Development of an Autonomous Humanoid Robot Team

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Hamid Mobalegh

Berlin
Oktober 2011

Development of an Autonomous Humanoid Robot Team

Dissertation

Autor Hamid Mobalegh

Verlegt im Oktober 2011

Disputation am 16. Dezember 2011

Erstgutachter Prof. Dr. Raul Rojas

Zweitgutachter Prof. Dr. Felix von Hundelshausen

Acknowledgements

I thank my parents for my education and their strong support in my life. Thanks to my wife Zahra Vedaei for her motivation and all her support during our immigration to Germany and the time I worked hard on my project. My special thanks to Prof. Dr. Raul Rojas, for giving me the opportunity to research in his work group, strongly supporting my work and my family and advising me trough out my PhD project. Then I want to thank all my RoboCup team members for their hard work and their support of the project, especially Greta Hohl, without whose administrative support it was not possible to have success in RoboCup competitions. I also like to thank Daniel Seifert for his valuable work in porting the code from our 2008 platform to the 2009 robot and Naja von Schmude for her contribution in the implementation of my GVG idea. Finally I thank Leko Murphy and Daniel Seifert for correcting my English.

Hamid Mobalegh
Freie Universität Berlin
October 2011

Abstract

In this thesis I describe the design and development of a fully autonomous humanoid team to participate in the RoboCup, the world robot soccer championship. The team has had several successful participations in RoboCup competitions, and had won many awards including 3rd place at RoboCup2007, and twice second place at RoboCup2009 and 2010.

After a short introduction, I begin the thesis with the description of the robotic platform, where I explain my contributions in the mechanical, electrical and software design of the robots. The next part of the thesis is concerned with the stabilization methods for bipedal locomotion. I first developed a simulation platform. The control algorithms are designed based on this platform and then completed and fine tuned on the real robot. My methodology facilitates rapid and robust omnidirectional walking with a velocity of over 40 cm/s for a humanoid robot of 60 cm overall height. The method is much simpler than the current state-of-the-art methods and is capable of compensating large perturbations. The approach described here does not necessarily use accelerometers and relies on position feedback from the motors and ground contact of the feet.

Afterwards, I describe several computer vision solutions I developed for the robot. The development of a color-based object recognition module is presented first. The module uses on a small low-cost CMOS camera and a low power microcontroller and provides microcontroller compatible output, in form of serial access to the list of recognized objects.

Finally, I propose two new methods for shape-based object recognition. The first method uses a grid of cells and clusters the edge points based on their orientations and reports a connection graph of the edge structure in the image. The second algorithm uses the statistics of the edge orientations in the image to find a round object using a recursive method.

The ideas and methods presented in this thesis were implemented in the RoboCup humanoid team of the Free University of Berlin, the FUmanoids.

Contents

Part I

Introduction

Chapter 1

Introduction

This thesis is about the design and development of an autonomous humanoid robot team. The team has several successful participations in local and world RoboCup competitions, and has won the third place in RoboCup 2007 and the second place in 2009 and 2010.

Humanoid robots have been an attractive research area for the last two decades. A main reason for this is that these robots are theoretically capable of performing similar tasks and acting in similar environments as the human. Even more, there are also tasks which are too complex to be performed by simple robots and too hazardous for human to undergo. Therefore humanoid robots provide a generic platform for researching and developing technologies on a wide range of areas. Some examples are bipedal walking, stereo vision, self localization and human-robot interaction. Despite the long research on development of humanoid robots there is still a huge lack of functionality and performance, compared to an actual human.

At the time this thesis is being written, the greatest challenge is the stabilization of bipedal walking. This is a key problem in development of humanoid robots. There are several examples of humanoid robots which can walk acceptably stable but require one order of magnitude more power compared to human walking. On the other hand, the new studies in passive dynamic walking introduce a class of bipedal walkers with the energy efficiency of a human. Passive dynamic walkers are however pretty far from being robust[1, 2] and suffer from lack of controllability.

I therefore focus on the problem of bipedal walking, and suggest solutions for a simple closed loop stabilization of the motion system. The proposed solutions are simple and platform independent, so that they can be applied almost to every humanoid robot.

The second issue I address in the thesis is computer vision and object recognition. Due to

RoboCup Soccer	
	Simulation League
	Small-Size League
	Middle-Size League
	Standard Platform League
	Humanoid League
RoboCup Rescue	
	Rescue Simulation League
	Rescue Robot League
RoboCup Junior	
	RoboCup Junior Soccer
	RoboCup Junior Rescue
	RoboCup Junior Dance
RoboCup@Home	

Table 1.1: RoboCup Leagues

the specific guidelines in design of humanoid robots for RoboCup competitions, there are significant restrictions in the weight and available on-board processing power of them. Together with the special color/shape coding of the objects in RoboCup scenarios, implementation of new techniques becomes unavoidable for the computer vision. I follow two main approaches in computer vision. There I present my achievement in developing a small computer vision module with the capability of color-based object detection, and then present my suggested algorithms to reduce the role of colors in computer vision for RoboCup scenarios.

1.1 Motivation

The idea of soccer playing robots was proposed to the artificial intelligence community by Alan Mackworth [3]. The idea was to develop robots, which could perceive and interact with the ball, avoid obstacles, communicate and cooperate in real-time. After five years of discussion, the RoboCup initiative was founded in 1997. The first RoboCup leagues included "Small Size" with a field size of a ping-pong table and "Middle Size" with 3 times the area. Worldwide RoboCup competitions have been held annually since then, adding more and more leagues and advancing the existing ones. The latest RoboCup was held with close to 4,000 competition participants in over 500 teams. The competitions are distributed in 15 major leagues in 4 categories. Table 1.1 shows the main RoboCup leagues.

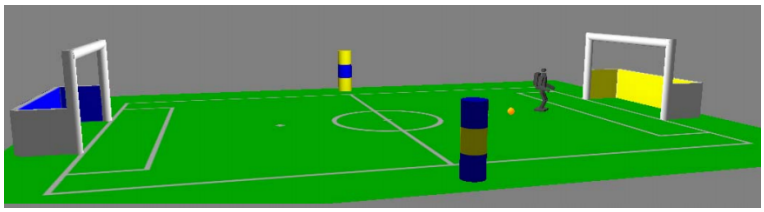


Figure 1.1: Field of RoboCup Humanoid Teen Size League[4]

1.2 RoboCup Humanoid League

RoboCup Humanoid is one of the leagues of *RoboCup Soccer*. It was originaly divided into two sub-leagues based on the size of the robots. Robots of the overall height below 60 centimeters compete in the *Kid-Size* class. Others were categorized as *Teen-Size* robots. In 2009 a new *Adult-Size* class was introduced. A set of rules exists for each league, which defines constraints for robot design as well as how the games are played and the competition is organized [4]. Each RoboCup league has a technical committee which is responsible for updating the rules. The members are partly elected by the participating team leaders and partly set by RoboCup organization. Figure 1.1 shows the field of the teen size league with an example robot.

1.3 The FUmanoid Robots

The FUmanoid project belongs to the Artificial Intelligence group at Free University of Berlin. The research group has had a successful and long history in RoboCup with its FU-Fighters team. FU-Fighters won World Champion 2004, 2005, European Champion 2000 and German Open 2005, 2004, 2003, 2002. The team stopped its RoboCup activities in 2006 and started developing autonomous vehicles.

This is the time when FUmanoids were born. Feasibility studies were conducted before and during RoboCup 2006. As the required knowledge level in software and hardware development of multi-agent robotic platforms was already achieved with the wheeled FU-Fighters, humanoid robots could offer an advanced research area with many attractive unsolved problems. The development began with simulations, then moved on to a commercially available robot kit and finally continued with self -designed and constructed robots.

The team showed an excellent performance in its first year of activity by winning the 3rd place of the world RoboCup in the Kid Size humanoid league, presenting the lightest and the least expensive football playing robots in their class. Having improved the software and hardware and having intro-

duced new solutions for the existing problems, the team improved its rank by achieving the first place in RoboCup Iran Open 2008 and second place in RoboCup German Open 2008. In 2009 and 2010 the team became the vice champion in RoboCup Humanoid league. This was achieved by advancing several solutions in the areas of hardware and software, which I will present in my thesis.

1.4 Related Work

The thesis covers a wide area of research including several disciplines. The state of the art is therefore surveyed separately in each chapter.

1.5 Organization of the Thesis

The thesis is organized as follows. The next part describes the platform developed for the research. Chapters 2, 3 and 4 explain my developments in three major directions: mechanics, electronics and software. It includes all prototypes along with final versions and revisions made to them. Details of the design are presented using images, schematics and block diagrams.

Part III focuses on the problem of bipedal walking, its generation and stabilization. I present a simulation-based approach to stabilization of dynamic walking in chapter 5. This provides a basic tool for analysis of the bipedal walking as well as development of control methods. It is followed by chapter 6, in which I introduce an interface between methods developed using simulation and the real robot by conversion of the parameter space. Chapters 7 and 8 introduce different proposed methods for stabilization of the biped based on the results provided from the previous chapters.

Part IV of the thesis addresses the problem of computer vision for the developed humanoid platform. It is organized in three chapters. After an introduction to the problem of computer vision for RoboCup in chapter IV I present two different approaches in the following two sections: embedded color-based object detection is explained in chapter 10. Here my achieved results from a developed hard- and software are presented. Chapter 11 includes some suggested solutions for shape-based object detection.

Part II

Description of the Platform

In this part I describe the design of the robotics platforms used in FUmanoids team. It includes several versions of humanoid robots ranging from initial prototype bipeds used for testing and improving bipedal walking algorithms to fully functional versions designed and developed to participate in the RoboCup humanoid league.

The next chapter explains the hardware design of the platform in two main parts. Mechanical design of different versions is presented first. It is then followed by the description of the electronics. I discuss several design ideas and solutions in this context. Software solutions developed for the platform is described in the following chapter. Complete software structure of the system, together with some useful external tools for the development are described in the chapter.

Chapter 2

Mechanical Design

2.1 Introduction

In the last decade, many studies have been focused on development of humanoid robots. Honda R&D's humanoid robots[5], WABIAN series of Waseda University[6, 7], ASIMO[8], Partner, QRIO, H6 & H7[9], HRP-4C[10] and JOHNNIE[11] are well known examples.

It is theoretically possible to make a biped robot walk on an even surface with a very small number of actuators. It has been shown in a large group of publications that a biped model can walk without any joint actuation if it receives the energy of walking in some way[12, 13, 14, 15, 1, 16, 17]. This is known as passive dynamic walking [1, 16]. The research was pioneered by Tad McGeer, who built the first physical passive dynamic walking machine. Passive dynamic walking is known to be energy sufficient and have a human-like walking gait. On the other hand, a large number of actuated degrees of freedom is needed for a general purpose humanoid robot. There is a group of motions other than walking, which cannot achieve success with under-actuation. A good example of this is standing up.

Typically, 12 degrees of freedom are required in the lower limb of a humanoid robot: 2 (pitch and roll) in the ankle joint, 1 (pitch) in the knee joint and 3(pitch, roll and yaw) in the hip joint per leg. The upper trunk usually has a less important impact in the robot's locomotion, however arms can be used to compensate body yaw due to leg motion reaction [13] and a degree of freedom in the torso can be used for frontal plane stabilization as it is also the case in human walking . The arms can also be used to stabilize lateral or frontal plane motion as described in [12].

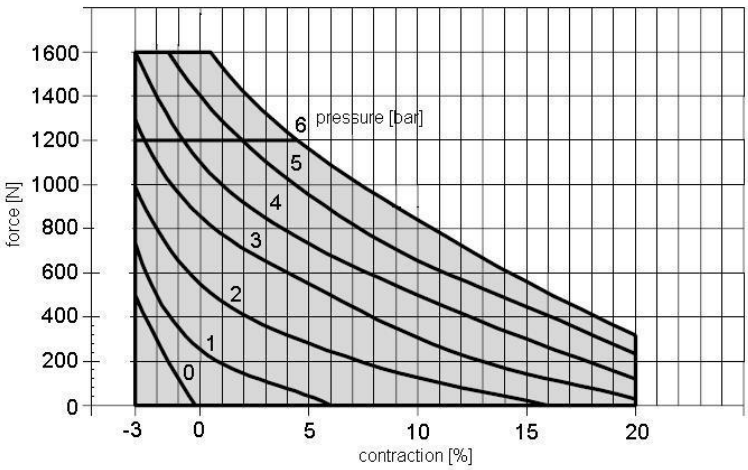


Figure 2.1: An Example Behavior Curve of a PAM from Festo.

2.2 Actuation

One of the very first considerations which should be taken in the mechanical design of a humanoid robot is which type of actuation should be used. A variety of actuators has become commercially available during the last decade. This section gives a brief introduction to several existing actuation methods, and continues with the discussion of which actuators are suitable for a humanoid robot. An overview of different actuation methods is also given in [18].

2.2.1 Pneumatics Cylinders and Air Muscles

Pneumatic actuators, usually cylinders, are widely used in industry for automation. Robotics also takes advantage of pneumatic actuation more recently. Pneumatics has become famous because of the low weight and the compliance of the actuators due to the compressibility of air. The degree of compliance is a direct function of the pressure. This is an important feature in different areas, such as the handling of fragile objects. Several types of pneumatic actuators are available today. Cylinders and pneumatic engines are common examples. A more interesting type specially for actuation of humanoid robots are the so-called *"Pneumatic Artificial Muscles"*[1]. PAMs contract as a result of inflation due to the air pressure applied to them. A non-linear spring-like behavior is shown. Properties of the force-length curve are adjustable with the pressure applied to the muscle[19]. An example behavior curve is presented in figure 2.1.

PAMs are extremely lightweight and they can transfer the same amount of energy as cylinders do.

[1] PAM

There are no sliding parts in PAMs. This makes them suitable for applications in robotics. However the application of PAMs and generally all pneumatic actuators in mobile robotics is seriously limited due to the drawback of having to carry enough compressed air for a reasonable time of operation.

There are several examples of complete or partial humanoid robots built based on PAM actuation. Björn Verrelst introduces in his Ph.D. thesis an improvement to existing PAMs, called *"Pleated Pneumatic Artificial Muscles"*[2][20, 21]. Further in the thesis he describes the design and construction of the planar bipedal walking robot "Lucy". Hosoda et al. [22] describe the design of a biped robot driven by antagonistic pairs of artificial pneumatic muscles with variable joint compliance. Simple controllers for realizing walking, jumping and running are also presented. [23]

2.2.2 Hydraulics

Hydraulic drive has in contrast to pneumatic drive a very rigid behavior and can only be made to act in a compliant manner through the use of relatively complex feedback control strategies [19]. In addition, use of hydraulic actuation is only possible with an external source of hydraulic fluid as well as a compressor to supply the required pressure. This makes the solution improper for self-contained mobile robots. If hydraulic fluid can be supplied externally, hydraulic actuators can be used for humanoid robots. [24, 25, 26] give examples of such platforms.

2.2.3 Electro-active Polymers[3]

The Polymer Gel Actuator is one of the candidates of artificial muscle actuators due to its compliance and compactness. Electro-active polymers, which respond to electric stimuli with shape change, were known for a few decades, however their actuating limitations prevented their use in serious robotic applications. Development of EAP materials in the last decade facilitates their application in many areas, e.g. robotics, medical service and toy industry.

Among other EAP materials, the Ionic Polymer-Metal Composite[4] actuator is one of the most promising actuators for applications. IPMC bends in response to electrical activation[27]. The actuator is produced by chemically plating gold or platinum on a perfluorosulfonic acid membrane which is known as an ion exchange membrane. The actuator bends rapidly when an input voltage is applied to metal layers of both sides. The phenomenon of this motion was discovered by Oguro et al. in 1992 [28].

[2] PPAM
[3] **EAP**
[4] IPMC

Applications of IPMCs are not limited to robotics. The new actuation method continues to be more recognized and used for different purposes in recent years. [29] describes the development of a fish-like, S-shaped swimming micro-robot using IPMC actuator. The swimming speed of the micro-robot can be controlled by changing the frequency of input voltage. There are, however, still too few humanoid platforms using this technology for the actuation. A miniature bipedal robot is manufactured using IPMC actuation, and a preliminary experiment is conducted in [30].

2.2.4 Shape Memory Alloys

The shape memory effect[5] is a unique property of certain alloys. These materials can recover their original shape by reaching a critical temperature. The history of shape memory alloys goes back to the 1930s. Today, several alloys with such a property are known[31]. The most commercialized material, nickel-titanium, was first developed in 1962–1963 by the United States Naval Ordnance Laboratory and commercialized under the trade name Nitinol[32]. SMAs have also other unique properties which are outside the scope of this section.

So far, there have been limited studies on using SMAs in robotic applications. This might be, among other reasons, because of several shortcomings of SMA actuators such as the lack of energy efficiency, slow response time, temperature dependence, and large hysteresis. Some implementations are published in [33, 34, 35, 36]. [35] has used SMAs for actuation of a micro biped robot. More recently, a new approach has been introduced by Kratz et. al. [37] which facilitates the use of SMAs in larger bipedal robots. He also proposes a control method for the actuator [38].

2.2.5 Electric Motors

A majority of today's humanoid robots use electric motors as their actuation method. Asynchronous AC motors and brush-less DC motors[6] are very reliable and have significantly long life times[39]. However due to the complex and expensive controllers needed for these type of motors, brushed DC motors have still the best reputation for low-cost humanoid platforms. DC motors are known to be simple to drive, have a linear behavior and are well understood. The mechanical properties of the load is linearly reflected on the electrical side of a DC motor. This phenomenon can be used to control the compliance of the actuation as well as to provide a controlled torque output.

[5] SME
[6] BLDC

2.2.6 Conclusion

Servomotors are the most used actuators in biped robot platforms. They have several advantages compared to the other groups of actuators, which follows:

Modularity and Compactness Servomotors are usually packed in small cases containing all needed components including the motor, gear box, drive and control electronics. They provide a well defined interface of both electrical and mechanical sides. It is therefore easy to replace a whole module in case of a malfunction or even replace a module with a compatible improved version.

Robustness Servos are very robust as long as they work inside the designed range. All parts of the system are integrated in a small area separated from outside. This reduces the risk factor further by avoiding dust as well as other mechanical and electrical disturbances.

Integrated drive and control Users don't have to struggle with drive and control issues. It is often enough to send commands including goal position, speed, etc. to drive the motor.

Linearity As also discussed above, the linearity of the DC motor provides many features such as torque feedback and control, adjustable compliance and so on.

In contrast, servomotors suffer from following disadvantages:

Maintenance of the Motor Most of the available servomotors are based on brushed DC motors. These motors have a limited life time and need to be replaced or repaired after a while.

Power to weight ratio Even though some new servomotors are lighter than their competitors in other classes, it is still a noticeable difference between servomotors and actuators such as PAMs. More recently there are some new technologies which lead to higher power to weight ratios. Some available, but still too expensive technologies include Harmonic Drive and direct drive BLDCs.

Reliability of the feedback Almost all low cost servos and many of the medium expensive ones on the market use potentiometer feedback. Potentiometers are easy to use, and can be calibrated to achieve acceptably high precision and linearity but they all suffer from loose contact in long term usage. This can lead to strong oscillations and cause permanent damage to the mechanical parts in feedback control. It is often observed that the problem can even be transmitted to other servos mechanically connected to the system.

Parameter	Value	
Weight (g)	55	
Dimension (mm)	32 x 50 x 38	
Gear Reduction Ratio	1/254	
Applied Voltage (V)	at 7V	at 10V
Final Reduction Stopping Torque (kgf.cm)	12	16.5
Speed (Sec/60 degrees)	0.26	0.19

Table 2.1: AX-12 Properties[40]

Parameter	Value	
Weight (g)	72	
Dimension (mm)	35.6 x 50.6 x 35.5	
Gear Reduction Ratio	1/193	
Applied Voltage (V)	at 12V	at 16V
Final Reduction Stopping Torque (kgf.cm)	28.3	37.7
Speed (Sec/60 degrees)	0.167	0.126

Table 2.2: RX-28 Properties[40]

Energy dissipation Principally, DC motors are designed for and therefore are most efficient in a continues rotation at the nominal power range. This is however not the case for a servomotor the majority of the time. Servomotors are either used in a torque holding semi-static mode or with a cyclic changing speed. In both modes DC motors have a weak power efficiency.

2.2.7 Selection of the Servomotors

Most commercially available servomotors are pulse driven. This means the desired position of the motor is determined by the width of a rectangular pulse applied to the unit. These products have two important disadvantages. First, each motor needs a separate pulse, daisy chaining is not possible with this type of servomotors, which in turn makes the cabling of the robot very difficult and reduces the reliability to a great extent. Second, there is no position/speed/load feedback available from the output shaft. The latter point is very important, as my approach to stabilize walking gate is dependent on the position feedback received from the joint servomotors.

In the last few years a new generation of servomotors has become commercially available, which can be easily daisy chained via a bus system. These actuators provide a digital interface which facilitates accessing several parameters for each unit connected to the bus. FUmanoid robots are designed based on the product series *Dynamixel* from *ROBOTIS Inc.* Hardware features of the servos are described in tables 2.1, 2.2 and 2.3. The communication protocol of the servomotors is described in section 3.2.4.

Parameter	Value	
Weight (g)	125	
Dimension (mm)	40.2 x 61.1 x 41.0	
Gear Reduction Ratio	1/200	
Applied Voltage (V)	at 15V	at 18V
Final Reduction Stopping Torque (kgf.cm)	64.4	77.2
Speed (Sec/60 degrees)	0.188	0.157

Table 2.3: RX-64 properties[40]

2.3 Mechanical Construction

In this section mechanical design of the FUmanoid robots is explained. This includes several prototypes and final versions used for the competitions.

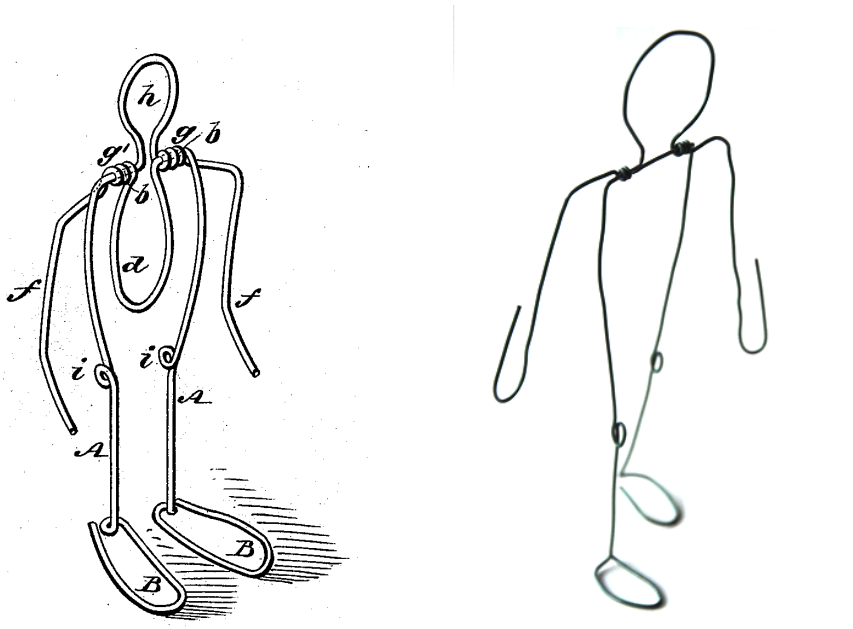
2.3.1 Prototypes and Test Platforms

Before I constructed the first functional robot, I verified several design ideas using prototype platforms. Most of the prototypes contained only the lower limb which was enough for testing the walking algorithms. The following prototypes have been constructed:

Fallis' Wired Biped One of the oldest known passive dynamic walkers is reported in the U.S. patent from Fallis [41]. His model is made from two pieces of wire hinged on each other in two points on the shoulders. Figure 2.2 shows the model. As the first step to get familiar with the problem of bipedal walking, I reproduced and tested this model. Despite its simplicity, the model could successfully make several steps on a gentle slope when it was appropriately launched.

Semi Passive Walker with Step Synchronization Based on the results reported in [17], I built a partially actuated model to study the effect of hip actuation, as well as step synchronization on the model. The model is presented in figure 2.3. The biped had a simple DC motor with a gear box connected to the hip joint. Two micro switches were installed on the feet to measure the foot to ground contact. These were then directly wired to drive the motor in opposite directions regarding to the stance foot. Maximum step length was determined using a mechanical limiter connected to the legs.

Telescopic Leg Biped I designed the prototype to verify the idea of walk stabilization by step length control using ankle joint feedback. The model was improved in several areas compared to its predecessor. The uncontrolled hip actuator was replaced with a servo motor. To achieve foot clearance



(a) The model is made from two pieces of wire, hinged on the shoulders. It works on a similar basis as today's passive dynamic walking models. Fallis has also implemented the counter-swinging arms.

(b) The reproduction of the model which surprisingly works well and can make several successful steps upon a good initial launch.

Figure 2.2: Fallis' Walking Model [41].

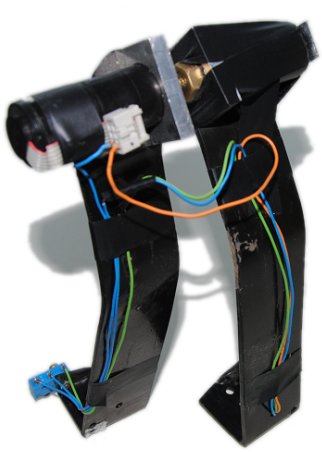
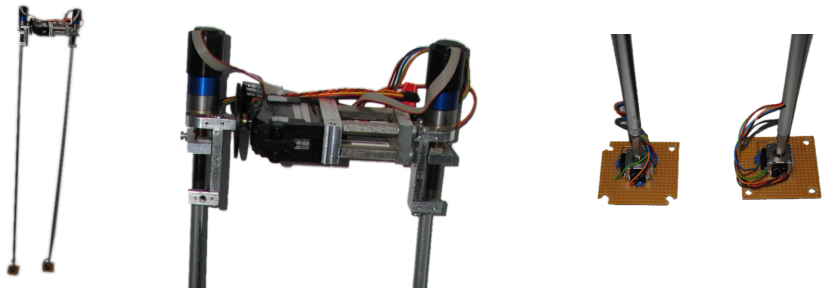


Figure 2.3: Semi Passive Walker with Step Synchronization



(a) Complete Model (b) Hip and Leg Actuation (c) Ankle Joint Sensor

Figure 2.4: Telescopic Leg Biped

and also to energize the model, each leg was coupled to a linear motion mechanism. A two dimensional potentiometer was built in each ankle joint to measure the pitch and roll angles of the foot. Ground contact could be measured using a micro switch integrated in the ankle joint sensor. The prototype is presented in figure 2.4.

Bioloid Leg Extension and Ankle Joint Feedback As it was often required to change the mechanical construction of the prototypes for testing different stabilization ideas, I decided to continue the experiments on a more flexible platform. I selected *ROBOTIS Bioloid Advanced kit* for this purpose. Bioloid provides a highly modular platform using the digital AX-12 servomotors, compatible intermediate parts and connections and a central processing unit. The platform was first used to implement the leg length control method. The requirement of non actuated, frictionless ankle joints was met by removing a gear from ankle joint servomotors, letting them function only as sensors.

2.3.2 2007 Platform

Robots built for the 2007 competitions are based on the robotics educational kit *Bioloid* from *ROBOTIS Inc.* Bioloid is a 18 DOF humanoid platform with an overall height of 34cm. The model is entirely actuated using Dynamixel AX-12 servomotors.

Figure 2.5 shows the robotic platform of 2007. I applied the following changes to the original kit to make it suitable for RoboCup. Two joints servomotors were removed from the elbows. These were then built in a neck pan/tilt mechanism for the head of the robot. The battery pack in the original kit contained NiMh cells. It was too heavy and hard to replace. It was therefore replaced with a Li-Poly pack. To reduce the height of the COM of the robot, the battery pack was placed between the hip servos by increasing the distance between them. As the original product did not contain any

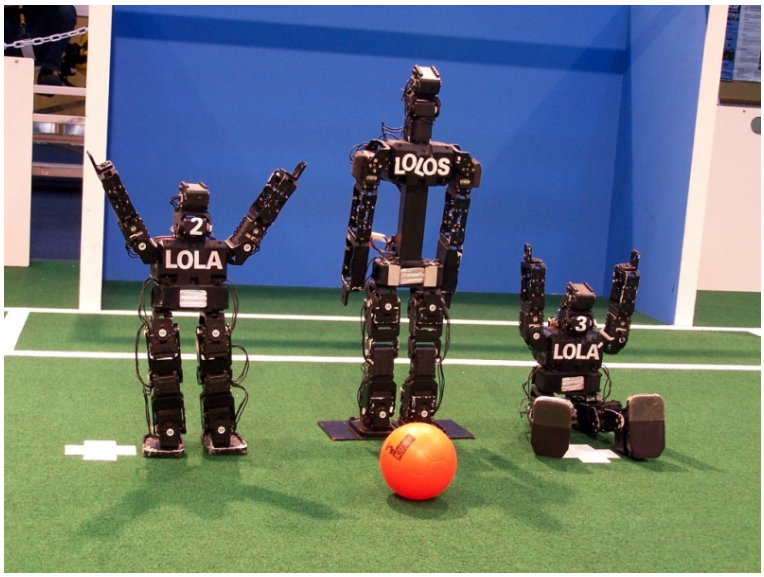


Figure 2.5: Robotic platform used for 2007 competitions

	LOLA	LOLOS
Overal height	37 cm	57cm
Height of the COM	21cm	27cm
DoF	18	18
Each Leg DOF	6	6
Head and neck DOF	2	2
Each Arm DOF	2	2

Table 2.4: Physical Properties of the 2007 Robots

solutions for computer vision, a vision module was developed and mounted on the head of the robot. Details of the vision module will be described in chapter 10. The module was mounted into the same casing as the servo motors to facilitate modularity and compatibility.

The overall height of the robot has reached 37 cm including pan/tilt mechanism and the camera. However for the goalkeeper to take advantage of the maximum allowed height to block the goal, an intermediate part was added between the hip servos and the upper limb. It is visible in figure 2.5. Table 2.4 presents the physical properties of the robots.

One of the shortcomings of the 2007 version was the low height of the robot, which in turn lead to lower placement of the camera. Since many other RoboCup participants build their robots with the maximum allowed height, they usually obstruct many objects from the smaller robots. On the other hand, the walking speed is, among other parameters, also proportional to the leg length, which is usually around half of the overall height.

Figure 2.6: Robotic platform built for 2008 competitions

Overall height	59cm
Height of the COM	34cm
DoF	18
Each Leg	6
Head and neck	2
Each Arm	2

Table 2.5: Physical Properties of the 2008 Robots

Another problem observed in the 2007 version was the insufficient torque of the servomotors in certain motions. This would worsen as the robot grew. AX-12 not only has a high friction in the gearbox, it also goes into a break mode when it receives a torque disable command. The driver shorts the motor terminals internally. As it will be discussed later in 7.4.3, energy is a very important concept in passive dynamic walking. So any loss of energy concerning any type of damping as well as any impact, should be taken into consideration. AX-12 was therefore not suited for some control rules.

2.3.3 2008 Platform

Figure 2.6 presents the robotic platform of 2008. I used a mixture of 12 RX-28s in the lower limb and 6 AX-12s in the upper limb in this design. RX-28 has much less damping compared to AX-12, especially when the torque is turned off. It produces more than two times the torque of the AX-12. The required intermediate parts and connections were designed in CAD and milled out of PVC sheets using a 3 axis CNC machine. Physical properties of the robot are presented in table 2.5.

Property	Value
Height (cm)	60
Weight (kg)	4.4
DoF	21-22
Each Leg DOF	7
Each Arm DOF	3
Neck DOF	1-2

Table 2.6: Physical Properties of the 2009 Robots

2.3.4 2009 Platform

After participation in 2008 competitions it was noticeable that the knee motors suffered from low speed for walking motions and also inadequate torque in stand-up motions. Some other joints such as ankle joints had also not enough torque for walking purpose. It was therefore decided to upgrade the servo motors of the robot. In the 2009 version, RX-64 servos replaced the RX-28 model in the legs, and AX-12 servos were replaced by RX-28 model in the upper limb. As presented in table 2.3, the RX-64 has more than twice the torque of the RX-28 but it is a bit slower.

It is very important that all the motors making a specific movement remain synchronous as they move to their goal positions. The knee motors are especially subject to lose their synchronization. As it will be discussed further in chapter 6, servos built in the knees move with two times the speed of the ankle/hip servos to lift the feet. In faster walking motions or under more load these motors reach their maximum speed and cannot follow the planned trajectory. The situation becomes further worsened using the RX-64, as it is slower than the RX-28.

A solution to this problem is to connect two motors in series for the knees. This doubles the speed, makes all motors move with the same speed and remain synchronized with each other. This is unfortunately at the expense of more weight. In addition, this method increases the power dissipation, as the load is not shared between series motors, but each of them receives the whole load.

The mechanical construction of the 2009 robot is described in the diploma thesis of Mariusz Kukulski [42]. The 2009 version had 21 degrees of freedom, 7 in each leg and 3 in each arm, and depending on the used lens for the camera, one or two in the neck. The elbow joint was added to make the robot able to pick the ball used for the first time for full automatic throw-in behavior. Table 2.6 shows the physical properties of the 2009 robot. The battery pack was placed in the trunk and consisted of 4 Li-Poly cells. The CPU, together with the camera board explained later in 3.2.3, was placed in the head of the robot.

Chapter 3

Electronic Design

3.1 Introduction

Several important issues should be considered in designing electronics for a humanoid robot. These are weight, power consumption and last but not least reliability and ease of maintenance. Based on these criteria the electrical design of the FUmanoid robots is described in the rest of this section.

3.2 Central Processor

According to the weight and consum criteria, ordinary processing boards cannot be used in the robots. These are usually equipped with heavy heat sinks or active cooling systems which reduce the reliability of the system. Additionally, a continuous need of around 10 Watts is far away from the expectations. In this section, two different approaches are followed which are implemented on two different generations of the FUmanoid robots. The first approach tries to reduce the need of CPU power by adding a vision module. A small microcontroller is shown to be enough for the rest of the tasks. The second approach uses a light-weight ARM processor module with a proportionally small power consumption.

3.2.1 On-Board Computer in 2007 and 2008 Versions

Most of the processing power of an on-board CPU of a humanoid robot is used for image processing. This means, if image processing could be done externally in some way, there is no more need

Architecture	8 bit RISC
Throughput	~16 MIPS @ 16 MHz
Flash	128 KB
RAM	4 KB
EEROM	4 KB
Peripherals	2x USART 4x Timer 8 channel ADC ...

Table 3.1: Features of ATMEGA 128

to have high processing capabilities. This is the idea followed in the versions of 2007 and 2008. In both versions, computer vision is done using an external module described in chapter 10. The original robotics kit contains a controller box called *CM5*. It is based on the Atmel ATMEGA 128 micro controller, which is an 8 bit RISC micro controller clocked at 16 MHz. A throughput of maximum 16 MIPS can be achieved. Table 3.1 shows a summary of the features of the CPU.

A simplified schematic of the CM5 is presented in figure 3.1. The control board includes the physical layer of the half duplex TTL communication protocol together with a power management module to work with a 9.6V NiMh battery pack. There are 5 buttons and 7 LEDs available on the board.

ATMEGA 128 is equipped with several peripherals, such as timers, UARTs, a multiplexed input ADC and an interrupt controller. The chip contains 2 UARTs, one of which is connected through a buffer to the half duplex TTL communication bus and the other is used for the cable connection to a PC. This interface is used as well to program the device and to access the camera for calibration. Recording and calibrating static motions are also done using this connection. The interface has also been used for debugging in the development phase.

The available LEDs were used to indicate the status of the system during the game and also for debugging purposes during the development. Start and stop signals were given to the robots of 2007 manually using the buttons. According to the rules of the RoboCup humanoid league, wireless LAN is the only allowed type of communication. Therefore the newer version was additionally equipped with a serial to WLAN converter. It converted the serial data transmitted from the robot into UDP packets and vice versa.

Among the other peripherals, one of the timers was used for synchronizing the vision module and processing the data received from it in the 2007 version. For the latter version, a scheduler was implemented to provide multi-threading. Following this change, vision owned an individual thread.

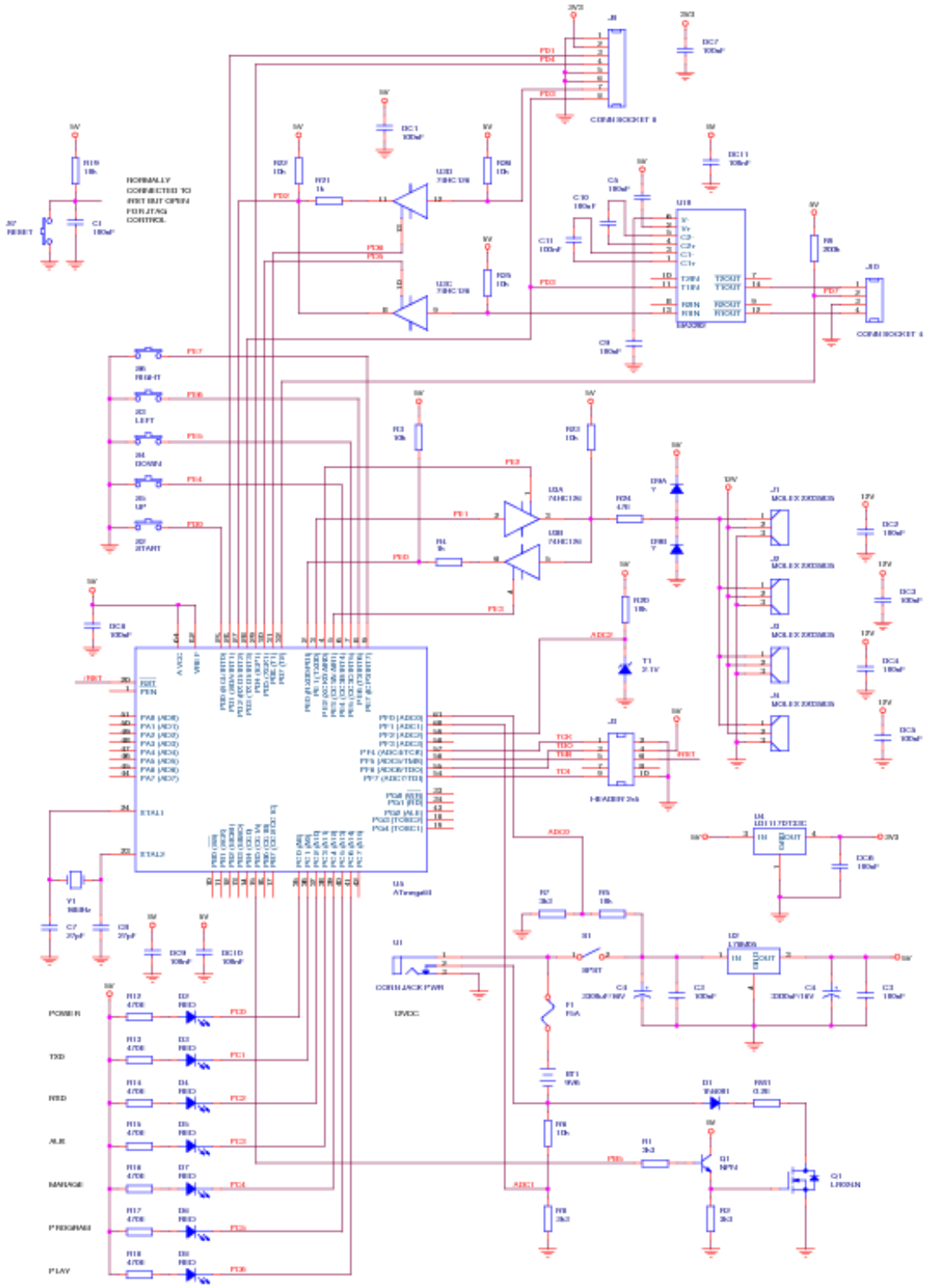


Figure 3.1: Simplified Schematics of the CM5[43]

3.2.2 On-Board Computer in 2009 Robot

Due to several reasons, the team decided to utilize a more powerful processor for 2009. The most important reason was inadequate details from the image processing module for self localization. In addition, the amount of RAM available in the ATMega128 was only 4KB which was obviously too small for some available techniques in robotics such as particle filtering. The throughput limit was also reached in some cases with this older chip.

There were plenty of alternatives available when selecting a processing unit. Some of these were PDAs, x86 embedded PCs, and ARM boards. In our point of view, a proper processor unit should be light-weight and small. It should consume low power, be easy to connect to peripherals and extension boards, specially, camera and serial devices. As the most important factor, the CPU should run an operating system which allows low level programming of real time features.

Gumstix is the name of an embedded computing product family. Gumstix products are ARM based microcontroller boards of very small dimensions, i.e. 2cm x 8cm. Available products range from 100 to 600MHz and have different features. For every CPU module there are also several available extension boards such as network extension, LCD connection, etc. The whole construction including the gumstix and its extension board sink about 200-300mA under 5v and therefore does not produce much heat. It is very advantageous as there is no need for any kind of cooling, or usually heavy heat sinks. The whole signal description of the expansion ports, together with the schematic and PCB data of all extension boards, are provided by the producer. It is therefore easy to design new extension boards.

The decided product for the 2009 robot was the "Verdex Pro XL6P". It was the fastest CPU board available at the time. For networking, the extension board "Netpro-vx" was selected together with the WLAN card mounted on it. Two extension boards were designed and built for the CPU module for connection to the cameras and motor bus which will be explained in the next two sections.

3.2.3 Camera Board

Computer vision was assigned to a stand-alone module in 2007 and 2008 versions. For 2009 the module could not meet the requirements. Therefore it was decided to assign this task to the central CPU by providing an effective connection between one or two cameras and the CPU. PXA270, the CPU on the gumstix verdex, is equipped with a peripheral called "Quick Capture Interface". This interface facilitates direct connection of one CMOS camera to the CPU using DMA to avoid any CPU load. Having the idea to use stereo vision, we had to design an extension board which was capable of multiplexing two CMOS cameras as if it would be one from the perspective of the CPU. Data bus

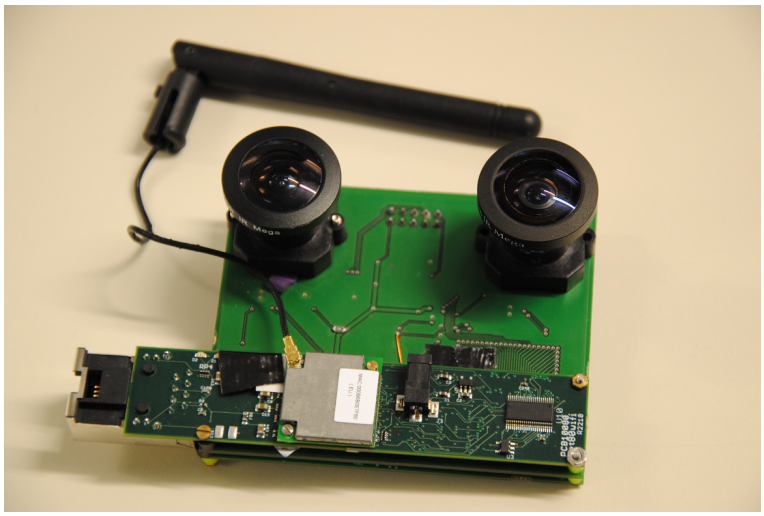


Figure 3.2: Camera Extension Board for Gumstix

multiplexing and other necessary timing was done in a CPLD block. An I^2C multiplexer connects the control signals of the both cameras to the CPU. Figure 3.2 shows the final design. Bennet Fischer[44] describes the development of the camera extension board in his diploma thesis.

3.2.4 Communication Bus

Complete documentation of the communication protocol used for the motor bus is available in [40]. A summary of the most important features are also presented in this section. The physical layer of the network consists of a half-duplex, TTL level asynchronous serial connection. Each module has a circuit similar to the one presented in figure 3.3. It is therefore important that as long as one of the modules on the bus is transmitting data, all others have to listen. Listen mode is the default mode for all modules. It is not obligatory but recommended to have a master/slave structure. Most of the time the master is the central processor. Each module is identified with a unique number ranging from 0 to 253. 254 is reserved for broadcast and 255 is used as packet header indicator. Figure 3.4 shows a typical packet sent to a servo motor and the response packet from the motor.

All servo products of ROBOTIS use the same communication protocol, however the physical layer is a little bit different. AX-12 has a 3 wire connector, 2 of which are supply lines. Data transmission is done using only one line. Other models use a differential RS485 interface. In the 2008 platform it was necessary to connect both servo models to the bus. I designed a simple converter for this purpose. It is presented in figure 3.5. The converter holds the inverted line in idle mode around 2.5v using a resistor voltage divider. So 4 wired devices connected to the bus can compare the voltage of their non

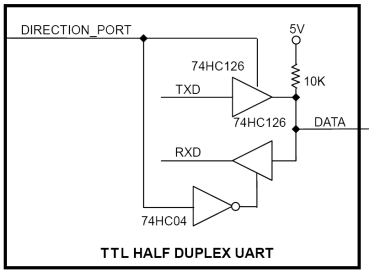


Figure 3.3: Full-Duplex to Half-Duplex Converter Built in the Modules

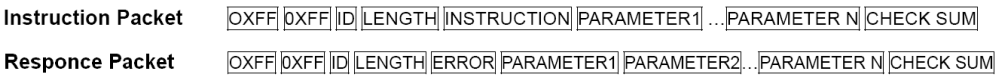


Figure 3.4: Overview of Dynamixel Communication Protocol

inverted input with the inverted one. In this case the result is the content of the non-inverted data line. A transmission from the 4 wired devices will of course work as 3 wired devices can easily ignore the inverted data line.

3.2.5 Power Management

Since the first version of the robots, Lithium Polymer batteries were selected as the energy source. The advantages of lower weight and greatly increased life times sufficiently justifies the price. However, Lithium polymer-specific chargers are required to avoid fire and explosion. Explosions can occur if the battery is short-circuited or over-charged.

Another difficulty with Li-Poly technology is ensuring that the pack is never discharged below its critical level. Otherwise it can no longer be recovered. This should also be guaranteed when the robot has been left on. It is therefore not possible to use a power management unit, which is still supplied from the battery pack in standby mode. For this purpose I designed a power management unit. It

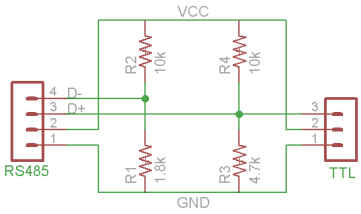


Figure 3.5: TTL to RS485 converter

consists of a self holding relay and a voltage comparator as presented in figure 3.6. In addition, a switching mode DC-DC converter was integrated in the circuit after 2008 to provide a 5 volt supply for the processor board.

3.2.6 Sensors

The robots were designed in a way that a minimum number of sensors were needed. To increase the reliability and to keep the system simple, the same communication bus of the servomotors is also used for the sensors. The following sensors were developed and used in the robots:

Vision Module In the 2007 and 2008 versions of the robots, the task of low level computer vision was assigned to an external module to reduce the load of the main processor. The hardware of the vision module contained a CMOS camera chip and an *ATMEL ATMega8* processor. A color based object recognition algorithm was developed for the module. The details of the software will be described in chapter 10.

In implementation mode, the camera module uses the same communication protocol as the servo motors. This makes it compatible with the bus system of the robot. For 2008 robots, a wide angle lens was mounted to the camera. Figure 3.7 shows different versions of the vision module.

General Purpose IO Module Other ordinary sensors can be classified into general input and outputs. These include analog, as well as digital sensors. For this group of sensors a sensor module was designed, which allowed connecting them to the communication bus. With this module 8 bits of selectable digital/analog IOs were available. This module was used in the 2009 version to connect 4 push-buttons per foot to detect ground contact. The same module was connected to a 5 DOF IMU sensor in the next versions. A Kalman filter was implemented in the microcontroller to measure the pitch and roll angles of the camera. Ground contact buttons were replaced with analog pressure sensors in this version. Figure 3.8 shows the schematics of the designed circuit.

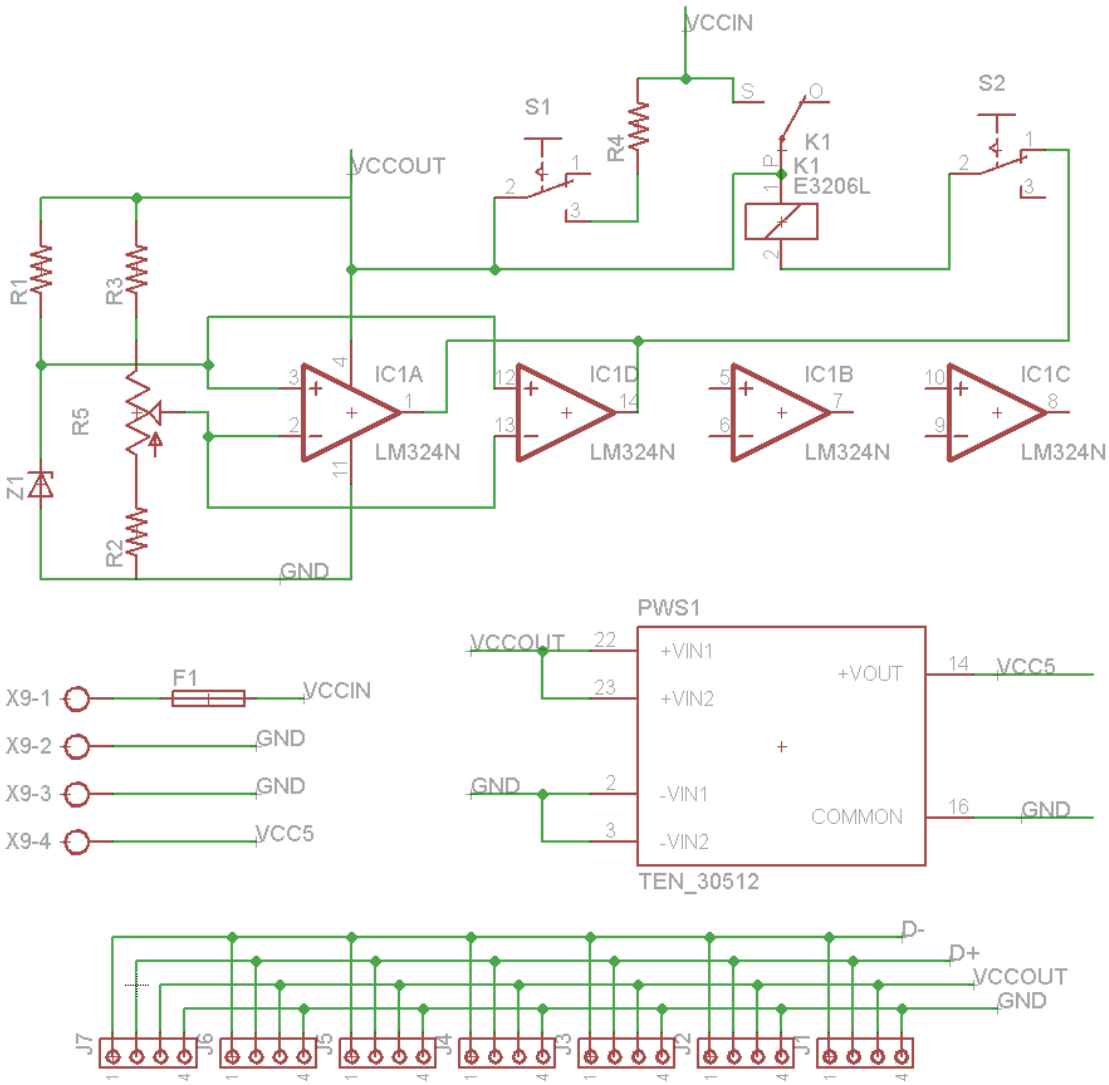


Figure 3.6: Schematics of the Power Management Circuit

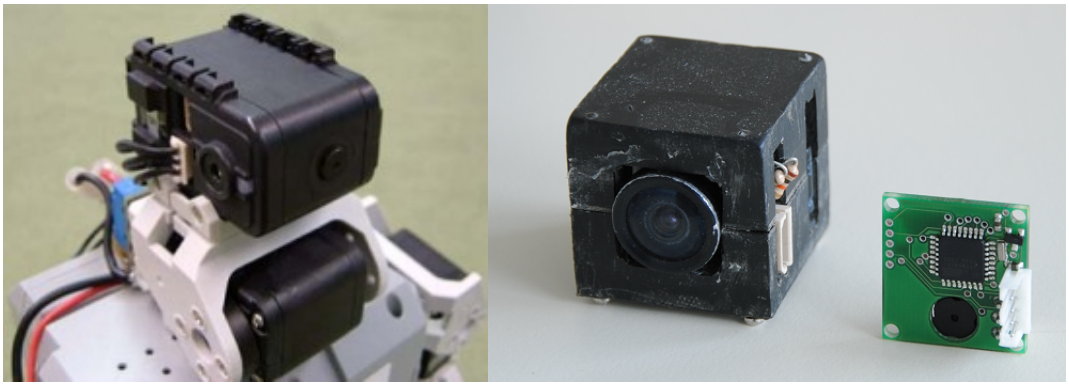


Figure 3.7: Vision Module, Different Versions

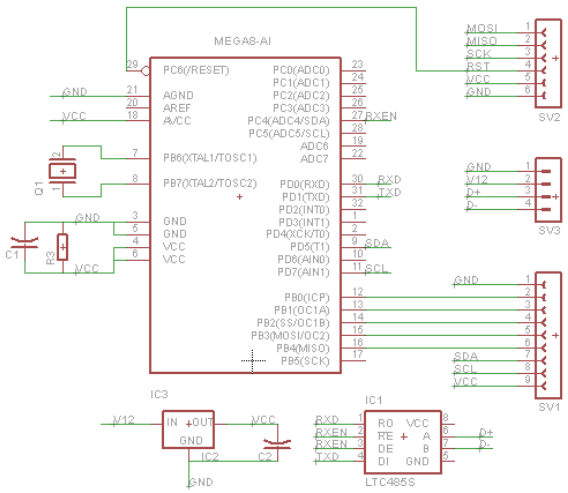


Figure 3.8: Schematic of the General Purpose IO Board

Chapter 4

Software Design

The software development for the FUmanoids ranges from low-level software for PID position control of the servos to high level programming to generate behaviors and make intelligent decisions. Most of the software development is done in C and C++ but for different target processors. In this section the software blocks of the robot are described briefly.

4.1 Low-level Software

4.1.1 Servo Motor Firmware Update and PID Control

ROBOTIS has developed a firmware for the servo motors. It is capable of controlling the position of the servomotor and regulating the speed to reach the goal position. The original firmware also calculates a set of feedback values, which are accessible using the available serial bus.

There were two main reasons motivating me to develop a new firmware for the servomotors. First the built-in controller is a simple P controller which limits the stability margins. For example due to lack of derivative feedback, overshooting happens for inertial loads. In addition, because of no integral feedback a residual error cannot be compensated. Figure 4.1 shows how the original controller of the Dynamixel servos can be adjusted.

The electrical design of the servos has been more or less reverse engineered and is available for programming purposes in several forums on the Internet. A conventional ISP interface is available on the PCB, which allows programming of the device. The control board of the AX-12 is shown in figure 4.2. The design is simple and easy to understand. The position feedback is obtained from a SMD potentiometer, connected to the output shaft of the motor. It is then fed to one of the analog

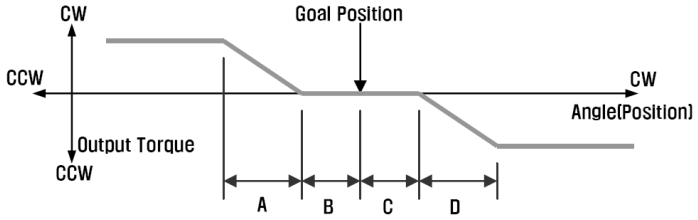


Figure 4.1: Feedback Curve of the Dynamixel Controller showing the adjustable parameters.

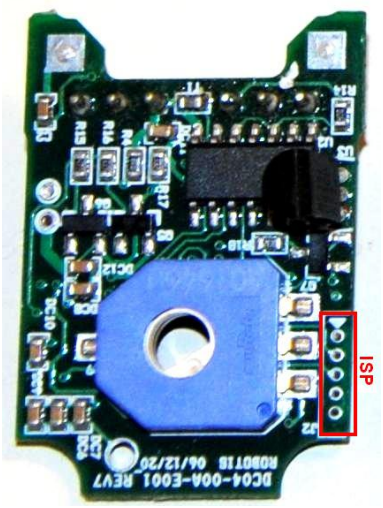


Figure 4.2: Control Board of AX-12. ISP interface is shown.

inputs of the ATmega8 microcontroller. Two PWM signals from the microcontroller are applied to the motor through the power stage.

The developed control software consists of two main parts. The interface carries the communication protocol and register table out and the controller realizes the PID control. The same implementation of the communication protocol for the motors has also been used in the camera module as well as in the general purpose IO module.

4.1.2 Operating System

In two versions of the robots, namely the 2007 and 2008 versions, the control software was run on an ATMega128 microcontroller. As the software complexity increased in 2008, it became necessary to organize the resources. Especially the complete implementation of the CSBP, which will be described later in this chapter, was not possible without enabling multi-threading. So a tiny operating system

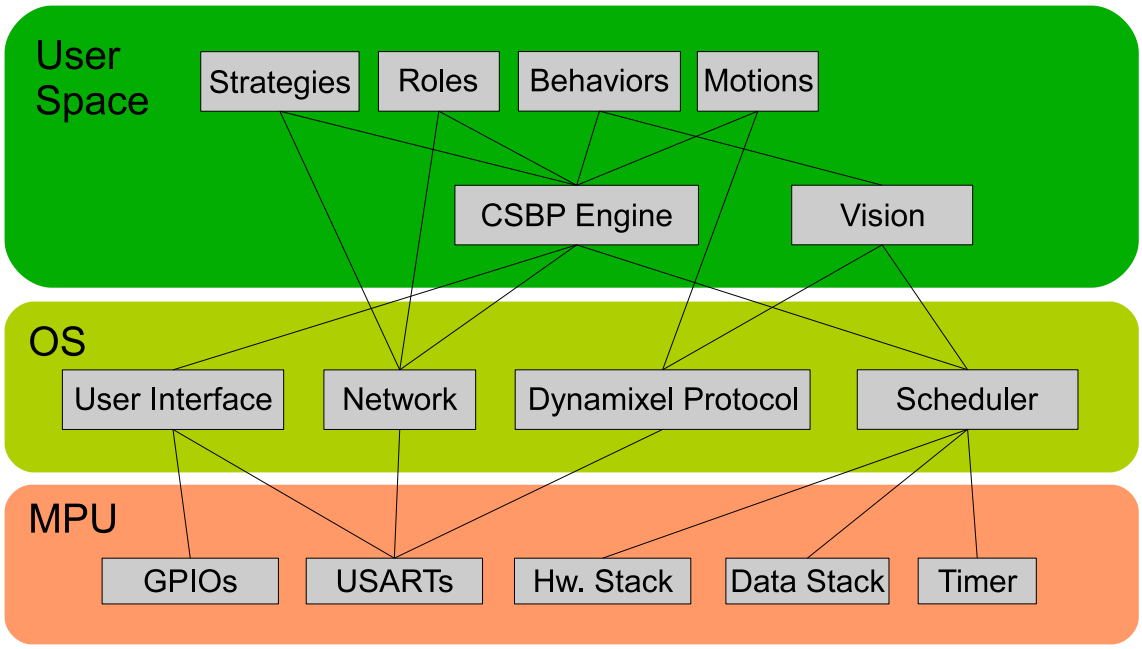


Figure 4.3: Block diagram of the operating system developed for 2008 robot

was designed and developed. Figure 4.3 shows a block diagram of the operating system. As shown in the figure, the scheduler is capable of running a static limited number of threads by assigning two stack areas to each thread and switching between them using a timer interrupt [45]. Each thread goes, or can be sent, to sleep mode by other threads. There are some tools available for assigning tasks to the threads or killing them. Communicating tools with the motor bus, vision module and the required tools for avoiding resource conflicts are included.

As already discussed, the 2009 robotic platform is based on a Linux compatible ARM processor. Therefore there was no extra need to develop an operation system. Hence, an abstraction layer for the needed resources was developed in user space instead. [46] describes the work.

4.2 High Level Software

The high level software of the robots is built up from 3 major parts: computer vision, communication and planning. Figure 4.4 shows an overview of the software architecture. In this section a summary of the high level software is given. Different parts of the software will be described in detail in further parts of the thesis.

4.2.1 Computer Vision

In 2007 and 2008 versions, object recognition was partly assigned to an external module. The module reported a list of colored blobs found in each frame. This list should be post-processed to standard field objects using existing logical relations. For example, a pole was distinguished from a goal by checking whether the color blobs of yellow and blue were vertically overlapping each other or a goal separated into two parts with a goal keeper in the middle was merged and the goal keeper was identified. Post-processing was done in the main microcontroller as part of the control program. Later in the 2009 version, the whole task of image processing was assigned to the main CPU. This included blob tracking using particle filtering, region growing, and post processing. Self localization and obstacle map generation proceeded using direct access to the pixel level. In part IV of the thesis I describe several developments I made in the robots' computer vision and their results.

4.2.2 Communication

No wireless communication was implemented in the 2007 version. The robot was therefore set up manually for starting a game. For this goal, a rather simple user interface was developed, which received the commands from the buttons on the CM5 and showed some status information using the LEDs and a serial terminal connection. The next version was equipped with a serial to WLAN converter. In this version the robot could be set into the game mode, in which it broadcasted its status periodically out and could receive status packets from other robots. Game start and stop signals together with other configuration commands could be sent from an external computer in the form of server packets. Status packets contained several fields carrying information about the state of the robot such as its position and ball position, as well as the decision state, i.e. what is the active role, behavior and motion. Each robot gathered status packets from other robots and updated a kind of world model, accessed in several layers of the planning program. On the server side, the same was done but the information was just visualized for debugging. The same concept is followed in the 2009 version with some features improved.

4.2.3 Planning, Behavior and Motion Control

RoboCup experience shows that in the early years of each league the focus lies on the hardware and control issues such as performance and reliability. This relation reduces rapidly with time as solutions and technologies improve in this area thanks to the well working idea exchange in the RoboCup community. Further the hardware and control systems of the participating teams become

more or less similar and the focus of the development shifts to the more abstract software side, where the cognitive behavior control plays the most important role. Planning and behavior control forms the major part of the software of the robot. This is also the most dynamic part of the control software in terms of development.

In this section I describe the planning system I designed for the FUmanoids. It is a multi-layer architecture called CSBP[1]. As shown in figure 4.4, each layer is designed to increase the abstraction of the sensors as well as the actuation commands for the higher layer. There are different signal paths connecting the layers together. Several control functions called *Scenarios* run concurrently in all layers. Each function is assigned to a certain task and starts as the task is about to be performed, and ends as the agent has either performed it successfully or failed to perform it. It was shown that this approach is capable of increasing the clarity of the control source code and reducing errors.

The developed behavior control consists of 4 layers called, from bottom to top, "Motion", "Behavior" , "Role" and "Strategy". As it can be understood from their names, behavior control is organized in layers from low level to abstract.

Motion This layer contains functionalities that produce the movements of the robot. Some examples are walking, stand up and kick. Some of the motions are static, these are pre-programmed movements which run independently from the outside world feedback. On the other hand, reactive motions can change dynamically to match the environment. Walking is a highly dynamic motion, which is discussed in detail in part III of the thesis.

Behavior Different skills of a soccer player are realized in this layer by activating and adjusting parameters of existing motions. Example behaviors are "Go to Ball", "Dribble ball" and "Block Opponent". Although all layers have access to the sensory data, called "world model", the majority of references to this data occur in behavior layer.

Role In the upper layer, "Roles" make high level decisions by activating proper behaviors to form a specific soccer player. There are different roles available such as: "Attacker", "Defender" and "Goal keeper".

Strategy The top-most layer is responsible for decisions related to the coordination of the whole team. These are developed in packages named "Strategies". A strategy decides which arrangement of the roles is needed for the current game situation and how the robots should change their roles in order to avoid conflicts. As in real soccer, there are different possible arrangements. Game strategy is set manually before the game start.

[1] Concurrent Scenario Based Planning

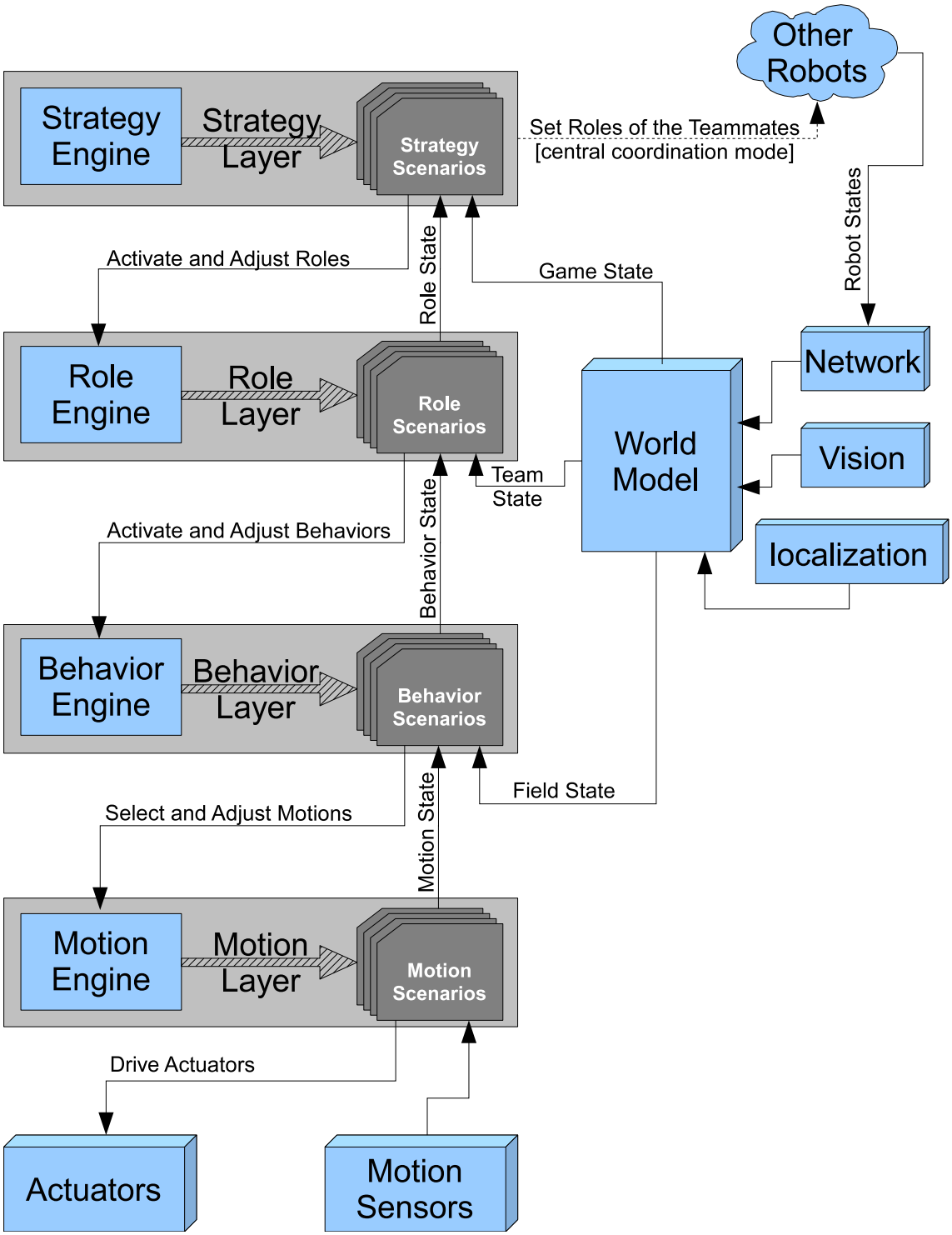


Figure 4.4: Architecture of Robot Control Software Including Concurrent Scenario Based Planning

4.2.4 Implementation of CSBP

My idea behind CSBP is to increase the readability of the control source code and decrease the fault rate by making the control code straight forward. This is done by organizing the control tasks into units called *scenarios*. By allowing the active scenario to run uninterrupted as long as the task it is assigned to is being performed, an almost direct relation between the task and the control source code can be established.

A scenario is defined as a function which is assigned to perform a behavior control task at a given abstraction level which starts with the task, continues with its progress and finishes as soon as the task has either successfully finished or failed. The definition seems to be trivial, however this is not the only way and not usually the way a function is assigned to control a behavior. The difference between scenarios and other behavior control methods becomes more clear as the number of parallel tasks or the number of abstraction levels increases.

According to CSBP, the programs should run simultaneously in all layers of the planning system. To manage the execution of "Scenarios" in different layers, there is one engine for each planning layer. This engine provides tools for assigning scenarios to the layer in addition to an inter-layer communication system. For example the behavior engine has a method called "switchBehavior()" this method is called by role layer in order to activate a new behavior scenario. Communication between layers is done using a status-feedback mechanism.

The planning system is based on a multi-layer architecture presented in figure 4.4. In this architecture each layer tries to increase the abstraction level of the perception as well as the actuation. I explain this abstraction process with some examples. In the field of actuation, the joint trajectories are abstracted to the desired motion of the robot in the motion layer. For example the higher layer only activates the *Walking* motion with a set of parameters. The required control and stabilization process is performed internally in the *Motion* layer. The actuation is further abstracted in the *Behavior* layer to a skill such as *GoToBall* with a set of parameters so that the higher layer has nothing more to do with walking and its parameters. Issues such as how to approach the ball and how to interact with the obstacles are handled in the behavior layer. The *Role* layer abstracts different skills to roles. There, an *Attacker* is formed out of several skills such as *GoToBall*, *Drible* and *Kick*. The role layer handles issues such as a fall down. A further abstraction layer is needed to organize the individual players into a team. This is called the *Strategy* layer in CSBP.

As shown in figure 4.4, the perceptual information is fed to each layer in two ways. The first set of sensory measurements are provided directly from the sensors. For instance, the relative position of the obstacles and the ball to the robot is directly used in the behavior layer. The second source of

perceptual information is the feed-back abstraction from the lower layers. For example whether the robot owns the ball is an abstraction of the ball and robot positions, which is calculated in behavior layer and fed to the role layer. A general feedback provided by each layer to the higher layer is the status of the active scenario. It indicates the progress level of the scenario as well as the the possible fail reason.

Each layer of CSBP is managed using an engine which also provides an interface to both top and bottom levels. Using multi-threading, the active scenario of each level is assigned to a thread to facilitate uninterrupted execution of it.

Part III

Control and Stabilization of Dynamic Walking

In this part, the techniques used to generate and stabilize a bipedal locomotion are described. There are plenty of approaches towards realization of dynamic walking, many of which require high precision actuation and sensorics [8]. There are also some, which try to rely on less precise hardware constructions and focus on development of simple techniques for correction of the errors produced instead[12, 13, 24]. The latter group considers the approach to be more human-like as human walking is also not based on the precision of actuation but rather on a trial and error based correction. The stabilization method discussed in this chapter is more or less inspired from a well discussed approach, called *Passive Dynamic Walking* [47, 1, 16]. Although the described method is very far from being passive after all improvements applied to it, it is still based on a passive dynamic walker model. In this chapter, a simulated robotic platform is initially presented, on which the walking algorithm is developed and further improved. Stability analysis is performed in the lateral plane as well as in the saggital plane. In both parts the system is modeled two-dimensionally and the effect of the third dimension is neglected. Final simulation in three dimensions, and experimental tests have later verified the correctness of this assumption under the test conditions. Finally, simulated and experimental results are presented and discussed.

The rest of this part deals with analysis and improvement of the behavior of the robot in lateral and frontal planes. Using a simple model of the bipedal robot, the methods are analyzed, discussed and improved. Theoretically achieved results are then verified using the simulator and finally fine tuned and tested on the real robotic platform.

Chapter 5

Modeling and Simulation of the Robotic Platform

In order to develop and test control rules for bipedal walking, it is necessary to simulate the platform to save time and hardware costs. However a whole bipedal robot as needed for the RoboCup humanoid league is too complicated to be fully simulated. Available physics engines have a limited capacity and precision. Some features are realized in low cost physics engines using large simplifications and wide estimations. Especially, significant inaccuracies exist in modeling joints and impacts.

For development of a control rule, it is usually enough to reduce the walking model to a few joints and masses. Hybrid simulation of a whole robot is another approach to reduce the degree of complexity. It is done by simulating a simplified model, derived from the whole robot by constraining it. For example, the foot to ground contact can be assumed as fixed, or some actuated joints are taken as rigid.

This section introduces the simulation tool used for the experiments and continues with modeling the components of the biped platform.

5.1 Mathematical Modeling

An important step towards analyzing biped walking is to model the biped. There are several suggested models with different complexities. Most of the differences are in the weight distribution and the way the leg length reduces to make ground clearance. Whether the model is analyzed 2D or 3D is also an important point. Figure 5.1 shows a widely used model called simplest walker [14].

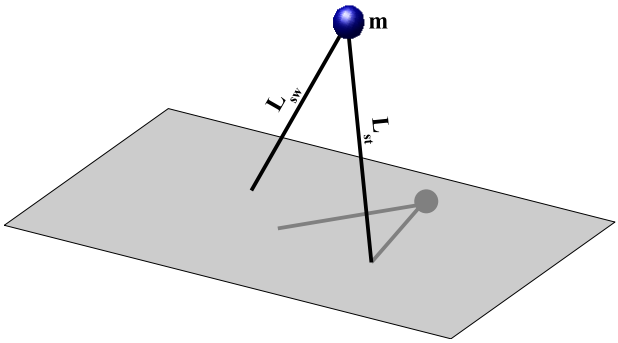


Figure 5.1: The simplest walker model.

The model is made of a point mass and two massless legs. There is no knee in the model. Two different approaches are followed for simulating the knee effect. In the first approach the leg length is considered as variable. In the second one the collision of the swing foot with the ground is ignored during the swing phase.The model can be analyzed in two dimensions as well as in three. In two dimensional analysis, the same model can be used in the frontal plane as well as in the lateral plane. Simplest walker model is easy to use as the robot can be easily modeled as an inverted pendulum in single-support phase. Elimination of the knee helps simplifying the analysis without great loss of generality. However, point mass distribution and massless legs are somehow less realistic and can make significant differences between the simulated results and the ones of the real system.

5.2 Numerical Simulation in Physics Engine

The core component of biped simulation is a physics engine. The function of this part is to calculate the dynamics, as well as to detect collisions between rigid bodies in the simulated world. Generally there are two classes of physics engines: real-time and high precision. High precision engines are capable of calculating very precise physics at the expense of more processing power and possibly loss of real-time response. On the other end, real-time engines simplify the calculations and reduce the accuracy to achieve an endurable real-time capability. Real-time physics engines are mostly used in computer games to improve realism [48]. There are three major paradigms for physical simulation of solids:

- Penalty methods: Interactions are commonly modeled as mass-spring systems. The method is popular for deformable objects.
- Constraint based methods: Constraint equations are solved that estimate physical laws.

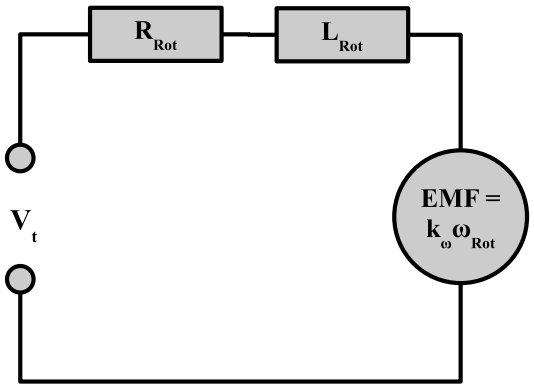


Figure 5.2: Electrical Model of a DC Motor

- Impulse based methods: Impulses are applied to object interactions.
- Hybrid methods: A combination of the above methods.

ODE[1][49] is one of the most common physics engines for simulating robotic environments[50]. It is available under BSD license and the LGPL. The two main components of the software are rigid body dynamics simulation engine and collision detection engine. Many 3D rigid bodies with arbitrary mass distribution as well as several joint types are definable in the engine. ODE is used in this research together with other simulation techniques to verify the stability of the proposed control rules.

5.2.1 Simulation of the Actuators

The only type of actuators used in FUmanoid robots are DC servomotors. A DC motor is an almost linear system. A simplified electrical model of the motor is presented in figure 5.2.

An applied voltage to motor terminals causes a current flow in the rotor. This produces a torque which leads to rotation of the motor shaft. A back EMF relative to the angular velocity of the motor is therefore induced in rotor windings. The higher the speed of the motor, the less the current flows in the rotor circuit. In steady state, the motor reaches a limit speed in which the sum of input and output torques becomes zero. The following equations describe the behavior of the motor.

$$i_{Rot} = \frac{V_t - EMF}{R_{Rot}} \tag{5.1}$$

$$EMF = k_\omega . \omega_{Rot} \tag{5.2}$$

$$\tau_{Rot} = k_i . i_{Rot} \tag{5.3}$$

[1] Open Dynamics Engine

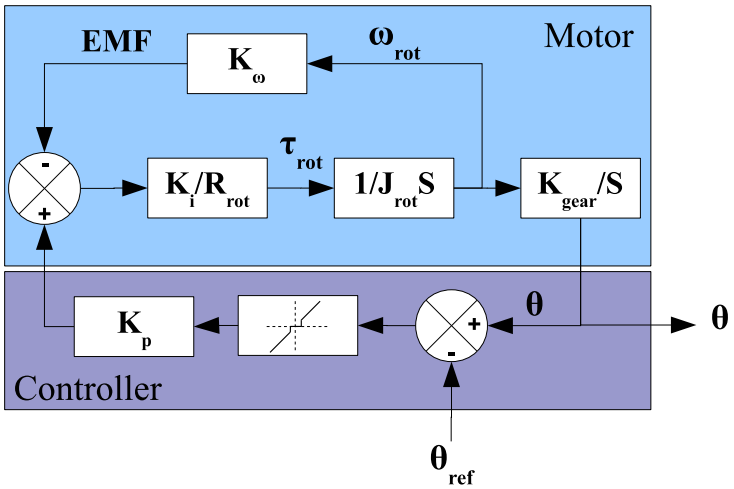


Figure 5.3: Block Diagram of the Control System of the Servos

$$\tau_{Rot} = J\frac{d\omega_{Rot}}{dt} \tag{5.4}$$

Figure 5.3 shows the block diagram for the servo motor. The internal feed-back loop for position control of the output angle of the shaft is presented in a separate block. It is an ordinary P controller with adjustable dead zone and stiffness.

Note that in equation 5.4 the inertia of the motor, gears and the load were summarized in parameter J. The first two parameters can however be neglected as long as the load inertia is big enough.

5.2.2 Simulation of the Sensors

As discussed in the description of the platform, the robot gains measurement data from 3 sources besides the camera. It is necessary to provide similar information from the simulated environment. The most important sensory data used to control the walking is the bearing angles of the servomotors. As servo motors usually apply a proportional feedback to control the position, a small deviation usually exists between the reference position and the current one. This measurable deviation is proportional to the output torque and is a side product of the angle measurement of the servos. In ODE, the current angle of the joints is directly readable. In some cases, for example the ball joint, the needed values can be calculated using simple vector operations.

Tactile sensors, or switches used in the bottom of the feet to sense a ground contact are important to distinguish between the stance leg and the swing leg. The collision detection engine of ODE helps in this case by triggering an event upon a collision and providing a list of colliding objects.

IMU is also used in some control rules to calculate the tilt angles of the body. All needed values can simply be calculated having the position and velocities of all objects. Furthermore it is also possible to generate feed-backs impossible to gain in a real environment such as the absolute position and angles of the bodies. This can simplify the problem and accelerate the development by allowing intermediate control solutions.

5.2.3 Simulation of the Mechanical Model

The aim of the numerical simulation is to verify and improve the control paradigm used for bipedal walking. The simulated platform is therefore an intermediate step between the extremely simplified mathematical model and the very complex real humanoid robot. Making the simulated model too complex would not only worsen the performance of the simulation, but could not also help to bring it closer to reality. Indeed lots of simulation errors would accumulate and impact the behavior of the system. It is therefore necessary to select the detail depth of the simulation carefully.

The simulator I developed for this purpose is based on the simplest walker of figure 5.1. Some features are added to increase the realism of the platform which are as follows.

- Mass distributions belonging to thighs and shanks are applied to make the model more realistic.
- A sliding mechanism replacing the knee model allows gaining foot clearance as well as lifting the center of mass.
- Foot planes with adjustable torques can be used to energize and stabilize the system.

5.3 Visualization of the Simulated Results

Visualization of the simulated platforms is very important as it accelerates the development of the simulator, helps finding and removing bugs and simplifies the interpretation of the results generated from the simulator. A widely used library for this purpose is OpenGL[2]. OpenGL is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics [51]. The other reason for using OpenGL is that it is fully compatible with ODE, saving a considerable amount of development time.

Figure 5.4 shows a screen-shot of the output generated by OpenGL. In this example the visualized model is similar to the simplest walker of figure 5.1. The robot is simulated as a combination of rigid bodies and joints.

[2]Open Graphics Library

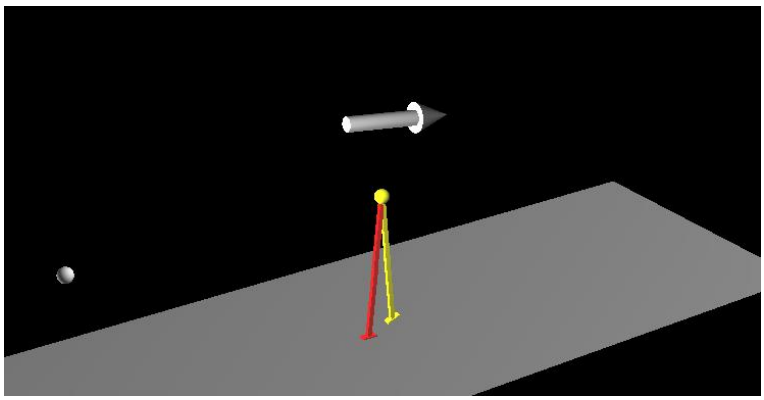


Figure 5.4: Visualization of the Simulated Platform using OpenGL

Chapter 6

Parameter Space Conversion and Inverse Kinematics

In synthesis of bipedal walking gaits, there is a huge parameter space to struggle with. It is often hard to find an appropriate combination of parameters to cause a certain effect. For example if the robot leans backward, it is not clear which joint angles, at which times and to which extent should be changed. In addition there are several possible ways to produce the same effect without clear view to the side effects caused by them. Some values must be changed respectively to guarantee the consistency. For example to increase the foot distance to the ground, at least 3 joint values should be changed, else the robot becomes completely destabilized.

To address the complexity problem of the kinematics of the biped, I followed two different approaches. In the first approach I apply a simple linear parameter space conversion called *Pseudo Inverse Kinematics*[1] to the joint position space. This helps achieving two important goals. The dimensions of the parameter space can be significantly reduced by defining a parameter space, in which several parameters remain constant and do not contribute in a normal walking motion. The parameter set can be chosen so that the values describe intuitive, simple relations of the geometrical properties of the robot such as leg length, step length, lateral lean and so on.

The second approach followed in this chapter is the use of common methods for calculation of inverse kinematics. In this approach the desired state of the robot is described with the coordinates of a set of specific points on its body, based on which the joint angles are calculated. Here, the work is concentrated on optimizing the calculations without much loss in the generality of the solution. The

[1] PIK

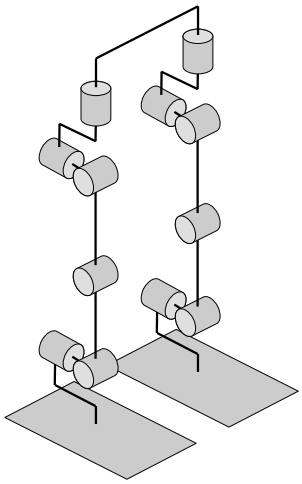


Figure 6.1: Kinematic Model of the Biped

chapter also describes the implementation of both approaches in FUmanoid robots.

6.1 Pseudo Inverse Kinematics

In figure 6.1 a model of the kinematics of the robot is presented. Concerning the symmetry of the system, it is enough to calculate the conversion only for one step. Let's call the legs "*Stance leg*" and "*Swing leg*". A step is defined as the movement of the robot starting from the state with both feet on the ground and ending in the mirrored state by moving the swing leg from back to front including the lifting needed to achieve foot clearance. To serve the consistency, the conversion is defined so that the final state of the robot is the same as the initial state under exchange of stance and swing legs.

In pseudo inverse kinematics a linear conversion is defined around a neutral working point which is in the case of bipedal walking a vertically standing robot. The conversion can be defined as follows:

$$\mathbf{S} = \mathbf{CP} + \mathbf{S_0} \tag{6.1}$$

$\mathbf{S}$ contains the calculated joint positions, $\mathbf{S_0}$ is the initial position explained above, $\mathbf{C}$ is the conversion matrix, and $\mathbf{P}$ contains the desired parameter set in the new space. The conversion matrix does not necessarily have to be quadratic and invertible but it can be very advantageous to calculate the parameter vector for an arbitrary pose of the robot possibly read back from the servos.

To avoid complications let's ignore the upper trunk and define the joint position matrix, **S** as follows:

$$\mathbf{S} = \begin{bmatrix} RightHipYaw \\ LeftHipYaw \\ RightHipRoll \\ LeftHipRoll \\ RightHipPitch \\ LeftHipPitch \\ RightKnee \\ LeftKnee \\ RightAnklePitch \\ LeftAnklePitch \\ RightAnkleRoll \\ LeftAnkleRoll \end{bmatrix} \tag{6.2}$$

Assume the set of parameters to be as follows:

$$\mathbf{P} = \begin{bmatrix} SwingLegLength \\ StanceLegLength \\ FrontalCOMShift \\ LateralCOMShift \\ FrontalStepLength \\ LateralStepLength \\ BodyRotation \\ TorsoAngle \end{bmatrix} \tag{6.3}$$

A key feature needed for bipedal walking is shortening the swing leg to achieve foot clearance from the ground. In the present model, as also in the human model it is done by bending the knee joint. Parameter "*SwingLegLength*" is defined for this purpose. It has a similar effect to the parameter l in figure 5.1. To maintain consistency, this parameter should affect three joint values shown darker in figure 6.2d. Assuming the right leg to be the stance leg, a partial presentation of the conversion

matrix applying this parameter is as follows:

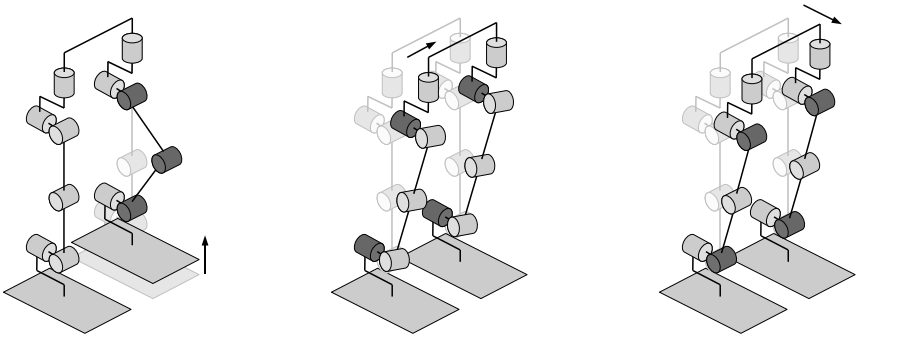
$$\mathbf{S} = \begin{bmatrix} 0 & . & . & . & . & . & . & . & . \\ 0 & . & & & & & & & \\ 0 & & . & & & & & & \\ 0 & & & . & & & & & \\ 0 & & & & . & & & & \\ 1 & & & & & . & & & \\ 0 & & & & & & . & & \\ 2 & & & & & & & . & \\ 0 & & & & & & & & . \\ -1 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \end{bmatrix} \mathbf{P} + \mathbf{S_0} \tag{6.4}$$

Coefficients are selected so that changing the stance leg length affects neither the angle between the feet and the ground, nor the horizontal position of the feet. The parameter does only have an impact on the effective length of the leg. Note that the sign of each coefficient depends on how the corresponding joint actuator is located and connected to the chain. Signs can be determined experimentally using a few trials.

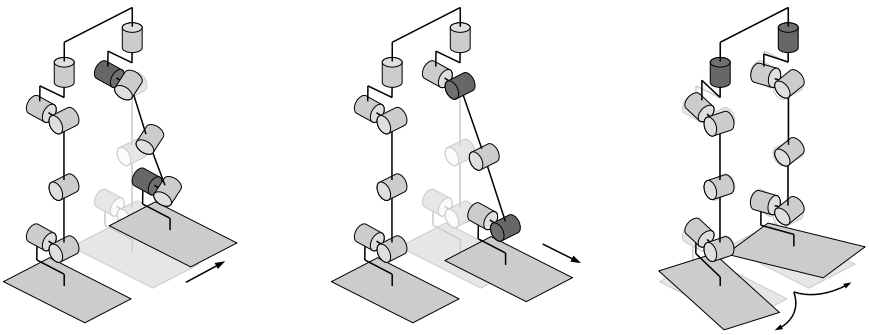
The parameter "*StanceLegLength*" is defined in a similar way. This parameter can be used to pump energy into the system as described later in section 7.4.1.

The parameter pair "*FrontalCOMShift*" and "*LateralCOMShift*" are used to move the upper trunk in either of the planes. This is done parallel to the swing leg motion. Each parameter affects two joints of the stance leg and two of the swing leg in order to guarantee the parallelism of the swing foot to the ground and balance of the upper trunk. The affected joints are shown darker in figures 6.2b and c. A step is made by lifting the swing foot and moving the center of mass and the swing foot in the desired direction. There are two parameters for the movement of the swing foot, each of which have a simultaneous impact on two joints. These parameters are presented in figures 6.2d and e. Parameter "*FrontalStepLength*" is used in conjunction with "*FrontalCOMShift*" to produce sideways steps. Parameters "*LateralStepLength*" and "*LateralCOMShift*" play a similar role in backward/forward walking. To apply a rotation to the body the parameter "*BodyRotation*" is defined. This parameter rotates both legs symmetrically using *HipYaw* actuators as presented in figure 6.2f.

The complete transformation is presented in equation 6.5.



a) SwingLegLength b) FrontalCOMShift c) LateralCOMShift



d) FrontalStepLength e) LateralStepLength f) BodyRotation

Figure 6.2: Effect of the PIK Space Parameters on Joint Angles

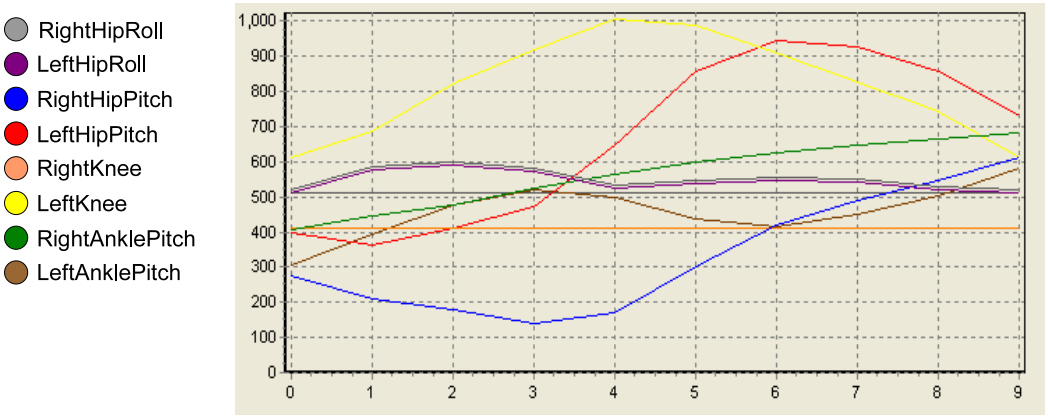
$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{P} + \mathbf{S_0} \tag{6.5}$$

The conversion matrix is not yet quadratic and therefore not directly invertible. As noted above, it is helpful to find an inverse or at least a pseudo inverse conversion which gives the best matching parameter configuration for an arbitrary pose of the robot. Theoretically, there are two possible ways to create an inverse conversion. In the first method, the pseudo inverse of the conversion matrix is calculated and used in the inverse conversion. The second technique tries to add dummy parameters to the space so that the conversion matrix becomes invertible. As the inverse solution is just partially needed by the stabilization algorithm, the second method is used as it will be discussed later in this part.

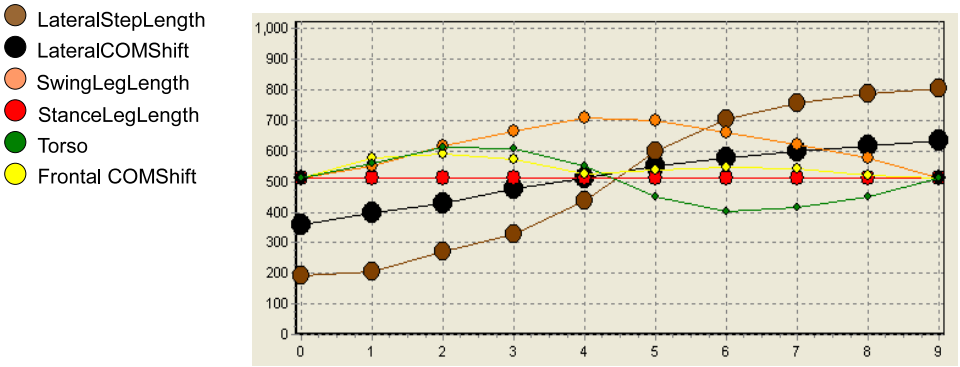
Pseudo inverse kinematics simplifies understanding of how gait parameters affect the motion. In figure 6.3a the recorded curves from the joint angles during a typical walking gait are presented. Figure 6.3b shows the same motion in PIK space. It can be clearly observed that the robot lifts the swing foot and simultaneously shifts the center of mass and the swing foot in the lateral plane.

In addition, pseudo inverse kinematics helps dividing the complex multi-input multi-output system of the robot into almost independent subsystems with less inputs and outputs. Such a system can then be controlled using well known classical control techniques. For example it is shown in [52] that the parameter *TorsoAngle* has a direct impact on walking speed. It is therefore possible to use a PID controller to adjust walking speed using this parameter based on the feedback from the speed of the center of mass.

Pseudo inverse kinematics is straight forward and effective. These features make the approach proper for implementation on low power CPUs. There are many commercially available platforms with 8 bit microcontrollers as the main processing unit. The 2008 FUmanoid robot was also a similar



(a) Joint Angles of a Typical Walking Gait



(b) Motion Data in PIK Space.

Figure 6.3: Comparison Between Joint Space and PIK Space.

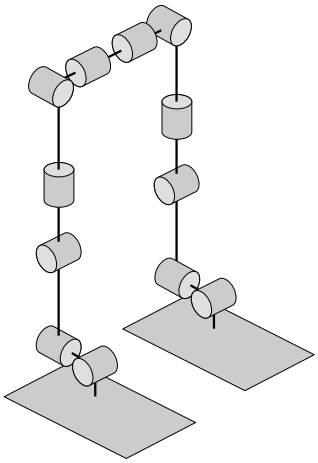


Figure 6.4: A Simpler Mechanical Design to Use With Inverse Kinematics

platform. On the other hand, the idea is only suitable for platforms with relatively simple kinematics. For example to be able to define the parameter *SwingLegLength*, it is required that three successive pitch joints with parallel axes are mounted in the legs as in 6.1. Moreover the assumed superposition holds true only in a relatively small part of the parameter space. This limits some important features such as step length. It is therefore important to implement a general inverse kinematics solution which is valid for a wider range of parameters. As the inverse kinematics is a rather complex problem, different solutions and optimizations are discussed in the next sections.

6.2 Inverse Kinematics

Several unavoidable shortcomings make PIK inappropriate for implementation on robots with complex kinematics. Having more CPU power available, it would be possible to simplify the mechanics, shift the complexity to the software part and use inverse kinematics. Figure 6.4 shows the kinematic model of such a robot. The design and assembly time of the robot and also the production cost is reduced to a great extent, however the prerequisites of the pseudo inverse kinematics can no longer be met. It is therefore essential to use the general form of inverse kinematics for parameter space conversion.

In this section I first define the parameter space. I solve the forward kinematics as the next step and discuss some solutions to the inverse problem based on the forward one. I present two different

numerical methods to solve the inverse kinematics and finally I describe the implementation of an analytical solution which is optimized enough to be implemented in the robot.

6.2.1 Parameter Space Definition for Inverse Kinematics

Using inverse kinematics, the parameter space of the robot is simply the position and orientation of a specific set of points on the robot in a reference Cartesian coordinate system. A commonly used convention for selecting frames of reference in robotics is the *Denavit-Hartenberg (D-H)* convention described in [53]. Definitions in this section are based on this convention.

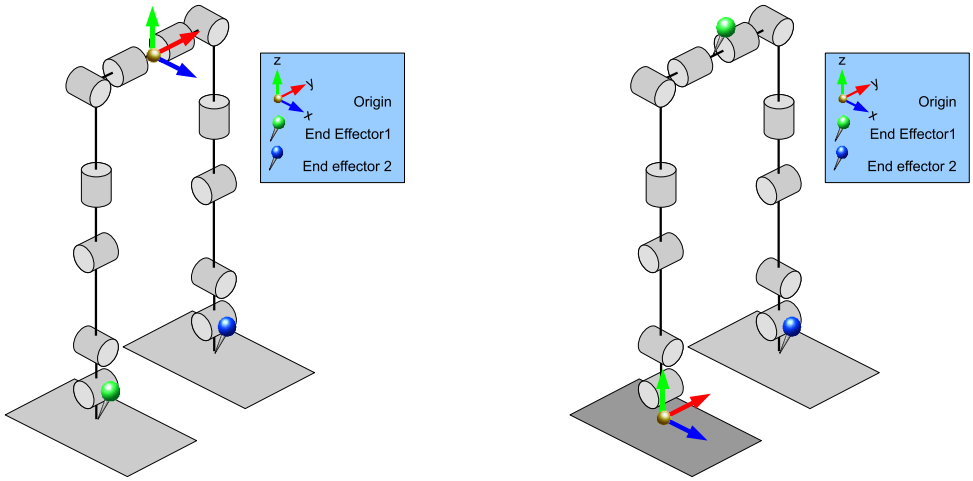
In *D-H* convention end effectors are coordinate systems fixed on certain parts of the robot. Each end effector includes the position vector of the origin, **O**, and three orthogonal normal vectors, **X**, **Y**, **Z**, representing three axes of a coordinate system fixed on the part. All coordinates are given relative to the base coordinate system. Following shows the homogeneous representation of an end effector.

$$\begin{bmatrix} X_x & Y_x & Z_x & O_x \\ X_y & Y_y & Z_y & O_y \\ X_z & Y_z & Z_z & O_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

The representation is straight forward, however the orientation has been represented with a degree of redundancy. It should therefore be ensured that the three axis vectors are unique and orthogonal to each other, forming a right handed coordinate system. A total of 24 not completely independent parameters describe the state of the end-effector in this case.

To place the coordinate system, there are different alternatives. Depending on gait production and stabilization methods used, a proper placement should be taken. The origin can be placed on the robot's center of mass so that the bottom of each foot can be defined as an *end effector*. This can simplify the calculations by taking advantage of the symmetry. However this leads to some discontinuities when stance and swing feet exchange roles.

In order to provide compatibility with pseudo inverse kinematics, the coordinate system can be placed at the bottom of the stance foot. End effectors can then be placed once at the center of mass and once at the bottom of the swing foot. Both models are presented in figure 6.5 a and b. End effectors are called *COM* for center of mass and *SWF* for swing foot in further calculations.



(a) Symmetric placement of the end-effectors (b) Asymmetric placement of the end-effectors

Figure 6.5: Examples of Parameter Space Definition for Inverse Kinematics

6.2.2 Forward Kinematics

The complete procedure for calculation of forward kinematics using D-H convention is described in [53]. A summary is provided here and the results are presented for the above discussed robot model.

In D-H convention a homogeneous transformation matrix is calculated for every joint in the chain. The transformation includes geometrical constants of the robot as well as variable parameters such as actuator angles and lengths. For that, an intermediate coordinate system is placed on each stage. Positions are determined regarding the placement and orientation of the successive actuators and therefore may not be necessarily placed on the center of each joint.

A joint transformation is a coordinate conversion from the n^{th} intermediate coordinate system to the $n-1^{th}$. This is done using four sub-conversions, which are as follows:

- Displacement along previous Z to the common normal

$$Trans_{z_{n-1}}(d_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.7}$$

- Rotation about previous Z to match the X axis

$$Rot_{z_{n-1}}(\theta_n) = \begin{bmatrix} \cos\theta_n & -\sin\theta_n & 0 & 0 \\ \sin\theta_n & \cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.8}$$

- Length of the common normal

$$Trans_{x_n}(r_n) = \begin{bmatrix} 1 & 0 & 0 & r_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.9}$$

- Angle about the common normal, from old Z axis to new Z axis

$$Rot_{x_n}(\alpha_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_n & -\sin\alpha_n & 0 \\ 0 & \sin\alpha_n & \cos\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

  where the common normal is the shortest distance between the rotation axis (always Z axis) of the successive joints. The following transformation is resulted by combining the above mentioned components.

$$T_n^{n-1} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

For each joint transformation, four parameters should be determined. These are d, θ, r and α. In the case of bipedal robots constructed entirely out of revolving actuators all parameters besides θ are geometrical constants and θ is the joint angle. Applying D-H procedure, joint parameters of the robot are listed in table 6.1 for both right and left legs as the stance leg. Values listed for θ are initial offsets determining the natural angle of the joints. These should be further matched to the angle values measured from the actuators using shifting and scaling. In addition, two dummy joints are inserted at the beginning and the end of the chain to match the design of the feet. These joints are assumed to be fixed, i.e. $\theta = 0$.

Stance: Left					Stance: Right			
r	θ	d	α		r	θ	d	α
35	-90	0	90		35	-90	0	90
0	90	0	-90		0	90	0	-90
43	0	96	90		43	0	96	90
-10	-90	0	90		10	-90	0	90
-111	90	0	-90		-111	90	0	-90
0	90	0	-90		0	90	0	-90
-65	0	0	180	COM	65	0	0	180
65	180	0	90		-65	180	0	90
0	-90	0	-90		0	-90	0	-90
111	-90	0	-90		111	-90	0	-90
10	-90	96	-90		-10	-90	96	-90
-43	0	0	90		-43	0	0	90
0	90	0	90		0	90	0	90
35	0	0	90	SWF	35	0	0	90

Table 6.1: D-H Parameters of the 2010 Robot

To calculate the pose of an end-effector, the pose of its origin in its coordinate system, which is of course an identity matrix, is transformed sequentially back through the chain to the base frame. This is shown in equations 6.12 and 6.13.

$$\mathbf{COM} = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6 \tag{6.12}$$

$$\mathbf{SWF} = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6 T_8^7 T_9^8 T_{10}^9 T_{11}^{10} T_{12}^{11} T_{13}^{12} T_{14}^{13} \tag{6.13}$$

It is also possible to calculate the pose of a stage based on the pose of a former one.

$$\mathbf{SWF} = \mathbf{COM} T_8^7 T_9^8 T_{10}^9 T_{11}^{10} T_{12}^{11} T_{13}^{12} T_{14}^{13} \tag{6.14}$$

Forward kinematics is the base of many numeric solutions of inverse kinematics. It is also used to back-transform the position feedback derived from the joints into the parameter space for stabilization and control of the robot.

6.3 Numerical Solution of Inverse Kinematics Based on Forward Kinematics

So far, forward kinematics provides a function computing the end-effectors in Cartesian space given the joint angles. The goal of inverse kinematics is to calculate the inverse of this function. Namely, given the desired pose of the end-effectors, it computes the required joint angles. As the

forward kinematics is a highly nonlinear function with a multi-input and output function, it is theoretically impossible to calculate its inverse analytically and in general form.

To struggle with this issue, there are many proposed solutions. These either try to solve the problem for special cases using analytical approaches or try numerical approaches towards a solution. A majority of the numerical solutions use the forward kinematics and its derivatives to solve an optimization problem. In the next sections, implementations of some of these methods on the bipedal mechanism are discussed.

6.3.1 Gradient Descent Method

To reduce the complexity of the output side of the forward kinematics, the gradient descent method suggests the use of an evaluation function. It maps the output vector into a single value, concerning the given goal pose. The evaluation function should be selected carefully because it plays an important role in the solution. The function should reach an absolute extremum where the end-effectors reach their goals. It should be as free as possible from local extrema.

The evaluation function used in the implementation of the gradient descent method is simply the Euclidean distance between each end effector and its desired value. However I noticed that some kind of weighting was necessary to equalize the effect of position and orientation components as the vectors representing the orientation of the end-effectors are normalized. A scale factor of 50 is applied to the orientation error.

The gradient Descent method is an iterative procedure, which guarantees the convergence to a local extremum. The method can only be used when the initial state is close enough to the goal state in order to guarantee the convergence to the proper result. Figure 6.6 shows the method. In each iteration, the gradient vector is calculated at the given position. It shows the direction with the steepest descent on the given point. The position is then advanced with regard to the gradient vector. It is very important how the step length is calculated, as it is a key to both precision and performance of the algorithm.

The method works fine for the first end-effector, but when including the second end-effector the performance of the method decreases significantly. An effective solution is to solve the first half of the inverse kinematics as it is independent from the second half. The rest can then be solved by assignment of the calculated joint values in the first half.

To calculate the gradient vector in each step, forward kinematics should be called as many times as the number of input degrees of freedom. This is in this case 12, which is computationally very expensive. Using the above described separation, the calculation of the forward kinematics can be

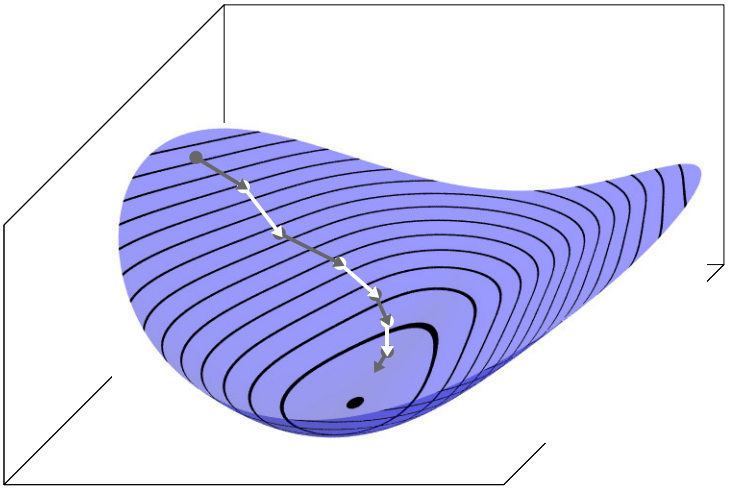


Figure 6.6: Gradient Descent Method

halved.

6.3.2 Stochastic Iterative Method

As discussed in the last section, it is too expensive to calculate the gradient of the evaluation function. Therefore another approach is presented in this section. The procedure is described in algorithm 1. In this method a shaped random step vector is generated in the joint-space. It is then verified whether the step improves the evaluation. A damped accumulator is used to smooth and accelerate the convergence to the solution.

The number of references to the forward kinematics is reduced to one per iteration in this method. A majority of the trials are rejected at the beginning but thanks to the damped accumulator, rejection rate is reduced significantly with growing number of iterations. On the other hand, much bigger steps can be made compared to the gradient descent method. This causes an overall improvement in the performance of the method.

The same technique described in the last section to separate the left and right side of the chain can also be used here to improve the performance.

6.3.3 Jacobian Pseudo Inverse Method

Using a vector representation of the joint angles and the end effectors, forward kinematics can be formulated as follows:

Algorithm 1 Stochastic Iterative Algorithm

```
init last evaluation
init joints
init accumulator
while optimal evaluation not reached
begin
    step = small random vector
    if eval(joints+accumulator+step) better than last evaluation then
    begin
        accumulator += step
        joints += accumulator + step
    end
    else
    begin
        accumulator *= damping factor
    end
    update last evaluation
end
```

$$\mathbf{E} = \mathbf{FK}(\theta) \tag{6.15}$$

where $\mathbf{E} = [e_1, e_2, ..., e_n]$ is a vector containing all components of all end effectors and $\theta = [\theta_1, \theta_2, .., \theta_m]$ is the vector of all joint angles. **FK** is the forward kinematics function defined in equations 6.12 and 6.13. Derivation of equation 6.15 results:

$$\dot{\mathbf{E}} = \mathbf{J}(\theta)\dot{\theta} \tag{6.16}$$

where the *Jacobian* matrix **J** is defined as follows:

$$\mathbf{J} = \left(\frac{\partial e_i}{\partial \theta_j}\right)_{i,j} \tag{6.17}$$

Based on the *Jacobian*, forward kinematics can be linearly approximated around a given working point.

$$\nabla\mathbf{E} \approx \mathbf{J}(\theta_\mathbf{0})\nabla\theta \tag{6.18}$$

As the inverse of the function is proposed, the linear system of equations 6.18 should be solved. In general form, the *Jacobian* is not square and therefore not invertible. Its pseudo inverse can be used instead. This results in:

$$\nabla\theta \approx \mathbf{J}^\dagger(\theta_\mathbf{0})\nabla\mathbf{E} \tag{6.19}$$

The *Jacobian* matrix can be calculated easily with as many references to the forward kinematics as the number of joints. Let's assume all joints to be revolving, which is the case in humanoid robots used in this thesis. By partially differentiating both sides of equation 6.13 upon $\theta_\mathbf{j}$:

$$\frac{\partial \mathbf{SWF}}{\partial \theta_j} = T_1^0 T_2^1 ... \frac{\partial T_j^{j-1}}{\partial \theta_j} ... T_{14}^{13} \tag{6.20}$$

this is possible because all other transformations are independent from θ_j. Moreover the transformation can also be separated to 4 parts, only one of which is a function of θ_j. It can be shown that

$$\frac{\partial Rot(\theta_j)}{\partial \theta_j} = Rot(\theta_j + \frac{\pi}{2}) \tag{6.21}$$

It is therefore possible to calculate the partial derivative of the end-effector by just calculating the forward kinematics and adding $\frac{\pi}{2}$ radian to the proposed component. This result can be generalized on a vector of several end-effectors, which allows calculating one row of the Jacobian using a single reference to the forward kinematics.

In comparison to gradient descent method, an optimal solution can be reached in this method in a noticeable lower number of iterations. It is generally possible to use much bigger step sizes in this method. Actually compressing the whole state space to a scalar evaluation value increases the complexity of the optimization problem and makes the selection of a proper evaluation function very hard with higher number of joints. The Jacobian pseudo inverse method is known to become unstable near the singularities. In [54] and [55] several improvements have been suggested to the method.

6.3.4 Jacobian Transpose Method

It is shown in [56] and [57] that the solution to the inverse kinematics can also be reached when the inverse of the Jacobian matrix is substituted with its transpose, i.e.:

$$\nabla\theta = \alpha \mathbf{J}^T(\theta_\mathbf{0}) \nabla \mathbf{E} \tag{6.22}$$

where α is a sufficiently small positive scalar. This is known as Jacobian transpose method.

The above described method was implemented on the robot. Under normal walking gaits it could always converge to the solution by setting the minimum allowed distance to 0.1 and using the distance measure described in section 6.3.1. An average of 62% CPU usage is reported on a Gumstix Overo platform (ARM OMAP3530 @ 800MHz). All computations are implemented in fixed point arithmetics, as the platform provides no FPU. The performance reached is not satisfying, as other processes such as computer vision and behavior control run on the same platform and need a significant amount of processing power.

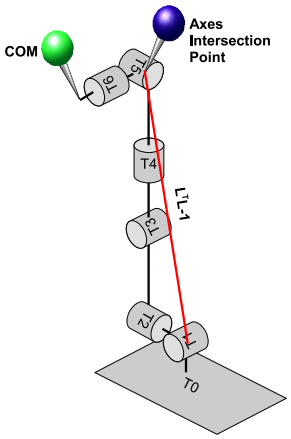


Figure 6.7: Pieper's Solution on the Stance Leg

6.4 Closed Form Solution of IK Based on Pieper's Method

As discussed in the last few sub-sections, numeric solutions of the inverse kinematics are computationally too expensive to be used on the on-board processor of the robot. An analytical solution, despite of how complex it is, can be much faster than any iterative method. Inverse kinematics of a high DOF chain can rapidly become so complex that there may exist no closed form solutions for it. Fortunately there are some design tips which help reducing the complexity.

Pieper has given a sufficient condition for providing a closed form solution to a 6 DOF system in his PhD thesis [58]. Pieper's solution is applicable when 3 consecutive actuators have a common intersection of their axes or the axes are parallel to each other and in a few other configurations.

Both versions of the humanoid robot studied in this thesis contain 3 revolutionary stages with a common axis intersection point in the chain. A useful property of such a construction is that the group of actuators only affect the orientation of an end-effector placed on the intersection point. Assuming such a system with six degrees of freedom, there remains just a system of three equations and three unknowns which is of course much easier to solve than a system with six degrees of freedom.

Figure 6.7 shows the stance leg of the robot, which is in this case the left leg. Equation 6.12 describes the pose of the end-effector placed in the center of mass in terms of the first six joint values. To be able to use Pieper's solution, the end-effector should be placed on the intersection point of the last three joints. This does not hold for the end-effector **COM**. Regarding the leg model, the end-effector has a displacement of one half of the hip lateral distance from the desired position. This is due to the latter two components of the transformation T_7^6, i.e. $Trans_{x_7}(r_7)$ and $Rot_{x_7}(\alpha_7)$. Applying the inverse of T_7^6 to both sides of the equation results:

$$\mathbf{COM}(T_7^6)^{-1} = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \tag{6.23}$$

To eliminate the position component of the end effector, both sides are simply multiplied with $\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ resulting:

$$\mathbf{COM}(T_7^6)^{-1}\mathbf{P} = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \mathbf{P} \tag{6.24}$$

It can be shown that both products $(T_7^6)^{-1}\mathbf{P}$ and $T_6^5\mathbf{P}$ are constant vectors as follows:

$$(T_7^6)^{-1}\mathbf{P} = \begin{bmatrix} 0 \\ 0 \\ -65 \\ 1 \end{bmatrix} = \mathbf{Q} \tag{6.25}$$

$$T_6^5\mathbf{P} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{P} \tag{6.26}$$

In addition T_1^0 is a constant matrix as described in 6.2.2. Applying these results in equation 6.27 it is reduced to:

$$\left(T_1^0\right)^{-1}\mathbf{COM}\,\mathbf{Q} = T_2^1 T_3^2 T_4^3 T_5^4 \mathbf{P} \tag{6.27}$$

The left side of the equation is known. On the right side, the result is independent from θ_4 because the production $T_5^4\mathbf{P}$ omits the corresponding terms. Using the program *Maxima*, the right side is calculated and simplified as follows:

$$\left(T_1^0\right)^{-1}\mathbf{COM}\,\mathbf{Q} = \begin{bmatrix} c_1(-111c_2s_3 - 10s_2 + 96c_2) - (111c_3 + 43)s_1 \\ s_1(-111c_2s_3 - 10s_2 + 96c_2) + c_1(111c_3 + 43) \\ 111s_2s_3 - 96s_2 - 10c_2 \\ 1 \end{bmatrix} = \mathbf{L} \tag{6.28}$$

where $c_n = \cos(\theta_n)$ and $s_n = \sin(\theta_n)$. The first three elements represent a vector from the first joint to the intersection of the last three joints axes. Due to the geometric model of figure 6.7 the length of this vector should only be a function of the knee joint. This is also verified using the following

substitutions:

$$\begin{aligned} s_1 &= \frac{2u}{1+u^2} \qquad c_1 = \frac{1-u^2}{1+u^2} \\ s_2 &= \frac{2v}{1+v^2} \qquad c_2 = \frac{1-v^2}{1+v^2} \\ s_3 &= \frac{2w}{1+w^2} \qquad c_3 = \frac{1-w^2}{1+w^2} \end{aligned}$$

$$\mathbf{L}^T\mathbf{L} = \frac{13941w^2 - 42624w + 33033}{w^2+1} \tag{6.29}$$

Equation 6.29 gives two candidates for the knee joint, just one of which is acceptable. It is then possible to calculate the second joint from the third component of $\mathbf{L}$ in equation 6.28. Finally the third joint value can be derived by substituting the results in either of the first two components of $\mathbf{L}$.

The next step is to calculate the rest of the joints. Equation 6.23 can be rewritten in the following form, so that all known terms are moved to the left side and the unknowns remain on the right side.

$$\left(T_4^3\right)^{-1}\left(T_3^2\right)^{-1}\left(T_2^1\right)^{-1}\left(T_1^0\right)^{-1}\mathbf{COM} = T_5^4T_6^5T_7^6 \tag{6.30}$$

The right side is expanded as follows:

$$T_5^4T_6^5T_7^6 = \begin{bmatrix} s_4s_6 + c_4c_5c_6 & c_4c_5s_6 - c_6s_4 & \mathbf{c_4s_5} = a & 65c_4s_5 \\ c_5c_6s_4 - c_4s_6 & c_5s_4s_6 + c_4c_6 & \mathbf{s_4s_5} = b & 65s_4s_5 \\ -\mathbf{c_6s_5} = d & -\mathbf{s_6s_5} = e & \mathbf{c_5} = c & 65c_5 - 111 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.31}$$

It is now possible to calculate the joint angles 4, 5 and 6.

$$\theta_5 = \arctan2(\sqrt{a^2+b^2}, c) \tag{6.32}$$

$$\theta_4 = \begin{cases} \arctan2(c,a) & s_5 > 0 \\ \arctan2(-c,-a) & else \end{cases} \tag{6.33}$$

$$\theta_6 = \begin{cases} \arctan2(e,d) & s_5 < 0 \\ \arctan2(-e,-d) & else \end{cases} \tag{6.34}$$

Chapter 7

Analysis and Stabilization of Dynamic Walker in Lateral Plane

7.1 Introduction

In this chapter I address the problem of stabilization of the bipedal robot in the lateral plane. My work followed an energy based approach, where the changes in the energy of the robot are considered. Some other researchers have also worked on energy based control of the bipedal walking. Asano et. al. [59] propose an effective gait-generation method, which imitates the energy behavior of a passive walker on a ramp. Franken et. al. [60] focus on the ankle joint and its effect on the energy level of the robot. A similar approach is also reported in [15] using a telescopic leg biped.

The chapter begins with an energy analysis of the biped model. Here the energy conversions of the model in single and double support phases are investigated. Different parameters are then suggested to affect the energy level of the robot. In the reminder of the chapter I propose controllers to stabilize the walking in the lateral plane based on the gained results. Control methods are then applied on the simulated platform and the results are presented.

7.2 Dynamic Walking vs. Static Walking

Dynamic walking is known as a kind of walking procedure with consideration of the impact of momentums and accelerations applied to the body elements. In other words one can define static walking as a procedure which achieves stable locomotion by neglecting the impact of dynamic compo-

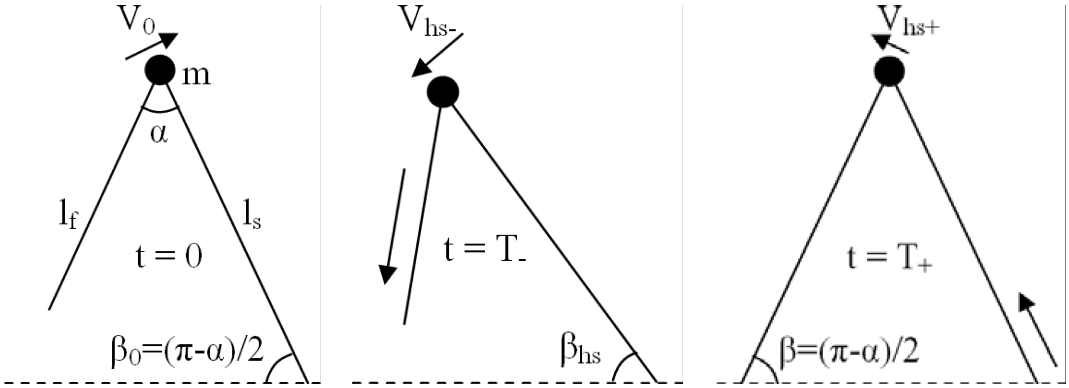
nents and uses only the static stability conditions. It is almost obvious that this assumption holds, as long as the biped makes considerably small movements. In addition, the joint drive mechanism must be strong enough to eliminate all momentums and hold the body as rigid as possible. Most of the designs based on static walking are urged to use over-powered actuators, which reduce the efficiency of the walking to a high extent. This approach was utilized by early researchers in the field of bipedal walking. My work in development of a control rule is focused on dynamic walking in this thesis.

7.3 Open Loop Dynamic Walking

One of the earliest approaches by bipedal walking researchers was to imitate human walking patterns and to try to find out to which extent these open loop motions were stable, respectively trying to increase the stability with trial and error. Actually, it is shown that there are many open loop gaits which can guarantee the stability of the biped even under a noticeable amount of perturbations [61, 62]. This is also the case for a majority of the participating robots in RoboCup humanoid league at the time this thesis is being written. Open loop dynamic walking can also be observed as the basis for development of closed loop walking methods. This approach tries first to create a stable open loop dynamic walking motion, then to control one ore more parameters of the system to improve the stability.

There are several known procedures to generate an open loop dynamic walking gait. The simplest method is to generate a static walking procedure by either key frame recording or direct calculation of the joint values, then to increase the play-back speed and respectively to try to improve the stability by manipulating the joint values. Another method is to record the trajectory of human walking and transfer the motion to a similarly constructed bipedal model.

Trial and error methods usually face serious difficulties during optimization of the gait by direct manipulation of the joints. There are seldom direct relations between the physical behavior of the system and the trajectories applied to the joint values. Often, several joints should be tuned at the same time with an eye kept on the consistency of the change. This is because of the complex kinematics of the robot. It is therefore suggested that generation and manipulation of the gait is done in more straight forward parameter spaces such as those discussed in section 6.



(a) swing phase (b) before heel strike (c) after heel strike

Figure 7.1: Energy analysis of the simplest walker.

7.4 Energy Analysis of the Biped

Energy analysis is a well known technique used to investigate bipedal walking [63, 59, 64]. The approach has several advantages to the other methods. It is simple to calculate and can describe many phenomena, which are rather hard to describe using conventional methods. In energy based methods, the whole energy of the system and its components, kinetic and potential energies, are focused. The key point in this concept is that the whole energy of the system is assumed to remain constant as long as there is no energy transfer with the outside world.

In energy analysis of dynamic walking it is assumed that the overall energy of the robot remains constant during a swing phase. This holds true for the simplest walking model described in 5.1 as the movement of mass-less legs requires no power and therefore applies no change to the energy of the system. Figure 7.1 shows the simplest walker in different phases. Energy of the system consists of two components: kinetic and potential energies. Because the whole mass of the system is assumed to be concentrated in one point, the components of the energy are simply calculated as follows.

$$E_p = mgh \tag{7.1}$$

$$E_k = \frac{1}{2}mv^2 \tag{7.2}$$

Possible energy exchanges occur either in a heel strike or in a deliberate change of the height of the center of mass. In the former event, the robot loses kinetic energy due to a sudden change in the velocity of the center of mass. The latter event can increase, as well as decrease, the overall energy of the system.

In the first step, it is assumed that heel strike is the only way the robot exchanges energy with the world. This means the length of the stance leg l_s is considered as constant. Also the model is

assumed to walk on an even surface. Equation 7.3 presents the overall energy of the model in terms of kinetic and potential energies during the swing phase, assuming v_0 to be the velocity of the robot as the walker reaches the vertical position.

$$E = E_p + E_k = mgl_s + \frac{1}{2}mv_0^2 \tag{7.3}$$

At the end of the step, and before the heel strike occurs, the height of the center of mass is reduced to $mgl_s \sin\beta_{hs}$. This conversion of potential energy into kinetic energy appears as an increment of the speed to the new value v_{hs-}. Assuming the steps to be all the same length or in other words $\beta_{hs} = cte.$, this speed is the same in value as that of the beginning of the step. In the following equation the energy is described before heel strike happens.

$$E = mgl_s \sin\beta_{hs} + \frac{1}{2}mv_{hs-}^2 \tag{7.4}$$

An important boundary condition for the model to be able to undergo a full step is that the minimum velocity v_0 remains positive. This holds:

$$v_{hs}^2 > 2gl_s(1 - \sin\beta_{hs}) \tag{7.5}$$

As long as l_s remains unchanged, a heel strike affects only the kinetic energy of the model. As discussed above, the effect is actually a loss of energy caused by the sudden change of the velocity vector, as the stance point of the robot flips from one feet to the other. Figure 7.2 shows the velocity vector of the COM before and after the heel strike as well as the path of the robot. The velocity vector before the heel strike, v_{hs-}, has a tangential component along the new path which remains unchanged. By contrast, the centrifugal component, which lies along the leg, will be eliminated due to the inelastic impact. The new energy is then calculated as follows:

$$v_{hs+} = v_{hs-}\cos\alpha \tag{7.6}$$

$$E_{k+} = E_{k-}\cos^2\alpha \tag{7.7}$$

$$E_{hs+} = mgl_s \sin\beta_{hs} + \frac{1}{2}mv_{hs-}^2\cos^2\alpha \tag{7.8}$$

where the lateral opening angle of the legs α is described as:

$$\alpha = \pi - 2\beta_{hs} \tag{7.9}$$

An interesting result derived from heel strike analysis is that the opening angle is a suitable parameter to stabilize bipedal walking by controlling the amount of energy of the robot. This approach is

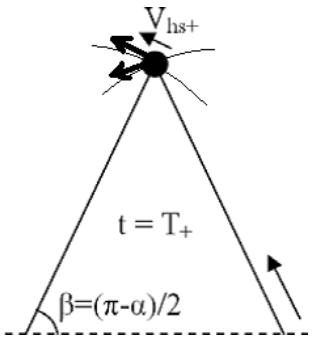


Figure 7.2: Biped Model in a Heel strike

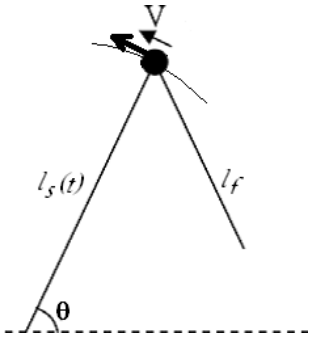


Figure 7.3: The Simplest Walker in Swing Phase

described in section 7.5. However it is quite obvious that the biped cannot walk continuously, unless the heel strike energy loss is compensated in some way. In the next few sections several methods are discussed to energize the biped.

7.4.1 Energizing the Walker Using Knees

As noted in the previous section, it is possible to influence the energy of the robot by varying the length of the stance leg. With this method, it is possible to both increase as well as decrease the energy of the model. It is just a matter of timing how it affects the energy of the model. In a kneed biped, the knee mechanism is used to reduce the length of the leg. Knees are usually used to provide foot clearance in human walking motion. In human walking, the knee of the stance leg remains always unbent. This holds no more as walking turns into running. In running, the knee of the stance foot bends not only to absorb the shock of the heel strike but also to re-energize the runner for the next step by raising the height of its COM.

Figure 7.3 shows the simplest walker in a swing phase. The length of the stance foot l_s is assumed to be variable. This addresses the classic control problem of variable length pendulum. To extract

the equation of the motion, the Lagrangian of the system should be calculated as follows:

$$L = E_k - E_p \tag{7.10}$$

$$E_k = \frac{1}{2}m(l^2\dot{\theta}^2 + \dot{l}^2) \tag{7.11}$$

$$E_p = mgl\sin\theta \tag{7.12}$$

using the Euler-Lagrange equation

$$\frac{d}{d_t}(\frac{\partial L}{\partial \dot{\theta}}) - \frac{\partial L}{\partial \theta} = 0 \tag{7.13}$$

$$m\left[l^2\ddot{\theta} + 2l\dot{l}\dot{\theta} + gl\cos(\theta)\right] = 0 \tag{7.14}$$

$$l\ddot{\theta} + 2\dot{l}\dot{\theta} + g\sin\theta = 0 \tag{7.15}$$

To study how leg length affects the motion, let's rephrase the equation of motion.

$$l\ddot{\theta} = -\left[g\sin\theta + \dot{l}\dot{\theta}\right] \tag{7.16}$$

If the change in the length is considerably small and the change in the length happens when the robot is near its vertical position, the term $\dot{l}\dot{\theta}$ can be neglected. So it is possible to assume that the change of the length has an ignorable effect on the kinetic energy level of the robot. Therefore the only change in the energy of the robot would be that of the potential energy.

The manner in which the stance leg changes are applied has a direct effect on how the robot is energized. It is also important to notice that different lengths of the stance and flying legs lead to different heel strike angles. It is however possible to select a scenario in which it is both possible to increase as well as decrease the energy of the system. In section 7.5.4 a controller is proposed to adjust the energy level of the robot by applying changes to stance leg length.

7.4.2 Energizing the Walker Using Ankle Joint Actuation

In the simplest walker, feet have point contact to the ground, whereas in human model feet form a supporting surface which can either be used to stabilize or to energize the walker. In many humanoid platforms, the foot plane is installed symmetrically so that it can be used to accelerate the robot in both directions, i.e. forward and backward. In this section the impact of the foot plane on the energy of the robot is explained.

In figure 7.4 the biped model is equipped with a driven foot plane. Assuming the drive to be a constant torque and the leg length remains constant, the energy given to or taken from the system in

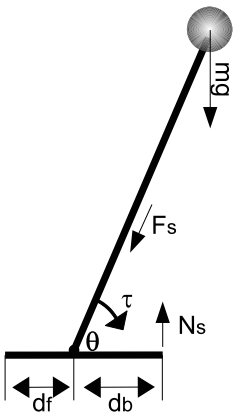


Figure 7.4: Biped model with foot plane

a step period is calculated as follows:

$$\Delta E = \Delta E_k = \int_{-\beta_{hs}}^{\beta_{hs}} \tau d\theta = 2\tau\beta_{hs} \tag{7.17}$$

Provided the step is symmetric:

$$\beta_{hs} = \frac{\pi - \alpha}{2} \tag{7.18}$$

Of course, the torque applied to the ankle joint is limited to some extents, one of the limiting factors is that the foot plane should remain on the ground. With higher torques the foot plane can flip around its edge and lift the whole model. Regarding figure 7.4, there are two torque components acting on the foot plane. These are the torque generated by the reaction force from the ground plane, τ_s, and the drive torque of the foot plane, τ. The foot plane does not flip over the back edge as long as the driving torque is less than the reaction torque. This results in:

$$\tau_s = F_s d_b \cos\theta = (mg\cos\theta - ml_s\dot{\theta}^2)d_b\cos\theta \tag{7.19}$$

A lower boundary can be similarly calculated in terms of d_f when the biped is decelerating using a negative torque to avoid flipping over the front edge of the foot plane. As seen in equation 7.19, the maximum applicable torque decreases as the model deviates from the vertical position. Experimental results have shown that in some cases the amount of energy given to the robot does not suffice for big steps.

It is now interesting to find out what happens if the torque exceeds the given limits and the model stands on the tip of the toes. This is basically similar to what occurs in human walking nearly at the

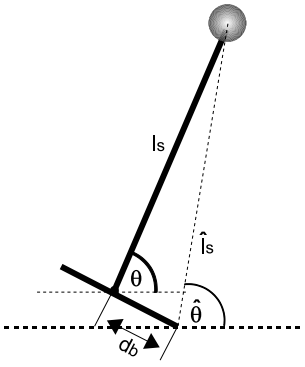


Figure 7.5: Plane foot biped model going on tip of the toe.

end of each step. Going on the tip of the toes is difficult to analyze in its general form, however it can be roughly separated into two different actions shown in figure 7.5. First, the stand point of the biped is transferred from within the foot plane to its front, which leads to sudden change of the direction of the velocity vector. As discussed in 7.4 this is a potential energy loss, but under the assumption of $d_f \ll l_s$, the loss of the energy can be neglected. Second, the effective length of the stance leg is now changed to $\hat{l_s}$. The effective leg length can be extended to $l_s + d_f$ by incrementing the angle of the ankle joint θ. The new stance angle is called $\hat{\theta}$. This value can also be estimated with θ under the given assumption. The model can now be reduced to the simplest walker model with variable stance leg length, as already discussed in section 7.4.1.

7.4.3 Passive Dynamic Walking

Passive dynamic walking is a bipedal walking method discussed in [1, 16, 47]. In this method the robot is fully unactuated. The dynamic motion is caused by the natural properties of the system. The required energy for walking is gained form the loss of overall potential energy, while the biped model walks down a shallow heel. Several aspects of this model including the stability are well studied in [12, 13, 2, 17, 65, 66]. Therefore the focus of this section is only on energy conversions in the model.

Figure 7.6 shows a passive dynamic walking model walking down a ramp with slope γ. Δh is the height difference of the center of mass of the model at the end of the step and can be calculated in terms of γ and α as well as l_s.

$$\Delta h = 2l_s \cos\frac{\alpha}{2} \sin\gamma \tag{7.20}$$

Therefore, the kinetic energy gain of the model during the step is:

$$\Delta E_k = -\Delta E_p = mg\Delta h = 2mgl_s \cos\frac{\alpha}{2}\sin\gamma \tag{7.21}$$

Assuming E_0 to be the kinetic energy of the model at the beginning of the step, the kinetic energy after heel strike can be calculated using equation 7.7 as follows:

$$E_{k+} = (E_0 + 2mgl_s \cos\frac{\alpha}{2}\sin\gamma)\cos^2\alpha \tag{7.22}$$

For the model to reach the steady state it is required that:

$$E_{k+} = E_0 \tag{7.23}$$

which implies

$$E_0 = \frac{2mgl_s \cos\frac{\alpha}{2}\sin\gamma\cos^2\alpha}{\sin^2\alpha} \tag{7.24}$$

To guarantee a complete step, minimum kinetic energy should not reach zero. At the beginning of the step, the model converts its kinetic energy into potential energy until it reaches the vertical position. The converted amount of energy can be calculated in terms of $\Delta h'$ as follows:

$$\Delta h' = l_s(1 - \cos(\frac{\alpha}{2} - \gamma)) \tag{7.25}$$

$$E_0 > mgl_s(1 - \cos(\frac{\alpha}{2} - \gamma)) \tag{7.26}$$

By substituting E_0 from equation 7.24 into equation 7.26, a general relation between step angle and ramp slope results:

$$2\cos\frac{\alpha}{2}\sin\gamma\cos^2\alpha > (1 - \cos(\frac{\alpha}{2} - \gamma))\sin^2\alpha \tag{7.27}$$

7.5 Closed Loop Lateral Stabilization of the Bipedal Walking

Different techniques are described in the last section to affect the energy of a walking biped. Overall energy of the system is a key parameter determining its stability. In this section I describe different control paradigms I developed based on energy feedback to provide lateral stability for the biped model.

I used the system presented in figure 7.7 to stabilize the process of walking. In order to control the energy of the system, it is necessary to measure its current energy level in some way. Referring to the biped model, kinetic and potential energies can be calculated in terms of stance angle and stance

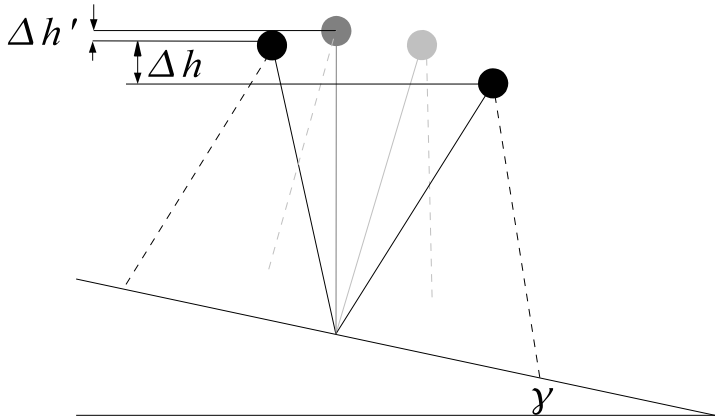


Figure 7.6: Passive Dynamic walking on a ramp

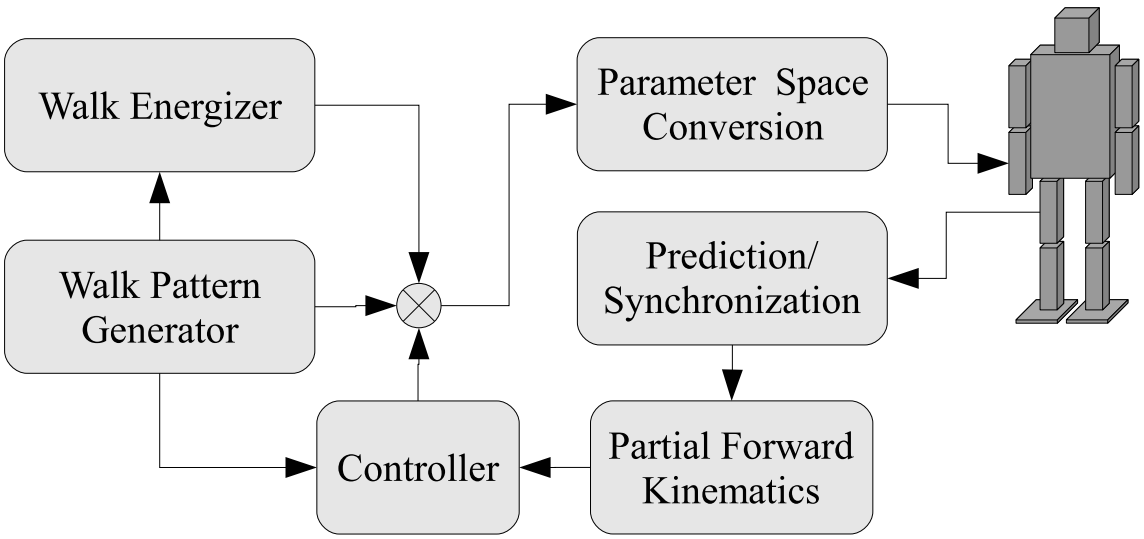


Figure 7.7: Block Diagram of Closed Loop Walk Controller

leg length as described in equations 7.11 and 7.12. Stance leg length is more or less insensitive to environmental effects and can be assumed to remain the same as it is set by the control software, whereas the stance angle is strongly variable. This parameter should therefore be measured using available sensors such as joint potentiometers or IMU modules. Partial solutions of forward kinematics are used to back-calculate the parameters from joint angle measurements. As there is usually an unavoidable delay in the measurements due to the data communication, a linear predictor can be used to compensate the delay.

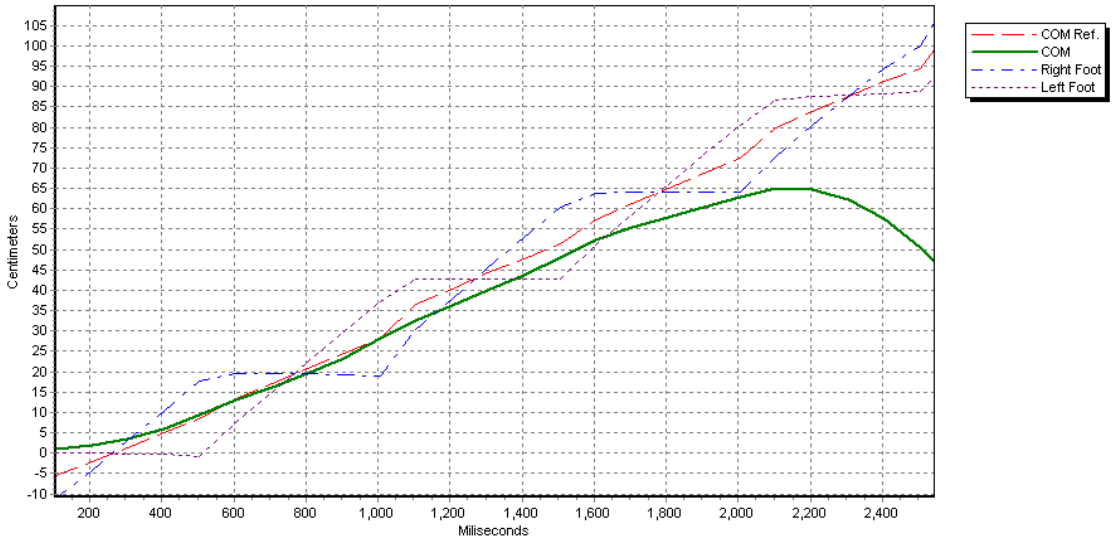
In the core of the system, *Walk Pattern Generator* computes the trajectory in parameter space as described in section 7.3. The result is then manipulated by two other modules. *Walk Energizer* applies one of the techniques described in sub-sections 7.4.1 to 7.4.3 to power the model and an additional *Controller* unit manipulates the walk pattern based on its control paradigm and the feedback it receives from the robot.

7.5.1 Stabilization Using Foot-Ground Contact Measurement

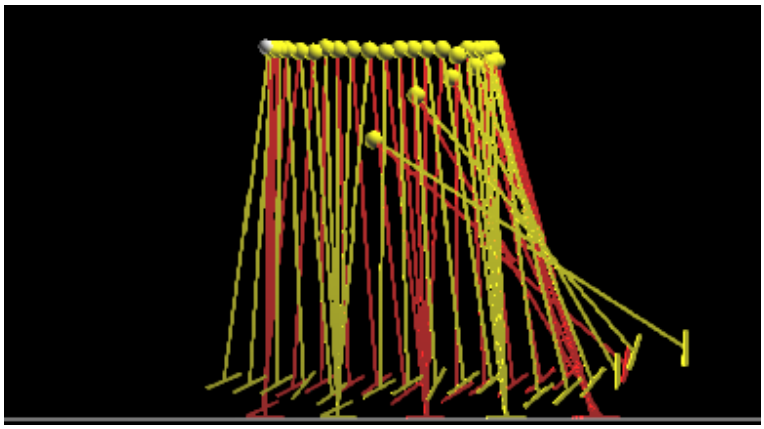
An often observed reason for destabilization of open loop dynamic walking is that the movement of the robot loses its synchronization to the dynamics of the body. The effect can be observed in the lateral plane when starting a new step while the last step is not fully finished yet, i.e. the robot is not yet completely supported by the swing leg. Loss of synchronization is a common reason for destabilization of the robot in frontal plane. An example of this effect is provided in figure 7.8 using the simulator. Figure 7.8a shows the position of the center of mass together with the positions of the feet. For reference, the average of the feet position is also shown in the diagram. The position of the center of mass first follows the reference more or less, but is however left behind after a few steps and finally the robot is destabilized. Figure 7.8b presents the visualization of the same process.

A solution to these problem can be to synchronize the timing of the open loop motion with the body movement using a kind of feedback. As the simplest approach, the ground contact of the swing foot can be used as the synchronization source. The beginning of a new step is delayed in this method until a heel strike occurs. In the opposite case, i.e. when the center of mass reaches the goal position earlier, the step is ended and a new step begins. Applying this procedure increases the stability of the system noticeably and reduces its sensitivity to perturbations.

It is observed that the system compensates a backward lag more successfully than a forward lean. It can therefore be useful to make a rapid swing movement and wait for both the step period to be elapsed and the desired stance angle to be reached. Simulated results show a considerable improvement.

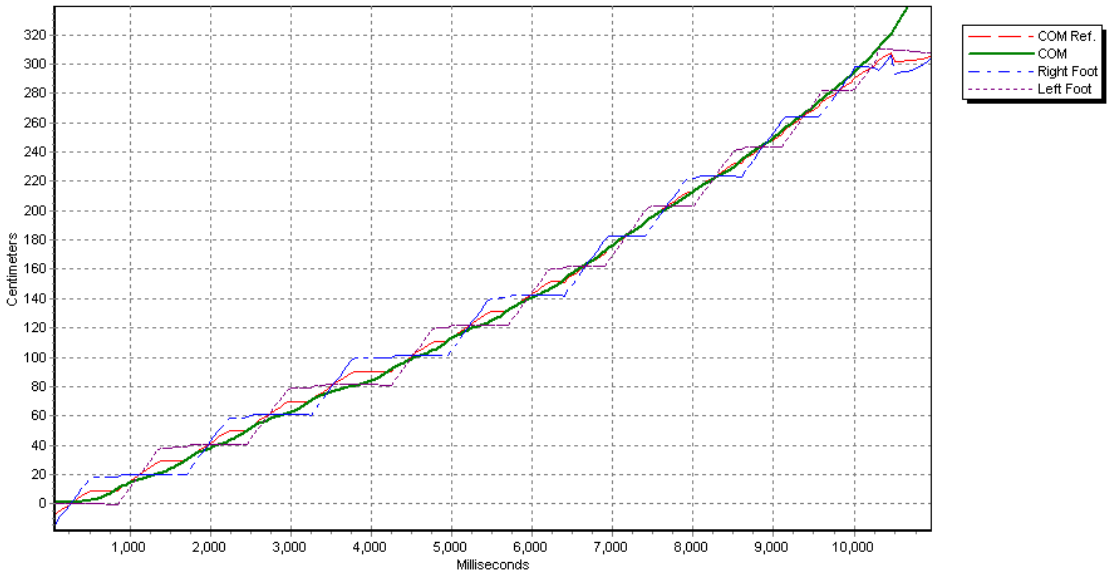


(a) Position of the center of mass, comparing the to positions of the feet along the walking direction. The COM is slower than the feet, causing the robot to fall down backwards.

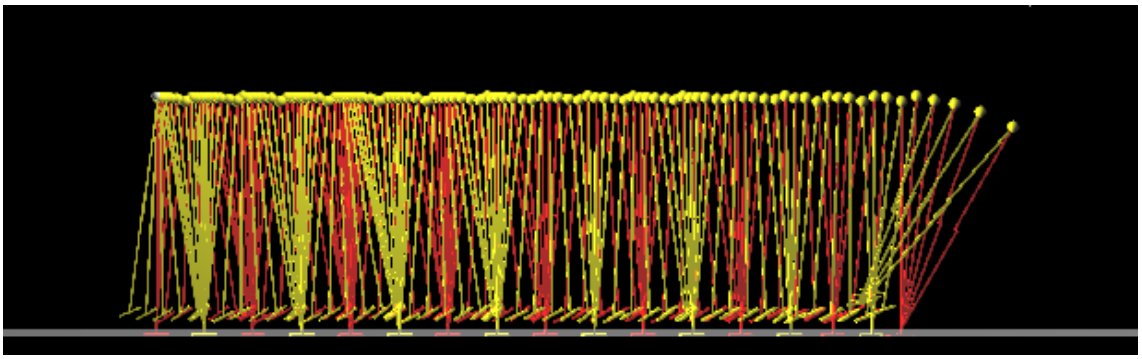


(b) Visualization of the results shows how the robot loses its synchronization.

Figure 7.8: Open Loop Dynamic Walking Often Loses the Synchronization



(a) COM position follows the reference value for a longer period, small perturbations are compensated.



(b) Visualization of the result shows a significant improvement of the stability.

Figure 7.9: Stability Improvement using Foot to Ground Contact Measurement

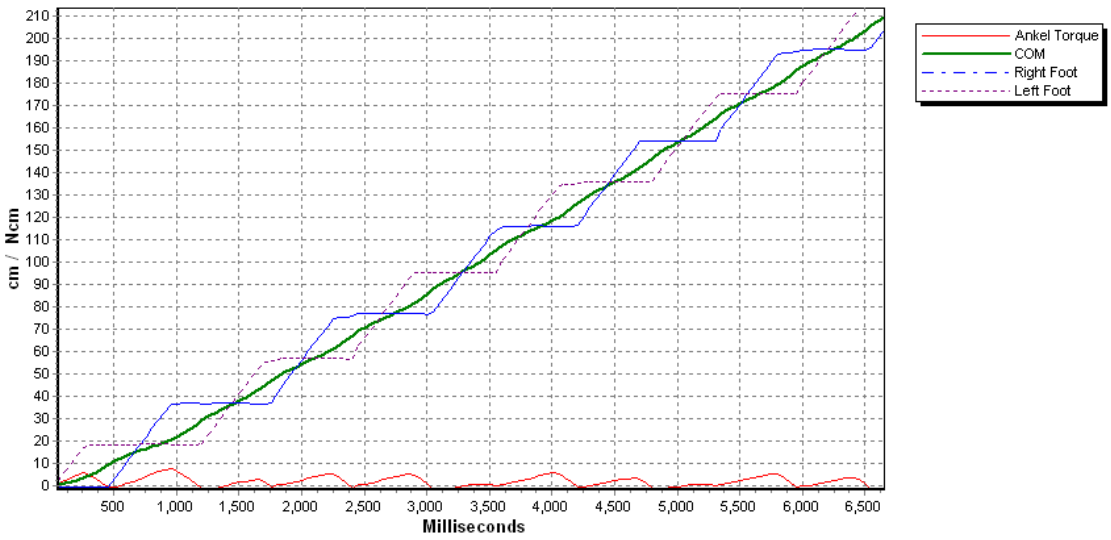


Figure 7.10: Ankle Joint Torque Control

7.5.2 Stabilization Using Ankle Joint Torque

As discussed in section 7.4.2, it is possible to affect the energy of the biped by actuating the foot plane. A simple P controller is applied to the open loop system of figure 7.9 upon stance angle feedback. The following is the relation of the controller,

$$\tau_s = k_p(\theta_{ref} - \theta_s) \tag{7.28}$$

where θ_{ref} is the reference angle rising linearly from $-\theta_{hs}$ to θ_{hs}. θ_s is the measured stance angle and k_p is the proportional control coefficient. The calculated torque τ_s is applied to the stance foot.

Figure 7.10 shows the simulated result of applying the controller. The system remains stable and can continue walking without falling down. The applied ankle joint torque is also visible in the diagram. It is positive most of the time and tries to recover the energy loss of the system due to heel strike.

Ankle joint method is a pretty straight forward approach, as there is a direct linear relation between the applied torque and the energy the biped gains. However, there are serious restrictions using this method. As also mentioned in section 7.4.2, there is a maximum limit for the applied torque which affects the stability margin of the controlled system. The method can therefore be combined with other stabilization methods for a better performance.

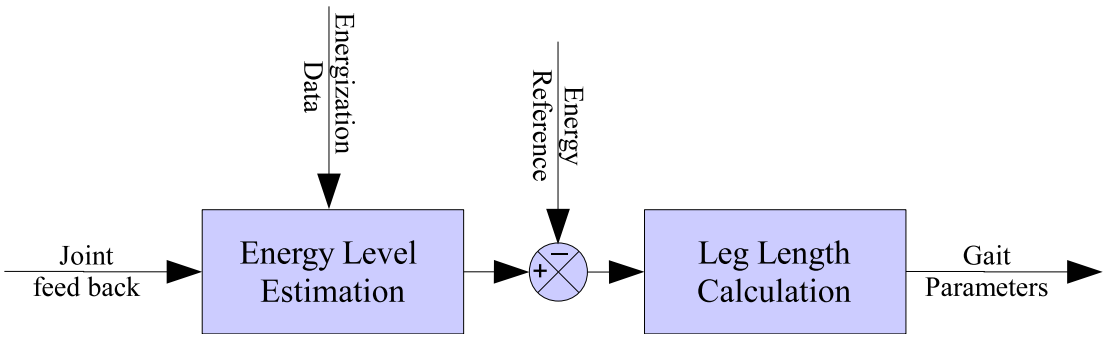


Figure 7.11: Block Diagram of the Energy-Feedback Step Length Controller.

7.5.3 Step Length Control

Step length control is based on the observed energy loss of the system due to the heel strike. It is summarized in equation 7.7. The stabilization method works in combination with an energizing method because it can only reduce the energy of the system.

Two approaches have been followed for stabilization of the bipedal walking using step length control. In the first, the average energy level of the robot is estimated using the available feedback. The step length is then adjusted to absorb the calculated amount of energy and stabilize the energy level of the system. A block diagram of the controller is shown in figure 7.11.

The second control method treats the system more or less similar to the classic control problem of an inverted pendulum on a cart. Actually, the problem of bipedal walking approaches the problem of the inverted pendulum on a cart when the step length approaches zero. In the latter problem, the goal is to keep the cart under the pendulum and respectively control the position/velocity of the combination to avoid it run away. The same goal is aimed in bipedal walking. Considering the analogy, the proposed solution is to keep the swing leg always symmetric to the stance leg or simply:

$$\alpha = \pi - 2\theta_s \tag{7.29}$$

Providing this condition, the system is marginally stabilized, however it is still required to have a mechanism to regulate the energy level of the system to avoid a divergence. As the method should anyhow be implemented in combination with an energizer, it is possible either to include energy regulation in the powering unit, or to apply it directly to the step length.

The controller shows a significant ability to recover from external perturbations as well as a wide dynamic range. Figure 7.12 shows the response of the simulated system to an external disturbance in form of a burst force. The biped remains stable, however it changes the walking speed and direction in response to the stimulation. A very similar behavior can be observed, when humans attempt to

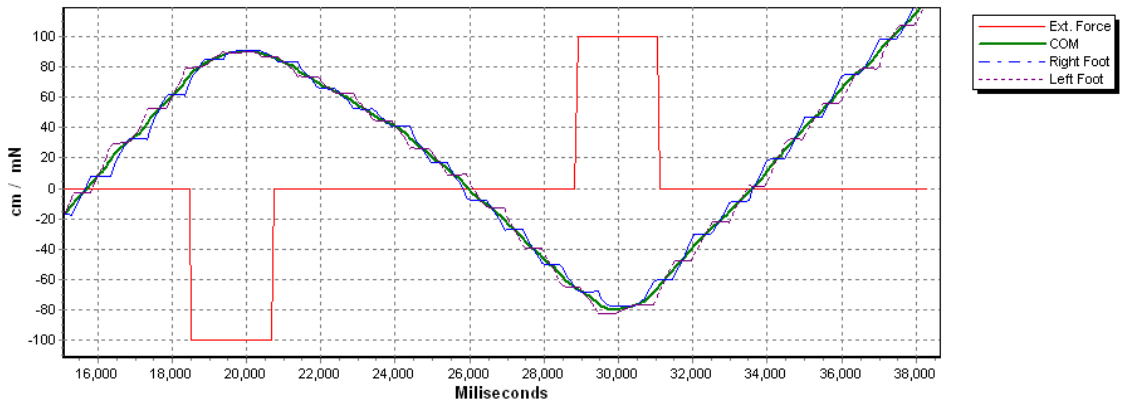


Figure 7.12: Step Length Control in Presence of an External Perturbation

compensate an external acceleration or force, i.e. standing in a bus as it spontaneously accelerates or decelerates.

Two important deficiencies can be observed when the proposed controller is used with a constant step time. First, it does not guarantee that the biped undergoes a full step, and second, it is possible to have asymmetric steps, i.e. a long step followed by a short one. Figures 7.13a and 7.13b show these shortcomings. Using the synchronization method described in section 7.5.1 minimizes both effects.

7.5.4 Stance Leg Length Control

Stance leg length is a suitable variable for actuation of the system. It has been shown in section 7.4.1 that it is possible to have full control over the energy of the robot by applying a proper trajectory to the length of the stance leg.

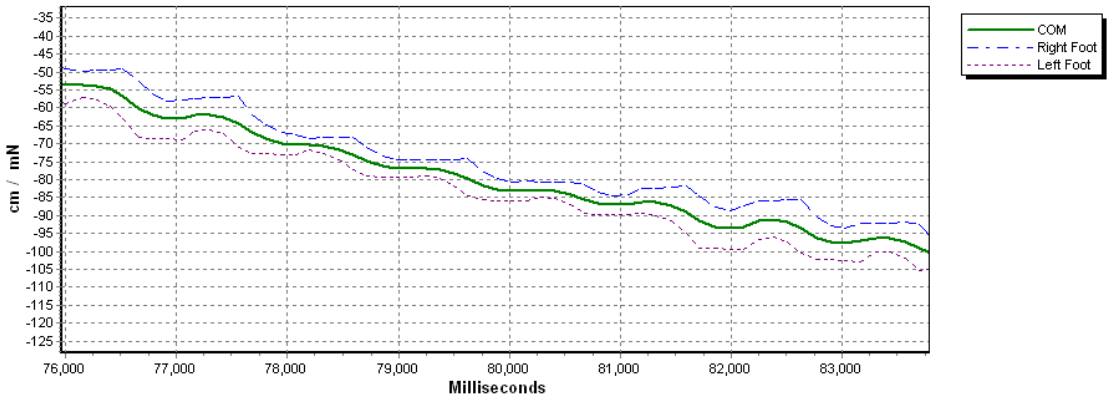
In the proposed control method, an error value is calculated using the feedback from stance angle. It should then be added to the default length of the stance leg. According to section 7.4.1 the effect of the leg length is a matter of timing. A simple approach is to apply the changes in a short period as the biped is staying almost vertically on its stance leg. For this purpose a sigmoid function is used which is defined as follows:

$$sig(t) = \frac{1}{1 + e^{-t}} \tag{7.30}$$

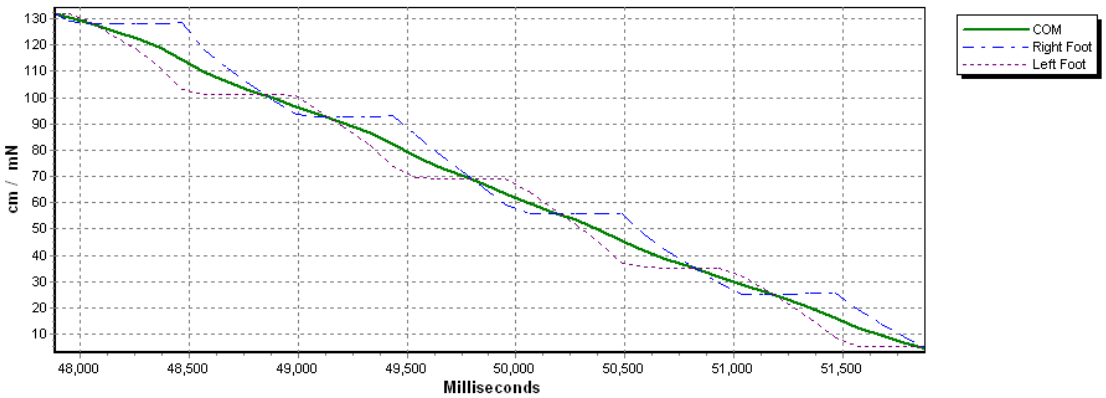
The resulting control rule is shown in equation 7.31.

$$l_s = l_0 + K_e(\theta_s - \theta_{ref})(1 - sig(K_\theta \theta_s)) \tag{7.31}$$

Figure 7.14a shows the motion curves. Variations in the length of the right leg during the stance phase are generated by the controller. The controller stabilizes the system, steps are equally distanced and

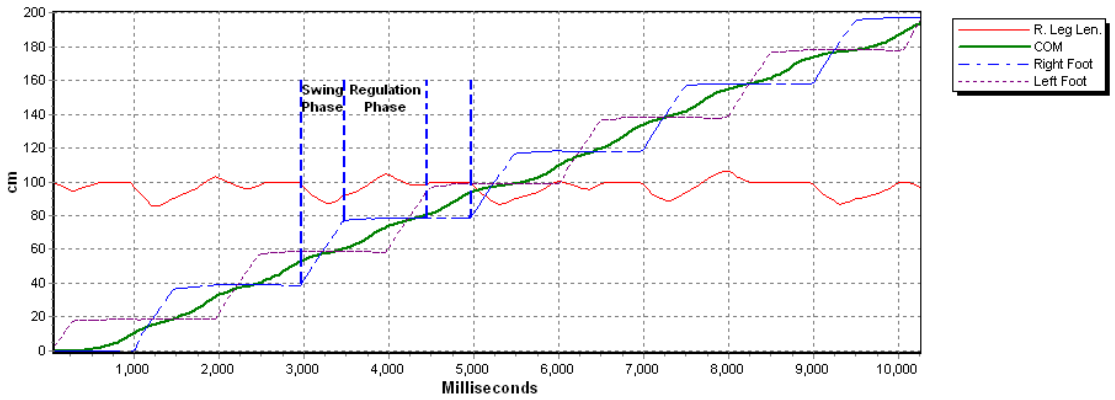


(a) The biped fails to undergo a full step in the given time. The order of the feet remains unchanged.

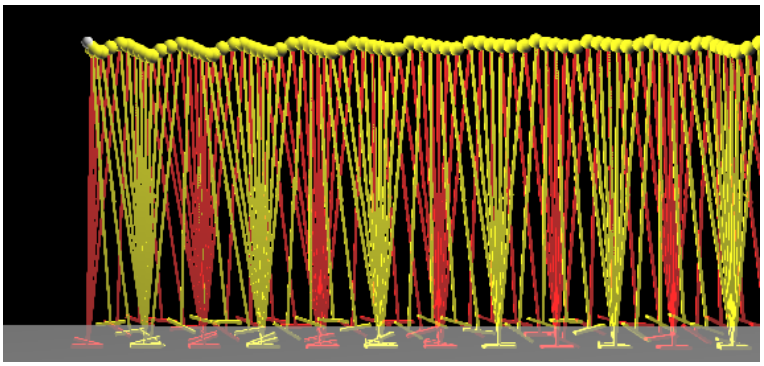


(b) Steps become asymmetric, i.e. a long step after a short step.

Figure 7.13: Two Deficiencies Observed in Step Length Controller with Constant Step Time



(a) Motion Curves: Right leg length is presented as one of the actuation variables. Regulation is possible in the indicated part of the step period for each leg.



(b) Visualization of the Motion: Note the variations in the height of the center of mass.

Figure 7.14: Simulation Results Using Stance Leg Length Control.

the system is mostly resistant to the disturbances. The motion is visualized in figure 7.14b. It can be observed how the controller varies the height of the center of mass to stabilize the system.

Chapter 8

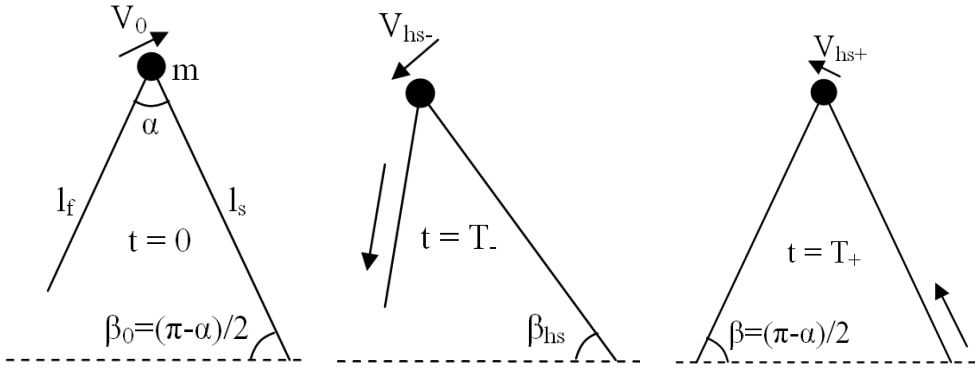
Analysis and Stabilization of Dynamic Walker Model in Frontal Plane

As reported in many of the published results, foot clearance is one of the key features in successful walking [12]. In some simulated 2D passive dynamic walking approaches, any premature ground contact known as foot scuffing is easily ignored [14, 67]. Some others have tried to make special test grounds which allow the swing foot to continue flying even if it goes below the ground surface [47, 64]. Adding knees or using telescopic feet can be useful to a great extent [12, 16], but synchronization of the foot clearance mechanism with the walking gait needs special solutions. In addition, in 3D form, side stability of the robot becomes a critical problem as the foot clearance increases. Some presented solutions to this problem are discussed in [2, 68].

In servoed walking one of the common solutions is to transfer the COM projection, or more generally the ZMP[1], to the stance foot support area before lifting the swing foot [69]. However, the calculation of these parameters and proper reaction times needs exact and reliable feedback data and control from the servos which cannot be expected from commercially available, low-cost products.

One of the foot clearance achievement methods is to excite the foot lifting mechanism in sequence with a harmonic function which results in the side vibration of the body. This method has been commonly used in RoboCup humanoid league, however a theoretical analysis of its behavior is still missing. In this section a mathematical model of the biped in frontal plane is suggested. The model is then energy-analyzed in "Swing" and "Heel Strike" phases. The steady state working point of the model is then calculated by applying steady state conditions. It is then shown that the steady state

[1] Zero Moment Point



(a) initial condition (b) before heel strike (c) after heel strike

Figure 8.1: Biped Model in Frontal Plane

working point is independent from foot clearance. Finally, the stability of the model is studied in the transient state and two methods are proposed to increase frontal stability of the physical platform[70].

8.1 Frontal Plane Analysis of the Bipedal Walking

Figure 8.1a shows the simplified biped model in frontal plane. The model is similar to the simplest walker described in [14] however it describes the behavior of the biped in frontal plane instead of lateral plane. The whole mass of the model m is assumed to be placed at the hip joint. Leg opening angle α stands for the distance between the feet contact points and remains constant. The swing leg is shortened to l_f for the period of T seconds, while the stance leg has the maximum length l_s. The role of the legs will be exchanged as the period is elapsed.

In swing phase the model behaves as an inverted pendulum hinged to the contact point of the stance foot to the ground. The swing phase starts with an initial angle β_0 equal to $\frac{\pi-\alpha}{2}$ (see figure 8.1a). The initial speed is normally positive, so that the stance angle β starts to increase. The center of mass decelerates and stops after a while if (and it is a necessary condition for stability of the model) it does not have enough energy to reach the vertical position. The center of mass swings back again until the period is elapsed. The stance angle at this moment is called β_{hs}. The state of the biped is presented in figure 8.1b.

In heel strike phase, the swing foot is re-extended to its original length and the stance foot is shortened. This usually leads to an impact between the swing foot and the ground surface, which can be simplified and modeled in two components of gaining and loosing energy. This simplification is possible since it can be shown that in steady state β_0 is very near to $\frac{\pi-\alpha}{2}$. Considering the inelastic

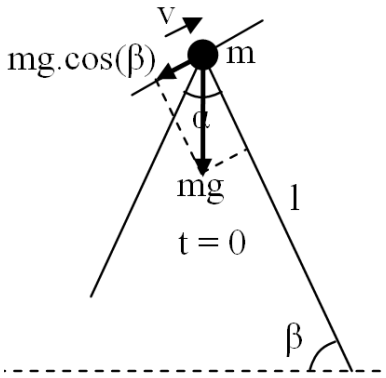


Figure 8.2: Biped Model in Flying Phase

impact between the heel and the ground surface, there is a loss of kinetic energy caused by the sudden change of the velocity vector of the center of mass. On the other hand the model gains potential energy due to the lifting of its center of mass. The state after heel strike is shown in figure 8.1c.

8.1.1 Analysis of the Swing Phase

Figure 8.2 shows the biped in the swing phase. As mentioned, the biped acts as an inverted pendulum in this phase. The model is similar to the one used for analyzing the lateral motion, however the frontal stance angle β is the angle between the leg and the ground. Assuming the stance leg length to remain constant, the equation of motion reduces to:

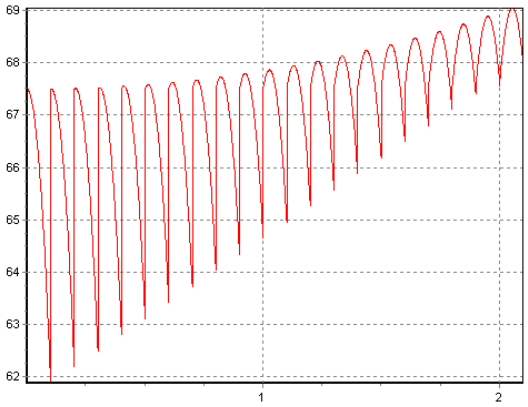
$$l_s\ddot{\beta} = g\cos\beta \tag{8.1}$$

where:

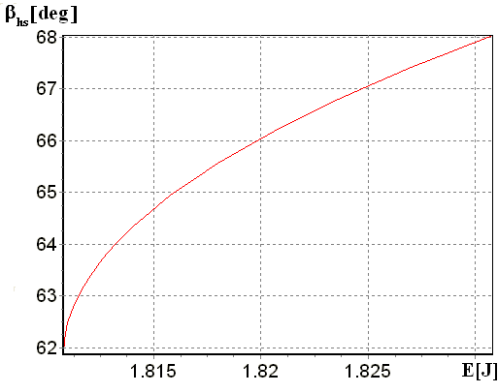
$$\beta_0 = \frac{\pi - \alpha}{2}, v_0 = v$$

The goal of analyzing the flying phase is to determine the value of β_{hs} as a function of the energy of the robot. The energy of the biped is a proper parameter showing the initial condition of the robot. As it remains constant during the flight, it can also be used in the analysis of the heel strike. The time of the flight is also constant and equal to the period of the leg length vibrations. To solve the differential equation, the normal numerical integration method over the period T is used. Figure 8.3a shows the simulation of several flying phases with different initial energies. In figure 8.3b β_{hs} is presented as a function of the energy of the model. The following values have been given to the fixed variables used in the calculations throughout the section:

$$m = 1Kg,\ l = 0.2m,\ T = 0.1s,\ \alpha = \frac{\pi}{4}rad$$



(a) Simulation of several flying phases with increasing initial energies.



(b) Heel strike angle as a function of the energy of the model. Angles are presented in degrees to increase the readability.

Figure 8.3: Analysis of the Flying Phase

8.1.2 Analysis of the Heel Strike Phase

In figure 8.1b and figure 8.1c the model is shown just before and after the heel strike. The changes in energy level of the model in the heel strike phase as discussed above can be considered separately over the kinetic and potential components of the energy of the biped. Equation 8.2 presents the energy of the model in terms of kinetic and potential energies in the initial state.

$$E = mgl_s \cos\frac{\alpha}{2} + \frac{1}{2}mv^2 \tag{8.2}$$

As the period time T is elapsed the height of the center of mass is reduced to $l_s \sin(\beta_{hs})$. The difference in the potential energy is converted into kinetic energy. The whole energy of the biped can be then presented as follows:

$$E = E_{hs-} = mgl_s \sin\beta_{hs} + mgl_s(\cos\frac{\alpha}{2} - \sin\beta_{hs}) + \frac{1}{2}mv^2 \tag{8.3}$$

The first term indicates the potential energy and the second and the third one show the kinetic energy of the system. Upon the occurrence of the heel strike, the potential energy of the COM will be increased to $mgl_s \cos\frac{\alpha}{2}$. On the other hand, due to substitution of the stand point, only the tangent component of the velocity vector remains unchanged. The centrifugal component will be eliminated due to the inelastic impact. The new energy can be shown as:

$$E_{hs+} = mgl_s \cos\frac{\alpha}{2} + (mgl_s(\cos\frac{\alpha}{2} - \sin\beta_{hs}) + \frac{1}{2}mv_0^2)\cos^2\alpha \tag{8.4}$$

Equation 8.4 can be simplified and rephrased in terms of E:

$$E_{hs+} = mgl_s \cos\frac{\alpha}{2} + (E - mgl_s \sin\beta_{hs})\cos^2\alpha \tag{8.5}$$

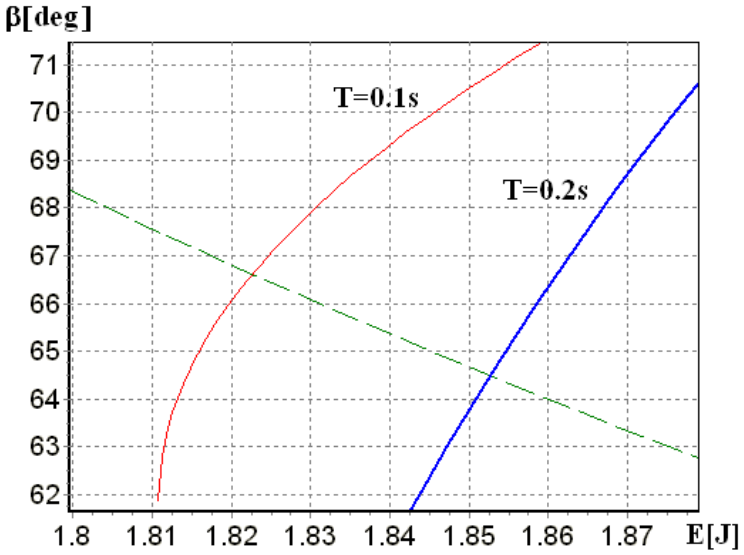


Figure 8.4: Steady State Working Point of the System: Steady state condition relating E to β_{hs} (dashed) intersected with the numerical solution of the flying phase equation (solid) for two different values of T

8.2 Steady State Condition and Working Point

In steady state, the energy of the biped should remain unchanged. This can be shown as:

$$E_{hs+} = E_{hs-} = E \tag{8.6}$$

By substituting equation 8.6 in equation 8.5, the steady state energy of the biped can be presented as the following function of the heel strike stance angle.

$$E(\beta_{hs}) = mgl_s \cos\frac{\alpha}{2} - (mgl_s \sin\beta_{hs}) \cos^2\alpha \sin^2\alpha \tag{8.7}$$

The function is shown in figure 8.4 together with the numerical solution of equation 8.1. The intersection point between these two functions determines the steady state working point. As it is observed, the length of the flying leg has so far no effect on the steady state working point. However, it will be further shown that this value is subject to a certain maximum. It means that the body vibration amplitude becomes independent from the foot clearance as the foot clearance exceeds a certain minimum.

8.3 Transient Analysis

So far it is shown that there exists a working point in which the amplitude of oscillations becomes independent from foot clearance. In order to prove that the biped can be self-stabilized, it is necessary to study the transient state. This section provides analysis of the system in transient state.

8.3.1 Delayed and Premature Heel Strike

When the initial velocity of the center of mass exceeds a certain value, the stance angle remains still higher than $\frac{\pi-\alpha}{2}$ at the end of the flying phase. The result is that the re-extended leg does not reach the ground level at the time of T. The heel strike is said to be "delayed" in this case. A delayed heel strike has only the energy loss component and therefore cannot occur in steady state. It is however a useful event in transient state, as it reduces the energy of the biped and stabilizes the process. An early heel strike can also happen in which the biped lands on its shortened flying foot due to the lack of energy. A premature heel strike is the only event in which the length of the flying foot plays a role. Actually, the length of the flying foot limits the minimum stance foot angle. As observed in figure 8.4, the less the stance foot angle becomes the higher amount of energy will be pumped in the system by the next heel strike. Therefore this parameter can be used to limit the energy of the biped to avoid instabilities as the biped goes through its transient condition. This method will be discussed further in the following sections.

8.3.2 Limit Cycle Stability and Transient Analysis of Frontal Plane Vibrations

As observed so far, the biped model seems to have an internal feedback loop which regulates the energy. If the biped has too much energy it takes longer for it to fly back and therefore the stance angle of the heel strike becomes larger which means less energy for the next flight and vice verse. However this internal feedback loop can also make the system diverge under certain circumstances. To avoid this, the amount of energy pumped in the system should be limited before the system reaches its steady state and also when it leaves the steady state under any disturbances. The peak-to-peak value of the stance angle can be used as a suitable indicator of the energy of the robot. It is then enough to calculate this value in each period and assign a proportion of it to the foot clearance. The value can also be low pass filtered to avoid rush changes of it. Applying this method in simulations shows a remarkable improvement in the convergence time and stability of the biped.

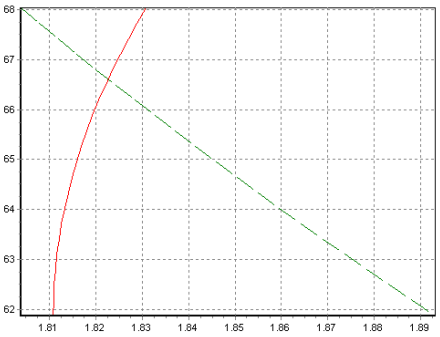
8.4 Results

8.4.1 Simulated Results

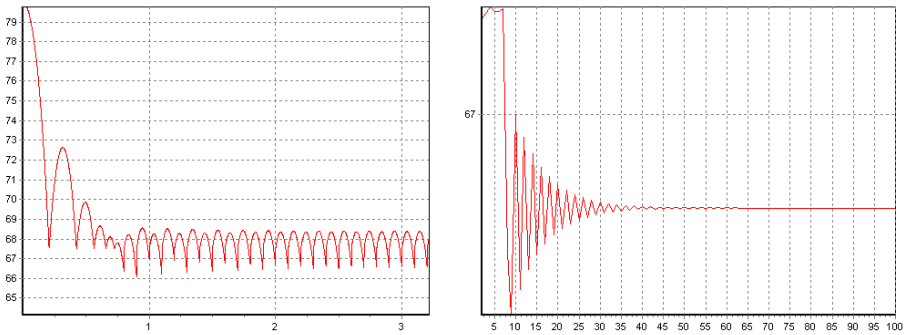
In order to test the convergence of the method and to compare the steady state working point with the one calculated in the previous analysis, the flying and the heel strike phases are simulated. The results are shown in figure 8.5 using two different leg opening angles. Other parameters remain unchanged. The model reaches its steady state after a few cycles of vibration and the steady state working point in both cases matches the values calculated in analysis.

8.4.2 Experimental Results

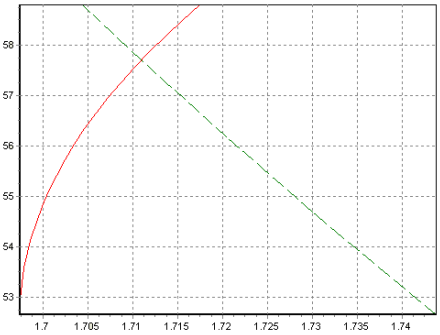
To be able to measure the stance angle of the robot, the stiffness of the servo motors belonging to the frontal plane movement of the feet is reduced to its minimum. This gives the possibility for the robot to act more similar to the point foot model. The stance foot area remains on the ground surface during the flying phase. The peak-to-peak value used in the stabilization method discussed in the previous section can be therefore extracted directly from the side servos. Figure 8.6 shows the roll value of the ankle joint of the robot together with its knee joint positions as an indicator of foot clearance. Recording starts from initial rest condition until the system reaches its steady state. The independence of the side vibration amplitude from the foot clearance can be observed in the figure. Vibration amplitude remains almost constant as foot clearance increases during the first five seconds.



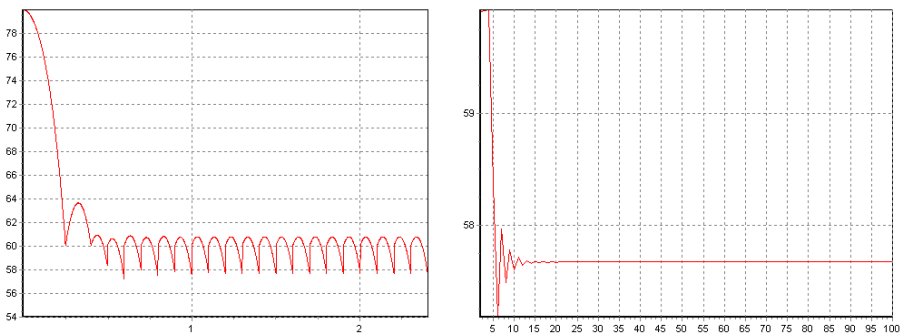
(a) Calculated working point for $\alpha = \frac{\pi}{4}$.



(b) Left: stance angle (β) and right: heel strike stance angle (β_{hs}) for several simulated steps with $\alpha = \frac{\pi}{4}$.



(c) Calculated working point for $\alpha = \frac{\pi}{3}$.



(d) Left: stance angle (β) and right: heel strike stance angle (β_{hs}) for several simulated steps with $\alpha = \frac{\pi}{3}$.

Figure 8.5: Simulated Results: the model converges to the calculated steady state working point.

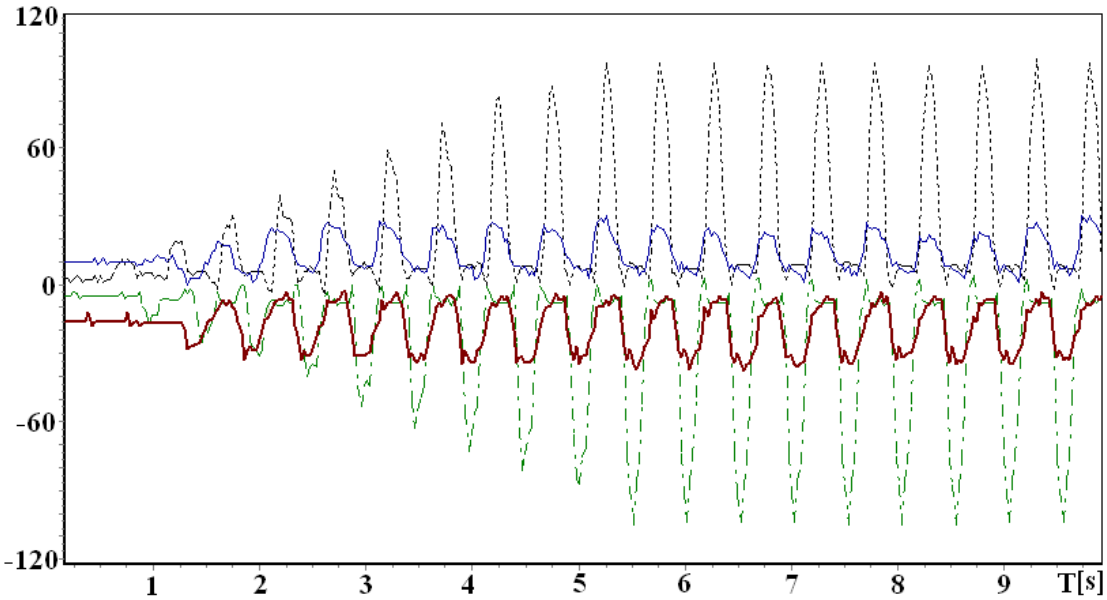


Figure 8.6: Experimental Results: Roll angle of the ankle joints (solid) indicates the vibration amplitude of the body. Knee joint position is used as an indicator of foot clearance (dashed).

Part IV

Computer Vision and Object Recognition

Based on the rules of the RoboCup humanoid league, robots are only allowed to use passive sensors, i.e. sensors without any kind of transmitters. Visual sensors get more important as it is possible to receive a much larger amount of information using visual sensors rather than other types of sensors. In order to simplify the computer vision, several considerations have been meet. First, all objects are coded with different colors so that the colors are easily distinguishable from each other and second, the lighting is held as constant and as homogeneous as possible. There are, however, many reasons, which make object recognition harder than it seems to be. For example, shadows make borders of objects hard to recognize. Irrelevant colors appear on the edges of objects due to limited pixel and color resolution of the cameras. As cameras can only be mounted in the head of the robots based on the rules, there is a considerable amount of vibration which causes blurring and deformation of the objects. In addition, with the improvement of computer vision techniques, RoboCup rules are updated so that the standard lighting and well distinguiashable colored marking is less guaranteed every year to finally reach the human soccer environment.

This part of the thesis reviews the developed vision system for RoboCup humanoid team FUmanoids and describes methods and techniques applied in different parts of the system. The part is organized as follows: In the first chapter, I survey the computer vision in the context of RoboCup humanoid league and discuss two basic approaches towards feature extraction from the captured images. In the next chapter I describe a vision module, I developed for the humanoid robots. The hardware and software of the module are described and together with two color based object recognition methods an on-the-fly algorithm is suggested for clustering the edges. The final chapter addresses the problem of shape based object detection in the field of RoboCup. I present two solutions. The first solution is an edge grouping method for extraction of the shape boundaries in the images. The second method is an entirely shape-based ball detection algorithm based on the histogram of edge orientations.

Chapter 9

Computer Vision in RoboCup Scenario

An important issue in development of computer vision for humanoid robots is the restricted ability of the robot to carry on-board equipment, such as computers and batteries. This limits the processing power available on-board the robot. As a noticeable part of the processing power is usually used for image processing algorithms[71], the focus of optimizations should lie on this part. Embedded vision, which can also be called modularization of computer vision, becomes currently more interesting in the field of robotics [72, 73], as well as in many other industrial fields [74, 75, 76]. A stand-alone hardware unit with a well-defined interface for adjustments, as well as for normal use, is encouraged. In addition, there is a wide range of robotics enthusiasts, for whom it is not possible to integrate vision in their robotic systems. These users have either not enough knowledge about computer vision, or they develop their systems based on processors not capable of connecting to a camera. For examples many hobby robotic projects are based on simple 8 bit microcontrollers. Providing a vision module is therefore a great advantage for educational robotics.

The task of computer vision in RoboCup humanoid league is divided into three main problems. These are ball detection, obstacle detection and self localization. For the first issue, the vision system should be capable of detecting a ball and distinguishing between the ball and any other objects with the similar color range outside the field. The known difficulties in this area are far balls which usually occupy just a few pixels and therefore cannot be detected robustly. Some works propose tracking methods for ball detection. This becomes impractical for humanoid robots as the position of the ball in the image becomes hardly predictable due to the rapid movements of the camera.

Obstacle detection has become more important as the number of robots has been increased, leading to more frequent collisions in the last years[77]. As the image captured by the camera of a humanoid robot is a 2 dimensional perspective projection of the 3D world, it usually becomes challenging to calculate the correct position of a robot by establishing a logical connection between its body parts. The problem becomes even more complicated if a robot gets partially occluded by another. In this case, the two objects are almost impossible to separate using a non stereo vision system.

The third problem in computer vision for humanoid robots is self localization, i.e. calculation of the pose (position and orientation) of the robot relative to the field. Many solutions exist to this problem, among which particle filtering is a widely used technique. Here, the known difficulties are perspective projection with almost unknown or just roughly known parameters, too few landmarks, which are also getting fewer due to RoboCup rule updates every year, and occlusion of the landmarks themselves. Self localization is very important, since it provides the basic knowledge for building a world model. Having a precise self localization, it is possible to predict the position of the ball or other objects in the image and optimize the vision system using tracking methods. Moreover, having a global world model, great improvements in game planning and cooperative behaviors can be achieved.

9.1 Color Based Object Recognition

Figure 4.3 presents a typical color based object recognition system. In the core of the system, a function is placed which interprets each color to a set of possible objects. To increase the performance, the function can be pre-stored in form of a lookup-table. It is sometimes useful to perform some preprocessing to enhance colors or to smoothen the surface of the objects. This should be done before the interpretation of the colors to the objects, as a majority of the image information is lost in this step. After interpretation, the pixels of the image contain only the information to which object(s) they can belong. Clustering or region growing is then used to find contiguous regions of the same color. In order to improve the results, it is necessary to remove quantization noise appearing in some areas of the objects. This is done using morphological operations. Figure 9.2 presents the intermediate results of color based object detection.

It is trivial that the quality of the detection is a direct result of how well the color to object interpretation is performed. The color interpretation should be capable of eliminating the effects of lighting and shadowing. Color interpretation can be assumed in its general form to be a transformation from the color space into a fuzzy vector of limited dimensions. Each entry in the target vector is the

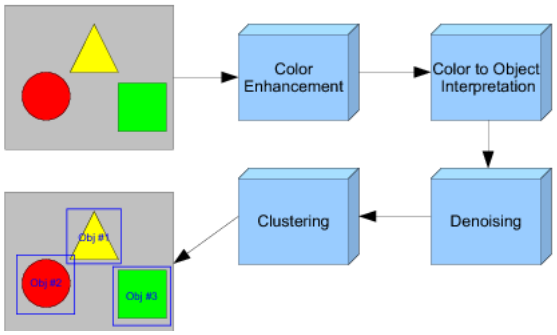


Figure 9.1: Block Diagram of a Typical Color-Based Object Recognition System

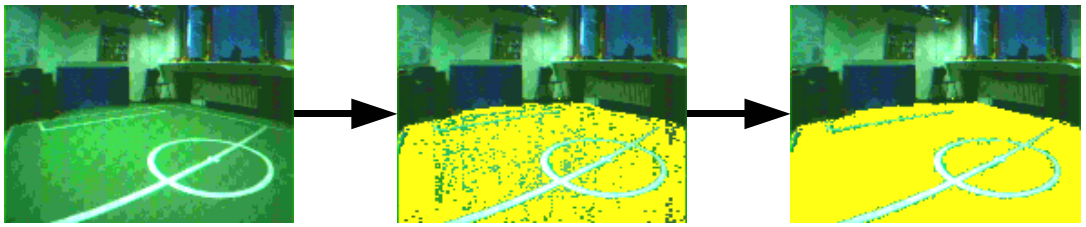
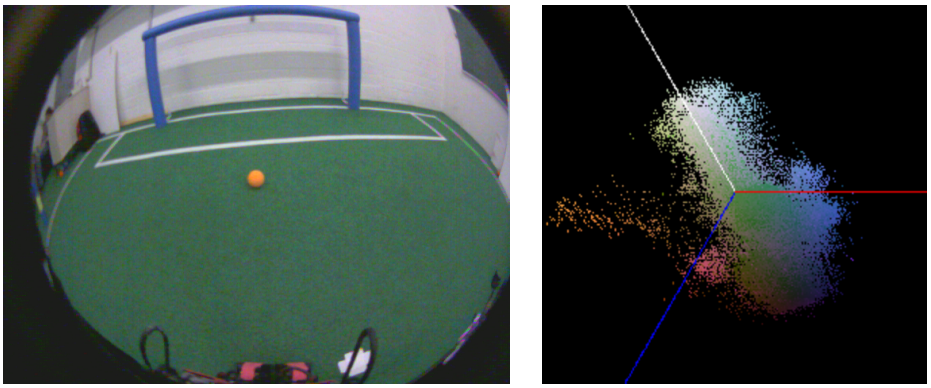


Figure 9.2: Intermediate Results of Color Based Object Recognition



(a) RoboCup Humanoid Field (b) Color Space Transformation

Figure 9.3: An Image From RoboCup Humanoid field and its transformation into color space.

membership of the given color to a certain object. Binary entries are often used to minimize the resources needed to realize the function. To understand the function of color interpretation, a typical image from RoboCup humanoid field together with its transformation into color space is presented in figure 9.3. A color image can be considered as a five-dimensional geometry with two position and three color components. The transformation of the image into color space is equal to the projection of this geometry into the space with only the 3 color dimensions. As seen in figure 9.3, colored objects convert into point distributions in the color space. There are different techniques to define a color interpretation. Two important approaches are analytical and numerical interpretations.

In analytical methods, the color space is reduced into a one dimensional space, so that the color masses remain still separable. A simple thresholding can then identify the target object. Color space reduction can be done in different ways. An example is to project the space among the intensity axis in a 2D space and take the angle value of the resulting polar coordinates. This is similar to the hue value in HSV color space. This can separate many colors, however white and black are not separable this way.

Numerical methods use a look up table, which contains an entry for each possible color. The look up table can be then filled either automatically using clustering methods, or manually by clicking areas in the image or in the color space transformation of the image. Numerical methods have higher flexibility compared to analytical ones, however they consume a noticeable amount of memory and can also cause memory bottlenecks in systems with limited memory bandwidth.

The goal of region growing is to detect colored blobs on the image. The question is how the regions should be characterized after being detected. The simplest way is to define a region by the smallest rectangle surrounding the region, called a bounding box. A bounding box is however not

enough to sufficiently determine what is inside it. More statistics such as number of pixels detected in the box, centroid of the region and directional distribution of the pixels can improve the definition.

9.2 Shape Based Object Recognition

In color based methods, geometrical information of the objects is ignored. This leads to a lack of robustness in some situations. There are different reasons for a malfunction in color based object recognition. Variability of the colors due to the changes in the lighting conditions or shadowing is the most common cause. In the case of using low resolution cameras or low quality optics, artifacts appear at the edges of objects in form of wrong colors, which can accidentally match with associated colors to other objects. This is very odd if, for example, a robot detects balls on the borders of the field lines.

It can also happen that the same color associated to an object appears outside the field in the audience. The last problem can be more or less solved by assigning a color set to the field and searching for the existence of both colors in a neighborhood. A precise color calibration takes too much effort as it still needs a high amount of manual interaction. Many of the automatic methods suffer from lack of reliability.

To decrease the dependency of object recognition methods to colors, it is necessary to use the geometrical information of the objects. Almost all objects to be recognized in RoboCup have fixed, defined shapes. Moreover, thanks to the use of standard lighting, unpatterned surfaces and well distanced colors, all objects have sharp visible edges which can be easily extracted without the need to use complex algorithms. Although it is theoretically possible to detect the objects regardless of their colors, it would save a rather great amount of time and work to use colors as well. This reduces the need to a perfect calibration of the colors and saves the effort toward that.

A requirement to almost all shape based object detection algorithms is the edge detection. There are some well known methods for that, among which Canny algorithm is the most famous one [78]. Almost all algorithms derive their results from the gradient of the image. Different heuristics are then applied to finalize the binary result assigning an edge/not edge property to each pixel. In RoboCup scenario, it is however enough to calculate the gradient magnitude and binarize it using thresholding. Figure 9.4 shows the result of applying this method on a typical image from the HL field. It is then the question of the object to be detected as to which techniques should be applied to the edge points to detect the object.

In the following chapters I describe two shape based methods I developed for the FUmanoids.

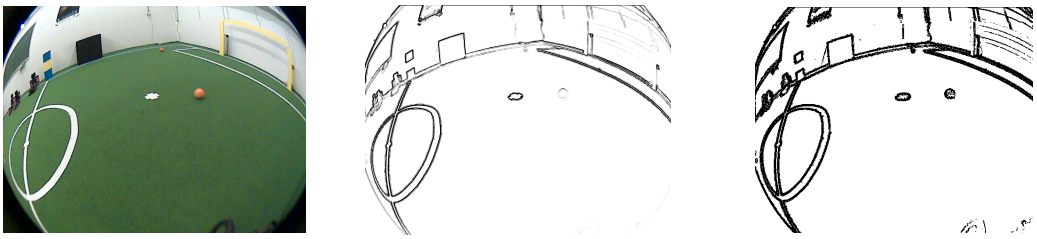


Figure 9.4: A typical image from the RoboCup Humanoid Field and its gradient magnitude before and after thresholding.

The first method is an algorithm for detection and clustering of the connected edges in the image. The second technique detects a ball in the image by examining the local distribution of the edge directions derived from the gradient image.

Chapter 10

Embedded Object Detection

In this section I describe the object recognition module, I developed for the humanoid robots. The goal of the development was to facilitate computer vision for low power microcontrollers. This was performed by assigning the pixel level processing, which is a resource consuming task, to a specific processor and generating intermediate results to be further processed by the main MCU of the robot. Several parameters were considered in the development of the system. These were modularity, simplicity and compactness of the results and low production costs. The software development got an enormous importance because the available resources were extremely limited.

Figure 10.1 shows a recent version of the module. The module is capable of detecting up to 16 colored regions at 19 frames per second. Up to 255 color sets can be defined in a 12 bit color look up table. The results can be accessed via full/half-duplex serial communication using different baud rates. In half duplex mode it is compatible with the TTL communication protocol from ROBOTIS and can therefore be connected to the same bus as the servomotors.

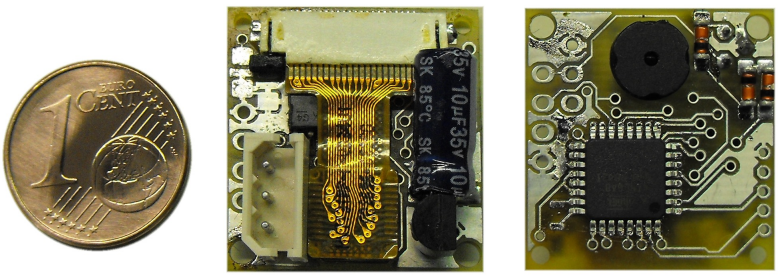


Figure 10.1: Developed Vision Module

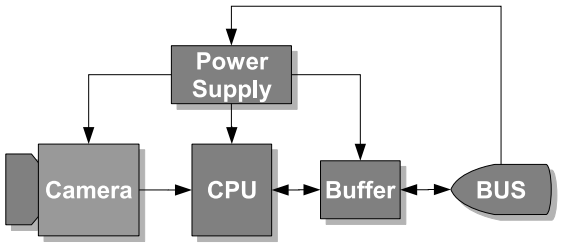


Figure 10.2: Block diagram of the vision module

Feature	Value
Max. Clock frequency	16 MHz
Architecture	8 bit RISC
FLASH	8 KB
RAM	1KB
Peripherals	UART, Timer, ADC, ...

Table 10.1: Features of ATMega8

10.1 Hardware Description

Figure 10.2 shows the block diagram of the vision module. The hardware is pretty simple. The imaging device is a CMOS camera module. It is directly connected to an Atmel ATMega8 which is responsible for image processing and communication. Table 10.1 gives a brief description of the features of the MCU. The camera is clocked from the same source as the micro controller and is therefore synchronized with it. Less handshaking is required using this method, which helps to save the processing power. The line and frame synchronization signals can be either polled or used to trigger an interrupt on the MCU. As the MCU does not contain enough resources for processing the whole color depth of the camera, only the most significant 4 bits of the pixel data are connected to the microcontroller. Besides of the parallel pixel data and synchronization signals, an I2C path connects the camera to the microcontroller, which is used to send configuration data such as camera parameters and image format to the camera. Technical details of the camera module can be found in [79].

10.2 Software Architecture

In the design of the vision module, the complexity of the system has been shifted as much as possible to the software side. This is, of course, to reduce the mass production costs. The software of the system is therefore somewhat complex. In figure 10.3 a block diagram of the developed software for the system is presented. The software interacts with two interfaces: The camera interface and the

serial communication interface. The camera interface provides 3 group of signals:

- Pixel data

  Connected to one of the 8-bit ports of the MCU and can be read in one clock cycle. The pixel data is delivered as a stream with YCrCb 4:2:2 format. This means for two pixels, four bytes are needed. This is, however, reduced to 2 bytes by ignoring the lower 4 bits of the data.

- Synchronization signals

  The camera chip produces 3 synchronization signals, one for each data byte, one per line and one at the beginning of each frame. The module uses only line and frame synchronization signals. Pixel synchronization is not necessary as the MCU and the camera chip receive their clock from a common source. Once the capture program is synchronized with the data stream, it remains synchronous at least for one line.

- I2C

  Camera chip has an internal register table, which includes a large number of image capture as well as image preprocessing parameters and settings. These include parameters such as exposure and white balance as well as image resolution and color format. The internal register table is accessible through an I2C port connected to the Module MCU.

The communication interface facilitates the connection of the module to the robot. It accepts several commands to invoke available image processing algorithms, access the results of the image processing and to get and set the camera settings. Additionally, accessing the built-in boot loader is also done through this interface. The function of the boot loader is to store calibration data such as the color look up table in the flash memory of the MCU. It is however limited so that the block used by the firmware cannot be accessed.

The heart of the image processor is the READ LINE function. It waits for the beginning of a line, then reads the pixel data of the line from the camera interface and stores it in the line buffer. The image format used by the module is YCrCb 4:2:2. The data stream is therefore "Y Cr Y Cb". To have the complete color information of a pixel, it is also necessary to read the neighbor pixel. The 12 bit color code of each pixel can be used to address the color look up table, which occupies half of the flash memory of the MCU. The line data is then passed to the image processing algorithm. Different image processing algorithms are developed for the module, which are described in the next sections.

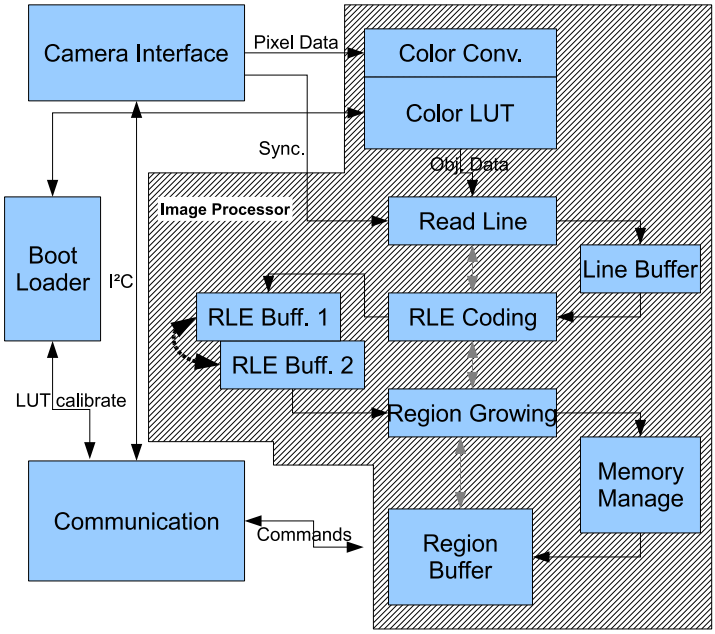


Figure 10.3: Block diagram of the software of the vision module

10.2.1 Color Based Region Growing Algorithm

The region growing method described in this section is based on the work described in [80, 81]. A summary of the method is presented here, followed by the improvements applied to the algorithm. After translation of the pixel colors into associated object codes, successive pixels of the same object are grouped together in the form of a RUN and stored in the RLE BUFFER. RLE BUFFER contains two sub buffers, which are toggled every line. The information regarding the previous scan line of the image is required for region growing. A RUN contains the following information:

1. position of the run in the line
2. number of pixels
3. associated object
4. pointer to the region

The region growing algorithm scans the runs of the current line, compares them to the ones from the last line, finds connected regions and updates runs as well as the entries in the REGION BUFFER. Entries of the region buffer are structures of the following fields:

1. father region

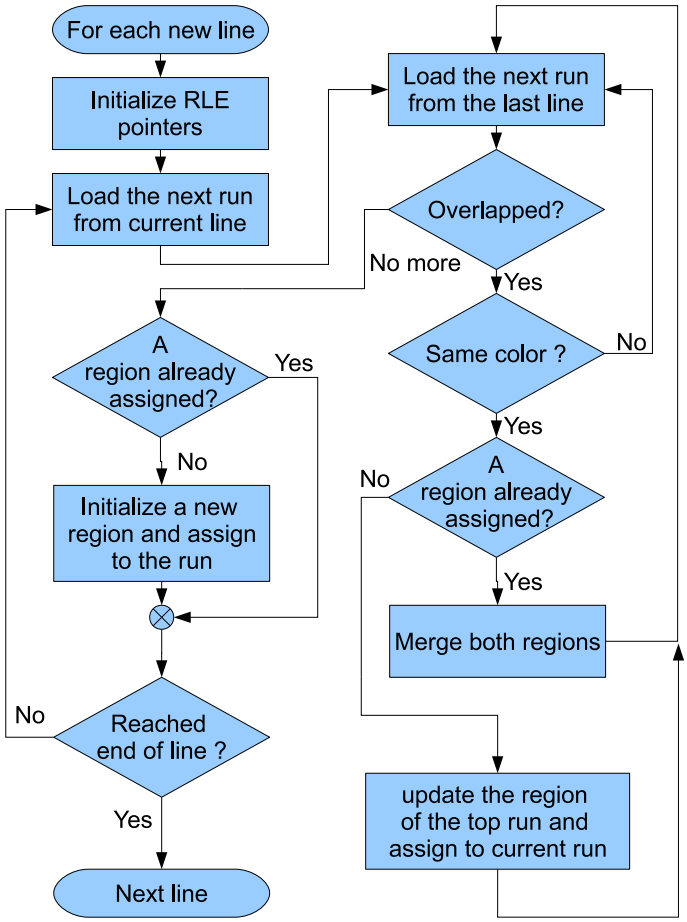


Figure 10.4: On-line region growing algorithm used in the module

2. associated object
3. number of pixels
4. sum of x values
5. sum of y values
6. bounding box

The region growing algorithm applied in the module is presented in figure 10.4. It is based on a simple concept, however the implementation has become a bit complicated. Conventional region growing techniques cannot be applied, because due to the limited memory of the MCU, no random access to the image data is possible. The algorithm tries to connect overlapping runs of the same

color together. Difficulties appear when two or more regions built separately get finally connected by a run being overlapped by these regions. In this case, the regions should be merged together. This leads to a highly dynamic use of the memory considering the limited available memory of the MCU. The memory management problem is solved using three techniques. First, a pointer field has been added to the region structure called FATHER REGION. Initially the father of each region points to its own. However, for each access to the region the father region is accessed instead of the child. Upon recognition of a multiple overlap, the father of the first region is set to point to the second one. This is called a LOGICAL MERGE. The PHYSICAL MERGE takes place after the processing of the line is finished.

The second technique is the use of a buffer of pointers, pointing to the unused regions. It can either be a FIFO or a FILO. Initially, all available regions are included in the buffer. Upon the requirement of a new region, it is popped out of the buffer. Regions modified during the process are marked. Any unmarked region is isolated from the rest of the image. It can therefore be taken out of the process and stored in the final results. The region becomes free again and can be pushed back in the buffer. In addition, all regions are scanned after the line is processed. Regions with a different father than themselves are merged in their father regions and become free again.

10.2.2 Image Griding Algorithm

The region growing algorithm discussed in the last section is capable of detecting and localizing convex and concentrated objects such as the ball or the goals. An indispensable amount of information can be extracted from the field lines, which cannot be detected perfectly using this algorithm. This information is required for robust self-localization. To solve this problem, an online algorithm was developed, which compressed the acquired image into a considerably smaller grid of found objects. In this method the detection rate is as high as the previously described method, however the position accuracy is reduced. In contrast, the geometrical information of the found objects does not get lost.

The image griding algorithm seems to be pretty simple however there are several problems to struggle with. The first problem is the limited available RAM. To have an acceptable grid resolution, a minimum dimension of 32x24 cells is required. Considering the whole data of each cell to be stored in a single byte, this needs at least 768 bytes, which is 75% of the available memory of the MCU. A cell in the grid should at least provide the information about which object(s) have been detected inside the cell. Not less important is also the number of pixels of the given object as it is possible to suppress the noise by thresholding this value. Due to the limited memory, there are two possible approaches. The available byte for each cell can be divided into 2 fields of each 4 bits. It is then either

	Priority	Foreground	Background
highest	1	Red	Green
	2	Blue	White
	3	Yellow	Black
	4	Cyan	Magenta
	5	Magenta	Cyan
	6	Black	Yellow
	7	White	Blue
lowest	8	Green	Red

Table 10.2: Priorities assigned to the objects for griding.

(a)

BG	Green
FG	Black
ignore	White

(b)

BG	Green
ignore	Black
FG	Orange

(c)

BG	Green
FG	Blue
ignore	White

Figure 10.5: Examples of appearing more than two objects in a cell

possible to store one object code and its number of pixels, or two object codes. The second approach has the advantage of storing a foreground and a background object. This can be used to filter the objects outside the field, by verifying the background object to be the field.

The other challenge in the development of the algorithm is finding a solution for the case when several objects appear within a cell. A known solution is to assign priorities to the objects so that minimum information is lost due to lack of memory. In figure 10.5 some examples of this case are presented. The assigned priorities are different for foreground objects and background objects. The highest priority for the foreground objects is assigned to the ball because it cannot be ignored in any case. Table 10.2 lists the priorities assigned to the objects.

Figure 10.6 shows the algorithm. A local history containing the found objects and the number of pixels of each color found in each cell is calculated for 5 consecutive lines. Finally a row of the grid is updated after denoising and identifying the highest priority foreground and background objects. The function is repeated until the end of the frame is reached. The image processing function produces 768 bytes of data for each frame. Each cell of the 32 x 24 grid summarizes 25 pixels of the image in one byte containing the highest priority foreground and background object found in the cell. The

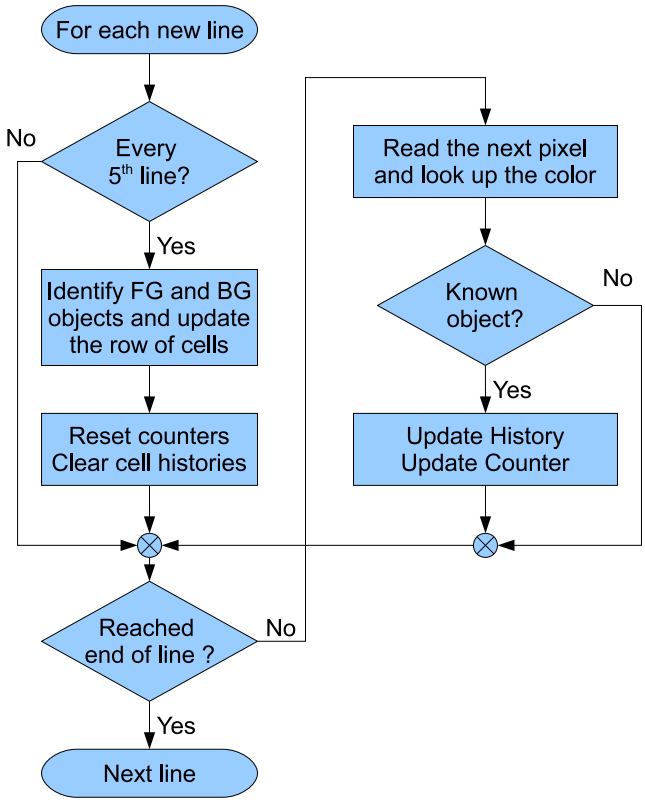


Figure 10.6: On-line Image Griding Algorithm

results of this algorithm are used for self localization.

10.2.3 On-Line Edge Clustering Algorithm

In contrast to images captured from natural scenes, images from the RoboCup field contain clear edges, most of which are straight. Figure 9.4 shows an example. An algorithm to find and group straight edges can extract a majority of the information needed for object recognition and self localization. Such an algorithm increases the reliability of object recognition by reducing the need for well defined colors. This section describes an online algorithm developed for the vision module to detect and cluster straight edges for further use in object recognition.

Edge detection is the first step in shape based object recognition. There are several well known methods to detect edges. Regarding existing hardware limitations and the on-line nature of the algorithm, methods are applicable which require a local and sequential access to the image data. There is a restricted possibility of buffering the image data. Large kernels should therefore be avoided

for providing image gradients. Robert's cross operation [18] described in equations 10.1 to 10.4 is suited for this purpose. The kernel used in the operation is 2x2 and therefore only one line has to be buffered. Moreover, the operation is very simple and can be performed in a few clock cycles. As an edge is not only the result of a sudden intensity change but also a color change, all 3 channels should be used for the calculation.

$$G_x : \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{10.1}$$

$$G_y : \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{10.2}$$

$$|G| = \sqrt{G_x^2 + G_y^2} \text{ or } |G| = |G_x| + |G_y| \tag{10.3}$$

$$\theta = \arctan(G_x/G_y) + \frac{\pi}{4} \tag{10.4}$$

Applying Robert's cross operation, the magnitude and direction of the gradient vector can be determined for each pixel. To increase the performance, Manhattan distance is used instead of euclidean distance and a look up table is implemented to accelerate direction calculation. Edge points are detected by thresholding the magnitude of the gradient vector. The direction component is used to group the pixels. In the next step, the edge points should be grouped together. As there is no guarantee for the thresholded edges to be only one pixel thick, it may lead to multiple detection of the same segment. To avoid this effect, edge points are grouped not only along the edge direction, but also orthogonal to it.

The next step is similar to the one used in the region growing algorithm. Here the edge points belonging to a line segment should be grouped together. Because the algorithm has only access to one line of the image data at a time, grouping should be first done for the horizontal neighborhood. A similar run length coding is used for this goal, however the condition *same object* is replaced with the *similar direction*. As the direction value is not as deterministic as the object code and is subject to deviations and measurement errors, a range check is used instead of a direct value comparison. The runs can then be compared to the ones from the upper image line to check for a connection. A connection is detected upon a geometrical neighborhood with a direction match.

Theoretically it is enough to store both end coordinates of a detected line segment. For optimization purposes, three other fields are also added to the object. These are normalized direction, predicted position of the next point and the number of pixels joined. The normalized direction is the average direction of the first few runs of the segment. Averaging is needed as the direction of a single

point does not have enough accuracy to be used for the whole line. On the other hand, averaging the direction for the whole length of the line causes the wrong detection of a curve as a single line segment. A direction match is checked in two steps. First by comparing the direction of the run with the one of the segment and second by checking if the predicted next point of the segment lies close enough to the borders of the run.

Upon joining a run to the segment, the parameters of the segment should be updated. The update procedure for the end coordinates of the line segment is a bit different from the one used for the bounding box in the region growing algorithm. There, the four values (min_x, min_y, max_x and max_y) could be updated separately. In contrast, the coordinates are updated in the current algorithm in couples. In other words, only one component of each coordinate couple is updated as a maximum or minimum which is called the *master* component. The second component stores the position in which the maximum or minimum occurs. The direction of the segment determines which component becomes the master. The master component is the x value for almost horizontal lines, and the y value for almost verticals ones.

Chapter 11

Shape Based Object Detection

As noted at the beginning of this part, shape based methods are often more reliable than color based ones. This can, however, only be achieved with the cost of more processing power. In this chapter some shape based algorithms developed for object recognition of the FUmanoid robots are described. The developed methods can be used separately, or in some cases, in conjunction with the color based methods to increase the reliability of the results.

In the first section I present an optimized method for edge detection and grouping, based on which the form-color based object detection is implemented. The method summarizes the form information of the image using the so-called *griding*. The edge structure is then extracted from the information available from the last stage. Finally, the color information is additionally used to recognize different objects.

In the next section I describe an algorithm I proposed to detect the ball only based on its form. The idea is inspired from *Histogram of oriented Gradients*[1], introduced by Dalal and Triggs in 2005 [82]. The algorithm was originally focused on the problem of pedestrian detection in static images. Today it is expanded to other objects such as animals and vehicles and also other media like video streams. The method is of great importance as it is only based on the shape data of the image rather than other more environment-dependent information like brightness and color. The presented method extends the idea using integral images and an overlapped binary search.

[1] HOG

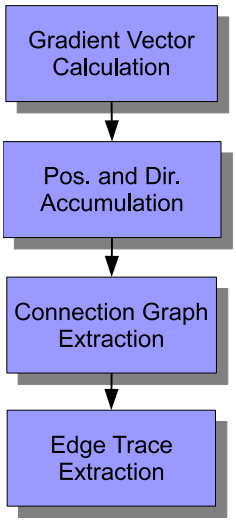


Figure 11.1: Gradient Vector Griding

11.1 Object Detection using Gradient Vector Griding [2]

This section describes a method developed to derive geometrical data from an image. The method is performance optimized for RoboCup scenario and is capable of clustering connected edges up to an adjustable curvature. The method is used in RoboCup 2010 as the low level stage of the object recognition module. Geometric results are then reinforced by adding color information and finally used for object detection. This technique facilitates a more precise self localization by increasing the number of features.

Figure 11.1 shows the basic implementation of *GVG*. The procedure begins with calculating one or two main gradient directions and positions called *edge representers*[3] for each cell in a rough grid on the image. This is unlike *HOG*. There, a histogram of orientations is extracted for each cell. Each set of position and orientation in *GVG* determines an edge passage through the cell. The next stage finds connected edges passing through neighbor cells. In the third pass, complete edge traces are extracted out of the cell connectivity graph.

11.1.1 Gradient Vector Calculation

Like every shape based object recognition method, the procedure begins with the calculation of the gradient vector for each pixel in the image. The two components of gradient vector are the result of differentiating the image in horizontal and vertical directions. In the case of Robert's cross operator

[2] **GVG**
[3] *ER*

described in equations 10.1 to 10.4, these are the result of diagonal differentiating. Sobel operators are less sensitive to noise, but however due to accumulating the algorithm is pretty noise tolerant. There are two important points to consider while calculating the gradient vector for *GVG*. First, it is necessary to calculate the direction in a -180° to 180° range, i.e. using the function arctan 2(y,x) instead of arctan(y/x), if the vector is presented in polar form. This prerequisite is discussed in detail in the next section. The second point is that the gradient direction is perpendicular to edge direction. Both directions are needed in the algorithm. The following convention is used in the algorithm to convert gradient vector $\mathbf{V}$ to edge vector $\mathbf{E}$, which is a simple 90° rotation.

$$\mathbf{E} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{V} \tag{11.1}$$

If Robert's cross operator is used for differentiating, edge direction should be calculated as follows.

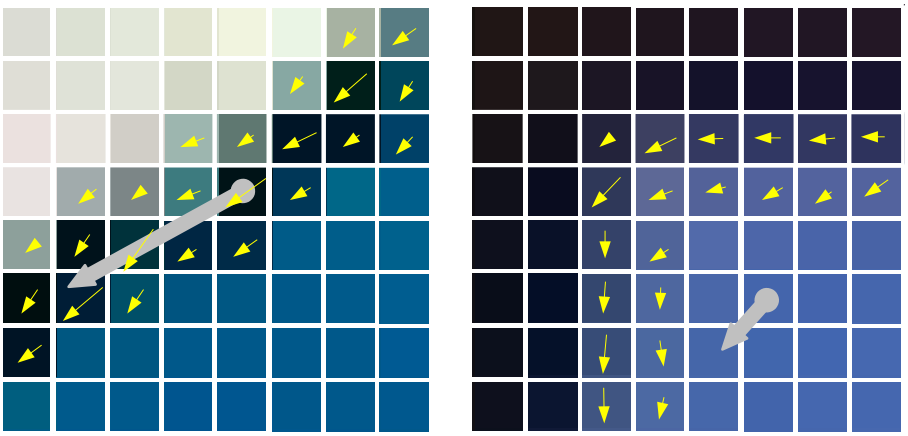
$$\mathbf{E} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \mathbf{V} \tag{11.2}$$

which is a rotation of 135°.

11.1.2 Position and Direction Accumulation

The next step is to calculate a set of *ER*s for each cell, this is done by a special form of accumulation described in this section. An *ER* should refer to an edge passing through the cell. It is possible to calculate the *ER* by simply averaging the position and edge vector of the points inside the cell belonging to the edge. The method works as long as there is only one straight edge inside the cell. As the complexity of the cell contents increase, simple averaging fails. It is then required to distinguish between multiple edges using a more general clustering algorithm. This can increase the difficulty of the problem. Some examples are shown in figure 11.2.

The problem cannot be solved in its general form without a noticeable increase in the amount of calculations. However a partial solution should be enough for RoboCup use. Figure 11.3 presents some examples commonly observed in RoboCup scenario. Samples are overlayed with edge direction vectors. An often observed case is a field line or a side pole included with both side edges in a cell. The other but less frequent observation is a more or less 90° corner as a result of either an intersection between two lines or a part of a rectangular object. Thanks to a 360° representation of the gradient direction, it is possible to separate the edges in a majority of the cases using only the direction information. Note that the edges of an object such as a field line form two complementary



(a) Single Edge Case (b) Multiple Edge Case

Figure 11.2: Averaging Fails When Multiple Edges Appear Inside a Cell

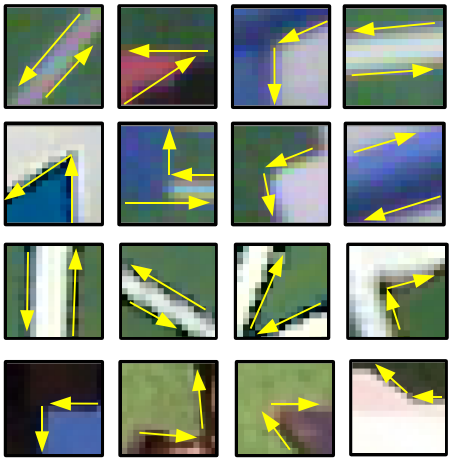


Figure 11.3: Common Examples of Multiple Edge Distribution

directions although they are parallel in the image. The angle distance between two angle values is defined in equation 11.3.

$$|\theta_1 - \theta_2| = \begin{cases} |\theta_1 - \theta_2 + 360| & \theta_1 - \theta_2 < -180 \\ |\theta_1 - \theta_2 - 360| & \theta_1 - \theta_2 > 180 \\ |\theta_1 - \theta_2| & else \end{cases} \tag{11.3}$$

The algorithm described in figure 11.4 is developed based on this idea. As a matter of optimization, it is preferred to have a one-pass algorithm so that it can also be implemented on systems without whole image buffering capability. The algorithm functions as follows. The image is scanned pixel by pixel. Two *ER*s are used in this implementation. Each *ER* contains a position and orientation

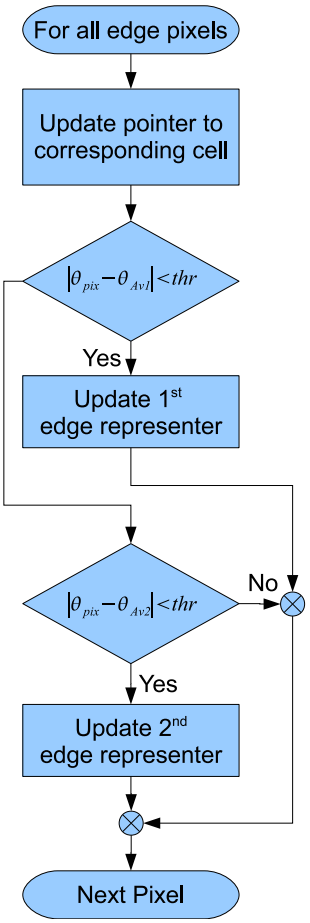


Figure 11.4: Position and Direction Accumulation Algorithm

accumulator and a pixel counter. An edge pixel is joined to the first *ER*, provided that its gradient orientation is closer than a certain distance to the average orientation of the *ER*. If this is not the case, the orientation is compared to the second *ER*, and in the case of no match it is ignored. An empty *ER* will obviously be filled with the first edge pixel met.

As a further optimization to the algorithm, it is possible to replace the polar representation of the gradient vector with a Cartesian one. The implementation of angle distance thresholding will then be replaced with thresholding the inner product of the vectors as presented in equation 11.4.

$$\mathbf{V_1}.\mathbf{V_2} > |\mathbf{V_1}|\,|\mathbf{V_2}|\cos(thr) \tag{11.4}$$

It saves the dynamic calculation of arctan2 function. According to the CPU documentation 32 bit multiplication can be performed in one cycle. However normalization of the vectors could cause performance problems. These can also be optimized away using the z component of the cross product

of the vectors as described in equation 11.5.

$$|(\mathbf{V_1} \times \mathbf{V_2})_z| < \mathbf{V_1} \cdot \mathbf{V_2} \tan(thr) \tag{11.5}$$

Using this technique, vector angle thresholding is done with only four integer multiplications having a great impact on the performance of the algorithm.

The grid structure can be chosen overlapped, i.e. each cell also covers half of every neighboring cell. This increases the smoothness of the results but is computationally more expensive.

11.1.3 Connection Graph Extraction

A list of *ER*s are produced in the previous stage through a single image scan. A connection graph should be calculated by scanning the grid and comparing *ER*s of the neighboring cells. The graph is implemented using an extra field in the cell structure pointing to the following neighbor *ER* called *"outbound"* and a Boolean field indicating that the *ER* is added to the trace called *"inbound"*.

The algorithm is shown in figure 11.5. Upon two conditions, two neighbor *ER*s are marked as connected. The first condition verifies that both *ER*s are in the same direction. This condition is however not enough as it also holds true for separate parallel edges. Therefore, the second condition verifies that the vector connecting the *ER*s is also in the same direction as the self edges. By adjusting the thresholds, the maximum accepted curvature of the edge trace can be determined. Both conditions can be optimized using the technique described in the last section. Note that for the second condition the edge direction should be used. Figure 11.6 demonstrates different examples of the neighboring *ER*s. A connection is only accepted in example 11.6d.

The internal loop of the algorithm breaks as soon as a connection is found. This guarantees that every node in the connection graph has an out-degree of maximum one. It is also encouraged to reduce the in-degree of the nodes to a maximum of one. This is implemented by refusing a connection if the destination node has already been connected by checking its *inbound* property.

It is theoretically not guaranteed that the connection graph becomes free from loops, however if a loop exists it should be the result of an uninterrupted semi-circular edge in the image, which does not usually appear in RoboCup images.

11.1.4 Edge Trace Extraction

The final stage of the algorithm produces an array of edge traces. Each component of the array is a connected *ER* chain. The algorithm is demonstrated in figure 11.7. It searches the graph for

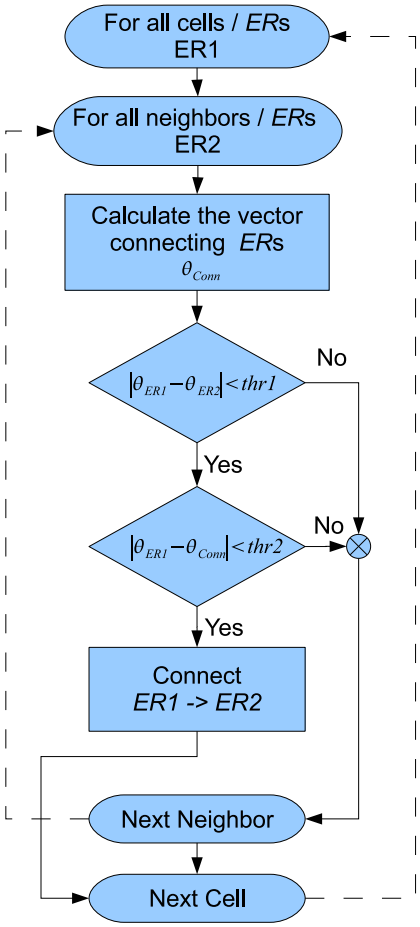
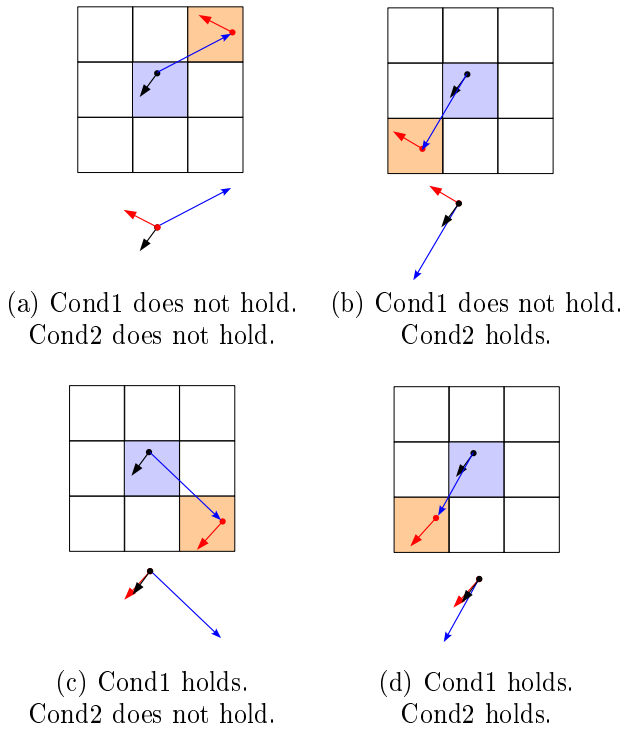


Figure 11.5: Connection Graph Extraction Algorithm



(a) Cond1 does not hold. Cond2 does not hold.

(b) Cond1 does not hold. Cond2 holds.

(c) Cond1 holds. Cond2 does not hold.

(d) Cond1 holds. Cond2 holds.

Figure 11.6: Examples of Connected and Unconnected Neighbors

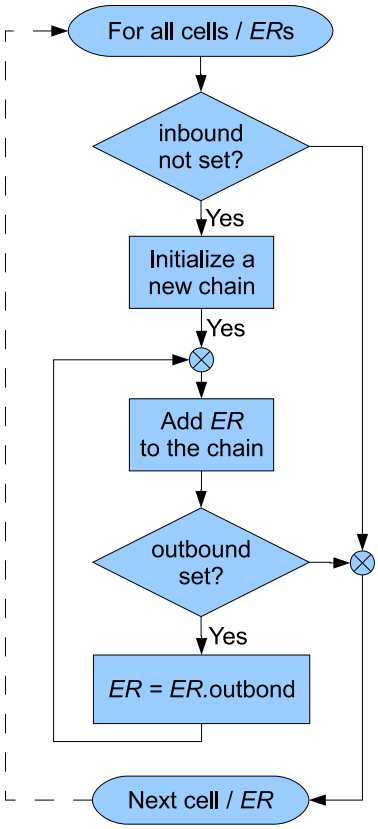


Figure 11.7: Edge Extraction Algorithm

source nodes, which are simply *ER*s with *inbound* not set. Upon a match, the trace is followed using *outbound* pointers and the matching *ER*s are pushed in the trace.

11.1.5 Implementation and Experimental Results

The algorithm is implemented as the low level part of the vision software. It is tested on two available computer platforms using different optimizations. To minimize the amount of computations, one *ER* is calculated per cell. The grid contains 40 x 30 non-overlapping cells, each of which cover 256 pixels of the captured image.

To achieve the required frame rate and still have enough CPU power free for other processes running, the following optimizations are applied to the algorithm.

- Quarter resolution scan: Color digital cameras usually provide images with a so called *"Bayer pattern"* . A VGA image contains therefore 640 x 480 single channel values. This is 1/3 of the information recorded in common RGB pattern. The remaining values are interpolated in such

	Gumstix Overo		Gumstix Verdex	
	Frame rate (FPS)	CPU usage (%)	Frame rate (FPS)	CPU usage (%)
No optimization	12	100	10	100
Quarter res. scan	18	100	14	100
+Over horizon skip	20	100	17 (max)	80
+Out of circle skip	26 (max)	80	17 (max)	75
+Random line skip	26 (max)	65	17 (max)	60

Table 11.1: GVG Implementation Results

representations. Hence it is possible to skip every other pixel and very other image line without a distinct loss of information.

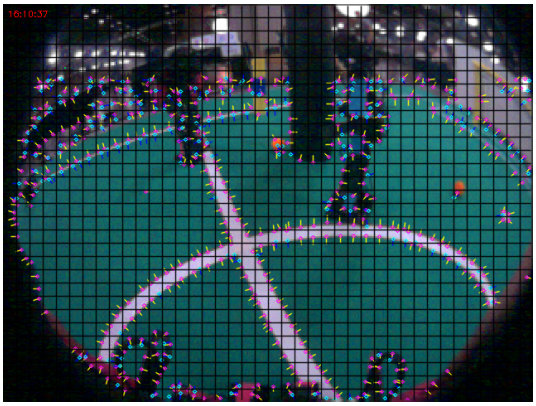
- Over horizon skip: The only objects which partly appear over the horizon are the landmarks and other robots. To detect and calculate the position of these objects, it is always enough to find the lowest point belonging to them. Therefore there is no need to detect objects over the horizon. As the camera is equipped with a wide angle lens, horizon is projected as a curve in the captured image. This can however be estimated with a horizontal line touching the actual projection in the top-most point. There are different methods to calculate the horizon in the image. Two solutions currently used in FUmanoids are IMU sensors and field range detection. Upon detection of the horizon, scanning the image can vertically begin from this line.

- Out of circle skip: Fish eye optics used in FUmanoid robots projects the image inside a circle. The rest of the imaging surface is covered with black pixels. This effect can be observed in figure 11.8. It is possible to skip these pixels by calculating the horizontal extents for each image line.

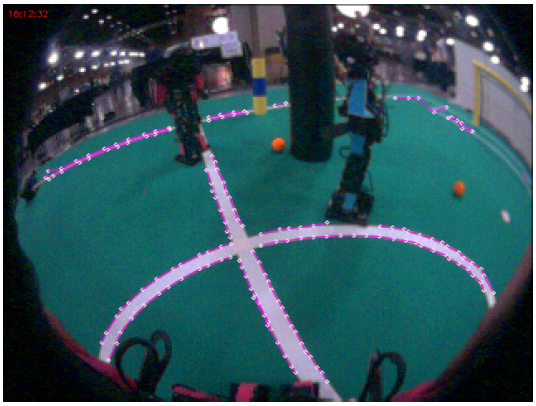
- Random line skip: Apart from the fish-eye optics deformation, the camera provides a perspective view of the field. Therefore much more information is available from near objects, observed in the lower area of the image, than the far ones appearing in the upper area. Due to this fact it is possible to ignore more and more lines as the scan gets closer to the bottom of the image. It is done by comparing a uniformly distributed random number with a dynamic threshold calculated form scan line Y coordinate. Following formula is used to skip a line:

$$r < Y/640 \tag{11.6}$$

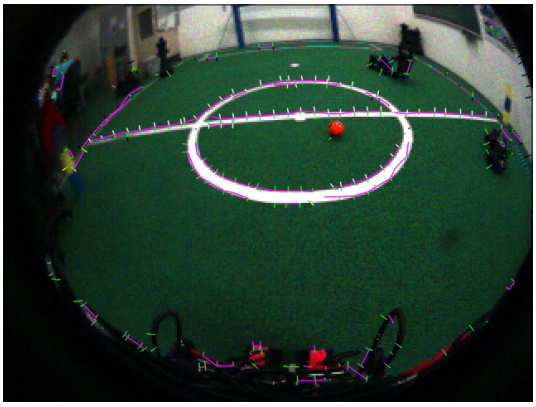
Where r is a random real number between 0 and 1 and Y is the vertical component of the scan line coordinate.



(a) Visualization of the ERs.



(b) Extracted edges overlaid on the test image.



(c) Typical results of the test case.

Figure 11.8: GVG Implementation Results

	Gumstix Overo		Gumstix Verdex	
	Frame rate (FPS)	CPU usage (%)	Frame rate (FPS)	CPU usage (%)
Canny	2.7	100	1.5	100

Table 11.2: Canny Implementation Results

Implementation results are summarized in table 11.1 on two available processor platforms. The first platform is Gumstix Verdex Pro which is equipped with a 600 MHz PXA270 and the second is a Gumstix Overo with a 720 MHz OMAP3 3530 processor. Each row of the table shows the optimization added to the last state. The last row shows the highest optimization level achieved by applying all described techniques. A standard test procedure is used for this performance measurement. The robot is placed upright on one of the penalty points, directed to the goal placed on the other side of the field as shown in figure 11.8c. Two parameters are measured, which together show the performance of the algorithm. These are processed frame rate and CPU usage. Since the camera delivers a limited number of frames per second and the CPU is also capable of performing a limited amount of processing, there are two possible scenarios.

1. A frame can be entirely processed before the next frame gets ready. In this case the processor goes idle and the CPU shows less than 100 percent of usage. Frame rate remains constant in this mode and CPU usage is used as the indicator of the performance.

2. A new frame gets ready as long as the last frame is still being processed. This leads to a frame buffer overflow, which is in turn handled with dropping frames. The CPU is never released in this case and the usage indicator shows always 100 percent. Frame rate is then used as the performance indicator.

Using the Canny algorithm from the Open CV implementation as the control case, the described test procedure shows the results listed in table 11.2. Compared to the *GVG*, the Canny edge extraction is slower by a factor of 10. Note that Canny algorithm only results in a bitmap of the edges and doesn't perform any edge grouping, so this has to be done additionally. It can therefore be concluded that the *GVG* algorithm significantly outperforms the standard Canny procedure in the RoboCup scenario.

Figure 11.8a shows the visualization of the ERs for a typical image captured by the robot. Despite the rough grid, a relatively high level of detail is detected. In figure 11.8b the result of the edge trace extraction is demonstrated for field lines.

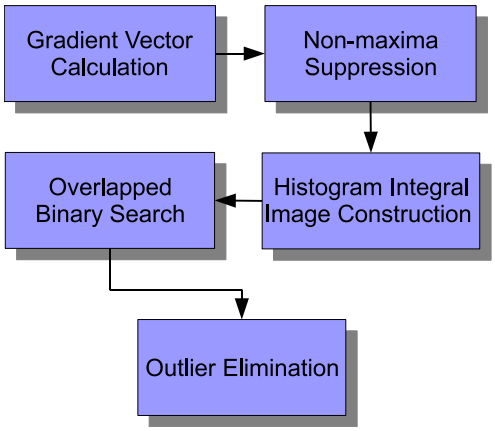


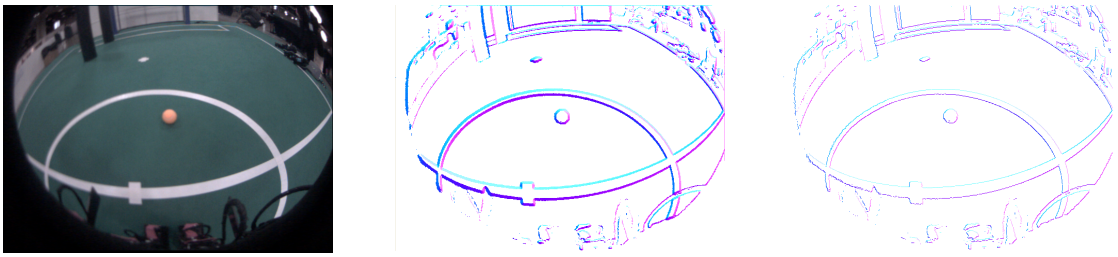
Figure 11.9: Structure of the Ball Detection Method

11.2 Shape Based Ball Detection Using Edge Orientation Histogram

In this section I introduce a method to detect the ball based on its shape. The method can also be used for other round objects with detectable edges. The method is motivated by two previous works: intensity integral images introduced by Viola et al. [83] and *HOG* introduced by Dalal and Triggs[82]. The same concept of integral images has been used, however integration is applied to a vector representation of the gradient orientation histogram of each pixel. As an extension to the original idea, an overlapped binary search is used to locate ball candidates in the image.

11.2.1 Structure of the Method

The method is structured as presented in figure 11.9. It includes five stages starting with the calculation of the gradient vector, which is common in shape based object detection methods. Using non-maxima-suppression, the edges are thinned and so normalized. A so-called "*histogram integral image*" is then constructed based on the orientation of the gradient vector in each pixel. This representation of the original image helps to accelerate the search algorithm. An overlapped binary search recursively scans the pyramid down and finds the best-fitting box around the object using edge orientation statistics from the histogram integral image. Final results are once more filtered using further statistical criteria.



a) Original Image b) Thresholded Edges c) Non-maxima-suppressed

Figure 11.10: Edge Points Selected for Ball Detection Algorithm

11.2.2 Gradient Vector Calculation and Thresholding

Gradient vector can be calculated as described in section 11.1.1. It is encouraged to smoothen the image before calculating the gradients to remove the noise. Results are then non-maxima-suppressed along the gradient direction, similar to the method used in canny edge detector [78]. This reduces the edge thickness and provides normalized results for the procedure introduced in section 11.2.6. Too weak edge points are then omitted using a simple thresholding. Figure 11.10 shows the intermediate results.

11.2.3 Histogram of Edge Orientations

Histogram of edge orientations is an important measure used in shape-based object detection algorithms as described in the beginning of this chapter. In the present method, the histogram is calculated by counting the number of edge pixels representing each direction range. Direction of the gradient vector is extracted in a 360° range, which is quantized into 18 groups of each 20 degrees. The detection algorithm relies indirectly on the fact that an ideal round object has a uniform distribution of edge orientations. This feature gets however rapidly lost as the window size grows and other contents are added to the image.

Additional contents of the image always add positive values to the histogram. A direct result gained from this is that large regions of the image can be entirely rejected if the orientation histogram contains zeros in more than a given number of directions. Theoretically, an edge orientation histogram belonging to a window containing the round object must be zero free. This can however not always be fulfilled in real test conditions due to shadow, partial occlusion or some other effects. Therefore a certain number of zeros are allowed in the histogram, which can be adjusted in accordance to the image condition.

To visualize what kind of information an edge orientation histogram provides and how strong this

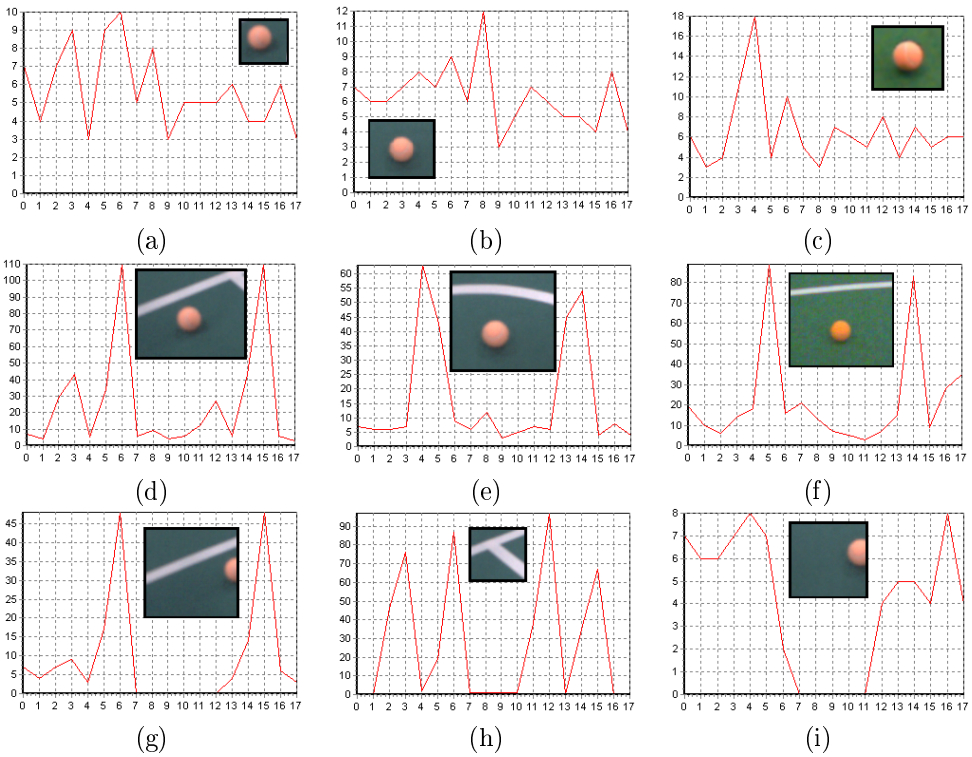


Figure 11.11: Histogram of Edge Orientations: Examples

measure is, several example histograms of the images captured from the robots are presented in figure 11.11. In figure 11.11a, b and c, the histogram of a just-ball window is presented. The histogram shows a more or less uniform distribution of the directions. In figure 11.11c, d and e, the window size is grown and some other content has been added. The histogram has lost the uniformness, however all components have remained non-zero. In figure 11.11g and i, the ball is partially visible which has lead to a significant number of zero components and finally the resulting histogram of a ball-free window is shown in figure 11.11h. As it can be seen, a round object placed on the border of a window cannot be detected. A solution to this problem is discussed in section 11.2.5.

11.2.4 Integral Histogram Image

The idea of intensity integral images is introduced in [83]. The same idea is extended to the histogram of edge orientations in this work.

Assume an image with 18 channels, i.e. each pixel value is an 18 dimensional vector. The vector describes the histogram of edge orientations according to a rectangle, stretched from the origin of the source image to the given coordinates. This is described in equation 11.7 and presented in figure 11.12.

$$\mathbf{I}(x,y) = \sum_{\substack{x\prime < x \\ y\prime < y}} \mathbf{H}(x\prime, y\prime) \tag{11.7}$$

$\mathbf{H}$ is a vector filled with zeros except for the component corresponding to the direction of the gradient vector at $(x\prime, y\prime)$, which is filled with 1 if the pixel is identified as an edge pixel.

According to [83], an integral image can be computed in a single scan by using the following recurrence relation:

$$\mathbf{S}(x,y) = \mathbf{S}(x-1,y) + \mathbf{H}(x,y) \tag{11.8}$$

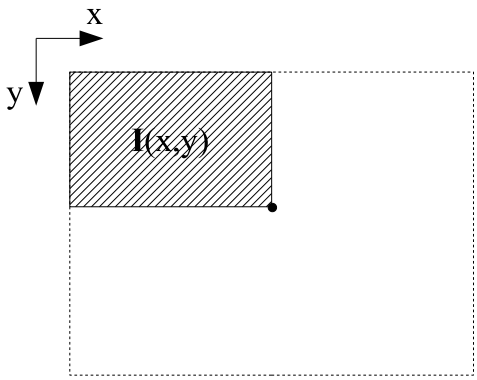
$$\mathbf{I}(x,y) = \mathbf{I}(x,y-1) + \mathbf{S}(x,y) \tag{11.9}$$

where $\mathbf{S}$ is a temporary vector holding a histogram of the current line of the image. It is enough to store $\mathbf{S}$ as a single accumulator because there is no reference to its history.

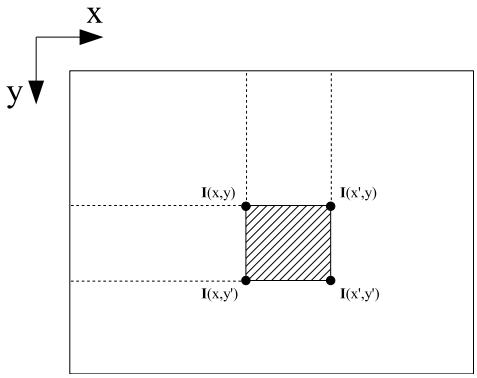
Integral representation reduces the computation of the histogram for any given window to two additions and one subtraction as follows:

$$\mathbf{Hist}(x,y,x\prime,y\prime) = \mathbf{I}(x,y) + \mathbf{I}(x\prime,y\prime) - (\mathbf{I}(x\prime,y) + \mathbf{I}(x,y\prime)) \tag{11.10}$$

This accelerates the operation to a great extent.



(a) Calculation of Histogram Integral Image



(b) Computation of the Histogram Based on the Integral Representation

Figure 11.12: Computation of Integral Histogram

Algorithm 2 Overlapped Binary Search

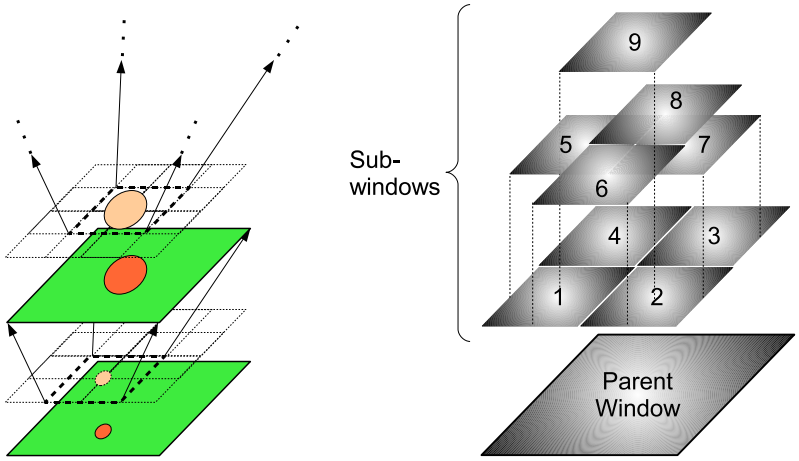
```
Boolean search_ball(window, level)
begin
    if window is already scanned then
        return true
    window <- scanned
    calculate_histogram(window)
    if histogram has at least one zero component then
        return false
    if (level>5) then
        return false
    b <- false
    for all sub windows
        b <- b or search_ball(sub window, level+1);
    if not b then
        push_ball_candidate(window)
end
```

11.2.5 Overlapped Binary Search

So far, a measure is defined to determine areas in the image not covering an entire ball. However, It does not mean that an area contains a ball if it is not rejected using this measure. Furthermore, it does not guarantee that if the ball exists in the area, it is the only object surrounded by the area. In this section a recursive function is suggested to find the best ball candidates using an edge orientation histogram measure.

The function is presented in algorithm 2. It examines the given window using above described measure. The function first verifies if the window could contain a ball by scanning the histogram. If the window is not rejected it is divided into several overlapping sub-windows, which are recursively processed by the function. Base cases of the recursion are windows, which either have more than a certain number of zero components in their histogram or are smaller than the ball is expected to be. This is demonstrated in figure 11.13a.

A problem could occur when the ball lies on the border of neighboring sub-windows if sub-windows were non-overlapping. But thanks to the overlapped searching, the object can be entirely covered by at least one sub-window. As demonstrated in figure 11.13b, sub-windows are a quarter of the parent window in area and are distributed using a grid, both horizontally and vertically, one fourth of the parent window edge length. A window is thus divided into 9 sub-windows.



(a) Recursive Binary Search (b) Overlapping Sub-windows

Figure 11.13: Overlapped Binary Search

It can be observed that overlapped searching can reference a window more than once, due to the overlapping in the algorithm. The solution suggests a look up table with an element for each possible window, down to the desired depth storing the search result for that window. A repeated reference can be detected at the beginning of the function and replied with the pre-stored search result.

11.2.6 Outlier Elimination

According to the algorithm, a ball candidate is reached if all sub-windows of an accepted parent window are rejected to contain an entire ball. The results are still subject to false positives. It is therefore required to further filter the output of the algorithm using a geometrical criterion. Two measures are suggested, both of which can be obtained from the histogram so that no further reference to the image is needed.

The first measure computes standard deviation and average of the histogram. The following condition verifies how uniform the distribution is.

$$\sigma < \alpha\mu \tag{11.11}$$

where σ is the standard deviation and μ is the average of the histogram. α is a constant, which determines the accepted uniformness of the distribution. The higher α becomes, the more candidates are accepted as balls.

The second measure verifies whether the number of edge points the window contains matches the circumference of the window. Assuming d to be the edge length of the window and n the number of

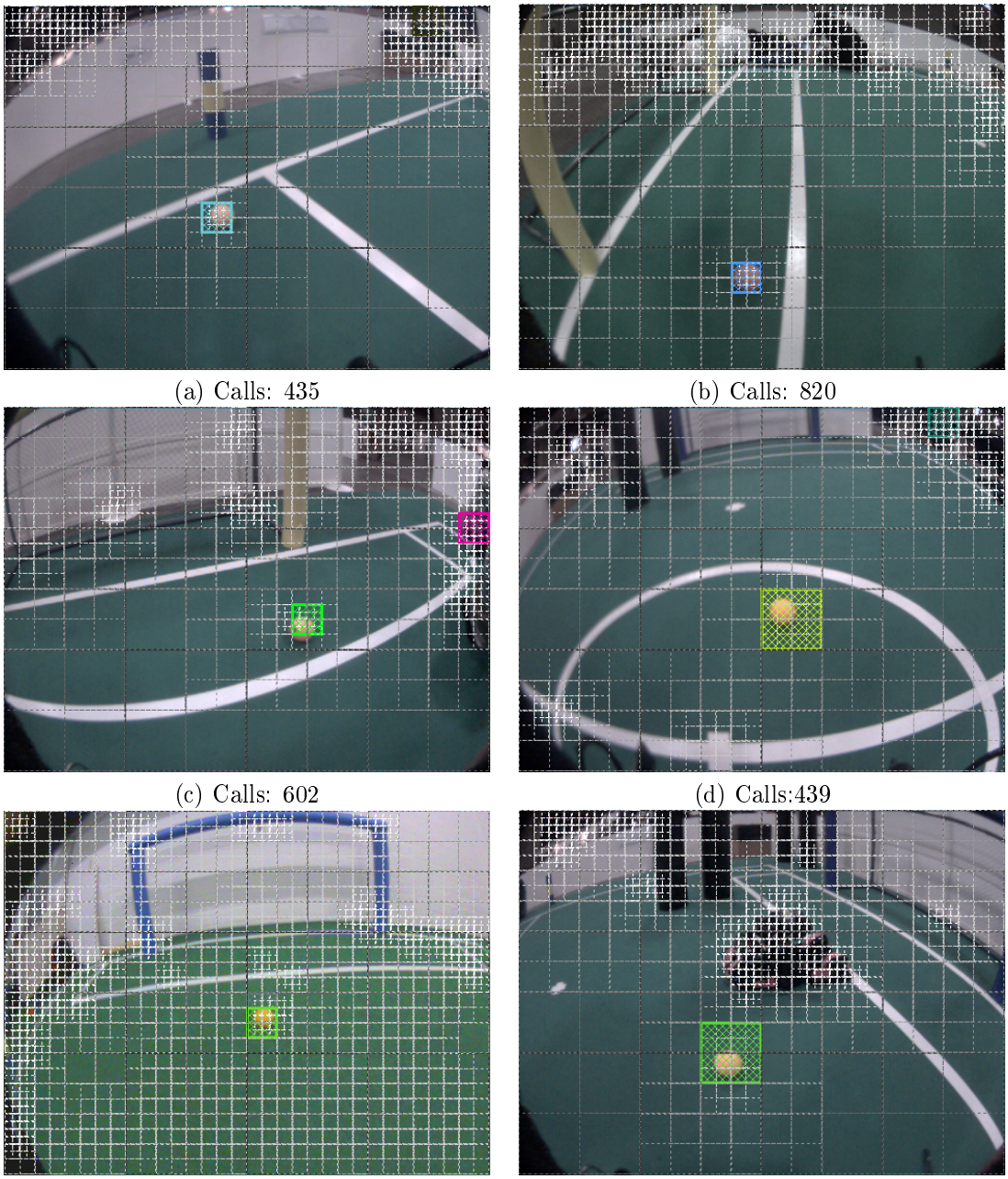
edge pixels found in the window, the following is the condition to find balls.

$$\frac{\pi}{2}d(1-\beta) < n < 4d(1+\beta) \tag{11.12}$$

As window size is halved in each level, the window can be up to double the ball size. This sets the lower bound of the pixel count. The upper bound is set to the circumference of the window. β is an adjustable tolerance added to the condition. As a prerequisite to this condition the edge thickness should be reduced to one pixel using non-maxmima suppression.

11.2.7 Results

The algorithm has been tested off-line using images recorded from RoboCup humanoid field. Some example results are presented in figure 11.14. Results are overlayed with windows that the algorithm has accessed, which shows how the method approaches the image. As a performance metric, the number of references to the recursive function is counted. Comparing this to the number of pixels of the image (in this case 512x512) this shows a promising optimization of the search method. However the initial image scan to calculate the *histogram integral image* should also be considered.



(a) Calls: 435 (b) Calls: 820

(c) Calls: 602 (d) Calls:439

(e) Calls: 1338 (f) Calls:658

Figure 11.14: Results of Ball Detection using Histogram of Edge Orientations.

Part V

Summary, Conclusion and Future Work

In this thesis, I described the development of the "FUmanoids". The FUmanoids are a team of intelligent soccer playing humanoid robots, founded in 2006 and participated up to the publication of this thesis yearly in RoboCup world competitions. During the presence of the FUmanoids in RoboCup, the team has won several valuable prices including the 3rd place and two times the second place of the world RoboCup competitions.

I explained the robotics platform, including the mechanical and electrical design of the different versions of the robot in part II. The structure of the software used in the 2007-2009 versions of the robot is also explained in this part. Here I presented several ideas and solutions I implemented to solve the problems and improve the state of the system.

In part III I focused on the stabilization of the bipedal walking, which was my main contribution in the software development of the robots. I explain a simulation platform I developed, based on which I studied the control of the robot. Using this platform and by further development of the methods on the real robots, I achieved a combination of control strategies which guaranteed the fast and robust bipedal walking of the robots. To develop the control mechanism, I used an energy-based concept. The approach I followed in the development was inspired from passive dynamic walking. I have then extended the idea to be used on active and compliant joints in the robots.

Part IV described my contribution in the development of computer vision solutions for the humanoid robots. I presented a computer vision module capable of providing color-based object detection results to low power microcontrollers. Different image processing algorithms I developed for this module were explained in this part. I further introduced two shape-based algorithms for detection of the field lines and the ball. My shape based edge grouping method was successfully implemented in the FUmanoid robots in years 2010 and 2011.

11.3 Future Work

The development of the FUmanoids continues further. I suggest the following improvements for the future:

11.3.1 Mechanics

Parallel leg kinematics is recently introduced in humanoid league. The technique suggests the coupling of the successive parallel joints to guarantee that the feet planes stay always parallel to each other. Using parallel leg kinematics can reduce the software expense and make walking more stable.

Further improvement of the hands for making them capable of manipulating objects can be useful

for the goal keeper. Currently the goal keeper is only capable of touching the ball with its feet or with its body but the interaction with the ball is rather uncontrolled. There are many situations, in which the ball stays near to the goal keeper for a long while. It is therefore useful if the goal keeper can take the ball and through it far away using its hands. This is also allowed in the rules.

Another idea for the future is to make changes in feet of the robots in order to allow high kicks. High kicks has been used for a long while in other RoboCup leagues, such as middle size. This can make the goal keeper unable to calculate the correct path of the ball. The idea can also be useful to score a goal when the goal keeper lies in front of the goal.

Adding force/torque sensors in the feet plates is the next improvement I suggest for the future. This can provide a more reliable feedback for control and stabilization of the robot.

11.3.2 Electronics

I suggest adding an intermediate processor between the main computer and the motor/sensor bus. This module should form an abstraction layer and can undertake basic motions. This can also improve the reliability of the robot by taking necessary actions if the robot temporarily loses its processing capability. A reboot phase of the main CPU can be such a case.

The next suggestion is the use of stereo cameras to increase the distance measurement accuracy and correct the lag of the IMU sensors. The head of the 2009 Robot was designed to allow a stereo vision, however there were some difficulties, such as insufficient resolution of the cameras, especially after using a wide angle lens. For an effective stereo vision the resolution should be at least a few mega pixels.

11.3.3 Control Software

For the bipedal walking I suggest the integration of the IMU and foot pressure in the control methods. This can smoothen the behavior of the joints and reduce the loss of energy. The implementation of the walking algorithm on an intermediate processor can also improve the performance.

Currently only a limited feedback linearization is done for walking motion. Back-calculation of the end-effector coordinates using the joint angles can provide a better feedback for the control of the walking. The same method can also be used for the static motions, which can improve their reliability.

11.3.4 Behavior Control

I suggest the implementation of CSBP on a data-flow platform. This can increase the parallelity and also make the implementation faster and more clear.

For future there is a need for development of more reactive behaviors. These require more reliable and real-time sensory data and can therefore be implemented in lower levels of the software.

11.3.5 Computer Vision

I suggest further work on shape based methods. It is possible to reduce the dependency of the object recognition methods of the colors. Tracking methods both in the image and in the state space can be used in future to follow the objects and reduce the amount of calculations. Implementation of stereo-vision is another area which can be improved in conjunction with high resolution cameras.

Another idea is the optical calculation of the camera orientation based on the observation of the field lines. Theoretically it is possible to calculate the complete orientation matrix of the camera having recognized a pair of perpendicular lines. This can avoid problems such as errors in IMU measurements and delays between camera and the IMU measurements.

References

[1] T. McGeer, "Powered flight, child's play, silly wheels and walking machines," in *Proc. Conf. IEEE Int Robotics and Automation*, 1989, pp. 1592–1597.

[2] M. Wisse and A. L. Schwab, "Skateboards, bicycles, and three-dimensional biped walking machines: Velocity-dependent stability by means of lean-to-yaw coupling," *The International Journal of Robotics Research*, vol. 24, no. 6, pp. 417–429, 2005. [Online]. Available: http://ijr.sagepub.com/content/24/6/417.abstract

[3] A. K. Mackworth, "On seeing robots," Vancouver, BC, Canada, Canada, Tech. Rep., 1993.

[4] "Robocup humanoid rules." [Online]. Available: http://www.tzi.de/humanoid/bin/view/Website/Downloads

[5] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of honda humanoid robot," in *Proc. IEEE Int Robotics and Automation Conf*, vol. 2, 1998, pp. 1321–1326.

[6] J.-I. Yamaguchi, A. Takanishi, and I. Kato, "Development of a biped walking robot compensating for three-axis moment by trunk motion," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems '93 IROS '93*, vol. 1, 1993, pp. 561–566.

[7] Y. Ogura, H. Aikawa, K. Shimomura, A. Morishima, H. ok Lim, and A. Takanishi, "Development of a new humanoid robot wabian-2," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2006*, 2006, pp. 76–81.

[8] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: system overview and integration," in *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, vol. 3, 2002, pp. 2478–2483.

[9] 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," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2000)*, vol. 3, 2000, pp.

1559–1564.

[10] K. Kaneko, F. Kanehiro, M. Morisawa, K. Miura, S. Nakaoka, and S. Kajita, "Cybernetic human hrp-4c," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots Humanoids 2009*, 2009, pp. 7–14.

[11] M. Gienger, K. Loffler, and F. Pfeiffer, "Towards the design of a biped jogging robot," in *Proc. ICRA Robotics and Automation IEEE Int. Conf*, vol. 4, 2001, pp. 4140–4145.

[12] S. H. Collins, M. Wisse, and A. Ruina, "A three-dimensional passive-dynamic walking robot with two legs and knees," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 607–615, Jul. 2001. [Online]. Available: http://ijr.sagepub.com/content/20/7/607.abstract

[13] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/307/5712/1082

[14] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman, "The simplest walking model: Stability, complexity, and scaling," *Journal of Biomechanical Engineering*, vol. 120, no. 2, pp. 281–288, 1998. [Online]. Available: http://link.aip.org/link/?JBY/120/281/1

[15] J. K. Holm and M. W. Spong, "Kinetic energy shaping for gait regulation of underactuated bipeds," in *Proc. IEEE Int. Conf. Control Applications CCA 2008*, 2008, pp. 1232–1238.

[16] T. McGeer, "Passive walking with knees," in *Proc. Conf. IEEE Int Robotics and Automation*, 1990, pp. 1640–1645.

[17] M. Wisse, A. L. Schwab, R. Q. van der Linde, and F. C. T. van der Helm, "How to keep from falling forward: elementary swing leg action for passive dynamic walkers," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 393–401, 2005.

[18] R. Knight, U. Nehmzow, and C. C. Sq, "Walking robots, a survey and a research proposal," CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States), Tech. Rep., 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.1983

[19] F. Daerden and D. Lefeber, "Pneumatic artificial muscles: Actuators for robotics and automation," *Europian journal of mechanical and enviromental engineering*, vol. 47, pp. 11–22, 2002.

[20] B. Verrelst, R. V. Ham, B. Vanderborght, F. Daerden, D. Lefeber, and J. Vermeulen, "The pneumatic biped "lucy" actuated with pleated pneumatic artificial muscles," *Autonomous*

Robots, vol. 18, pp. 201–213, 2005, 10.1007/s10514-005-0726-x. [Online]. Available: http://dx.doi.org/10.1007/s10514-005-0726-x

[21] B. Verrelst, "A dynamic walking biped actuated by pleated pneumatic artificial muscles: Basic concepts and control issues," Ph.D. dissertation, Vrije Universiteit Brussel, Brussels, Belgium., 2005.

[22] K. Hosoda, T. Takuma, and A. Nakamoto, "Design and control of 2d biped that can walk and run with pneumatic artificial muscles," in *Proc. 6th IEEE-RAS Int Humanoid Robots Conf*, 2006, pp. 284–289.

[23] M. Sugisaka, "An approach for soft humanoid robot with artificial muscles," in *Proc. IEEE Int. Symp. Industrial Electronics ISIE 2009*, 2009.

[24] S. O. Anderson, M. Wisse, C. G. Atkeson, J. K. Hodgins, G. J. Zeglin, and B. Moyer, "Powered bipeds based on passive dynamic principles," in *Proc. 5th IEEE-RAS Int Humanoid Robots Conf*, 2005, pp. 110–116.

[25] S. O. Anderson, C. G. Atkeson, and J. K. Hodgins, "Coordinating feet in bipedal balance," in *Proc. 6th IEEE-RAS Int Humanoid Robots Conf*, 2006, pp. 624–628.

[26] S.-H. Hyon and G. Cheng, "Passivity-based full-body force control for humanoids and application to dynamic balancing and locomotion," in *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, 2006, pp. 4915–4922.

[27] M. Shahinpoor and K. J. Kim, "Ionic polymer-metal composites: Iv. industrial and medical applications," *Smart Materials and Structures*, vol. 14, pp. 197–214, 2005.

[28] K. Oguro, Y. Kawami, and H. Takenaka, "Bending of an ion-conducting polymer film-electrode composite by an electric stimulus at low voltage," *Trans. J. ofMicromachine SOC, 5*, pp. 27–30, 1992.

[29] S. Guo, Y. Ge, L. Li, and S. Liu, "Underwater swimming micro robot using ipmc actuator," in *Proc. IEEE Int Mechatronics and Automation Conf*, 2006, pp. 249–254.

[30] M. Yamakita, N. Kamamichi, T. Kozuki, K. Asaka, and Z.-W. Luo, "Control of biped walking robot with ipmc linear actuator," in *Proc. Conf. IEEE/ASME Int Advanced Intelligent Mechatronics*, 2005, pp. 48–53.

[31] K. Otsuka, *Shape memory materials*, C. Wayman, Ed. Cambridge: Cambridge University Press, 1998.

[32] W. J. Buehler and F. E. Wang, "A summary of recent research on the nitinol alloys and their potential application in ocean engineering," *Ocean Engineering*, vol. 1, no. 1, pp. 105 – 108, IN7–IN10, 109–120, 1968. [Online]. Available: http://www.sciencedirect.com/science/article/pii/002980186890019X

[33] K. J. DeLaurentis, C. Mavroidis, and C. Pfeiffer, "Development of a shape memory alloy actuated robotic hand," Neural Networks, Tech. Rep., 2000.

[34] K.-Y. Tu, T.-T. Lee, C.-H. Wang, and C.-A. Chang, "Design of fuzzy walking pattern (fwp) for a shape memory alloy (sma) biped robot," in *Proc. IEEE Int Systems, Man, and Cybernetics Conf*, vol. 4, 1998, pp. 3266–3271.

[35] M. Nishida, K. Tanaka, and H. O. Wang, "Development and control of a micro biped walking robot using shape memory alloys," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2006*, 2006, pp. 1604–1609.

[36] E. T. Esfahani and M. H. Elahinia, "Stable walking pattern for an sma-actuated biped," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 5, pp. 534–541, 2007.

[37] R. Kratz, M. Stelzer, M. Friedmann, and O. von Stryk, "Control approach for a novel high power-to-weight ratio sma muscle scalable in force and length," in *IEEE/ASME Intl. Conf. on Advanced Intelligent Mechatronics (AIM)*, Zürich, CH, September 4-7 2007.

[38] R. Kratz, M. Stelzer, and O. von Stryk, "Macroscopic sma wire bundle actuator/sensor system: design, measurement, control approach," in *Proc. 4th IFAC-Symposium on Mechatronic Systems*, Heidelberg, September 12-14 2006.

[39] H. M. F. Khorrami, P. Krishnamurthy, *Modeling and Adaptive Nonlinear Control of Electric Motors*. Springer-Verlag, New York, 2003.

[40] Robotis, *Dynamixel, High-performance networked actuators for robots fully integrated with feedback function and programmability.* [Online]. Available: http://www.robotis.com/xe/dynamixel_en

[41] G. T. Fallis, "Walking toy, improvement in walking toys," USA Patent 376588, 1888.

[42] M. Kukulski, "Entwurf und bau einer humanoiden bewegungsplattform für fußball spielende roboter," Master's thesis, Freie Universität Berlin, 2010.

[43] "Cm5 schematics." [Online]. Available: http://robosavvy.com/site/index.php?Itemid=&id=robotis_bioloid&option=com_openwiki

[44] B. Fischer, H. Mobalegh, and R. Rojas, "Low cost synchronized stereo aquisition system for single port camera controllers," in *Proc. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[45] M. Oertel, "Aufbau und implementierung einer multithreading technologie in einer 8 bit risc-architektur umgebung." Master's thesis, Freie Universität Berlin, Institut für Informatik, 2008.

[46] D. Seifert, "Portierung der fumanoids-software," Study Research Project, 2009. [Online]. Available: https://www.fumanoids.de/wp-content/uploads/2009/07/studienarbeit-portierung-fumanoids.pdf

[47] T. McGeer, "Passive dynamic walking," *International Journal of Robotics Research*, vol. 9(2), pp. 62–82, 1990.

[48] A. Seugling and M. Rölin, "Evaluation of physics engines and implementation of a physics module in a 3d-authoring tool," Master's thesis, UmeA University Department of Computing Science, 2006.

[49] R. Smith, "Open dynamics engine." [Online]. Available: http://www.ode.org/

[50] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science, N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. von Stryk, Eds. Springer Berlin / Heidelberg, 2010, vol. 6472, pp. 38–50. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17319-6_7

[51] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[52] M. Haruna, M. Ogino, K. Hosoda, and M. Asada, "Yet another humanoid walking - passive dynamic walking with torso under simple control," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, 2001, pp. 259 –264 vol.1.

[53] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices." *Trans. of the ASME. Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955. [Online]. Available: http://ci.nii.ac.jp/naid/10008019314/en/

[54] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, no. 3, pp. 163–171, 1986. [Online]. Available: http://link.aip.org/link/?JDS/108/163/1

[55] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," vol. 16, no. 1, pp. 93–101, 1986.

[56] A. Balestrino, G. D. Maria, and L. Sciavicco, "Robust control of robotic manipulators," in *Proceedings of the 9th IFAC World Congress*, vol. 5, 1984, pp. 2435–2440.

[57] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Proc. 23rd IEEE Conf. Decision and Control*, vol. 23, 1984, pp. 1359–1363.

[58] D. L. Pieper, "The kinematics of manipulators under computer control," Ph.D. dissertation, STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, 1968.

[59] F. Asano, M. Yamakita, N. Kamamichi, and Z.-W. Luo, "A novel gait generation for biped walking robots based on mechanical energy constraint," vol. 20, no. 3, pp. 565–573, 2004.

[60] M. Franken, G. van Oort, and S. Stramigioli, "Analysis and simulation of fully ankle actuated planar bipedal robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2008*, 2008, pp. 634–639.

[61] K. D. Mombaur, H. G. Bock, J. P. Schloder, and R. W. Longman, "Human-like actuated walking that is asymptotically stable without feedback," in *Proc. ICRA Robotics and Automation IEEE Int. Conf*, vol. 4, 2001, pp. 4128–4133.

[62] S. Aoi and K. Tsuchiya, "Self-stability of a simple walking model driven by a rhythmic signal," *Nonlinear Dynamics*, vol. 48, pp. 1–16, 2007, 10.1007/s11071-006-9030-3. [Online]. Available: http://dx.doi.org/10.1007/s11071-006-9030-3

[63] F. M. Silva and J. A. T. Machado, "Towards efficient biped robots," in *Proc. Conf. IEEE/RSJ Int Intelligent Robots and Systems*, vol. 1, 1998, pp. 394–399.

[64] F. Asano, M. Yamakita, and K. Furuta, "Virtual passive dynamic walking and energy-based control laws," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2000)*, vol. 2, 2000, pp. 1149–1154.

[65] M. Wisse, C. G. Atkeson, and D. K. Kloimwieder, "Swing leg retraction helps biped walking stability," in *Proc. 5th IEEE-RAS Int Humanoid Robots Conf*, 2005, pp. 295–300.

[66] M. Wisse, D. G. E. Hobbelen, and A. L. Schwab, "Adding an upper body to passive dynamic walking robots by means of a bisecting hip mechanism," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 112–123, 2007.

[67] A. Goswami, B. Thuilot, and B. Espiau, "Compass-like biped robot part i: Stability and bifurcation of passive gaits," INRIA, RR-2996, Tech. Rep., October 1996.

[68] R. Tedrake, T. W. Zhang, M. fai Fong, and H. S. Seung, "Actuating a simple 3d passive dynamic walker," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '04*, vol. 5, 2004, pp. 4656–4661.

[69] 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," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '03*, vol. 2, 2003, pp. 1620–1626.

[70] H. R. Moballegh, M. Mohajer, and R. Rojas, *Increasing Foot Clearance in Biped Walking: Independence of Body Vibration Amplitude from Foot Clearance.* Berlin, Heidelberg: Springer-Verlag, 2009, pp. 157–165. [Online]. Available: http://dl.acm.org/citation.cfm?id=1575210.1575226

[71] W.-H. Chang, C.-H. Hsia, Y.-C. Tai, S.-H. Chang, F. Ye, and J.-S. Chiang, "An efficient object recognition system for humanoid robot vision," in *Proc. Joint Conf.s Pervasive Computing (JCPC)*, 2009, pp. 209–214.

[72] D. Herrero-Perez and H. Martinez-Barbera, "Robust and efficient embedded vision for aibo in robocup," in *Proc. IEEE Latin American Robotic Symp. LARS '08*, 2008, pp. 8–13.

[73] M. Bader, M. Albero, R. Sablatnig, J. E. Simo, G. Benet, G. Novak, and F. Blanes, "Embedded real-time ball detection unit for the yabiro biped robot," in *Proc. Int Intelligent Solutions in Embedded Systems Workshop*, 2006, pp. 1–9.

[74] Y. Liping and S. Kai, "Design and realization of image processing system based on embedded platform," in *Proc. Int Information Technology and Applications (IFITA) Forum*, vol. 2, 2010, pp. 446–449.

[75] G. Yang and K. Shen, "Arm9 embedded system of the image acquisition and processing," in *Proc. Int Anti-Counterfeiting Security and Identification in Communication (ASID) Conf*, 2010, pp. 138–141.

[76] Y.-L. Chen and C.-Y. Chiang, "Embedded vision-based nighttime driver assistance system," in *Proc. Int Computer Communication Control and Automation (3CA) Symp*, vol. 2, 2010, pp. 199–203.

[77] M. Asada and M. Mayer, "Robocup humanoid challenge," *International Journal of Humanoid Robots*, vol. 5, pp. 335–351, 2008.

[78] J. Canny, "A computational approach to edge detection," no. 6, pp. 679–698, 1986.

[79] *CMOS Image Sensor with Image Signal Processing HV7131RP*, MagnaChip Semiconductor Ltd., 2005.

[80] A. Rowe, C. Rosenberg, and I. Nourbakhsh, "A low cost embedded color vision system," in *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, vol. 1, 2002, pp. 208–213.

[81] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2000)*, vol. 3, 2000, pp. 2061–2066.

[82] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2005*, vol. 1, 2005, pp. 886–893.

[83] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2001*, vol. 1, 2001.