMO had a weight of 43 kg, the walking gait had implemented cycle adjustable and stride adjustable, and had a walking speed between 0 to 1.6 km/h. The hand operation had a grasping force of 0.5kg/hand. The actuators used are servomotor, harmonic speed reducer and drive unit. It had a Walk/Operating Control Unit and Wireless Transmission Unit, i.e. walking autonomously or human control. For the sensor

system, there is 6-axis Foot Area Sensor at each leg, Gyroscope and Acceleration Sensor in the Torso. It is power with a 38.4V/10AH (Ni-MH).

ASIMO had 26 Degree of Freedom. 2 in the head, 5 for each Arms (3 DOF for shoulder joint, 1 at the elbow, and 1 at the wrist for each arm), 1 for each hand, and 6 for each leg (3 DOF for the hip joint, 1 for the knee joint and 2 for ankle joint).

2.3.10 PINO

Figure 2.10 – Humanoid: Pino

Kitano Symbiotic Systems was the project developed PINO, contributed with ERATO, JST, Osaka University and Sony CSL [10]. It is design to be the platform research for robotics and AI. There are four major issues in Pino's design;

- high DOF system to realize various kind of behaviors
- sensory system in order to recognize the exteroceptive and interoceptive
- information consisting of cheap and off-the-shelf components
- It has adequate size exterior to be able to consider the interaction with the environment

Pino had a configuration of 26 DOFs. They are distributed in such way, each leg has 6 DOFs, each arm has 5 DOFs, the neck has 2 DOFs and the trunk has 2 DOFs. The actuators it uses are cheap, compact and high torque servomotors. These SMs built in gearhead and its position controller, and these servo loops run at 50Hz. Two kinds of SMs (vender Futaba) are adopted for Pino which torque is 20 kg/_cm and 8 kg/_cm. And all of their gears are changed to metal for reinforcement them against high torque.

Pino has a sensors system consist of force sensor, posture sensor, joint angle sensor, and vision sensor. Pino uses 8 force sensors on each foot, which can obtain foot forces, a posture sensor on chest so that it can sense its body's posture. It also uses potentiometers in SMs as joint angle sensors to keep track of the dynamics. Pino also has a vision sensor mounted on head. All information of these sensors are sent to host computer via A/D converters (AD12-64(PCI), vender CONTEC) and frame grabber (GV-VCP2/PCI, vendor I/O DATA Japan).

CHAPTER 3

MECHANICAL DESIGN

This Chapter discusses the biped hardware used in this project. Although three humanoids were created for the project, nearly all the testing and development were done on Zaku and Tao-Pie-Pie.

3.1 Eyebot Microcontroller

The platform is based around the 32 bit Motorola 68332 central processing unit running at 16-33MHz. The 68332 also contain a timer processing unit (TPU) running at one-quarter clock speed. The TPU channels are used to control much of the hardware shich can be connected to the EyeBot platform.

The controller communicates through a number of digital input and output lines, an analog to digital converter, and a serial and parallel port controller. User interaction is through four input buttons and a 64x128 pixel liquid crystal display (LCD), as shown in Figure 3.1.

The platform has one megabyte of random access memory (RAM) and 512 kilobytes of read-only memory (ROM). The specialized operating system RoBIOS is stored in ROM, while user programs are stored in RAM. The EyeBot platform also provides the facility of storing user programs in ROM so that they do not need to be download each time the platform is powered up.

Figure 3.1 – Eyebot microcontroller

The EyeBot platform is designed to be a generic microcontroller for mobile robotics applications. As such, it provides interfaces to a variety of hardware, from digital cameras, analog and digital sensors, servomotors and DC electric motors, many of which are frequently used in mobile robotics applications.

Low level access to the underlying hardware is supported through the use of a hardware description table (HDT). This table contains entries that allow RoBIOS to know which hardware components are connected to the EyeBot.

3.2 Sensors

In order to compensate the two largest challenge autonomous agents and intelligent robotic control, humanoid need to determine their position and orientation with respect to their environment through indirect means. Tao-Pie-Pie uses two types of sensor, gyroscope sensor and digital camera. In order to obtain more accurate measurement with respect to its surroundings, more sensors could be used. However, in keeping with the goal of producing an inexpensive experimental biped, only these two types of sensor were used. Other sensors have been used in experimental bipeds and commercial manipulators include force sensors for the biped feet and position sensors for the actuators.

So far we only uses digital camera for determining the orientation, and for obstacle detection. We also mounted the Gyroscope sensor onto Tao-Pie-Pie, but only receiving pulse width modulated (PWM) signal. And we are planning to use it in further works.

3.3 Digital Camera

EyeCam is a full color (24 bits), low-resolution (80 x 60 pixels) digital camera to be used with the EyeBot controller. This resolution is sufficient for mst robotics tasks and allows fast image processing. The camera uses the latest CMOS technology which has a much better birghtness range than conventional CCD technology. EyeCam can be used directly on an EyBot controller or with an adapter cable on a PC,

using the Improv software. Camera size (PCB size) is 3.0cm x 3.4cm, including 4 mounting holes as shown in Figure 3.2.

Figure 3.2 – EyeCam

3.4 Actuator

The actuators used in the biped to control the joints are servos. These are small electric motors that are often used in remotely controlled vehicles to provide position control.

3.4.1 Servo Control

The servos are controlled using PWM digital signal of constant period. The duty cycle sent to the servo dictates the absolute angle it should turn to. For a typical servo, a 10% duty cycle will send the servo to the midpoint of its range. The servo performs this task internally by employing a PD controller, which ensures a low amount of overshoot, and fast response. Assuming there are no impediments to the rotation. More detail on the servos, and their calibration procedure can be found in

Nicholla's thesis [8]. The PWM signals are generated using a separate TPU channel for each servo. This alleviates load from the main processor.

3.4.2 Problems

(A) Servos – jitter

The servo update period is 20ms. When the servos are under load they move slightly away from the desired angle, and are then forced back during the next period. This problem is known as "jitter", which is noticeable as vibrations. Since the biped is small and it does not have a position sensor. If insufficient angle or overturned angle, may end up with an unbalanced system.

(B) Insufficient Torque

The servos are required to move the links of the biped quickly enough to maintain balance when walking. For most joints, the speed that the links are moved is adequate. In some instances, the torque that must be supplied by a servo is insufficient. This is very obvious with the ankle servos, sometimes it cannot lift the leg off the ground or it drops the swing leg to the ground too early and makes the biped walk in an unknown direction.

3.5 Biped Structure

The humanoid Rx78 and Zaku were modifies from toy models of Gundum warriors' animations. They both got 4 DOFs, 1 servo at the hip and 1 servo at the knee for each leg. As shown in Figure 3.3, we gained benefits from the original toy model; there, the mechanical design provides us 6 DOFs, hip, knee and ankle for both legs, which are sufficient to produce a walking biped. A wide range of robots had been manufactured and they are off the shelf product in a low cost. Due to the problems of plastic model cannot stand for the stress during the walk, the knee joints were cracks, and also lack of supporting space for strongly connecting actuators at the ankle.

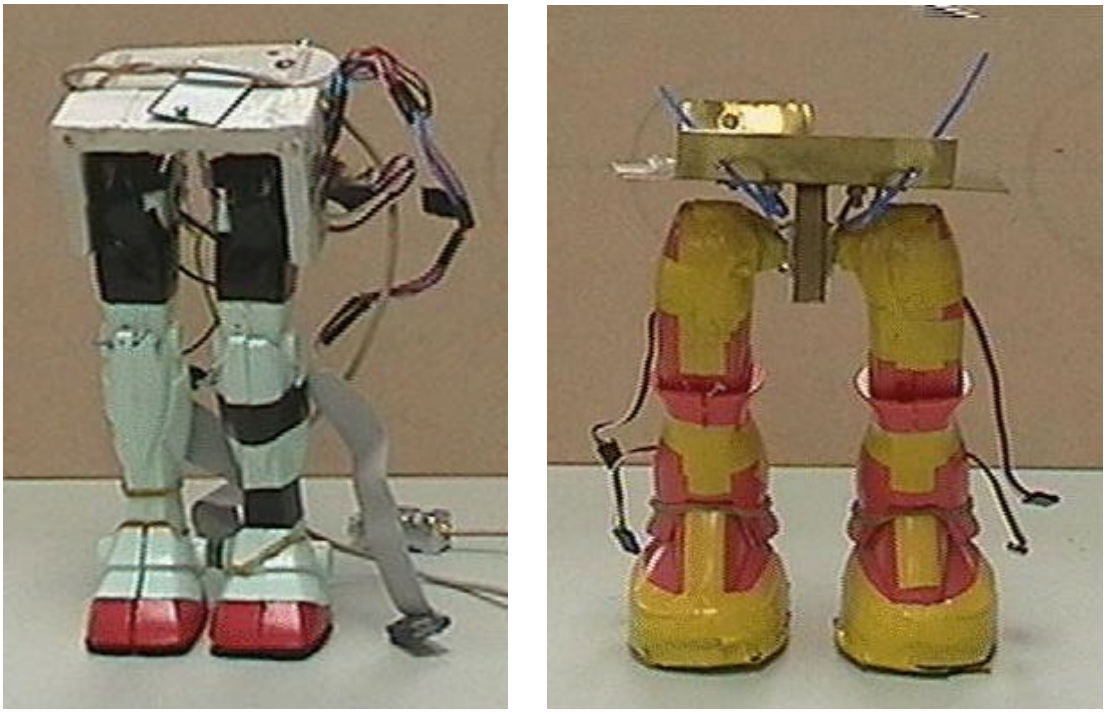

Figure 3.3 – Rx78 on the left and Zaku on the right

Tao-Pie-Pie was created using aluminium structure that is more robust to stress, as shown in figure 3.4. It was designed in conjunction with Dr. Nadir Ould Kheddal's group at Temasek Politechnic, Singapore. Appendix A9 shows a CAD drawing of

Tao-Pie-Pie. The drawing includes fundamental dimensions, all sensors, actuators, stand, and EyeBot controller.

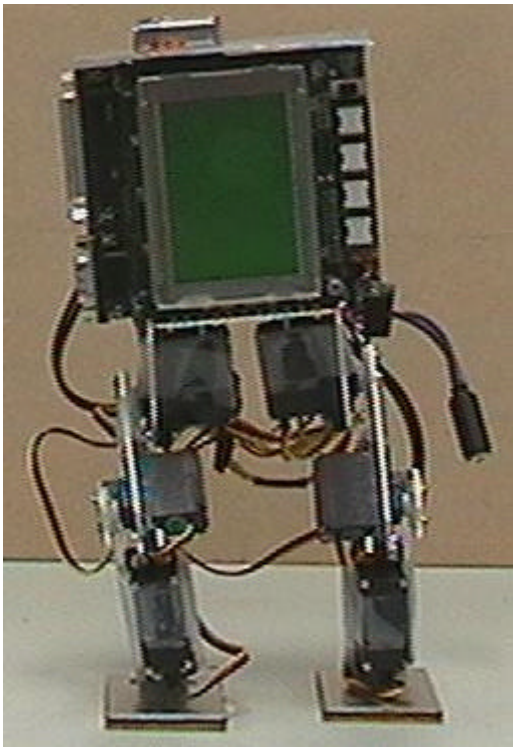

Figure 3.4 – Tao-Pie-Pie

CHAPTER 4

SIMPLE MODELS OF BIPEDAL WALKING

This chapter is going to use Inverted Pendulum Models [15] to exploit the behavior of a walking robot, before getting into the bipedal walking algorithm, let us examine some simple planar models of bipedal walking. We start with a simple inverted pendulum and evolve to more detailed models later. By understanding the influence of the dynamics the bipedal walking, we will gain a better understanding of important features of a walking gait.

4.1 Center of Mass and Center of Pressure

There are two facts that need to be considered before introducing the simple model of bipedal walking. Center of Mass (COM) and Center of Pressure (COP) [15]. Humans can walk easily without realizing where the COM and COP are and how important their roles are. For example, during advanced pregnancy and when nursing, a human female has an additional anterior load that moves her center of gravity forward and upward, making bipedal locomotion more difficult and energetically inefficient. A biped robot gait is said to be statically stable and a human posture is said to be balanced if the ground projection of its center of mass (GcoM), falls within the convex hull of the foot support area (the support polygon).

The center of mass is calculated according to its distance-weighted average location of the individual mass particles in the robot:

$$\mathrm{COM} = \frac{\sum Pm_i M_i}{\sum M_i}$$

(where Pm_i is the location of the mass particle i, and M_i is the mass of particle i)

COP is the pivot point of the human foot, the center point of the convex hull of the foot where it supports the most pressure. By moving the center of pressure, we have some freedom to control the rotational dynamics. A more detailed description will be presented later.

The center of pressure is calculated according to its distance-weighted average location of the individual pressures on the foot:

$$\mathrm{COP} = \frac{\sum Pp_i P_i}{\sum P_i}$$

(where Pp_i is the location of the pressure particle i, and P_i is the pressure of particle i)

4.2 Inverted Pendulum Models

When studying the human walking motion, we find that there are similarities between walking and the inverted pendulum. The pendulum pivot point is placed approximately at the center of pressure on the foot. The pendulum mass is placed approximately at the center of mass. Figure 4.1 shows a simple inverted pendulum of length l, with a mass m sitting on top, interacting with gravity.

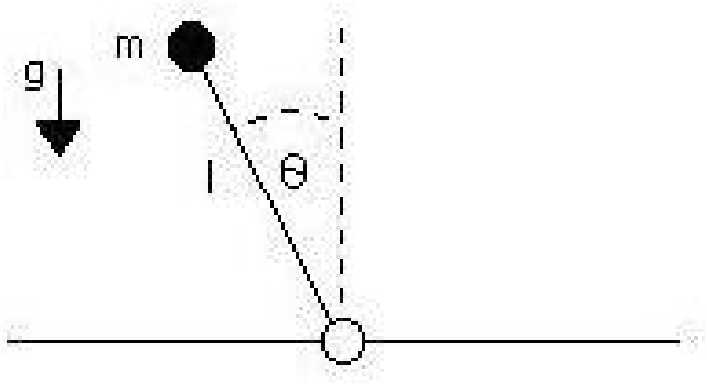

Figure 4.1 – A simple pendulum model of bipedal walking. m represents of the center of gravity. l is the length of the leg and q represents the stance leg angle

A pendulum has an equilibrium point in the straight up position and will accelerate in the direction of whichever side it is on. The further the mass is from the vertical, the faster it will accelerate. Note that there is no torque at the pivot point. Suppose the mass is travelling from left to right. If the mass is on the left-hand side, it will slow down towards the vertical. If the mass has passed the vertical, it will accelerate to its right. At this point the system is converting kinetic energy into gravitational energy when it travels from left to vertical and convert it back into kinetic energy from the vertical to the right. According to the *law of conservation of mechanical energy* the total mechanical energy remains constant during the motion of the particle.

The initial kinetic energy is:

$$E_k = 1/2mv^2$$

The change to potential energy is:

$$\Delta E_P = mgl(1 - \cos\theta)$$

By setting the change of potential energy equal kinetic energy:

$$\cos\theta = 1 - \frac{v^2}{2gl}$$

For small angle approximation, we get

$$\theta = \frac{v}{\sqrt{gl}}$$

4.3 Linear Actuator Pendulum Model

Now suppose we add a linear actuator along the length of the pendulum, as shown in Figure 4.2, the force on the point mass lies along the length of the leg. The acceleration of the mass in the radial direction depends on the actuator force, F, the gravitational force, and the fictitious centrifugal force due to the rotation of the pendulum. Again, assume the mass of the pendulum is travelling from left to right. With the actuator pulling the mass, the rotation motion will accelerate. Extending the mass will decelerate the speed. This is the same as sitting on a spinning chair where the pivot point is underneath the chair, when opening the arms during spinning, the rotation will slow down, while closing up the arms will increase the rotational speed.

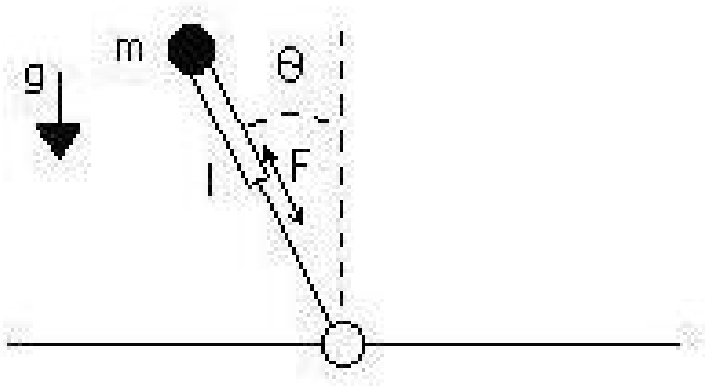

Figure 4.2 – A simple inverted pendulum with a linear actuator

4.4 Multi Joint Pendulum Model

This model shares same characteristics with the linear actuator model, but implemented in a different mechanism. In order to transform a linear actuator model into a multi joint pendulum model, we have to show that both dynamics are identical. Figure 4.3 illustrates the similarities, by using a two degree of freedom (DOF) multi joint model with the parameters mass point m, length l of the leg.

With a different mechanical design, to obtain the same parameters we have to use inverse kinematics to calculate the angles for each individual joint. The only difference between the two models is the force gain from the actuator, for linear actuator model it is a linear force. This model has a torque generated by the knee servo should be have minor influence.

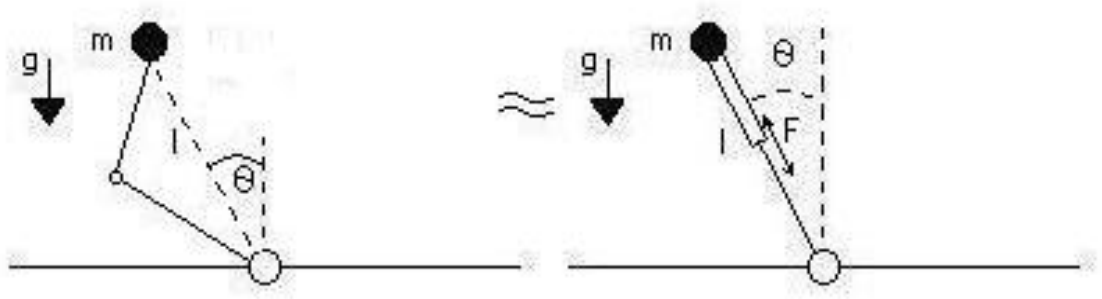

Figure 4.3 – Multi joint simple inverted pendulum module

4.5 Ankle Pendulum Model

So far we have show how to control the speed of the rotation motion of the inverted pendulum by adjusting the length of the leg but still it is not a well-balanced system. Figure 4.4 adds a foot and ankle brings many benefits. It gives us a larger supporting area (convex hull of the feet) for the center of mass to stay below when travelling. It also helps us to control the speed of the pendulum and leads us to a more stable system.

The most challenging thing to balance the mass is to shift it left and right from the vertical above the pivot point. As mentioned before, the pivot point of a human is the center of pressure, the distance-weighted average location of the pressure. If the COP is left of the COM then the mass point will accelerate to the right. If the COP is right of the COM then the mass point will accelerate to the left. By controlling the joints torque, we can arbitrarily control the location of the center of pressure.

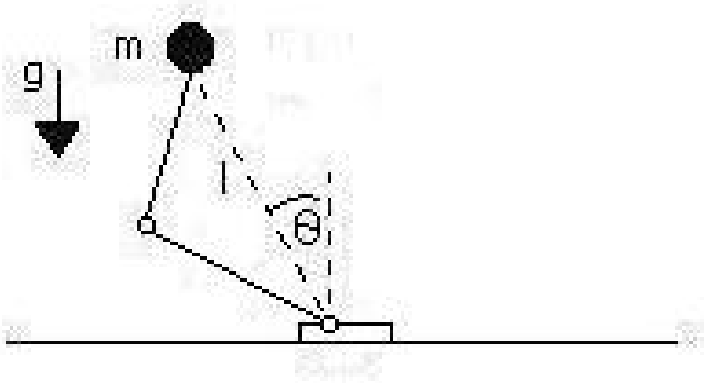

Figure 4.4 – A simple inverted pendulum with ankle and foot

CHAPTER 5

WALKING ALGORITHM

In Chapter 4 we had model the natural dynamics of a walking pattern by the Inverted Pendulum models, we know that there is some behaviors that are necessary for generating a walking gait. This Chapter is going to present Inverse Kinematics and Bezier Curve that had direct contribution to the Walking Algorithm, follow by the Software Design Scheme, the overview, then the implementation of the Walking Algorithm.

5.1 Inverse Kinematics

Inverse Kinematics computes the joint parameters necessary for a given destination point as shown in Figure 5.1. It is an important aspect since the most difficult problem in robot walking is stability. After working out the center of mass, we need to keep the COM within the supporting area. By controlling the joint angles, we can control the stability during the motions.

$$\theta_1 = \left| \frac{-(L_2 \sin(\theta_2)x + (L_1 + L_2 \cos(\theta_2)y)}{(L_2 \sin(\theta_2)y + (L_1 + L_2 \cos(\theta_2)x)} \right|$$

$$\theta_2 = \left| \frac{\cos(x^2 + y^2 - L_1^2 - L_2^2)}{2L_1 L_2} \right|$$

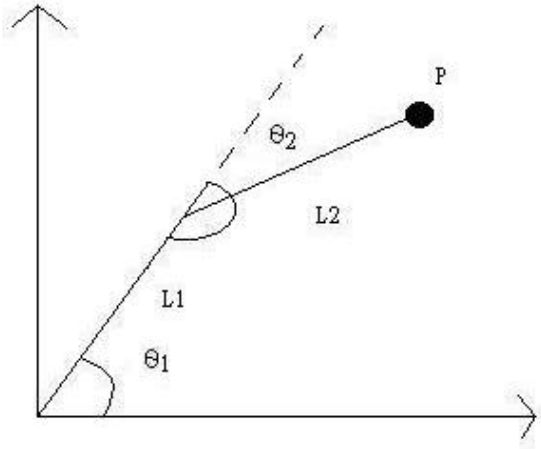

Figure 5.1 – Inverse Kinematics problem depicts the robot. Where P, represent center of mass, L2 – hip joint link, L1 – Knee joint link

5.2 Bezier Curve

Bezier curve in its most common form is a simple cubic equation. It was developed by Pierre Be'zier in the 1970's for CAD/CAM operations. It can be used for drawing model and also animation. The cubic equation requires four points parameters, the first point is the starting point (original end-point), second and third are the control points used to adjust the curve, the fourth point is the end point (destination end-point). It evaluates an arbitrary number of value t between 0 to 1. If t increases it will return a number close to destination endpoint, if t is closed to 0 it will return a number close to original endpoint.

Figure 5.2 illustrates a Bezier curve. The green circle is the start point, the circle is the end point, and the squares are the control points. By adjusting the control points we can achieve a smooth curve used to control the angles of each individual joint.

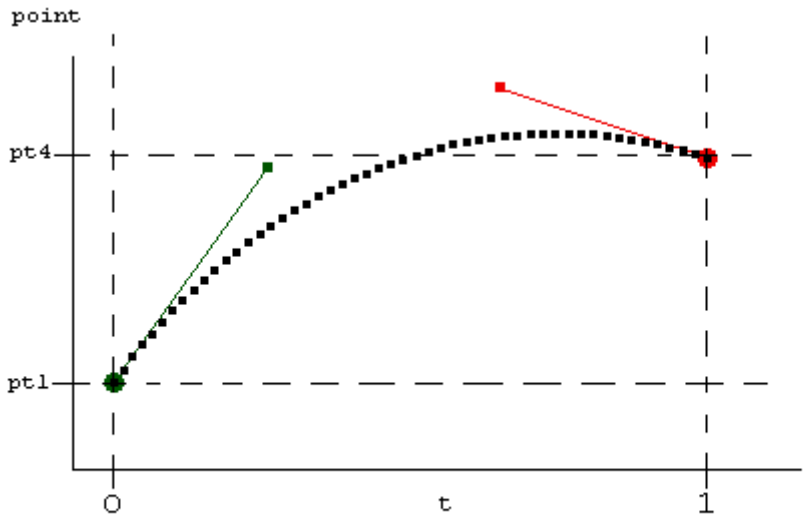

Figure 5.2 – Bezier Curve

5.3 Software Design Scheme

5.3.1 Overview

The overview of the implementation for generating a walking gait for small humanoid is shown in Figure 5.3, it consists of Six different classes, Main, Pattern, legservo, Bezier Curve, walkCam, and fastCam. The over all logic is divided into one program and two systems; The Main program is the design of the walking style. The Pattern Generation System, it invoke configuring the servos, setting up the control curve (Bezier Curve) and split the walking gait into six different phases. The Vision System, this is developed to capture images from the camera sensor, setting up the obstacle parameters, using fluid algorithm is to detect obstacle that the humanoid need to avoid or move toward.

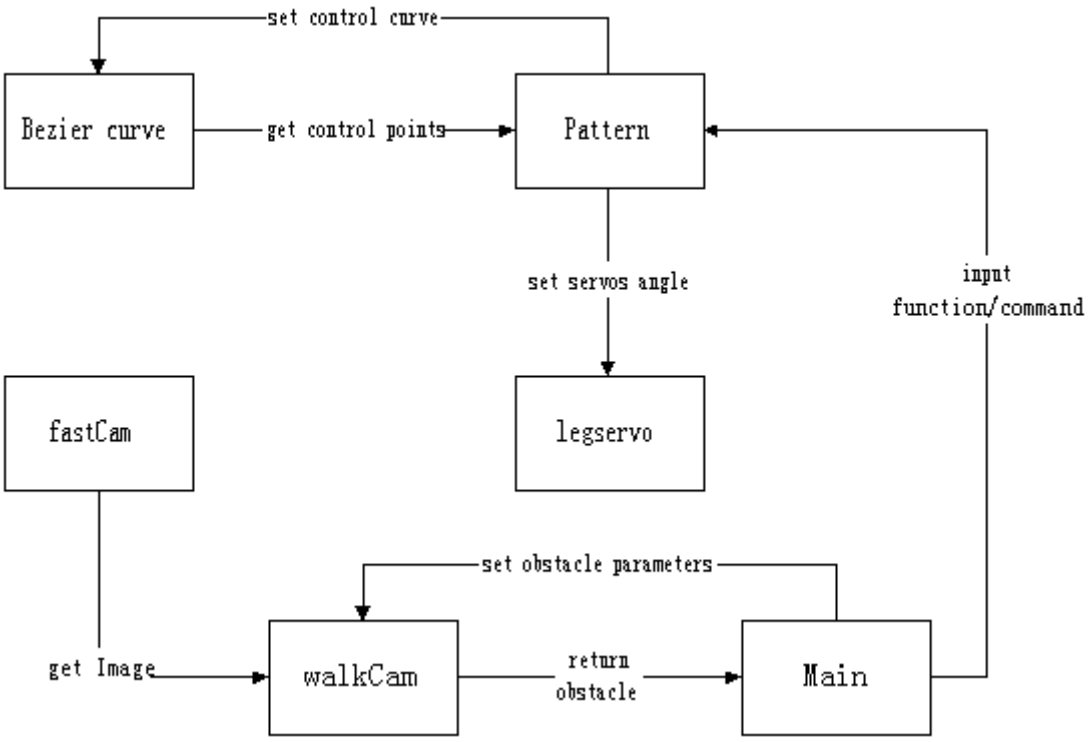

Figure 5.3 – shows a level 0 data flow diagram illustrating the overview of the Walking Algorithm

In Figure 5.3, Main is the executable class served as a higher-level path planner. It is use to determined the behavior of the robot. The behavior are controlling the walking speed, the number of steps for walking or turning, a left or right turn, how robot deals with obstacle - walk pass it or turn away from it, the step size (the distance between front foot to rear foot). There are all sort of behaviors; in general, Main class controls the overall flow of the gait for humanoid starting at position A and move to destination B.

Pattern class, legservo class and Bezier Curve class are part of Pattern Generation System. By describing the lowest level class first, then work the way up will gain a better knowledge for the Software design.

The legservo class is an interface between all the servos, it is used to adjust and constraint the range of the servos, the reason for this is to imitate human close as

possible. For example, it is only possible for human to bend the knee until the heel almost touches the hip, and extend it until lower leg and thigh are lined up together. The range will be divided evenly and valued from 0 to 255. By working out the desire value correspond to the desire angle for each joint, this class can also turn the servo into the proper location for generating the locomotion.

The Bezier Curve as mention in the beginning of this Chapter. It is a simple cubic equation, given a set of control points we can generate a curve. Applying an arbitrary value t between 0 to 1 into Bezier function, we can obtain the servo angle for different time instance. Bezier Curve class is an interface for setting up the control points and retrieving the servo value in a particular time instance. Different settings can achieve different gaits for different situation, and this will explain in more detail at Chapter 6.

The Pattern class is the heart of the Pattern Generation System, it is served as a listener for the Main class command, distinguish it into speed, patterns and phases. So far there are four different patterns implemented, they are Walk, Turn, Kick and Bow. We also adapted six different phases that contribute for a walking cycle, three phases for each leg and both are symmetrical. This class will initialize the predefined control points for the Bezier Curve class, initialize the joints constraints for each servo in the legservo class. Then according to the input command – what kind of pattern and which phase to start and end, and what speed. Pattern class will adjust the control curve and joints constraints respectively and start walking.

WalkCam and fastCam class contributes the Vision System. The major role for this system is to capture images during the walk and processing the images. To filter out the unnecessary information and only return the desire obstacle where the biped

required or satisfy the obstacle parameters, this include size of the obstacle and color of the obstacle.

The reason for choosing the closest obstacle amount the rest instead of the whole picture is because the humanoid only had one camera. This can only process image in two dimensions that can only give us the edge and size of the obstacle, but not the depth and distance from the robot. Even through there are algorithms that can retrieve 3D information using one camera, such as Light Plane Projection algorithm and we can capture images as fast as 30 frame per second. This enable us to use optical flow to extract moving object and also recover 3D information using disparity check, combine with Stereo-graphic Projection algorithm using two consecutive images. Due to lack of processing power, although we can capture images fast enough, we cannot process 3D information in real time. The Algorithm we use for obstacle detection is Fluid algorithm, that is a simple and quick processing algorithm for grouping an object with similar character.

The fastCam class is a low-level interface that enables fast image capturing, before hand we are capturing images 15 frames per second. That is the library supported by the CPU and we over clock the camera. The fastCam class has two main functions, initialize the camera in 15 frames per second or 30 frames per second, and getting image.

The walkCam class is the most complicated class for Vision System. It is use to store the obstacles parameters, the size, and the twelve color parameters. Initialize the fastCam handler, request image from fastCam, performed Fluid Algorithm to the captured image and return the obstacle information to the main program. When the

main program initialize walkCam class, it will set the obstacle parameters into the storage space, and the walkCam will return the detected obstacle to main program for AI purpose.

Now we should already understand the overview of the software design, it consists of one main program, Pattern Generation System and Vision System. In the remaining of the chapter, we will describe each System more details, and how the models discuss in Chapter 4 is applied.

5.3.2 Pattern Generation System

Pattern Generation System support the main program the natural dynamics during the walking gait, it ensure the stability of the robot during the walking cycle, and the stability of the robot during the transition between patterns (walk, kick, turn and bow). This system provide a very powerful tools for implementing advance technology such as, reinforcement learning, Optimal path planning, and Dynamic Brain …etc. Without the system, it requires a lot of time for the robot to train before it can walk in a natural manner. And the main contributions for the stability are the simple pendulum models, the Inverted Kinematics and the Six Phases.

Simple pendulum models help us to understand and exploit the dynamics during the walking cycle in the following ways. The pivot point is placed approximately the center of pressure. The pendulum mass is placed around the center of mass, and further the mass is from vertical (ground projection from the pivot point), faster the pendulum will accelerate. Rotation motion will accelerate when the actuator pulling the mass and rotation motion will decelerate when the actuator extends. By

controlling the joints torque, we can arbitrarily control the location of the center of pressure, and when COP is left of the COM then the mass point will accelerate to the right. These models help us to establish the walking motion, and understand ways to achieve stability by controlling the servos.

Inverted Kinematics is use to compute the joint parameters necessary for a given destination point, in our case it is the COM. In order to keep the walking balance; we had to ensure the COM lies underneath the supporting region during the trajectory. This also limits the joint range, and with the help of Bezier Curve. We can setup a proper walking pattern to enforce it. The Six Phases breaks the walking cycle into three different states per foot, and maintaining the stability become more feasible and easier to handle.

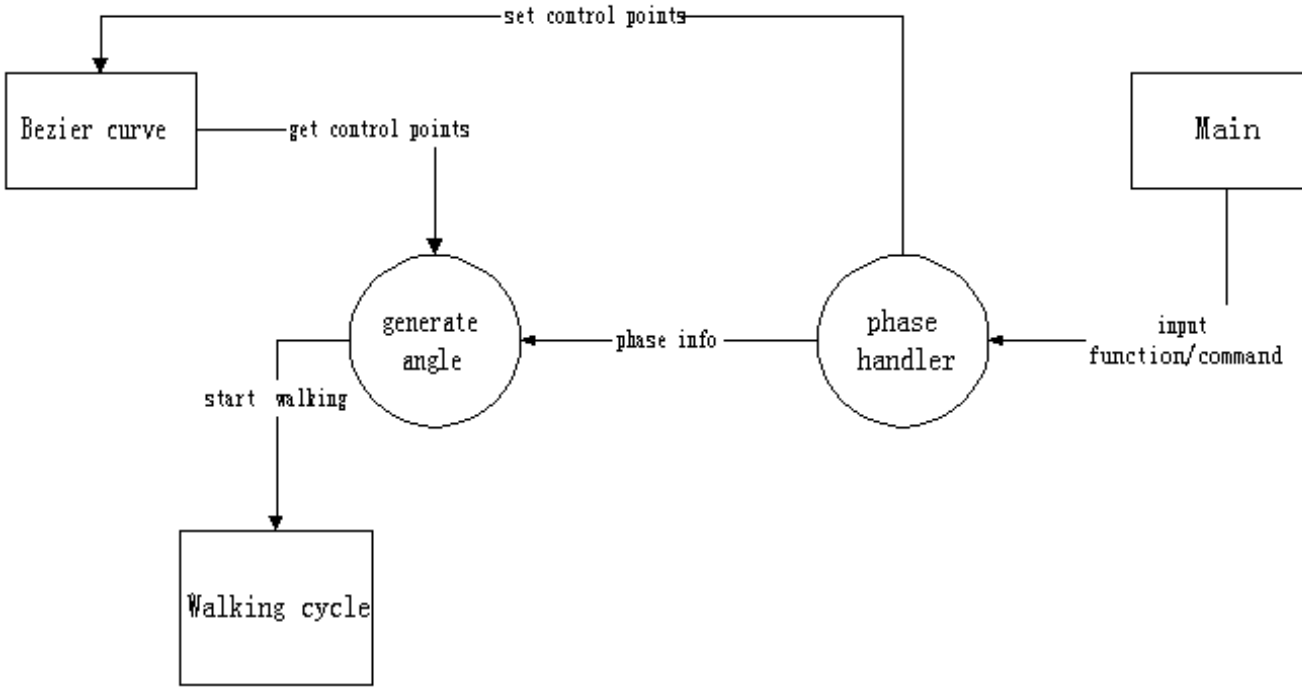

Figure 5.4 – shows a level 1 data flow diagram illustrating the role of Pattern Generation System

As shown in Figure 5.4, in order to initiate the walking cycle action, the Pattern Generation System needs to receive high-level command instructions from the Main

program. The basic command instructions include walking pattern (WALK, KICK, TURN, and BOW), starting phase, ending phase, and speed of the walk cycle. After receiving the command from the Main program, the phase handler turns the pattern flag on/off with respect to the input pattern, initialize Bezier Curve by setting up control points, then forward the phase info to generate angle process. This process then initialize the leg joints position into ready mode, get the servos value for each joints and trigger the walking cycle.

(A) Walking Cycle

Walking Cycle has six phases, three phases for each leg. As mention before, generate angle process will activate the cycle/phases, according to the given start phase and end phase. In other word, it will select either One leg Stand phase, or Ready for Landing phase, or Two legs Stand phase as shown in Figure 5.5. The number in the phases correspond the timing or state of the cycle between left leg be the supporting leg and right leg be the swinging leg or right leg be the supporting and left leg be the swinging leg.

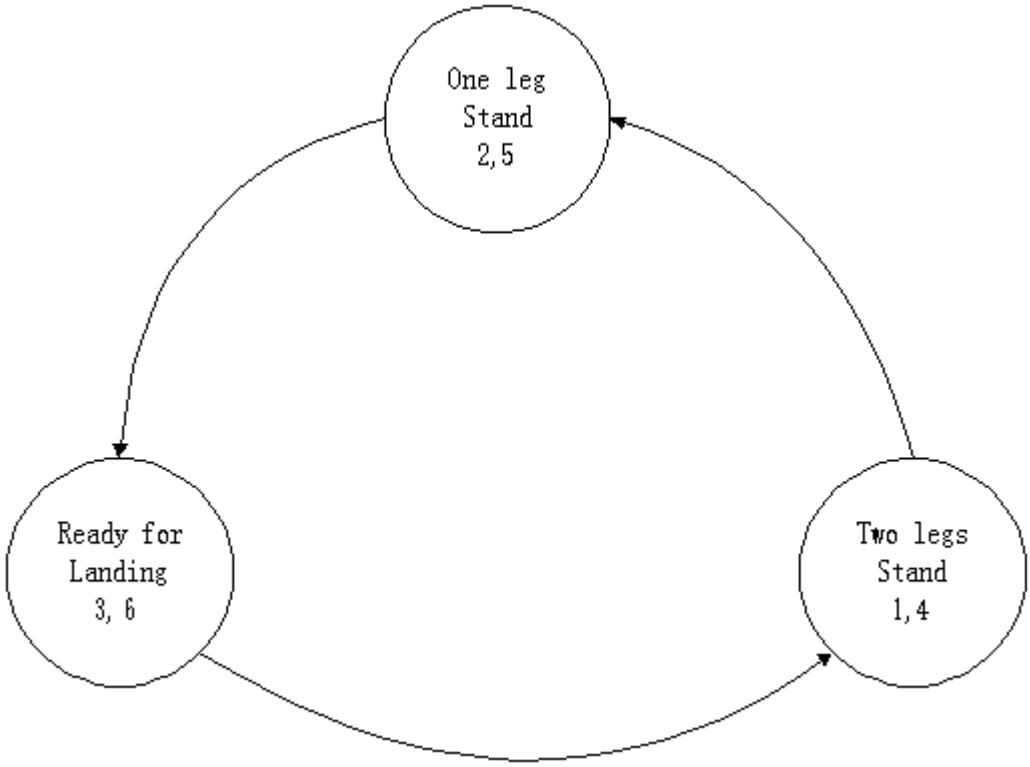

Figure 5.5 – shows a level 2 data flow diagram, illustrating the WALK pattern of the Walking Cycle

(A.1) WALK pattern

Figure 5.5 shows a WALK pattern of the Walking Cycle, during this pattern the robot will start at Two legs stand, that is one leg stand in front and the other stand rear. The reason for this is to move the COM forward in to the supporting area of the front leg and ready for the rear leg to lift over the ground. Since the robot is in static mode, the pivot point COP stays under the COM. Then next step is to transit forward into One leg Stand phase, during the transition the torque created by the turning of ankle servo will slide the COP towards the inner area of the support foot for a while. As mentioned before when the COM is not sit vertically above COP, so the robot will lend on either side and lift the rear leg above the ground. When the ankle servo reaches the desire angle, the torque will disappear then the COP will move back under the COM and this is the One leg Stand phase. The final step is moving the rear leg

forward, turn the joint angles to a landing position and this is the Ready for Landing phase. Switch back to Two legs Standing is simply turn the ankle servo of the supporting leg in the other way, in this case the COP will slide towards the outer side of the foot due to the torque generated by the servo. When COM is not line up vertically with COP, it will fall into the free leg direction and land back into Two legs Standing phase.

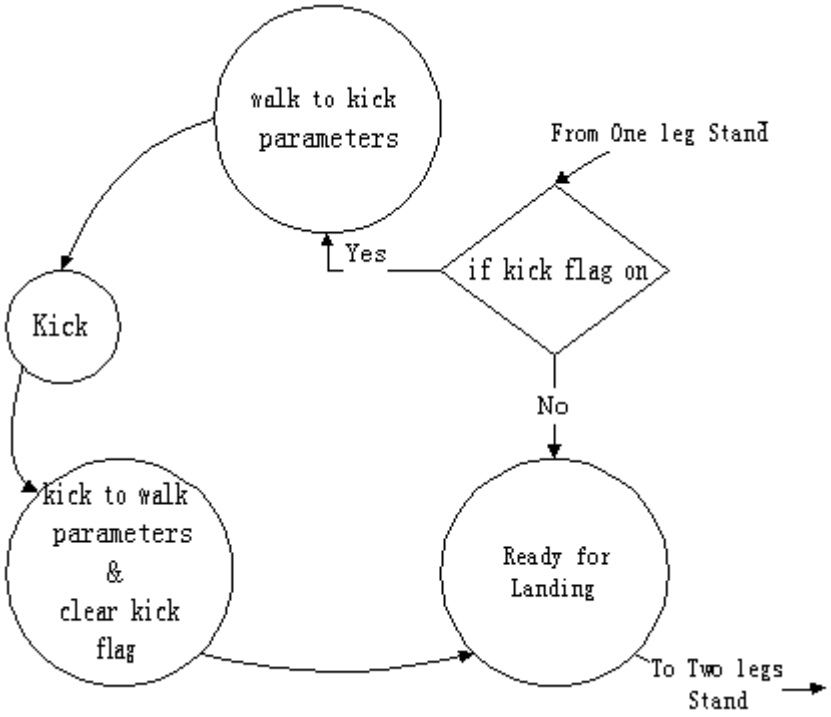

Figure 5.6 – shows a level 2 data flow diagram, illustrating the KICK pattern of the Walking Cycle

(A.2) KICK pattern

Figure 5.6 shows the KICK pattern of the Walking Cycle, this is similar to the WALK pattern. They had the same transition between Two legs Stand phase to One leg Stand, and Ready for Landing phase to Two legs Stand phase; except it will jump to another states that are outside the cycle. As Pattern Generation System section mentioned before, the phase handler would switch on the flag for kick motion when a

high-level input command request it for a KICK pattern. From One leg Stand phase onward, it will check if the kick flag is on, if it is not then carry on with the WALK pattern. Otherwise, it will jump to a process that will switch the walk parameters to kick parameters. In this process will also change the robot configuration by lending the torso forward and move the COM to the front edge of the supporting foot, that is important because the next process will start kicking. When pulling the free leg backward and ready to initiate the kick, it will move the COM back into the middle of the supporting leg. The next process is kicking, swinging the free leg forward and kick the ball. The relation of the COM will end up at the front edge of the supporting leg again. The final process will change the kicking parameters back to walking parameters, switch the turn flag off and carry on with the Ready for Landing phase.

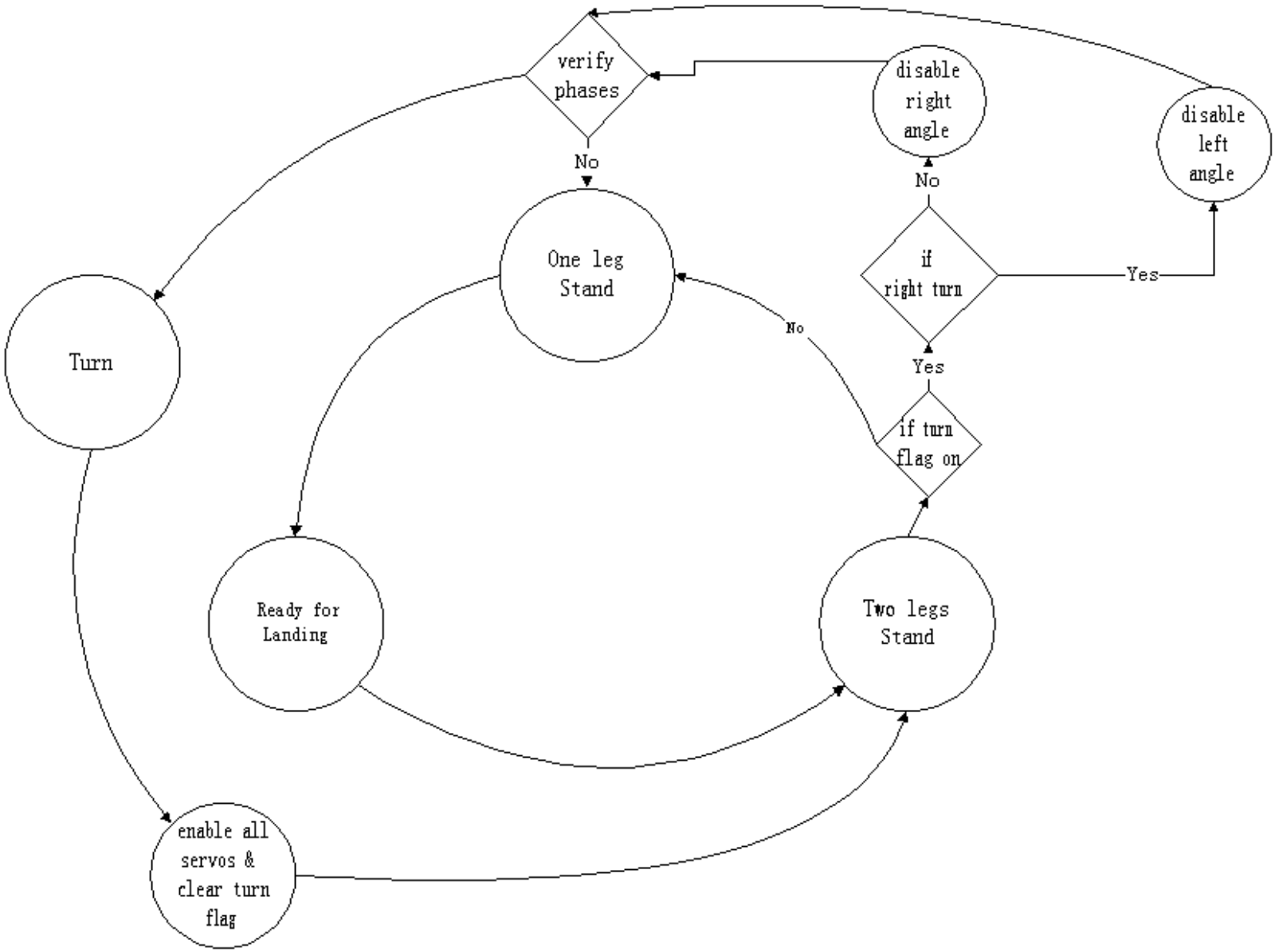

Figure 5.8 – shows a level 2 data flow diagram, illustrating the TURN pattern of the Walking Cycle

(A.3) TURN pattern

Figure 5.8 shows the TURN pattern of the Walking Cycle. For this pattern, there are no parameter changes because it shares the same parameters as walking, and an explanation is discussed in the chapter (Experimental robot). The phase handler process that is done in a higher level will turn the turn flag on, and this flag consist of two states: right turn and left turn. From Two legs Stand phase, it will verify the turn flag, and if the turn flag is off, it will transit to the normal Walking Cycle. Otherwise, it will check if the flag is a right turn, if so it will disable the left ankle servo, else it will disable the right ankle servo. It will perform another verification for the timing of the phases, if it is not the right phase for the correct leg then it will result a normal routine for transition between Two legs Stand to One leg Stand phase. Repeat again until the correct phase for the correct turn. When the verification for the timing of the phases had pass, it will jump to a Turn process. Inside this process, because the front leg ankle servo already disabled; by pulling the front leg back and pushing the rear leg forward, it will twist the body to the desire direction, and switch the front leg to rear and rear leg to front. The final step is to enable the blocked ankle servo and the dynamics will be ready for the next Walking Cycle.

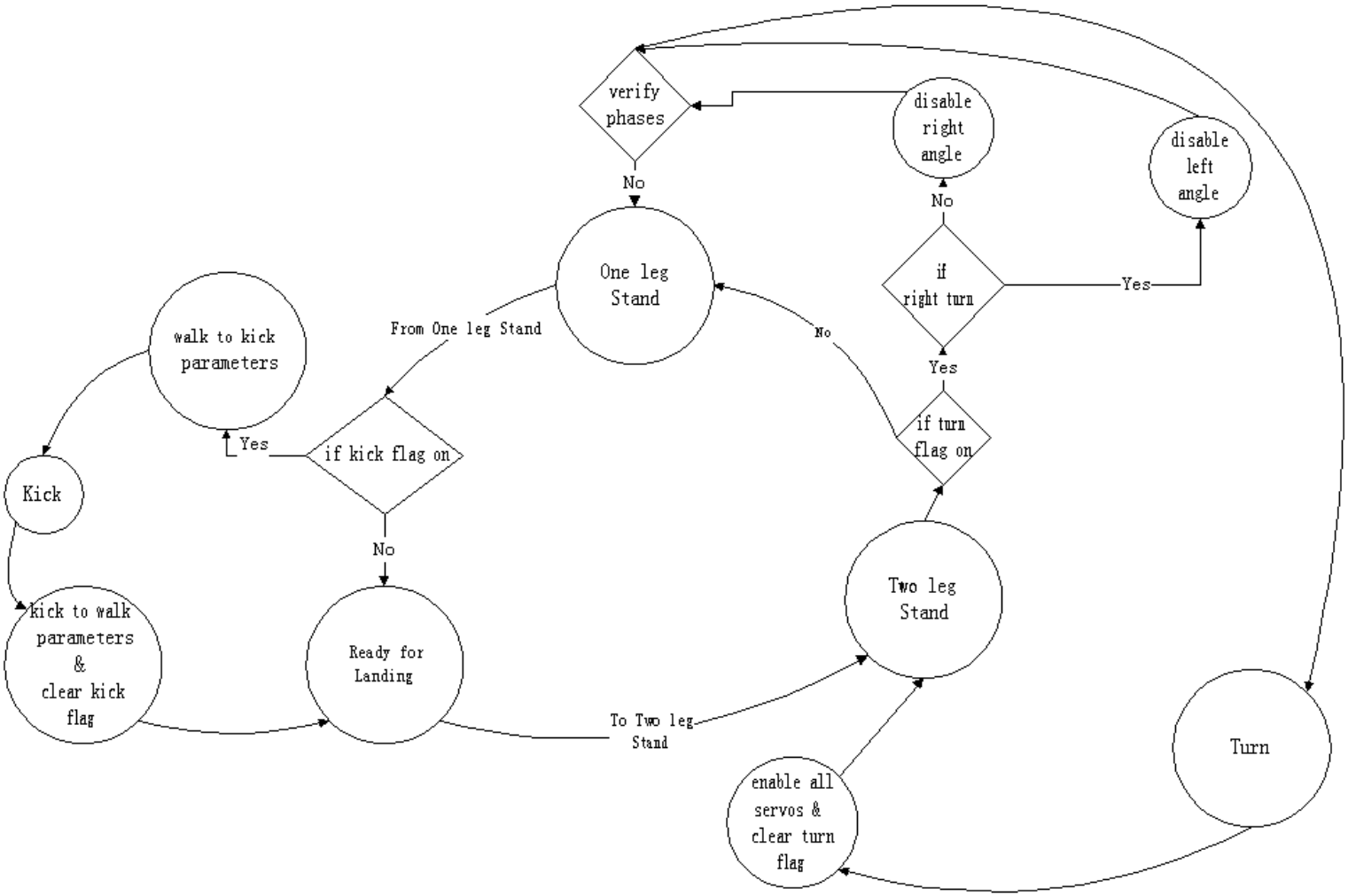

Figure 5.9 – shows a level 2 data flow diagram, illustrating the ALL pattern of the Walking Cycle

Figure 5.9 shows the summaries for Pattern Generation System consist of Pattern class, legservo class and Bezier Curve class. It handles the high-level instruction from Main program, setting the control curve for gait generation, limiting the driving range of each joint servo, controlling the dynamics of the humanoid, maintaining a Walking Cycle, and jumps between different states when dealing with different patterns.

5.3.3 Vision System

Vision System supports the main program Artificial Intelligent, and this is the only sensory device active during the walking gait now. It gives an interactive opportunity for the robot with the environment and human. For example, if the robot try to walk from one end of the room to the other, due to uneven surface, small offset of the servo calibration, low battery level… etc. the errors will accumulate and the robot may end up turning back in the middle of the room. With the help of a line marked on the floor, the robot can follow the line, correcting its direction and walk to the other end of the room. Vision System makes the robot more autonomous, it gives more space for the walking style. It helps protecting the robot when seeing obstacle, it can turn away and avoid collision that might damage the robot. Vision System also provides an entertainment opportunity for the robot, and this is mention in Chapter 6.

(A) Fluid Algorithm

Fluid Algorithm is used for image processing. It is a simple, easy and fast way to detect objects from an image. The algorithm processes the image from left to right, top to bottom for every pixel. Given the obstacle parameters, the size and the twelve color parameters, these are: Red Min, Red Max, Green Min, Green Max, Blue Min,

Blue Max, Red-Green Min, Red-Green Max, Red-Blue Min, Red-Blue Max, Green-Blue Min, and Green-Blue Max. It will search every pixel of the image to find the pixel that satisfies these color constraints. Then recursively fluid the 4 neighbor of the pixel and marked; the marked pixel only visited once and will not revisit again. Until there is no more neighbor that satisfies the constraints, it will calculate the size of the detected object, centroid, and stored in an obstacle structure.

As shown in Figure 5.10, in order to detect an obstacle, Vision System need to gather the information for the detective obstacle, that is given by the higher-level class, the Main program. When the system fetches a new image from the fastCam class, it will scan the image from left to right, top to bottom and passes the pixel to the next process. This process will check if current pixel is the last pixel remaining in the image, if so the system will finish processing for this image and return the detected obstacle to Main program for AI process. Otherwise, it will compare the pixel with the obstacle parameters given by the Main program, if it does not satisfy it will branch back to the next pixel process. If the current pixel matches the obstacle parameters, that means the pixel value is within the range of the color parameters for the detective obstacle. Then it will go to the next process that will apply the Fluid Algorithm to the pixel and return a detected obstacle. As mentioned in the Software Design Scheme – Overview, since we are lack of processing power and could not draw 3D information out from a single camera, we assume the larger obstacle in the image will be the closest obstacle. The final process will compare the detected obstacle and the stored obstacle, update the storage if necessary and repeat from the next pixel process again.

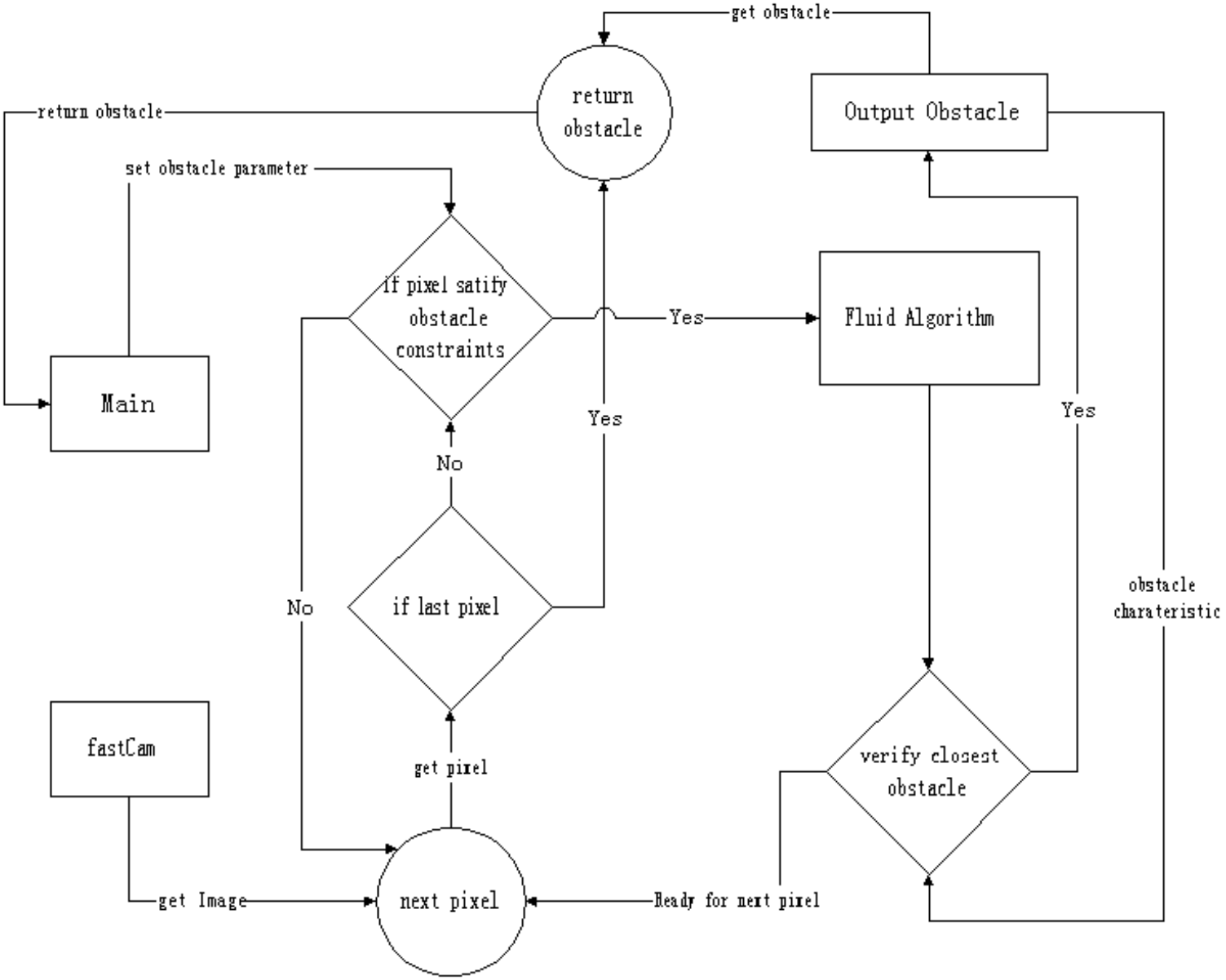

Figure 5.10 – shows a level 1 data flow diagram illustrating the role of Vision System

CHAPTER 6

SYSTEM EVALUATION

In Chapter 5 we had explained the Walking Algorithm that generates a walking gait for a small humanoid. We also understand the theory behind the Pattern Generation System and Vision System. Experiments on the real robots are emphasis in this Chapter. These composed of Bezier Curve, transition between the walking cycle, and obstacle detection.

Inverted Pendulum Models mentioned in Chapter 4 had influence our experiment a lot. It had exploited some of the behaviors necessary for controlling the stability of the humanoid. For our experiments, we assumed the center of mass is a point that lies above the center of the legs, and we learned that by controlling the length of the leg (length of the COM to COP), we can change the speed of the bipedal walking motion. We also learned that if sufficient amount of force by the joint torque is generated, we can move the center of pressure along the supporting feet in order to maintain center of mass vertically above the pivot point. This helps us control the humanoid postural stability as well as the static and dynamic stability during the walking.

6.1 Evaluation 1 – Bezier Curve for Pattern Generation System

Bezier Curve is a very power tools for generate a walking gait for small humanoid. It is a simple cubic equation that provides sufficient control points for implementing a simple, smooth, natural, and versatile gait patterns for the robot during the walk. A GUI was developed to facilitate this experiment. Figure 6.1 shows the Bezier Curve

GUI; it consists of four input spaces, point one the starting end, point two and three are the control points, and point four is the destination end. After the start button is press, the GUI will display the output curve. By inputting different control points, we can obtain different curves that are use to predict the behavior of the gaits before testing it onto the robot. This could also eliminate unnecessary exams that could possibly damage the robot.

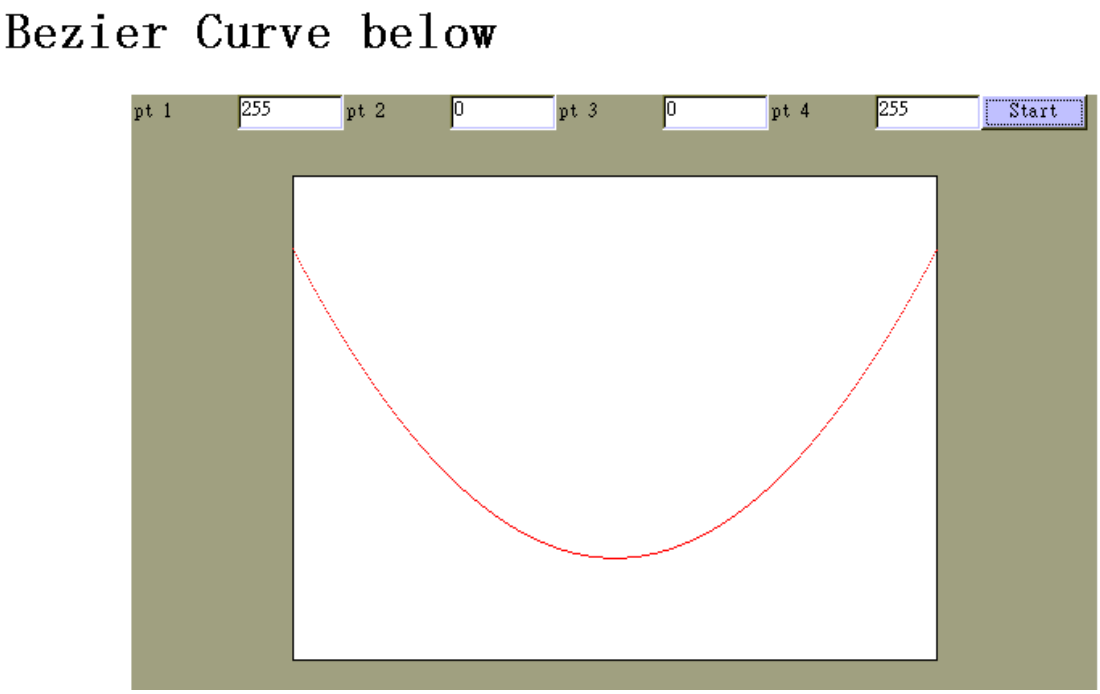

Figure 6.1 – Bezier Curve GUI

Figure 6.2 shows the control curve tested on our second-generation robot, Zaku. When t is between 0 to 0.25, it changes rapidly, at interval 0.25 to 0.75 it changes in a slow pace and at 0.75 to 1 it changes normally.

Zaku only had 4 Degree of Freedom (DOF), one in the hip and one in the knee for each leg. The software developed for Zaku has no Phases implemented it only use Bezier Curve for every walking cycle. The idea for this curve is to generate a walking gait. Even through the experiment fail, it led us to new directions and understands the

minimum DOFs that is required for a walking robot. Due to the limitation of Zakus mechanic, it requires a large force from the actuator to shift the COM. The predicted walking gait is as follow. At interval 0 to 0.25 the bend leg extend in an instance, and this provide sufficient force to move the Center of Pressure (COP) toward the outer edge of the foot of the stretching leg, which result a dynamic balance state (One leg stand). The free leg will swing in an inverted pendulum behavior. To control the speed during this transition, we had to stretch the supporting leg for as long as possible. As the Inverted Pendulum Models in Chapter 4 mention, the further COM is away from COP, faster the pendulum mass will accelerate.

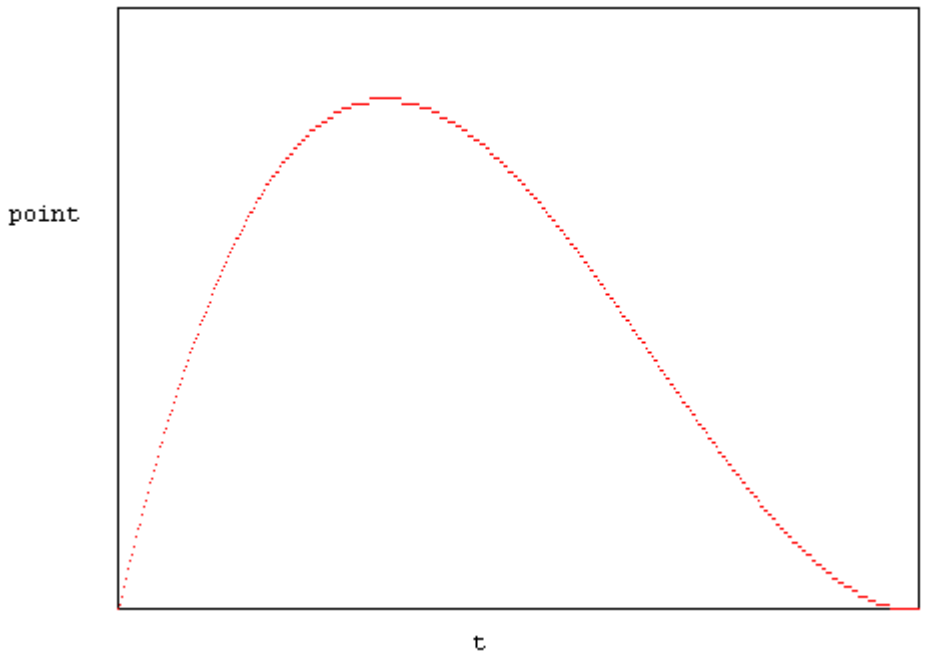

Figure 6.2 – Bezier Curve tested on Zaku

Using Bezier curves, we can form a smooth control curve for individual joints. Figure 6.3 shows the final control curve we had implemented for our latest robot, Tao-Pie-Pie. When t is between 0 to 0.25 curve changes rapidly, at interval 0.25 to 0.75 it changes slowly and at interval 0.75 to 1 it changes rapidly again.

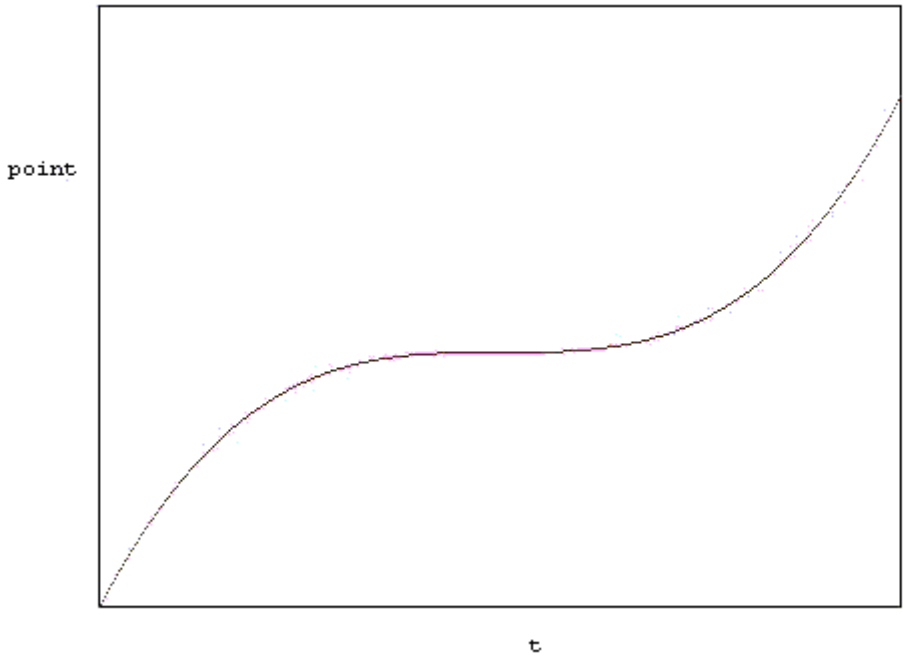

Figure 6.3 – Bezier Curve of our walking pattern

Tao-Pie-Pie Humanoid start with an arrow standing position where front leg bent, rear leg straight. To transit between the next step, position of both leg were swapped, the transition requires a large torque from the front leg joints to shift the COP into the heel of the front leg. Meanwhile the rear leg needs to bend or leave off the ground quickly so that the COP can move into the front leg (dynamic balance position) and the COM is in front of COP. Eventually the robot will swing forward. This is the phase when t is in the interval of 0-0.25 or 0.75 to 1. During the transition, we want to control the speed of the swing so that the humanoid walks naturally and also avoid crash landings. To maintain a slow frontal swing velocity, we have to keep the distance between COM and COP as far as possible. So during interval t equal 0.25 to 0.75, the front leg (standing leg), the leg with the COP will keep stretching for a while and the bending leg (free leg) will keep the bending motion as long as possible. Because the further the leg stretches, the more the COM moves away from the vertical. According to the Inverted Pendulum Models, we know that the further the

COM is away from the vertical, the faster the rotational pendulum and that is not what we want.

This experiment demonstrates a successful usage of Bezier Curve to control the walking motion for each joint. Bezier Curve GUI was developed to predict the behavior of the walking gait. Applying different control points will construct different control curve, which then alter the behavior of the robot during the walking gait. Figure 20 illustrate the final control curve we had implemented in Tao-Pie-Pie.

6.2 Evaluation 2 – Phases for Pattern Generation System

Phases were the latest strategy we had developed to improve the body balance of our humanoid during the motions. It is only implemented in our third generation robot, Tao-Pie-Pie. The walking cycle is divided into six different phases, three phases for each leg and these consist of “Two legs Stand”, “One leg Stand”, and “Ready for Landing” phases. The reason for developing these phases is to split the walking cycle into intervals, the smaller the problem, the easier it is to control and a more stability system become feasible. We also developed three patterns that are interpolated from these phases. These are “Walking pattern”, “Turning pattern”, and “Kicking pattern”.

6.2.1 Walking pattern

Figure 6.4 shows the movements of a straight walking cycle. In this case, the biped uses the final control curve given from the previous experiment to control the joint angles. Walking Cycle consists of 6 phases of movements, and successive walking gait is possible by repeating the phases from phase 1 to phase 6.

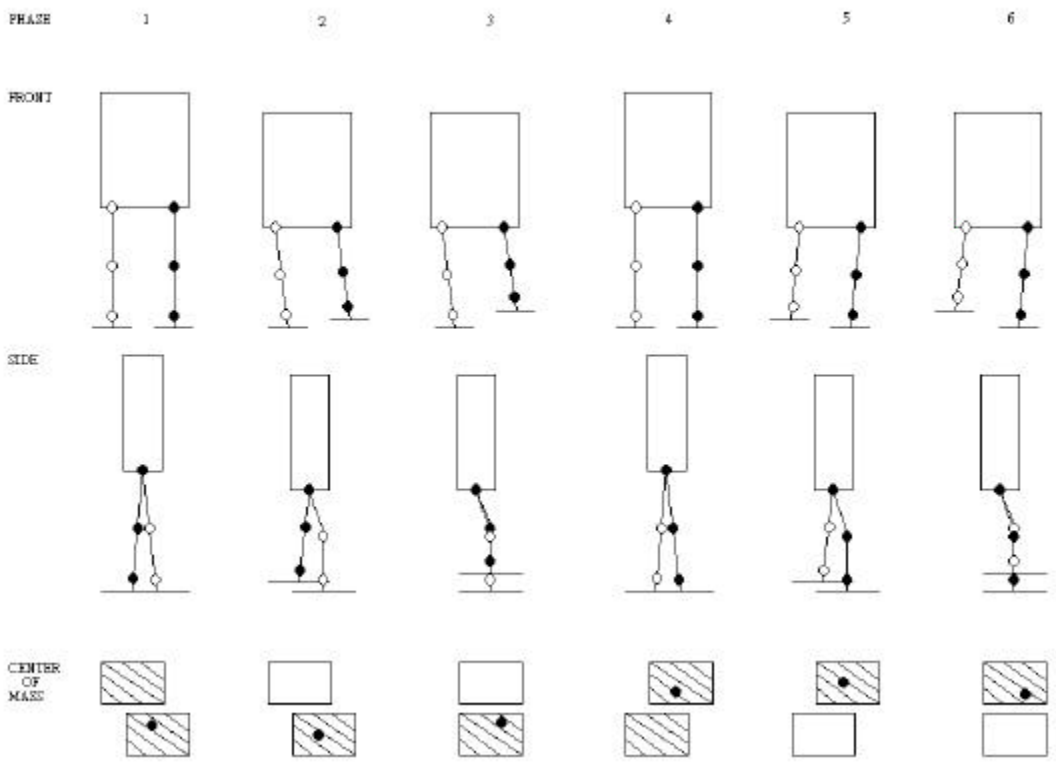

Figure 6.4 – Movement of Walking pattern

This figure also illustrates the COM position during the gait. Tao-Pie-Pie start from phase 1 – "Two legs Stand", where the right leg is in front and the left leg in rear. In this instance, both feet are placed on ground and the COM was kept in the middle of the supporting area, slightly to the inner side of the front leg. Transition between phase 1 to phase 2 – "One leg Stand" requires the torque generated by the ankle of the front leg. The torque wills arbitrary slides the COP towards the inner edge of the

front leg. According to the features we learned from Inverted Pendulum Models Chapter 4; COM will move to the opposite side of COP if they are not vertically lined up. Since the COM is on the right-hand side of COP, the biped will lean to its right, shift the COM over its right-hand side and lift the rear leg off the ground. Leaning will stop until the ankle servo reaches its desired value, and free the ankle torque. Here the COP will reposition back under the COM, which is in the middle area of the supporting foot and now we are in phase 2. Transition from phase 2 to phase 3 – "Ready for Landing" is done in static mode. The robot pulls the free leg forward and adjusts the kinematics into landing position. This setting will bring the COM onto the front end of the supporting foot, which makes the next transition more robust. A proper landing position can maximize the supporting area once both legs contact the ground and the ideal case is landing the free leg flat onto the ground. The final transition from phase 3 back to phase 4 – "Two legs Stand", which is a mirror reflection of phase 1 is done in dynamic mode. The supporting leg will extend its knee joints to shift COM outside the front end of the supporting area. The supporting leg will turn the ankle servo to the outer side, and moves its COP towards the outer edge of the foot. Now the biped swing towards the diagonal side from the COP to COM, and eventually it will land with both legs on the ground. Phase 4 till phase 6 repeat similar steps as phase 1 to phase 3 because they are symmetrically identical.

6.2.2 Turning pattern

We had achieved two different turning patterns. In the first pattern, if the robot want to turn right, reducing the step size of its right leg will results turning right. But this is not physically desired for our robot, because it takes a long range to walk before it turns. In the second case, due to the mechanical limitation of DOFs, we can only

performed a sharp turned by twisting its lower body and Figure 6.5 illustrate the movements of such Turning pattern. The turn phase will swap the front leg back, rear leg forward, and during this motion the COM and COP will sit in the middle of the whole supporting area.

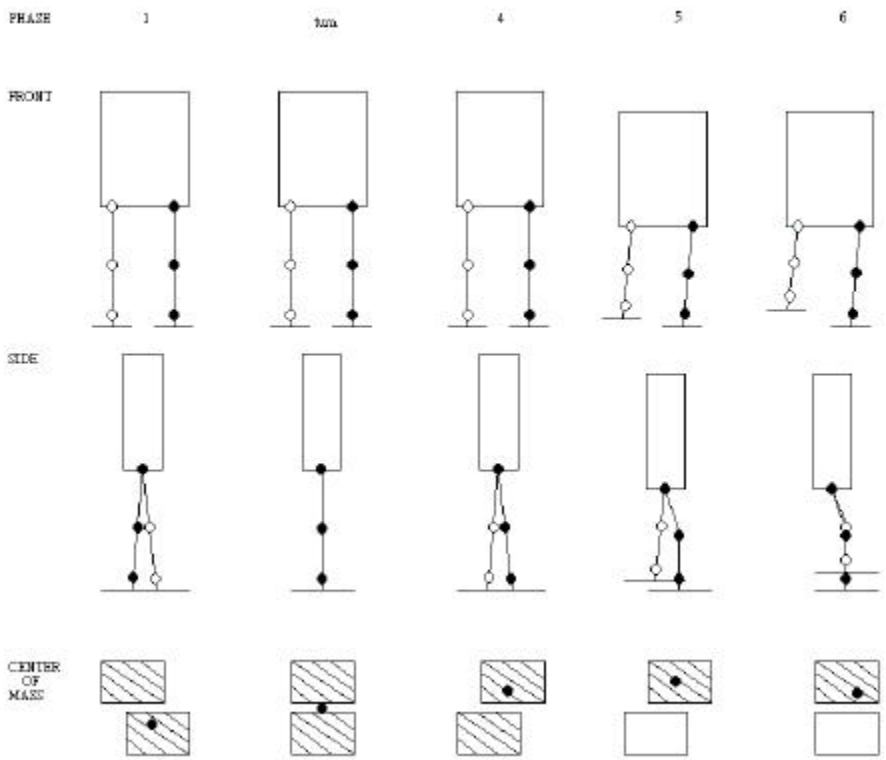

Figure 6.5 – Movement of Turning pattern

6.2.3 Kick pattern

Figure 6.6 shows the movement of kicking pattern. It is an transition stage from phase 2 to phase 3, pulling the free leg as rear as possible will increase the potential energy and provide a wider range for the strike in order to perform a stronger kick. To keep the consistency between the phase changes, the main task is to hold the COM in the same position that will be in phase 3 – "Ready for Landing". The biped leans forward the upper body to keep the COM in the front end of the supporting leg.

Straighten the upper body after the kick is performed which will result in landing position.

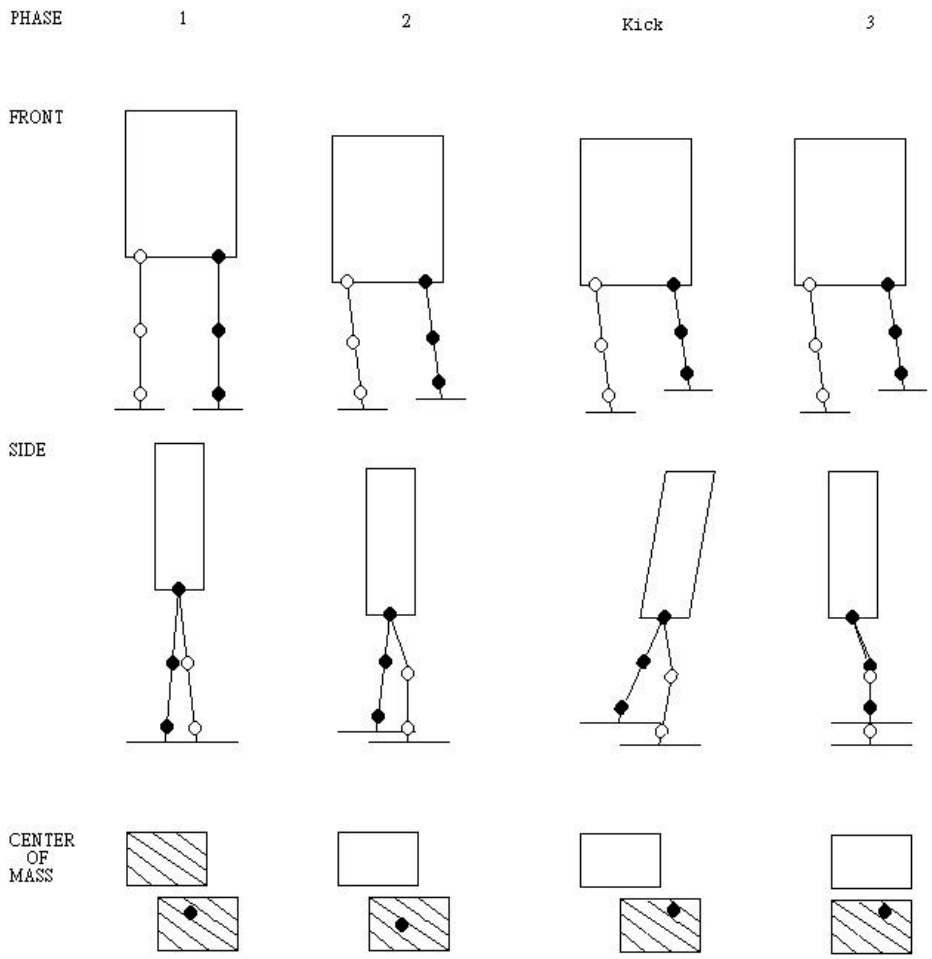

Figure 6.6 – Movement of Kicking pattern

6.3 Evalutation 3 – Obstacle detection for Vision System

Vision System provide the AI for the walking gait. It gives the walking gait more robust to the environment. For example, the robot can detect an obstacle, turn away to avoid collision which may damage the robot. The main challenge for it is image processing, detecting an obstacle. In chapter 5, Walking Algorithm had already describe the theory behind the Vision System.

In our lab, there is a color calibration library which had been developed and installed in Gimp, an image manipulation program under Linux. This section we are going to show some pictures of the detective ball, and giving a hint on how color calibrations work.

Figure 6.7 – First set of color detection images

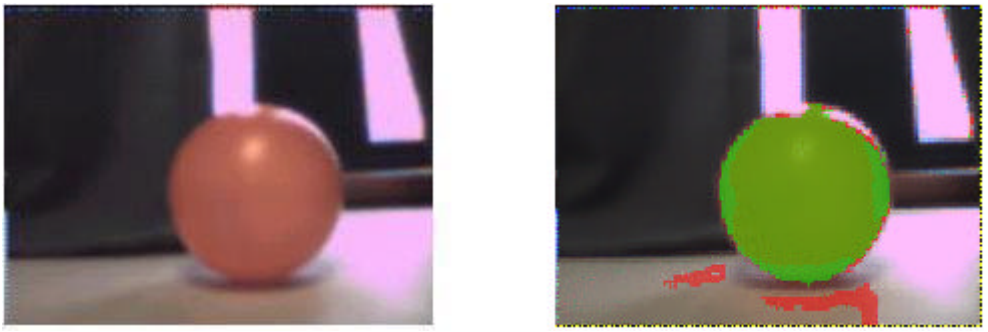

Figure 6.8 – Second set of color detection images

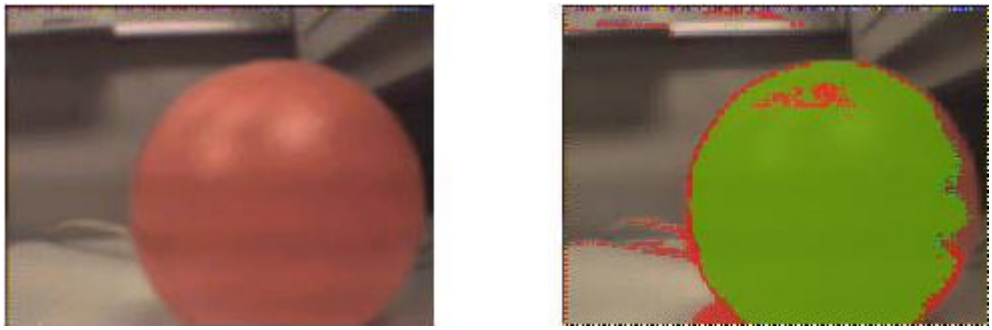

Figure 6.9 – Third set of color detection images

MIN	Red	Green	Blue	R – G	R – B	G - B
1st set	104	51	44	33	42	3
2nd set	153	81	67	50	46	-4
3rd set	136	66	53	67	78	8

Table 6.1 – Minimum range of the color parameters

MAX	Red	Green	Blue	R – G	R – B	G - B
1st set	168	108	84	70	90	26
2nd set	254	177	143	92	126	40
3rd set	224	136	111	95	120	38

Table 6.2 – Maximum range of the color parameters

Figure 24 to 26 are some successful color parameters used to detect the orange ball. Each set composed with left image, the original image and right image, the image processed image and the green block indicate the position of the obstacle detected by the Vision System. Calibrating the right parameters depend on the texture, the lighting of the environment, the color, the auto-focus of the camera, and the background. And Table 6.1 and 6.2 shows the parameters values.

6.4 Evaluation 4 – Fira2002 and Robocup2002

Currently, the humanoid competition is in its infancy. Fira2002 and Robocup2002 held the first humanoid league this year. The intension is to encourage research in practical, autonomous, highly mobile, flexible, and versatile robotic platforms. Reduce the steep learning curve toward fully autonomous soccer. As a benchmark

problem, the goal of the humanoid league is to develop humanoid robots that can play soccer well against other robots or even human opponents.

The physical challenges of the league for both Fira2002 and Robocup2002 is around three areas. Robot Dash, an event that require the robot to walk or even sprint as fast as possible from one starting line to the finish line. Penality Kick, the robot must approach to the ball and kick into the goal. Obstacle Run, similar to the Robot Dash challenge. The robot must move from one end of the playing field, that is distributed with obstacles as quickly as possible to the other end.

Tao-Pie-Pie was registered for both competitons. In Fira2002, it had demonstrate a stable walking in the Robot Dash, unfortunately one of the servo is not working fine, due to overworking for hours. The right leg was not strike in its full strength, in result our robot turns back just in front of the finish line, which make the crowd very exciting. In Robocup2002, there is a event so call Free Style. Every team can show off what they can do with their robot in five mins. With the combination of walking and obstacle detection, Tao-Pie-Pie had perform an awsome show. It had demonstrate its autonomous by letting a little girl helding a piece of pink paper with a Japanese word, “GO” one side and the other side is white with word, “Stop”. When the robot saw the page GO, it start walking. When it saw Stop, it stop immediately. In Penalty Kick event at Robocup2002, there are a goolie which is guarded by the competitive team. The stratgey we use is combining the walking pattern with turning and kicking. Tao-Pie-Pie approach to the ball directly, that confuses the opponents defense, then turn to left or right in front of the ball and shoot it into the goal.

This year, the humanoid competitions are very opened. It open up for the teams to use IR transmition, so human can control the robot and correct their path. Since this is the first year for humanoid league, most of the team focuses on developing a high speed walker. For our team, Tao-Pie-Pie a small humanoid with optimal DOFs and processing power, which limit our walking speed. Instead of building a fast walking humanoid, we had developed a fully autonomous robot that does all the processing and a feedback from the camera to avoid obstacle. Tao-Pie-Pie had win an autonomous award in the Fira2002 competition, a second place for Penalty Kick in Robocup2002 competition, and a third place for Free Style in Robocup2002 competition.

CHAPTER 7

CONCLUSION

Developing a scheme for controlling humanoid walking is a difficult task. This project is aimed to produce a prototype for biped robot for intelligent learning. We had already applied some intelligent control with our humanoid once the walking gait was developed. This project has model the walking behaviors by the use of Inverted Pendulum Models. Developed a Walking Algorithm and implemented using the experimental robots, Rx78, Zaku, and Tao-Pie-Pie that were created in our laboratory.

7.1 Modeling

Inverted Pendulum Models help us understand the behaviors of the robot during testing, and lead us to evolve to the final motion that is very stable and natural. We understand the relation between center of mass and center of pressure i.e. rotational motion will accelerate when the actuators pulling the mass, when COP is left of COM then the mass will accelerate to the right. The use of torque generated by the ankle to arbitrary shift the center of pressure.

7.2 Walking Algorithm

The Algorithm is composed with the systems, Pattern Generation System (PGS) and Vision System (VS). For PGS, we had make used of Bezier Curve to generate smooth control points to control the kinematics of the links. We had developed a Walking

Algorithm with these control points, and divide the walking pattern into six phases three for each leg. Two legs Stand, One leg Stand, and Ready for Landing. Inserting different transition phase will had created three different walking patterns, WALK, KICK, and TURN. For VS, we used the existing fastCam library that was developed in our lab for 4 stogges [13] to capture images. We developed the code for obstacle detection and use a simple image-processing algorithm, Fluid Algorithm.

This Algorithm also builds up library functions for the next humanoid project. It had supported functions for the use of Bezier Curve, functions for the walking cycle, control of the servos, obstacle detection and these are shown in Appendix A1 to A8.

7.3 Achievement

One of the initiatives for this project is Fira2002 and Robcup2002 competition. The goal is to Compete, find out the weakness of our approach and present us with opportunities for improving our design. We noticed the walking speed of our robot is slow compare with others, one of the reasons is its ankle only got one degree of freedom and we got no room for extra servo. Other teams had suggested solutions, such as mounting the servo on the knee joint and make use of timing belt to control the extra DOF, make use of gears to reduce the number of servos.

During both competition, we had gain an international reputation for our robot by winning an Autonomous Award in the Fira2002, a second place for the Penalty Kick event in Robocup2002 and a third place for the Free Style in Robocup2002.

7.4 Future Work

There is no accurate orientation for our robots, we had assumed the walking algorithm function properly, due to different environmental issues, such as slippery surface, uneven surface...etc. The orientation of the robot will off track after a time interval. Due to the jitter problem which had describe in the mechanical design. There exist a synchronization problem between the dynamics balance period with the walking algorithm. Lack of 3D information, the biped can not kick the right spot on the ball in order to gain the maximum power, also the robot can not do advance intelligent planning a field of obstacles. These induced the following fields for future projects.

- Orientation measurement
- Intelligent control
- Path planning
- Reinforcement learning

Two Gyroscopes can be used in different axis to draw feedback from the system and adjust the parameters in order to synchronize the dynamic balance period with the bipedal waking algorithm. Integration with the gyros reading can give use an estimation of the orientation and we can use the gyros for reinforcement learning to correct abnormal behavior for different surfaces. Image processing with two cameras can provide 3D information for the environment and we can use it for path planning and intelligent control.

7.5 Final Impression

Rx78, Zaku, and Tao-Pie-Pie had provided important stepping stones and insights into the design of a small humanoid robot. A Walking Algorithm had been developed and fully tested. The key concepts in the areas of controlling legged locomotion and robot manipulation have been learnt. The relationship between the Inverted Pendulum and a walking pattern had been studied. Practical problems interact with the real world been solved which will be remembered and likely be applied in the future.

References:

[1] Ambarish Goswami, Bernard Espian, Ahmed Keramane. Limit cycles and their stability in a passive bipedal gait. INRIA Rhone-Alpes, France.

[2] Ambarish Goswami, Bernard Espian, Ahmed Keramane. Limit cycles in a passive compass gait biped and passivity-mimicking control laws. INRIA Rhone-Alpes, France.

[3] Allen S. Parseghian. Control of a simulated, three-dimensional bipedal robot to initiate walking, continue walking, rock side-to-side, and balance. Technical report, Electrical Engineering and Computer Sciences University of California, Berkeley, 1998.

[4] Autonomous Humanoid Robots Androids. Homepage http://www.anice.net.ar/intsyst/artisBod.htm

[5] Chalmers Robocup Initiative. Homepage http://humanoid.fy.chalmers.se/

[6] Damien Kee. GuRoo: Humanoid Robot Project, in proceedings of the 2002 International RoboCup Symposium p474.

[7] Eric R. Dunn, Robert D. Howe. Foot Placement and Velocity Control in Smooth Bipedal Walking. Division of Applied Sciences, Harvard University, Cambridge Massachusetts 02138.

[8] F. Nichollas. Bipedal Dynamic Walking in Robots. Final year honours dissertation. Electrical and Electronic Engineering University of Western Australia, 1998.

[9] Federation International de Robotic Soccer. Fira hompage. http://www.fira.net, May 2002.

[10] Fuminori Yamasaki1;2, Tatsuya Matsui1, Takahiro Miyashita1, and Hiroaki Kitano1;3. PINO The Humanoid that Walk. Kitano Symbiotic Systems Project, ERATO, 2 Osaka University, 3 Sony CSL

[11] Greatest humanoid project. Homepage. http://www.androidworld.com/prod01.htm

[12] Honda technology humanoid. ASIMO Hompage http://world.honda.com/ASIMO/

[13] Jacky Baltes. The 4 stooges homepage. WWW, November 1999. http://www.citr.auckland.ac.nz/~jacky

[14] John Camp. Powered "Passive" Dynamic Walking. Masters of Engineering Project Report. The Sibley School of Mechanical and Aerospace Engineering Cornell Univeristy, Ithaca NY. August 1997.

[15] Jerry E. Pratt. Exploiting inherent robustness and natural dynamics in the control of bipedal walking robots. Master's thesis, Massachusetts Institute of Technology, 2000.

[16] Jerry E. Pratt and Gill A.Pratt. Exploiting natural dynamics in the control of a 3d bipedal walking simulation. Technical report, MIT Leg Laboratory, 1999.

[17] Mariano Garcia, Andy Ruina, Michael Coleman, Anindya Chatterjee. Some Results In Passive-Dynamic Walking. Department of Theoretical and Applied Mechanics Cornell University, Ithaca, NY 14853 USA.

[18] R. Stojie, C.Chevallereau. On the Stability of Biped with point foot-ground contact. Institut de Recherche en Cybernque de Nantes, France.

[19] Richard Quint van der Linde. Passive Bipedal Walking with Phasic Muscle Contraction. Delft University of Technology, Netherlands, 2628CD.

[20] W. T. Miller, Real-time neural network control of a biped walking robot. IEEE Control System, pp.41-48, February 1994.

[21] Wired archive 8.09 – Sept 2000. Homepage http://www.wired.com/wired/archive/8.09/robosapiens.html?pg=3

APPENDIX

A.1 WalkCam.h

```
typedef struct {
  int size, cx, cy;
  unsigned char wMax, wMin, hMax, hMin;
} OBSTACLE;

void error (char *str);
void InitCam ();
void InitCam2();
void setMenu ();
void param_menu ();
void scaleImage(frame *src, colimage *dest);
void createImage (frame * fp, int width, int height);
void destroyImage (frame * fp);
void displayImage ();
void sendString (int port, char *s);
void sendRGBData (int port, frame * fp);
int SendPic (frame * fp);
void CopyImage (frame * from, frame * to);
int TakePic ();
void getImage();
void ulImage();
void destroyAll();
void flud_image(frame *src, unsigned char x, unsigned char y, OBSTACLE *obstacle);
OBSTACLE detectObject();
void setParam(int RMin1, int RMax1, int GMin1, int GMax1, int BMin1, int BMax1, int RGMin1, int
RGMax1, int RBMin1, int RBMax1, int GBMin1, int GBMax1);
```

A.2 ppwa.h

```
#define chram (0xffffff00)
#define ch6par0 (chram + 60 + 0x0)
#define ch6par1 (chram + 60 + 0x2)
#define ch6par4 (chram + 60 + 0x8)
#define ch6par5 (chram + 60 + 0xa)
#define ch7par5 (chram + 70 + 0xa)
#define ch10par0 (chram + 0xa0)
#define ch10par1 (chram + 0xa2)
#define ch10par4 (chram + 0xa8)
#define ch10par5 (chram + 0xaa)

extern int NewDataX;
extern int NewDataY;
extern int accinit();
extern int accrelease();
extern int accreadX();
extern int accreadY();
```

A.3 pattern.h

```
#define TAKE_IMAGE 1
#define NOT_TAKE_IMAGE -1
#define WALK 1
#define KICK 2
#define BOWL 3
#define RIGHTTURN 3
#define LEFTTURN 4
#define CENTERX 80
#define CENTERY 60

void smoothControl(double time, int start, int end, int phase);
void smoothControlBowl(double time, int start, int end, int pattern);
void Init_pattern();
void Free_pattern();
int getSpeed();
void bowlPattern();
void walkPattern();
void kickPattern();
void initBowl();
void initWalk();
void initKick();
void walkToKick();
void kickToWalk();
void walkToBowl();
void bowlToWalk();
int walkCycleBowl(int pattern, int start, int end, int speed);
int walkCycle(int pattern, int start, int end, int speed, OBSTACLE *obstacle, int takeImage);
```

A.4 legservo.h

```
#ifndef CAMERASERVO_H
#define CAMERASERVO_H

#define LEFT_SERVO SERVO12
#define LOWERLEFT_SERVO SERVO13
#define ANKLELEFT_SERVO SERVO9
#define RIGHT_SERVO SERVO2
#define LOWERRIGHT_SERVO SERVO4
#define ANKLERIGHT_SERVO SERVO3

void initServo();
void FreeServo();
void TurnTo(int value, int pos);
void TurnToAngle(int angle, int pos);
void setMinAngle(int angle, int pos);
void setMaxAngle(int angle, int pos);
void setDefault();
#endif
```

A.5 Gyro.h

```
#define INPUT_GYRO SERVO8
#define INTERFRAME_DELAY 1
#define EAR_DEAF -1
#define EAR_OK  0
#define UPPER_MID 16
#define MID_ROW  64
#define MID_COL  32
#define MAXRow 127
#define MAX_SAMPLES 120
#define MAX_ANGLE 30

typedef struct eyetim {
            int Hour;
            int Minute;
            int Second;
            int Ticks;
       double dt;
} EyebotTime_type;


void initGyro();
void releaseGyro();
int GetRawReadings(int * piAccX, int * piAccY);
int TransmitLine(char * sz);
int TransmitSample(int iAccX, int iAccY);
int getDegree(int * rgiAccX, int * rgiAccY, int iSampleNumber, EyebotTime_type * rgTimes);
```

A.6 fastcam.h

```
#ifndef _fastcam_h
#define _fastcam_h_

#include "eyebot.h"
#include "colour.h"

#define IMAGE_COLUMNS 160
#define IMAGE_ROWS    120

#define RAW_FRAME       1
#define DEMOSAICED_FRAME 0

typedef struct {
  int flags;
  int width, height;
  int bpp;
  int bpl;
  unsigned char *fptr;
  int next1x0y0;
  int next2x0y0;
  int next3x0y0;
} frame;

#define PIXEL_INVALID   0x00111213

enum fastcam_ret {
  okay = 0,
  ERR_fp_invalid,
```

```
  ERR_bpp_invalid,
  ERR_width_min,
  ERR_width_max,
  ERR_height_min,
  ERR_height_max,
  ERR_width_odd,
  ERR_height_odd,
  ERR_src_width_max,
  ERR_src_height_max,
  ERR_start_row_min,
  ERR_start_col_min,
  ERR_fastMode_unknown
};

int FCAMGetPlanes(image planes[4]);

enum fastMode {
  fps5,
  fps15_raw,
  fps30_raw,
  fps15_demosaiced,
  fps30_demosaiced
};

enum fastcam_ret FCAMGetColFrame(frame *fp, int colourAdjust,
                                 int start_row, int start_col,
                                 enum fastMode fastMode
                                 );

int FCAMInit(int mode, enum fastMode fastMode);
void FCAMClose();
void setRGB(frame *fp, unsigned char x, unsigned char y, unsigned char r, unsigned char g, unsigned
char b);
int demosaicPixel(unsigned int * pixel, frame * img, int x, int y);
int get_rgb(frame *image, unsigned char x, unsigned char y, COLOUR *col);
#endif
```

A.7 colour.h

```
#ifndef _COLOUR_H_
#define _COLOUR_H_

typedef struct {
  unsigned char r, g, b;
} COLOUR;

#endif
```

A.8 bezier.h

```
int bezierCurve( double t) ;
void setControlPt( int ang, int x1, int y1, int x2, int y2, double period);
void setScale( double value );
```

A.9 CAD drawing of Tao-Pie-Pie