An Analytical Motion Filter for Humanoid Robots

Karl J. Muecke

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Dennis W. Hong, Chair
Andrew J. Kurdila Co-chair
Steve C. Southward
Thurmon E. Lockhart
Scott L. Hendricks
Kevin P. Granata †

March 31, 2009
Blacksburg, Virginia

Keywords: Humanoid robot, Zero Moment Point, Biped, Motion Filter

This page is intentionally left blank.

An Analytical Motion Filter for Humanoid Robots

Karl J. Muecke

(ABSTRACT)

Mimicking human motion with a humanoid robot can prove to be useful for studying gaits, designing better prostheses, or assisting the elderly or disabled. Directly mimicking and implementing a motion of a human on a humanoid robot may not be successful because of the different dynamic characteristics between them, which may cause the robot to fall down due to instability. Using the Zero Moment Point as the stability criteria, this work proposes an Analytical Motion Filter (AMF), which stabilizes a reference motion that can come from human motion capture data, gait synthesis using kinematics, or animation software, while satisfying common constraints.

In order to determine how the AMF stabilized a motion, the different kinds of instabilities were identified and classified when examining the reference motions. The different cases of instability gave more insight as to why a particular motion was unstable: the motion was too fast, too slow, or inherently unstable. In order to stabilize the gait two primary methods were utilized: time and spatial scaling. Spatial scaling scaled the COM trajectory down towards a known stable trajectory. Time scaling worked similarly by changing the speed of the motion, but was limited in effectiveness based on the types of instabilities in the motion and the coupling of the spatial directions. Other constraints applied to the AMF and combinations of the different methods produced interesting results that gave more insight into the stability of the gait.

The AMF was tested using both simulations and physical experiments using the DARwIn miniature humanoid robot developed by RoMeLa at Virginia Tech as the test platform. The simulations proved successful and provided more insight to understanding instabilities that can occur for different gait generation methods. The physical experiments worked well for non-walking motions, but because of insufficient controllability in the joint actuators of the humanoid robot used for the experiment, the high loads during walking motions prevented them from proper testing.

The algorithms used in this work could also be expanded to legged robots or entirely different platforms that depend on stability and can use the ZMP as a stability criterion. One of the primary contributions of this work was showing that an entire reference motion could be stabilized using a single set of closed form solutions and equations. Previous work by others considered optimization functions and numeric schemes to stabilize all or a portion of a gait. Instead, the Analytical Motion Filter gives a direct relationship between the input reference motion and the resulting filtered output motion.

This work received support from the National Science Foundation for part of this work under Grant No. 0730206.

This work also received support from the JOint Unmanned Systems Test, Experimentation, and Research (JOUSTER) facility.

Acknowledgments

This dissertation is dedicated to the memory of Kevin P. Granata. He signed on to my advising committee April 12, 2007, which was the only time I met him. Four days later on April 16, he and 31 other Virginia Tech students and faculty were taken from this world. His research in biomechanics and movement dynamics promised to help and change the world. This work hopes to serve and honor Dr. Granata's memory and legacy.

My greatest thanks go to my wife Tanya Muecke, who has encouraged and supported me through my graduate studies and has played a large part in my will to complete my graduate education. Temporarily setting aside her career goals and supporting me has been an inspiration for myself. Her dedication and love for me will always be cherished.

Many thanks to Dennis Hong for supporting and encouraging me as my advisor throughout my education at Virginia Tech. The senior design project he started in 2005-2006 to design a humanoid robot has been the basis for my research and much of my fascination with robotics. Some of the incredible opportunities he has created for me have been: travel to Korea, China, Canada, and across the US; networking opportunities to explore my field; and countless presentations and demonstrations.

Thanks to Charles Reinholtz who allowed me to get involved with robotics as just a freshman in Virginia Tech's Autonomous Vehicle Team. Undoubtedly, my exposure to unmanned vehicles at the Intelligent Ground Vehicles Competition is what got hooked me on robots.

The members of my dissertation committee, Andrew J. Kurdila, Steve C. Southward, Thurmon E. Lockhart, and Scott L. Hendricks, who have given me their time and concern to better my education and research. An additional thank you to Dr. Lockart and the Locomotion Research Laboratory for providing human motion capture data.

The author would like to thank everyone in RoMeLa for their support. A special thanks to Robert Mayo and Ivette Morazzani who have been pivotal in motivating and supporting me during my graduate education. Also, Seungchul Lim for his help with my understanding of gait generation and stability. The 2005-2008 senior design teams who designed and fabricated the humanoid robot platforms that have allowed the realization of my research.

The Darmstadt Dribblers and Jesse Hurdus for their support in my involvement in the RoboCup competition.

Contents

List of Figures

List of Tables

Nomenclature

$\ddot{\Omega}$	Angular acceleration relative to earth (rad/s^2)
ω	Natural frequency (rad/s)
θ_K	Knee joint angle (rad)
θ_{A1}	Ankle joint angle in the roll or X-direction (rad)
θ_{A2}	Ankle joint angle in the pitch or Y-direction (rad)
θ_{W}	Waist joint angle in the pitch or Y-direction (rad)
θ_{H1}	Hip joint angle in the roll or X-direction (rad)
θ_{H2}	Hip joint angle in the pitch or Y-direction (rad)
θ_{H3}	Hip joint angle in the yaw or Z-direction (rad)
$C(t)$	Time scaling function. Scales time/speed of a motion
C_i	Discrete form of $C(t)$
C_s	Spatial scaling factor
C_T	Constant speed/time scaling value for a motion
g	Earth's gravity
H	$\frac{z_{\mathrm{COM}}}{\ddot{z}_{\mathrm{COM}}+g}$
L_{shk}	Length of shank link (m)
L_{thi}	Length of thigh link (m)
T_s	Step time (s)
v_{xD}	Velocity of the COM of the robot in X direction during the double support phase

v_{xS}	Velocity of the COM of the robot in X direction during the single support phase
$x_{\mathrm{COM}}^{\mathrm{AMF}}$	X direction of AMF or CATF COM trajectory (m)
$x_{\mathrm{COM}}^{\mathrm{Ref}}$	X direction of reference COM trajectory (m)
$x_{\mathrm{ZMP}}^{\mathrm{AMF}}$	X direction of AMF or CATF ZMP trajectory (m)
$x_{\mathrm{ZMP}}^{\mathrm{High}}$	Upper x ZMP limit–corresponds to support polygon (m)
$x_{\mathrm{ZMP}}^{\mathrm{Low}}$	Lower x ZMP limit–corresponds to support polygon (m)
$x_{\mathrm{ZMP}}^{\mathrm{Ref}}$	X direction of reference ZMP trajectory (m)
$y_{\mathrm{COM}}^{\mathrm{AMF}}$	Y direction of AMF or CATF COM trajectory (m)
$y_{\mathrm{COM}}^{\mathrm{Ref}}$	Y direction of reference COM trajectory (m)
$y_{\mathrm{ZMP}}^{\mathrm{AMF}}$	Y direction of AMF or CATF ZMP trajectory (m)
$y_{\mathrm{ZMP}}^{\mathrm{High}}$	Upper y ZMP limit–corresponds to support polygon (m)
$y_{\mathrm{ZMP}}^{\mathrm{Low}}$	Lower y ZMP limit–corresponds to support polygon (m)
$y_{\mathrm{ZMP}}^{\mathrm{Ref}}$	Y direction of reference ZMP trajectory (m)
AMF	Analytical Motion Filter
CATF	Constrained Analytical Trajectory Filter
CMA	Center of mass based Motion Adaptation
COM	Center Of Mass
COP	Center of Pressure
DARwIn	Dynamic Anthropomorphic Robot with Intelligence
DOF	Degrees of Freedom
DSP	Double Support Phase
FFT	Fast Fourier Transformation
FOV	Field of View
GA	Genetic Algorithm
HMCD	Human Motion Capture Data

J	Moment of inertia m^2kg
KAIST	Korea Advanced Institute of Science and Technology
LRL	Locomotion Research Laboratory
m	Mass (kg)
Nom	Nominal
RBFNN	Radial Basis Function Neural Network
Ref	Reference
RoMeLa	Robotics and Mechanisms Laboratory
SSP	Single Support Phase
W	Weighting factor applied to time or spatial scaling factors
ZMP	Zero Moment Point

Chapter 1

Introduction

"I can envision a future in which robotic devices will become a nearly ubiquitous part of our day-to-day lives" –Bill Gates. Whether we recognize it or not, robots are integrating into our lives more directly and personally than ever before. In the early stages of robotics, industry took advantage of the attractive benefits of automation for increasing production and decreasing costs. As the field has evolved, simple robots are available as toys from local electronic stores. Robots have also been taken to an entirely new level with a presence on land, in water, in the air, and in space. Robots' prevalence, success, and intelligence is dictated and grown by the minds that create them. No longer is there even a clear distinction between an ordinary machine and a robot. The Model-T Ford automobile was no more than a machine. However, with the advent of the DARPA Grand Challenge and DARPA Urban challenge, the public has seen that now even cars can be considered "robots." Robotic-like features and concepts creep their way into our everyday lives under the guise of something else. Luxury cars can now autonomously parallel park and vacuum cleaners can autonomously vacuum your house while recharging itself as needed. It seems that the fictional world Isaac Asimov painted of humanoid robots closely interacting with humans is not far off with the development of sophisticated robots like the Honda ASIMO.

Being egotistic creatures, we tend to envision the pinnacle of robotic progress and achievement as the fruition of a humanoid robot with equivalent human intelligence and capacities. Thus, humanoid robots hold a special place in the realm of robotics. It is very easy to think of uses for a fully capable humanoid robot. Anything a human can do could be supplemented or supplanted by a robot. Though not as popular for research and funding as in Asia, humanoid robots still hold a high level of intrigue for Americans. As an answer to the world's fascination with robots and in hopes of contributing to the humanoid robotic community, this work will attempt to add to the knowledge and understanding of humanoid robotics.

1.1 Humanoid robotics

Humanoid robotics is a growing area of interest and research. Thus far, Asia has arguably led the world in the development of sophisticated humanoid robotics with Sony's QRIO, Honda's ASIMO, and Korea's HUBO [2–4]. Part of their vision for humanoid robots is for them to evolve and become as sophisticated and as capable as humans so they can take care of the elderly or disabled. Additionally, humanoid robots hold the potential to serve as general helpers in the household. Another area of research in humanoid robotics examines the relationships between robot and human dynamics. With attempts to make robots move just like a human, we understand more about how humans move. If a robot can move just like a human, then it could be used in studies, experiments, and applications that would be hazardous to humans; such as those examining falling, tripping, or injuries. Additionally, the research could lead to improved prostheses for the disabled. Creating a more sophisticated humanoid robot naturally implies that we will learn more about ourselves in the process of trying to have a robot emulate a human.

1.2 Walking designs

Within humanoid robotics, there are many different designs, approaches, and fields of study. Walking holds an important place within humanoid robotics since it is one of the essential capabilities a humanoid robot must demonstrate in order to appear "more human". Within humanoid walking robots, there are different schools of thought and fields of study. The class of robot's that most closely resemble the futuristic robots depicted in science fiction novels and movies are similar to Honda's ASIMO, Sony's Qrio, and Korea's HUBO [2–4]. These robots have the same shape as and are similarly sized to humans. In some cases, as with the HUBO (Fig. 1.1), sophisticated artificial skins create the appearance of a life-like robot. All of these robots use similar methods to create walking patterns. The gaits are generally generated by specifying a path for each joint or link to follow. When the robot walks, it attempts to closely follow the desired trajectories while maintaining stability.

Another class of humanoid robots focuses more specifically on the walking aspect. More generally meaning a two legged robot, biped robots do not have to be humanoid; meaning they do not have to have essential human features. Instead, many biped robots are simply two legs with no upper body. A popular group of bipeds for researching humanoid/bipedal walking are passive dynamic walking robots, which were pioneered by Tad McGeer [5]. These robots use the dynamics built into the system to walk naturally, with little or no actuation. Gravity is sufficient input to keep the robot walking down a slight incline. The swinging of the robot's legs is natural and unforced. Each step occurs because the robot starts to fall down and a leg swings forward to catch the fall. This class of robots is very interesting because they seem to closely mimic human walking. When we walk, we try to exert as little effort as possible, which is exactly what the passive dynamic walking robots accomplish.

(a) Honda's ASIMO Robot (b) Sony's QRIO Robot (c) Korea's HUBO Robot

Figure 1.1: Pictures of some of the most advanced humanoid robots in the world to date

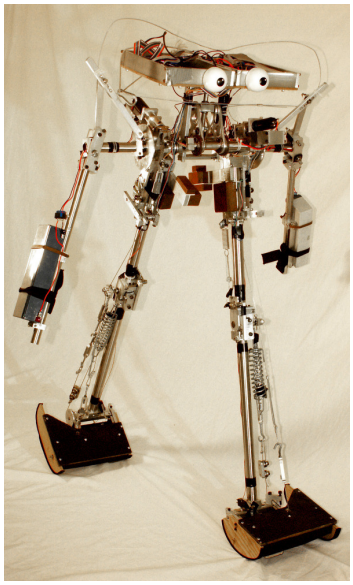
Figure 1.2 shows two well known passive walkers from DELFT and Cornell.

In a class by itself, Marc Raibert's robots do not so much walk as they do "hop" [8–10]. Figure 1.3 shows a single and two legged version of his hopping robots. Instead of planning a gait with trajectories in space or designing the robot to have dynamics to naturally create a gait, the hopping robot plans it's gait in an ad hoc fashion. When airborne, the robot uses its inertial sensors to determine first where it needs to land with its foot in order to stay upright, and second, how it will move its foot once hitting the ground in order to advance the robot in a desired direction. This much like bouncing on a pogo stick down a sidewalk. While Raibert's "walking" robots may be unconventional, they are very successful. In fact, Marc Raibert started his own company, Boston Dynamics, that created a quadruped, Big Dog, that uses the same walking principals and can carry a 340 lb payload at 4 mph over rough terrain [11]. No other walking robot has come close to matching those kinds of capabilities.

1.3 Justification of dissertation topic

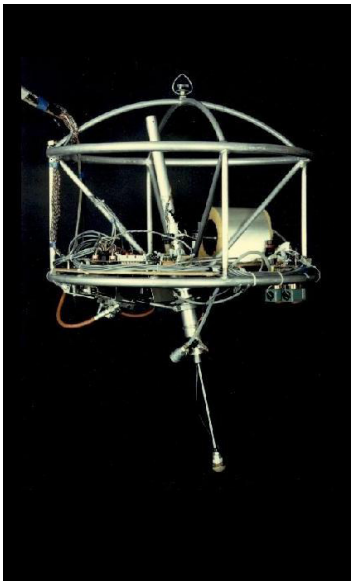
One of the essential aspects in humanoid robotics and walking, is the creation of the walking gait. There are a few different schools of thought when it comes to creating walking gaits for humanoid robots. Many researchers take the approach of creating trajectories for the robot's

(a) DEFLT's passive walking robot, Denise [6]

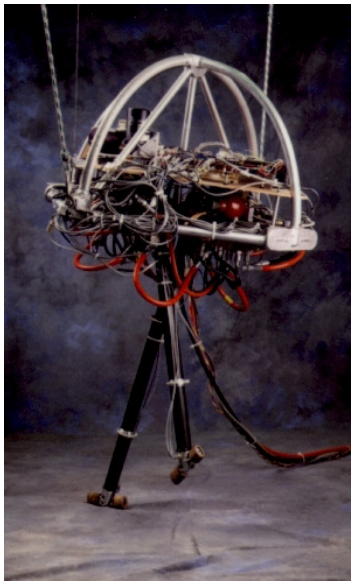


(b) Collin's passive walking robot [7]

Figure 1.2: Pictures of some of the more famous passive dynamic walking bipedal robots



(a) Raibert's one legged hopping robot



(b) Raibert's two legged hopping robot

Figure 1.3: Pictures of two of Raibert's hopping robots [1]

joints to follow. These trajectories are usually based on optimized parametric equations, stability equations, or numerically optimized trajectories through space. In all cases except for gaits created from stability equations, little is known about *why* the resulting stable walking gaits are stable. Much is hidden by a layer of numeric abstraction. Gaits based on stability equations are very straight forward in conveying *why* the resulting gait is stable and how it was achieved. Unfortunately, these gaits do not leave much leeway for variations in the walking pattern.

There seems to be a general lack of understanding of the role of stability criteria in the methods used for stabilizing gaits. Currently, stability criteria is primarily used as a cost function for optimization for stabilizing gaits. Motions are usually based on artificial equations and are then numerically optimized. Methods for generating gaits that do not use optimization techniques, but stability equations, are not very flexible in allowing arbitrary gaits.

1.3.1 Research questions

The first question to address is: Are there methods to stabilize gaits that do not rely on numeric optimization schemes? The follow up question is: Can we develop closed form solutions that can stabilize and arbitrary gait, and do these solutions provide some more meaning and insight to gait stability for humanoid robots?

1.3.2 Proposed solution

The goal of this work is to develop the foundations for an analytical motion filter that is used to stabilize a reference motion for a humanoid robot using analytical solutions. The tasks of the research are:

1. Analyze the relationship between stability and motion for a humanoid robot
2. Develop analytical solutions that can modify and stabilize an otherwise unstable motion
3. Apply and test solutions in simulation and with hardware for a variety of motions to verify the solutions and evaluate the results for further insight to the relationship between stability and motion.

The Analytical Motion Filter will be used to stabilize motions for humanoid robots in the Robotics and Mechanisms Lab. Future research may focus on expanding the analytical solutions to address more constraints, situations, environments, and robotic platforms.

1.4 Dissertation outline

The dissertation begins with a literature review of previous works covering gait generation, control, and filtering. It then outlines the Analytical Motion Filter (AMF). Chapters 4, 5, and 6 detail and explain the different parts of the AMF. The following two chapters (7 and 8) present and discuss the results of applying the AMF to some example motions. The last two chapters (9 and 10) detail the humanoid robot used as the experimental platform and additional work related to humanoid robotics, including: RoboCup, senior design, vision software, motion software, and interface software.

Chapter 2

Review of Literature

There are at least three primary areas of study in humanoid robot motion: Gait generation, motion control, and motion filtering. Gait generation is primarily concerned with coming up with the trajectories needed to create a dynamically stable motion or deriving the necessary dynamic properties of the robot needed to create a stable gait. Gait generation can be accomplished by starting with a known physical system, and then using its dynamic properties to help create a stable motion. Another way of generating gaits is to start with an undetermined system and design it in such a way that its natural dynamics create a stable gait (passive walking or passive motion). Motion control is primarily concerned with handling disturbances encountered during the robot's motion. These disturbances occur due to collisions with other objects, ground contact, or joint errors. Motion filtering, like gait generation, creates trajectories needed for a dynamically stable motion, but requires an input or reference gait. Motion filtering attempts to modify the reference gait as needed in order to achieve a stable gait.

2.1 Gait generation methods

There are many different kinds of gait generation methods in the field of humanoid robotics. Most of the methods eventually involve some kind of numeric optimization to create a stable gait. However, some methods involve using more interesting approaches that do not require trial and error to generate a stable walking motion.

2.1.1 Parametric gait synthesis

One of the most common gait generation methods in humanoid robotics, parametric gait generation, is one of the easiest generation methods to implement, but is generally not

computationally efficient. Generally a set of arbitrary parameterized functions describe the motion of the robot. The functions usually describe the ankle, hip, or torso trajectory and orientation. Typically, the functions consists of sines, cosines, and combinations of linear functions [12–19]. In order to create a stable gait, the function parameters are varied in a numeric optimization algorithm whose cost function usually includes: stability, speed, and/or joint torques. There is no consensus on what the "best" or most "optimum" cost function is that is needed to create a gait. However, there is a general consensus that the Zero Moment Point (ZMP) criterion is the best for determining the robot's stability.

Specifically, Qiang Huang's parametric piecewise gait is discussed in more detail in Sec. 7.2.1.

2.1.2 Using genetic algorithms and neural networks

A variation on parametric gait synthesis, genetic algorithms and neural networks have been used to generate stable humanoid robot motion [20–23]. The genetic algorithm (GA) uses initial, final, and intermediate conditions as the GA variables. The cost function considers joint torques, stability, and kinematic limits of the robot. The neural network (Radial Basis Function Neural Network RBFNN) is used for online/real-time gait generation. The RBFNN learns from the GA and is then utilized while the robot is in motion. The RBFNN accepts step length and step time as inputs and then generates the same parameters as the GA, but much faster. The GA generates an optimal solution requiring about 10 minutes of processing. However, the RBFNN taked approximately 200 μs with only about a 3.2% difference in the cost function compared to the GA.

2.1.3 Human motion capture

Another popular method for generating gaits is using human motion capture data (HMCD) [24]. Similar to Sec. 2.3 and filtering, Dasgupta developed a method to stabilize a reference motion, but focuses on reference motions from human motion capture data. Again, a numeric optimization function is used to modify the HMCD to follow a desired stable ZMP trajectory. The angles of the joints are modified with a Fourier series function described as

$$\delta\theta_i = \sum_n a_{in} \sin\left(\mathrm{n}\omega_{\omega\mathrm{i}} t\right) + b_{in} \cos\left(\mathrm{n}\omega_{\omega\mathrm{i}} t\right) \tag{2.1}$$

where $\delta\theta_i$ represents the corrective motion at the i^{th} joint, a_{in} and b_{in} represent the correction amplitude, and $\omega_{\omega\,\mathrm{i}}$ is the frequency of the corrective motion. The various parameters are optimized so that the resulting motion closely follows the desired ZMP trajectory while considering some other cost functions (ground reaction forces, etc).

HMCD can also be used directly to determine joint angles trajectories for the robot. However, when doing this, the motion is not guaranteed to be stable. This is discussed in more detail in Sec. 7.2.4.

2.1.4 Cycloids

A somewhat unique gait generation method, the cycloid-function-based gait, used for the HUBO robot at KAIST university in Korea, does not explicitly use a numeric optimization function to determine the appropriate parameters governing the robot's motion [4, 25–27]. Instead, Kim claims that the values were determined experimentally. Presumably by either changing the parameters manually or by some other undisclosed method. The cycloid function is a combination of a cosine and linear function. The cosine function creates accelerations necessary for a stable ZMP trajectory while the linear function actually moves the robot forwards. This gait is discussed in more detail in Sec. 7.2.2.

2.1.5 Inverted pendulum

The inverted pendulum model is one of the most common simplifications used for generating a walking gait for a humanoid robot [28–33]. By ignoring moments of inertia and the coriolis effects, the robot can be approximated as a collection of point masses. By further assuming that the majority of the mass is located in the upper body of the robot or at least follows the same trajectory as the upper body, then the robot can be simplified to an inverted pendulum if the gait is generated properly. An inverted pendulum is very easy to model and derive equations for. You can imagine that an inverted pendulum could "walk" if it oscillated while changing its contact point with the ground.

A slight variation on the inverted pendulum, the table-cart model depicts the robot as a cart able to roll on a table affixed to an inverted pendulum [34]. This accomplishes the same result as allowing the inverted pendulum to change in length as many others have done.

2.1.6 Stability equations

An alternative to using optimization functions to tweak gait parameters in order to achieve a stable walking gait is to use dynamic stability equations to generate a gait. Takanishi showed in [35] that by using a Fast Fourier Transformation (FFT), a gait motion could be derived from a stable ZMP trajectory. Since a walking gait is inherently cyclic, it makes sense that an FFT is an appropriate tool for deriving a solution. Lim showed in [36–39], that hyperbolic sines and cosines could be used as a solution for formulating gaits with a desired ZMP trajectory. Hyperbolic sines and cosines are simply combinations of exponential functions,

which reflect the dynamics of an inverted pendulum. Lim's hyperbolic gait is discussed in more detail in 7.2.3.

2.1.7 Passive

A more popular method for generating bipedal gaits in America is utilizing passive dynamics. Instead of designing a stable gait for a given robot, a robot is designed to naturally perform a stable dynamic gait. Tad McGeer's passive dynamic biped walker is one of the most well known passive dynamic bipeds [5, 40]. McGeer states, "There exists a class of two-legged machines for which walking is a natural dynamic mode: Once started on a shallow slope, such a machine will settle into a steady gait quite comparable to human walking, without active control or energy input." Essentially, McGeer designed a two legged robot that could walk down a shallow slope with no additional energy input into the system besides gravity. Collins in [41] showed that a passive dynamic walking robot could walk on level ground using small actuators that essentially substituted for gravity's input to the system. Ono showed in [42] that a biped mechanism with actuators at the hip and no actuation at the knee could overcome gravity and walk up inclines. Kuo and other extended the passive dynamic walking mechanism to include passive dynamic motion in the lateral direction [43, 44]

2.1.8 Hopping

Another class of gait generation blurs the line between what is gait generation and what is gait control. Hopping robots essentially implement a control algorithm that is then modified to move the robot. One of the most well known hopping robots is Raibert's machine [8–10]. There are 5 discrete states the robot goes through during hopping. During the Flight phase, the active leg leaves the ground and shortens, idle and active legs exchange roles, and the new active leg lengthens for landing and is positioned for landing. Afterwards, the active leg touches the ground during the loading phase. Once the leg touches the ground, the compression phase takes place and the active leg spring shortens while the pitch of the robot is adjusted by the active hip controlled by a motor. Afterwards, the thrust phase consists of the active leg spring lengthening while the active hip is still actuated. Finally, during the unloading phase, the active leg spring approaches full length and no torque is applied at the hip. During flight, the landing position of the foot is chosen based on the desired motion for the robot. Kajita implemented a version of a hopping robot in an experiment on creating a running gait for a robot in [45]. A sort of inverted pendulum was used to model the robot's motion, but the length of the pendulum consisted of a spring and damper. Consequently, in addition to the typical inverted pendulum motion, vertical oscillation was added to move the center of mass (COM) up and down. With the center of mass moving up and down, it is easy to create a running motion by lifting the legs off the ground when the COM is at its peak in oscillation.

2.2 Control methods

In addition to gait generation, stability control is oftentimes needed to create a successfully stable walking robot. Frequently, stability control is closely linked with the gait generation method used. However, since the two can be implemented independently, it is worthwhile to examine control methods separately. Many robots, without an additional layer of stability control, will fall over while performing an initially planned stable walking pattern. The fall occurs because of external disturbances and their perturbation throughout the system. "Therefore, we have to measure the walking state online by using information [from sensors] and change the [reference] walking pattern gradually [4]."

2.2.1 HUBO's control methods

The KHR-2 and KHR-3 (also named HUBO) developed at the Korea Advanced Institute of Science and Technology (KAIST) use a combination of control algorithms to achieve stable humanoid walking in the presence of disturbances [4, 25–27, 46]. HUBO uses a force/torque and pressure sensor in the feet to measure external loads directly. Additionally, accelerometers and rate gyros are used to determine the robot's orientation. Finally, a CCD camera is used to process and recognize its environment. The HUBO robot implements many different types of controllers in order to achieve a stable walking gait. There are many other variations of gait control that are similar to HUBO's, but will not be detailed in this work [47, 48].

Kim classifies gait control into three categories: balance control, walking pattern control, and predicted motion control. The balance control stabilizes the robot while walking in real-time. The walking pattern control modifies the walking pattern after each step to create a "better" walking pattern for the next step. The predicted motion control is used to prevent instabilities that cannot be corrected by the balance or walking control.

Real-time balance control

Kim further divides real-time balance control into 5 categories: damping controller, ZMP compensator, soft landing controller, landing timing controller, and upright pose controller. The damping control helps to prevent oscillations in the robot's gait during SSP caused by the robot's links flexing. The controller is applied by using the actuators at the ankle joints. The ZMP compensator controller ensures that the ZMP path follows the desired ZMP trajectory. The controller is applied by modifying the hip trajectories during the gait. The soft landing controller reduces impact when the foot strikes the ground when ending the SSP. The soft landing controller is implemented by orienting the foot to be perfectly parallel to the ground during contact using the torque sensors in the foot, and also by modeling the robot as an inverted pendulum sitting on an spring damper system. "The landing timing controller prevents the robot from being unstable during landing by [modifying the] walking

pattern schedule." If the foot does not hit the ground when expected, the landing timing controller ensures that the schedule of the main walking gait is modified so that when contact is made, the gait can continue as planned. Finally, the upright pose controller uses the height of each leg and the pitch of the ankles to ensure that that the upper body remains vertical at all times–even when walking up inclines. This is done slowly (frequency of 0.1 Hz) because the inclination of the ground changes slowly.

Walking pattern control

Kim divides the walking pattern control in two sections: pelvis swing amplitude controller and torso roll/pitch controller. The pelvis swing amplitude controller modifies the amplitude of the lateral swing of the pelvis during walking motions in order to attempt to achieve an average ZMP location that follows the reference ZMP trajectory. By swinging the pelvis more or less laterally, the ZMP can also be controlled to move more or less in the lateral direction. The torso roll/pitch controller operates by considering the robot as a double inverted pendulum. The torso will naturally want to sway forwards and back while walking. In order to reduce the sway, the hip location can be adjusted to balance the upper torso, which would be similar to balancing the upper pendulum on a double inverted pendulum.

Predicted motion control

Kim divides the predicted motion control into two parts: landing position controller and tilt over controller. The landing position controller searches for the best landing point of the swing foot on the ground. The landing point is chosen by measuring the angular momentum of the robot's torso. If the torso is tilting forwards, then the landing point of the foot is adjusted to be slightly further or longer than initially planned. You can imagine getting shoved in the back and extending your foot out further in front of you to catch your fall. The tilt over controller is supposed to prevent the robot from falling over in the lateral direction. The controller is implemented by modifying the roll of the ankle joint according to an experimentally derived sinusoidal compensation algorithm.

2.3 Filtering methods

Filtering methods differ from generation and control methods in that they generally rely on a previously generated gaits as an input and the control is not used online, but rather in the process of filtering the motion. Filtering motions is not a completely novel concept. Animators and motion researchers commonly modify and filter gaits so that the final subject can successfully perform a version of the original gait [49, 50]. For a time, the dynamic stability of the motion was ignored by animators because physically accurate models were not

as important. However, recently in an effort to create more realistic animations, physically accurate models and simulations have become more desirable and spurred on more research in the area. Gait researchers are also interested in the idea of filtering motions and have always been concerned with the dynamics and physical meaning behind the gaits. However, most, if not all, methods for filtering gaits are based on numeric algorithms or filters, which can conceal some of the other insight behind why the filtering works.

Witkin describes an animation tool that forms motions for a physically valid human model that obeys constraints and optimization criteria [51]. The method "entails the numerical solution of large constrained optimization problems, for which a variety of standard algorithms exist. These algorithms, while relatively expensive...". In previous efforts, the character animation was created by solving for the character's motion and time-varying muscle forces progressing sequentially through time. Instead, by solving for the muscle-forces over the entire time interval, he formulated a model called *spacetime constraints*.

Witkin's key contribution was to allow the effects of constraints to propagate forwards and backwards in time, unlike some previously developed algorithms that only considered the effects of a constraint at a given moment. The animation model character, called "Luxo," was able to perform a variety of physically valid motions specified by the animator. However, for all of the motion generated, numeric solutions were used with optimization functions in order to generate the final motion. Though Witkin helped contribute to creating a more efficient and useful tool for animators, the process does not provide much insight into the stability of the motion.

Tak describes "a constraint-based motion editing technique that on the basis of animator-specified kinematic and dynamic constraints, converts a given captured or animated motion to a physically plausible motion" [52]. The method is used by viewing the motion as sequential frames. The filter is "basically a concatenation of unscented Kalman filters[s] and ... least-squares filter[s]...we can add or remove some or all of the kinematic and dynamic constraints depending on whether they significantly affect the type of motion being animated." Position, kinematic, balance (using Zero Moment Point), torque limit, and momentum constraints are all proposed as available constraints for this filter. The Kalman and least-squares filters converge to form the final motion. However, Tak states that "it is virtually impossible to characterize the analytical condition under which the repeated applications of the Kalman filter and least-square filter converge...[When] the editing involves dynamic constraints, we must deal with the velocities and accelerations, which are much more sensitive than positions." Though Tak's filter is able to create fantastic motions like dancing, wide stepping, golf swings, limbo walking, and jump kicks, the filtering methods use numeric filters and require iterative processing to create the stable filtered motion. Additionally, as Tak admits, there is not even a way to analytically determine how long it will take the filtering to converge, let alone create an analytical solution.

Yamane, describes a "dynamics filter, an online, full-body motion generator that converts a physically infeasible reference motion into a feasible one for the given human figure" [53].

The filter is able to account for collisions and contacts while creating a stable motion online, without prior knowledge of the rest of the motion. Reference motions that could be used include human motion capture data, motions created using animation software, or motions synthesized based on kinematics. The motion is generated based on an optimization scheme that computes joint values that satisfy the motion as well as dynamic constraints, which are expressed as constraints on the joint values. Contact and collisions are included in the filter using numeric trial and error procedures. Again, as Tak had expressed in his work, Yamane states that he could not find a systematic method for tuning the parameters for the filter. Both Tak and Yamane used numeric methods to filter motions and also could not determine a way to analytically describe the parameters used in the filters.

Chapter 3

The Analytical Motion Filter (AMF)

3.1 Introduction

Humanoid robots that can mimic human motions can be very useful in many areas and applications. A bipedal robot that walks like a human and has similar dynamics to a human could be used for gait studies in place of a human subject when the study would otherwise be dangerous; such as studies involving falling down or tripping. Humanoid robots can also be used for entertainment, imitating human motions or actions such as dancing. Advancing humanoid robotics could also lead to insight into improved prostheses for the handicapped. Additionally, a capable life size humanoid robot could be used to assist the elderly or disabled at home or in hospitals since the robot would be able to carry out any task a human could.

Making a humanoid robot precisely mimic a desired human motion is not always possible because of the differences in dynamic properties between the two: size, mass distribution, moment of inertia, actuator performance, etc. A robot may be able to copy the joint trajectories of a human's motion, but may lose balance because of the difference in mass distribution. Additionally, certain parts of a motion may be too fast or require too much torque for the robot's actuators. Current approaches exist such as "Dynamic Filters," which use a reference motion such as, human motion capture data, motions created by hand using animation software, or gait synthesis using kinematics, and convert an otherwise infeasible or unstable motion into a stable feasible motion [53]. These approaches filter reference motions using numeric trial and error optimization methods.

The work presented here proposes an analytical approach to filter (or create) a stable motion from a reference motion. The analytical solutions are faster than trial and error solutions and provide insights into the stability of the system that would otherwise be difficult to identify in a numeric optimization scheme. The resulting filtered motion should be as similar to the reference motion as possible while satisfying constraints. The definition of "similar" dictates the filtering methods used so as to achieve an optimum similarity. The proposed analytical

motion filter (AMF) presented here currently does not consider the physical limits of the actuators (torque, speed, etc) or the possibility of link collisions. Additionally, it is assumed that the reference motion is possible for the robot to perform without violating the robot's kinematic and physical range of motion.

3.2 Stability

The analytical solutions and algorithms used in the AMF are derived and based on the stability of the robot. With a few exceptions (i.e. the Honda ASIMO, the Sony QRIO, the KAIST HUBO, and Tad McGeer's passive walker [2–5, 41]), most legged robots today walk using what is called the static stability criterion. The static stability criterion is an approach to prevent the robot from falling down by keeping the center of mass of its body over its support polygon by adjusting the position of its links and the pose of its body very slowly to minimize dynamic effects [4]. Thus, at any given instant in a statically stable motion, the robot could "pause" and not fall over. Statically stable walking is generally energy inefficient since the robot must constantly adjust its pose in such a way to keep the center of mass of the robot over its support polygon, which generally requires large torques at the joint actuators and/or awkward motions. Humans do not walk in this manner. Rather, we naturally walk dynamically, with the center of mass almost always outside our support polygon. Human walking can be interpreted as a cycle of continuously falling and recovering: a cycle of exchanging potential and kinetic energy in the system–much like the motion of a pendulum. We fall forward and catch ourselves with our swinging foot while continuing to walk forward. This falling motion allows for our center of mass to continually move forward and does not expend energy to stop the momentum. The lowered potential energy from this falling forward motion is then increased and recovered by the supporting leg lifting up the rest of our body.

One natural question that arises when examining dynamic walking is how to classify the gait's stability. Since the center of mass can travel outside of the support polygon, the static stability criterion is not useful. To address this, dynamic stability is commonly measured using the Zero Moment Point (ZMP), which is a point defined as "where the influence of all forces acting on the mechanism can be replaced by one single force" without a moment term [54]. If this point remains in the support polygon, then the robot can apply some force or torque to the ground, which in turn means the robot can have some influence over the motion of itself (the system). Once the ZMP moves to the edge of the foot, the robot is unstable and can do nothing to recover without extending the support polygon (planting another foot or arm). Another way of interpreting the ZMP is to consider the center of pressure (COP), which is the same as the ZMP when a foot is in contact with the ground. Figure 3.1 shows a visualization of the COP. The vector drawn as "N" starts at point "P", where if there is no moment, coincides with the COP and the ZMP. It is also important to note that the ZMP stability criterion depends on friction or a no-slip condition.

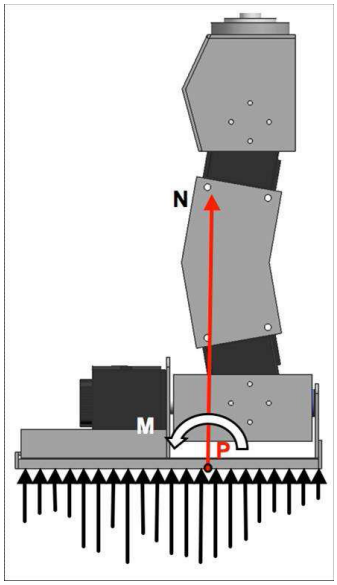


Figure 3.1: Diagram showing the center of pressure on a robot's foot, which corresponds to the ZMP (M=0 in diagram)

Yet another way of understanding the ZMP is to compare it to the COM as a measure of static stability. If the COM is within the support polygon of the robot and the robot "freezes," the robot will not fall over. This is the definition of static stability. In truth, the actual ZMP of a robot can never leave the support polygon. However, a planned ZMP trajectory may leave the support polygon due to the planned robot motion. The result will be that the robot cannot physically follow the desired planned motion since it would require the ZMP to exist outside of the support polygon. Therefore, in order to ensure that it is possible for the robot to follow a desired COM trajectory and consequently a desired motion, the planned ZMP trajectory must remain within the support polygon of the robot for the entire motion.

While most robot's use the static stability criteria, there is a lot of research interest in dynamic walking robots. In most cases, the stable dynamic gaits generated are parameterized gaits that are optimized using the ZMP as a stability criterion or stable hyperbolic gaits generated by solving the ZMP equation for a center of mass trajectory. In addition to being used to generate gaits, the ZMP can be measured directly or estimated during walking to give the robot feedback to correct and control its walking [4, 18].

3.3 Analytical Motion Filter approach

A typical humanoid robot has 20+ degrees of freedom (DOF). With so many DOF, analytical closed form solutions are extremely difficult, if not impossible, to come by when generating stable walking gaits. This has given rise to many numeric brute force approaches that simply seek to solve an optimization function that includes stability in the cost function [18]. Instead of generating gaits blindly, the AMF approach seeks to generate and form gaits based on closed form analytical solutions that are based on ZMP stability. Instead of convoluting the filtering process, the AMF will show a clear and consistent relationship between the reference motion and filtered motion that is known ahead of time, before filtering. These solutions can be obtained from a high DOF system because the ZMP can be simplified to depend only on the three dimensional location of the center of mass (COM)(See Sec. 4.1). Simplifying a 20+ DOF robot into a 3 DOF system allows for analytical closed-form solutions useful for creating stable motions [38]. The stability of the system then directly depends on the COM trajectory. The COM trajectory of a reference motion is easily calculated from the reference joint angles of the robot. However, instead of generating motions and gaits as Lim has, the analytical solutions in the AMF will be used to modify the COM reference trajectory (Constrained Analytical Trajectory Filtering, CATF, see Sec. 4). The stability of the entire system can be guaranteed as long as the robot's kinematic and physical configuration allows it to follow the resulting stable COM trajectory. The modified stable COM trajectory is used in conjunction with the reference joint angles from the reference motion to determine what the new set of joint angles (AMF joint angles) need to be in order for the robot to have a stable motion (COM-based Motion Adaptation, CMA, see Sec. 6). Figure 3.2 illustrates the process for the AMF as described. During the entire process, the resulting filtered motion should be as similar to the reference motion as possible with differences minimized.

Unstable motions are defined as those with a ZMP trajectory that goes outside of or to the edge of the support polygon. The ZMP is the stability criterion used in this work and is also the basis for the analytical solutions. In order to stabilize the reference motion, the CATF modifies the spatial and/or time information of the reference motion. The specifics for how the CATF modifies the reference motion depend on what characteristics of the reference motion are desired to carry over into the filtered motion, which are expressed as constraints. For example, if the robot's paths through space are paramount (i.e. when moving in a constrained space), then, if possible, the CATF will only modify the time/timing information of the motion, leaving the paths of the motion unchanged. This would involve either speeding up or slowing down the motion. Conversely, if the robot needs to appear to go through the same motion in the same amount of time, the CATF will only modify the spatial information of the COM reference trajectory, and only enough to stabilize the motion. Consequently, combinations of space and time modifications with space and/or time constraints create other unique algorithms the CATF uses for creating a stable motion (See Sec. 5). Additional intricacies to the algorithms appear when considering the kinematic constraints of the robot.

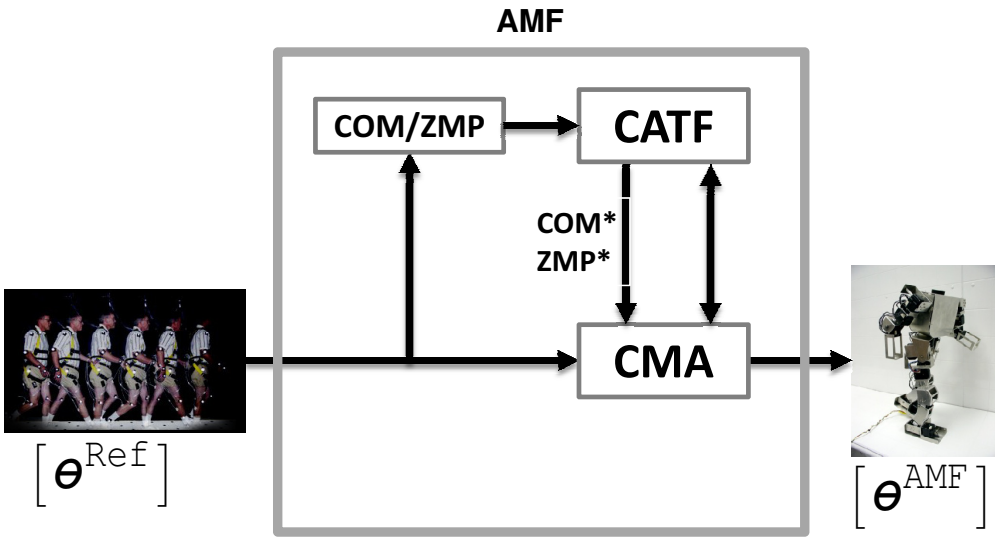


Figure 3.2: Flowchart outlining the Analytical Motion Filter (AMF)

Once the CATF filters the reference COM trajectory, the result is what will be called the "AMF COM trajectory". Though it is straight forward to calculate the reference COM trajectory from the reference motion, which consists of joint angle trajectories, there are an infinite number of solutions to do the reverse–calculate the new AMF joint angle trajectories from the AMF COM trajectory. Calculating the new AMF joint angle trajectories (the AMF motion) is accomplished in the CMA (COM-based Motion Adaptation). Since one of goals of the AMF is to try and minimize differences between the reference and filtered motion, calculating the *AMF* joint angle trajectories is accomplished using the *reference* joint angle trajectories as a starting point and deviating only enough to create a stable COM trajectory. When minimizing differences between the reference and AMF motion, the optimum solution depends on the definition of the difference between the reference and AMF motion. Hence, not only are there many different algorithms that could be used for calculating a set of joint angle trajectories from a COM trajectory, there are many different definitions of the cost function that relates changes from the reference motion to the filtered motion. Any number of algorithms could be used for the CMA, but the methods proposed here aim to try and create a starting point for a CMA that is based on analytical solutions and that minimizes changes between the reference and filtered joint angle trajectories.

The foot positions of the robot can completely determine the kinematic constraints of the system. Additional constraints can be enforced depending on algorithms used in the CMA. The kinematic constraints and constraints from the CMA are applied in the CATF so that the resulting filtered COM trajectory is kinematically feasible. The methods used in the CATF (space and/or time) also correspond to the kind of modifications/adaptations that occur in the CMA. For example, if time modification is used in the CATF, a form of time modification will be used in the CMA.

In this research the AMF will be verified by using reference motions from human motion capture data, motions created by hand using animation software, and gaits synthesized using kinematics. The filtered motions will be tested first in simulation, and then verified on a humanoid bipedal robot. The robot used for experimental verification is the Dynamic and Anthropomorphic Robot with Intelligence (DARwIn), which was developed at the Robotics and Mechanisms Laboratory (RoMeLa) at Virginia Tech.

Chapter 4

Constrained Analytical Trajectory Filtering (CATF)

Constrained Analytical Trajectory Filtering (CATF) is the portion of the Analytical Motion Filter (AMF) that takes a COM reference trajectory, which is calculated from the reference motion, and filters it so it is stable, creating the AMF filtered COM trajectory. Unstable motions are defined as those with a ZMP trajectory that goes outside of or to the edge of the support polygon. The ZMP is the stability criterion used in this work and is also the basis for the analytical solutions. In order to stabilize the reference motion, the CATF modifies the spatial and/or time information of the reference motion. The specifics for how the CATF modifies the reference motion depend on what characteristics of the reference motion are desired to carry over into the filtered motion, which are expressed as constraints. For example, if the robot's paths through space are paramount (i.e. when moving in a constrained space), then, if possible, the CATF will only modify the time/timing information of the motion, leaving the paths of the motion unchanged. This would involve either speeding up or slowing down the motion. Conversely, if the robot needs to appear to go through the same motion in the same amount of time, the CATF will only modify the spatial information of the COM reference trajectory, and only enough to stabilize the motion. Consequently, combinations of space and time modifications with space and/or time constraints create other unique algorithms the CATF uses for creating a stable motion (See Sec. 5). Additional intricacies to the algorithms appear when considering the kinematic constraints of the robot.

4.1 Zero Moment Point investigation

The equations governing the dynamic stability of a robot, the Zero Moment Point (ZMP), are also what govern the solutions in this work. To explain how the solutions for this work were derived, we will start with the basic definition of the ZMP. Using the axes from Fig.

4.1, the ZMP can be defined as

$$x_{\text{ZMP}} = \frac{\sum_{i=1}^{n} m_i \left(\ddot{z}_i + g\right) x_i - \sum_{i=1}^{n} m_i \cdot \ddot{x}_i \cdot z_i - \sum_{i=1}^{n} J_{\text{iy}} \cdot \ddot{\Omega}_{\text{iy}}}{\sum_{i=1}^{n} m_i \left(\ddot{z}_i + g\right)} \tag{4.1a}$$

$$y_{\text{ZMP}} = \frac{\sum_{i=1}^{n} m_i \left(\ddot{z}_i + g\right) y_i - \sum_{i=1}^{n} m_i \cdot \ddot{y}_i \cdot z_i - \sum_{i=1}^{n} J_{\text{ix}} \cdot \ddot{\Omega}_{\text{ix}}}{\sum_{i=1}^{n} m_i \left(\ddot{z}_i + g\right)} \tag{4.1b}$$

where m_i is the mass of each link i, g is gravity, J_{ix} and J_{iy} are the moments of inertia, $\ddot{\Omega}_{ix}$ and $\ddot{\Omega}_{iy}$ are the absolute angular accelerations of each link, and n is the number of links [54, 55]. If the angular accelerations are small or the moments of inertia are relatively small, the equation can be simplified, as many others have [29, 31–34], as:

$$x_{\text{ZMP}} = x_{\text{COM}} - \frac{\ddot{x}_{\text{COM}} \cdot z_{\text{COM}}}{\ddot{z}_{\text{COM}} + g} \tag{4.2a}$$

$$y_{\text{ZMP}} = y_{\text{COM}} - \frac{\ddot{y}_{\text{COM}} \cdot z_{\text{COM}}}{\ddot{z}_{\text{COM}} + g} \tag{4.2b}$$

where COM is the center of mass. This simplification is representative of an inverted pendulum, which is often used for modeling bipedal robot gaits. An additional simplification can be made when z_{COM} is relatively constant and $\ddot{z}_{\text{COM}}$ is relatively small compared to g:

$$x_{\text{ZMP}} = x_{\text{COM}} - \ddot{x}_{\text{COM}} \cdot H \tag{4.3a}$$

$$y_{\text{ZMP}} = y_{\text{COM}} - \ddot{y}_{\text{COM}} \cdot H \tag{4.3b}$$

where H is

$$H = \frac{z_{\text{COM}}}{\ddot{z}_{\text{COM}} + g} \tag{4.4}$$

The result, Eq. 4.3, is an ordinary differential equation, which can be solved as

$$x_{\text{COM}}(t) = a_1 e^{\frac{t}{\sqrt{H}}} + a_2 e^{-\frac{t}{\sqrt{H}}} + x_{\text{ZMP}} \tag{4.5a}$$

$$y_{\text{COM}}(t) = a_3 e^{\frac{t}{\sqrt{H}}} + a_4 e^{-\frac{t}{\sqrt{H}}} + y_{\text{ZMP}} \tag{4.5b}$$

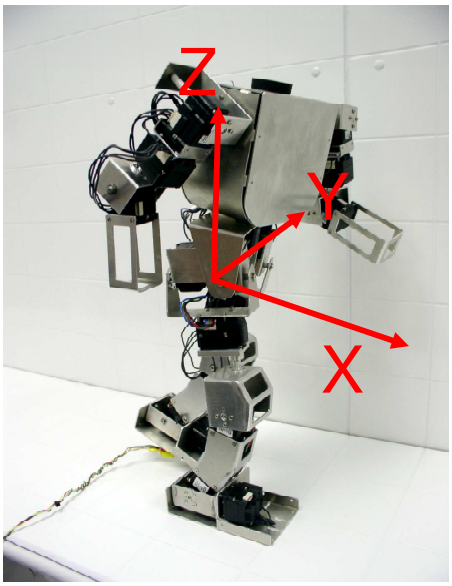


Figure 4.1: Diagram showing the axis labeling convention of a humanoid robot

where a_1, a_2, a_3, and a_4 are the respective coefficients of the exponentially increasing and decreasing parts of Eq. 4.5 and the ZMP values are held constant.

Equations 4.3 and 4.5 serve as the basis for how the CATF modifies the COM trajectory of the reference motion in order to stabilize the motion as a whole. More specifically, how the AMF modifies the reference motion depends largely on the prescribed desirable characteristics of the reference motion. The AMF has two distinct approaches to modifying the reference motion: spatial and time. The approaches are used in conjunction with each other based on the constraints placed on the AMF, which can be designed to preserve desirable aspects of the reference motion. Constraints typically include factors such as no-ground-penetration or joint limits. An example of a constraint that is designed to preserve a desirable aspect of the reference motion would be: the filtered version of the motion must start and end exactly as the reference motion.

To reiterate, some of the major assumptions of the simplifications to the ZMP equations are:

- Negligible moment of inertia of the robot
- The height of the COM is relatively constant
- The acceleration of the height of the COM is relatively small compared to gravity

How the AMF addresses these assumptions is discussed in Sec. 4.4.

4.2 Spatial scaling

The AMF can manipulate the COM trajectory in order stabilize the reference motion while leaving the motion's time information (dimension) unchanged. For simplicity, only the process for filtering the X-direction will be shown in this section. The x_{ZMP} and y_{ZMP} are decoupled as seen in Eq. 4.1. The same process shown for the X-direction can be directly applied to the Y-direction without impacting stability in the X-direction, except for the cases of non-rectangular support polygons and when at kinematic extremes (these considerations are discussed in Sec. 4.4). Additionally, in keeping with the overall goal of the AMF, the COM trajectory is changed only as much as needed in order to achieve stability for the entire motion so as to reduce the differences between the filtered and reference motions.

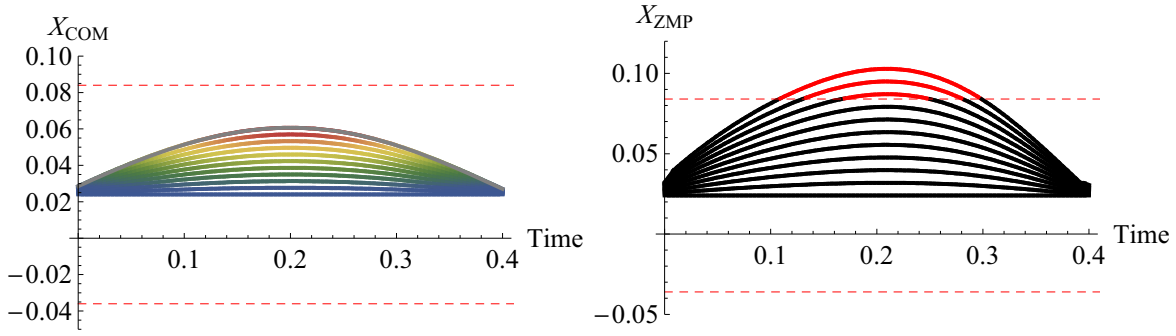
The principle behind the spatial modification with the goal of achieving stability, is to scale the COM trajectory down towards some nominally stable trajectory. To understand this concept, we first consider the motion of a robot standing perfectly still in a stable pose, which would result in COM path described as

$$x_{\text{COM}}(t) = D \tag{4.6}$$

where $x_{\text{ZMP}}^{\text{Low}} < D < x_{\text{ZMP}}^{\text{High}}$, and D is a constant. Therefore, the X-acceleration is 0 and $x_{\text{COM}} = x_{\text{ZMP}}$. $x_{\text{ZMP}}^{\text{Low}}$ is the lower limit that the ZMP is permitted to be, which is usually the back of the robot's grounded foot. $x_{\text{ZMP}}^{\text{High}}$ is the upper limit that the ZMP is permitted to be, which is usually at the front of the robot's grounded foot. Since standing still with the COM within the support polygon (ZMP Limit) is inherently stable, scaling the reference COM trajectory down towards a constant position in the support polygon will eventually lead to a stable motion since accelerations will approach 0 as the trajectory starts to resemble a constant and all of the COM positions approach the stable region. Using $x_{\text{ZMP}}^{\text{nom}}$, which is usually halfway between $x_{\text{ZMP}}^{\text{Low}}$ and $x_{\text{ZMP}}^{\text{High}}$ (or can be chosen elsewhere within the support polygon), the CATF scales the reference motion down towards $x_{\text{ZMP}}^{\text{nom}}$ until the filtered motion is stable. The COM trajectory from the reference motion, $x_{\text{COM}}^{\text{Ref}}$, is scaled, resulting in the AMF COM trajectory defined as

$$x_{\text{COM}}^{\text{AMF}} = x_{\text{COM}}^{\text{Ref}} + C_s \left(x_{\text{ZMP}}^{\text{nom}} - x_{\text{COM}}^{\text{Ref}} \right) \tag{4.7}$$

which can create a stable, scaled-down version of the reference motion that centers around the nominal ZMP location, where C_s is the spatial scaling factor and $x_{\text{COM}}^{\text{AMF}}$ is the filtered COM trajectory. Essentially, the COM trajectory is scaled so it is closer to the nominal ZMP value with smaller accelerations and C_s dictating how much scaling is needed to stabilize the motion. Figure 4.2 helps to illustrate this process with a fabricated example reference motion. When C_s is 0, $x_{\text{COM}}^{\text{AMF}} = x_{\text{COM}}^{\text{Ref}}$ and when C_s is 1, $x_{\text{COM}}^{\text{AMF}} = x_{\text{ZMP}}^{\text{nom}}$. Again, because $x_{\text{COM}}^{\text{AMF}}$ is attracted to $x_{\text{ZMP}}^{\text{nom}}$ and the trajectory is "damped out," resulting in lower accelerations, the ZMP trajectory, $x_{\text{ZMP}}^{\text{AMF}}$, also approaches $x_{\text{ZMP}}^{\text{nom}}$.



(a) $x_{\text{COM}}^{\text{AMF}}$ attracted to $x_{\text{ZMP}}^{\text{nom}}$.

(b) $x_{\text{ZMP}}^{\text{Ref}}$ attracted to $x_{\text{ZMP}}^{\text{nom}}$

Figure 4.2: Plots showing how $x_{\text{COM}}^{\text{AMF}}$ and $x_{\text{ZMP}}^{\text{Ref}}$ are attracted to $x_{\text{ZMP}}^{\text{nom}} = 0$ using Eq. 4.7. C_s varies from 0 to 1 by increments of 0.1. The reference COM and ZMP trajectories are the most arched trajectories plotted. The dotted red lines represent the ZMP limits.

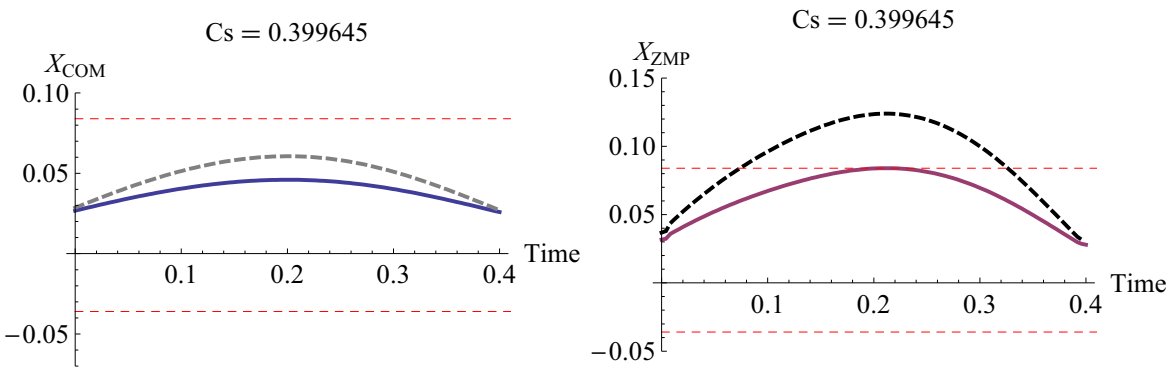
If we define the greatest instability as the most the reference ZMP trajectory, $x_{\text{ZMP}}^{\text{Ref}}$, leaves the ZMP range and denote it as $x_{\text{ZMP}}^{\text{err}}$, the motion can be stabilized by using the scaling factor:

$$C_s = \frac{x_{\text{ZMP}}^{\text{err}}}{x_{\text{ZMP}}^{\text{nom}}(t_m) - x_{\text{ZMP}}^{\text{Ref}}(t_m)} \tag{4.8}$$

where $x_{\text{ZMP}}^{\text{nom}}$ is the nominal x ZMP and t_m is the time when $x_{\text{ZMP}}^{\text{Ref}}$ is the furthest from the ZMP range (or when $x_{\text{ZMP}}^{\text{err}}$ occurs–derivation can be found in Sec. A.1). Since C_s scales the entire motion, it must be large enough to stabilize the most unstable portion of the motion. If the most unstable portion of the motion is stabilized using this method, then any other areas of instabilities are consequently also stabilized. Using Eqs. 4.7 and 4.8, the example motion from Fig. 4.2 is filtered resulting in Fig. 4.3, which shows a comparison of $x_{\text{COM}}^{\text{Ref}}$ to the final stable $x_{\text{COM}}^{\text{AMF}}$. Additionally, since the resulting AMF COM trajectory is a scaled version of the reference COM trajectory, the motion naturally retains some characteristics of the reference COM trajectory, such as curve or shape. If handled appropriately in the CMA, the resulting AMF motion (including joint angle trajectories) can also result as a scaled version of the reference motion.

4.3 Time scaling

The compliment to scaling the spatial information of a motion would be to scale the time information of the motion–essentially slowing down or speeding up the motion. Leaving the spatial portion of the motion unchanged is a desirable feature in instances where path following is very important. Robots manipulating tools or navigating an obstruction may have



(a) $x^{\text{Ref}}_{\text{COM}}$ (Gray dashed) and $x^{\text{AMF}}_{\text{COM}}$ (Blue) (b) $x^{\text{Ref}}_{\text{ZMP}}$ (Black dashed) and $x^{\text{AMF}}_{\text{ZMP}}$ (Purple)

Figure 4.3: Comparison of the reference and resulting filtered motion from spatial scaling. The red dashed lines are the ZMP limits.

to follow a very specific path in order to avoid collision, but the speed at which the task is accomplished may not be as important. One of the best features about only scaling the time information of the motion is that determining the joint angle trajectories for the filtered motion in the CMA is trivial–the joint angle trajectories maintain the same angular trajectories and inherit the same sort of " "speeding up" or "slowing down" that the COM trajectory goes through in the CATF. The downside to modifying/scaling the time information is that the X and Y directions are inherently coupled–all spatial dimensions share the same time dimension. Therefore it may be possible that speeding up or slowing down the motion may stabilize motion in the X-direction, but then destabilize motion in the Y-direction. These considerations are addressed in more detail in Sec. 4.4. Finally, in keeping with one of the overall goals of the AMF, the motion is only "sped up" or "slowed down" as much as needed in order to stabilize the motion so that differences between the reference and filtered motion are minimized.

We can describe speeding up or slowing down the reference motion, time scaling, as

$$x^{\text{AMF}}_{\text{COM}}(t) = x^{\text{Ref}}_{\text{COM}}(t') \tag{4.9a}$$
$$y^{\text{AMF}}_{\text{COM}}(t) = y^{\text{Ref}}_{\text{COM}}(t') \tag{4.9b}$$

where t' is the new time information, which will be described as

$$t' = \int_0^t C(t)dt \tag{4.10}$$

where $C(t)$ describes the amount of "slowing down" or "speeding up" applied to the reference motion. From A.2, the stability of the filtered motion can be shown as

$$x_{\text{ZMP}}^{\text{AMF}}(t) = x_{\text{COM}}(t') - H\left(\ddot{x}_{\text{COM}}(t')C^2(t) + \dot{x}_{\text{COM}}(t')C'(t)\right) \tag{4.11}$$

where $C'(t)$ is the derivative of $C(t)$ with respect to time. We can then rewrite the equation using discrete notation as

$$\left(x_{\text{ZMP}}^{\text{AMF}}\right)_i = \left(x_{\text{COM}}\right)_i - H\left(\left(\ddot{x}_{\text{COM}}\right)_i {C_i}^2 + \left(\dot{x}_{\text{COM}}\right)_i C_i'\right) \tag{4.12}$$

Since C_i' is included in the solution, there is no clean way to solve for C_i. However, if $\dot{x}_{\text{COM}} = 0$ or $C_i' = 0$, then it is much simpler to find C_i. Since for most motions, $\dot{x}_{\text{COM}} \neq 0$, we hold C_i as constant for the duration of the motion in order to obtain a solution for C_i.

$$\left(x_{\text{ZMP}}^{\text{AMF}}\right)_i = \left(x_{\text{COM}}\right)_i - H\left(\ddot{x}_{\text{COM}}\right)_i {C_i}^2 \tag{4.13}$$

Therefore, if we say $C_i = C_T$ for the entire, motion, we can solve for C_T as

$$C_T = \sqrt{\frac{\left(x_{\text{COM}}\right)_\tau - x_{\text{ZMP}}^{\text{Limit}}}{H\left(\ddot{x}_{\text{COM}}\right)_\tau}} \tag{4.14}$$

where τ occurs when C_T is calculated as furthest from 1. $x_{\text{ZMP}}^{\text{Limit}}$ replaces $\left(x_{\text{ZMP}}^{\text{AMF}}\right)_i$ since C_T is designed to stabilize the motion just enough so that the ZMP is within the support polygon. Therefore, the most unstable point will lie on a ZMP limit. There are certain conditions that must exist in order for there to be a valid C_T that can stabilize the reference motion since C_i is constant for the entire motion. First, the result of 4.14 must always be above or below 1 for all unstable portions of the motion. This is because if $C_T < 1$, the motion is slowed down and if $C_T > 1$, the motion is sped up. Since C_i is constant, the entire motion can *only* be slowed down or sped up. Therefore, if part of the motion needs to be sped up in order to be stable and another part of the motion needs to be slowed, it is not possible to have a constant C_i value. In order for this to be true, $x_{\text{COM}}(t) - x_{\text{ZMP}}^{\text{Limit}} - H\ddot{x}_{\text{COM}}(t)$ must be either greater than or less than zero for all unstable portions of the reference motion (solved by setting $C_T = 1$ in Eq. 4.14).

Holding C_i constant presents additional constraints on what motions can be stabilized from scaling the time information. Given Eq. 4.14 and 4.3, three cases of instability can occur with respect to the speed of the motion:

1. $x_{\text{COM}}^{\text{Ref}}$ is over the support polygon, and therefore within the ZMP range, but the magnitude of $\ddot{x}_{\text{COM}}^{\text{Ref}}$ is too high and causes instability by pushing $x_{\text{ZMP}}^{\text{Ref}}$ outside of the ZMP limit

2. $x_{\text{COM}}^{\text{Ref}}$ is outside of the support polygon and the sign of $\ddot{x}_{\text{COM}}^{\text{Ref}}$ is the opposite of $x_{\text{ZMP}}^{\text{Ref}} - x_{\text{ZMP}}^{\text{Range}}$. Instability occurs because $\ddot{x}_{\text{COM}}^{\text{Ref}}$ is not the correct magnitude.

3. $x^{\text{Ref}}_{\text{COM}}$ is outside of the support polygon and the sign of $\ddot{x}^{\text{Ref}}_{\text{COM}}$ is the same as the sign of $x^{\text{Ref}}_{\text{ZMP}} - x^{\text{Range}}_{\text{ZMP}}$. Instability is guaranteed no matter how fast or slow the motion is.

Case 1 is the simplest case to understand. If $x^{\text{Ref}}_{\text{COM}}$ is over the support polygon, the robot can be stable if the motion slows down and takes an infinite amount of time to occur, which drives the COM acceleration to zero. The resulting ZMP then coincides with the COM, which is over the support polygon and therefore stable.

Case 2 highlights the relationship between the COM position and the COM acceleration. If $x^{\text{Ref}}_{\text{COM}}$ is above $x^{\text{High}}_{\text{ZMP}}$, the acceleration must be positive (since $H > 0$ in Eq. 4.3) in order for the resulting ZMP, $x^{\text{Ref}}_{\text{ZMP}}$, to be under $x^{\text{High}}_{\text{ZMP}}$. Likewise, the acceleration must be negative if the COM position is below $x^{\text{Low}}_{\text{ZMP}}$ in order for Case 2 to occur. Adjusting the speed of the reference motion can result in a stable motion by changing the magnitude of the acceleration and bringing the ZMP within limits. In this case, stabilizing the motion may involve either speeding up or slowing down the motion.

Case 2 and 3 are opposites of each other; Case 2 can be stabilized while Case 3 can not. If the COM position is above $x^{\text{High}}_{\text{ZMP}}$ and the COM acceleration is negative, it does not matter how much faster or slower the motion is, there is no magnitude of the acceleration that would yield a ZMP below $x^{\text{High}}_{\text{ZMP}}$. The same situation occurs if the COM is below $x^{\text{Low}}_{\text{ZMP}}$ and the COM acceleration is positive. This presents an interesting characteristic that was not present in spatial scaling. When scaling the spatial dimensions of the reference motion, there was always a stable solution. However, when only scaling the time dimension of the motion, it is possible that the motion cannot be stabilized. An easy example to visualize this case would be if a robot was holding a very heavy weight beyond the front of its feet while stepping backwards; no matter how quickly or slowly the robot steps backwards, it will fall over towards its front.

Figure 4.4 shows a plot of a fabricated sample reference motion. The different types of instability can be seen along with the corresponding reference motion and reference motion acceleration.

As an example to show how changing the speed of the reference motion can stabilize the motion, Fig. 4.5 shows ZMP trajectories in the X and Y directions with their instabilities classified. Also shown are the corresponding C_i values calculated using 4.14 for each instance. As seen, C_i is calculated as constantly below 1 and no Case 3 instabilities are present. Since C_i is held constant, the most extreme C_i value is used first for each direction to show the effect of speeding up and slowing down the reference motion as seen in Fig. 4.6.

Since the X and Y directions are coupled, only one C_T value can be used. Again, the most extreme C_T value between the two directions is used. Since it is possible that the speeding up or slowing down the motion in one direction could destabilize the other direction, we must check the C_T value used in both directions. Figure 4.7 shows the resulting AMF ZMP trajectories using the the same and most extreme C_T value applied to slow down the motion. As seen, the ZMP trajectories in both direction remain within the support polygon,

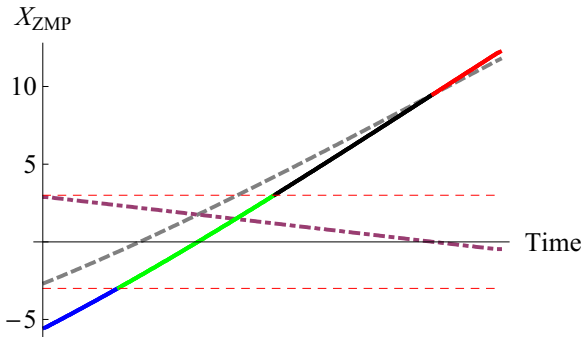


Figure 4.4: $x^{\text{Ref}}_{\text{COM}}$ (Dashed gray), $\ddot{x}^{\text{Ref}}_{\text{COM}}$ (Dot-dashed purple), $x^{\text{Ref}}_{\text{ZMP}}$: Categorized by instability; blue is Case 1, black is Case 2, red is Case 3, and green is stable. $H = 1$ for this example.

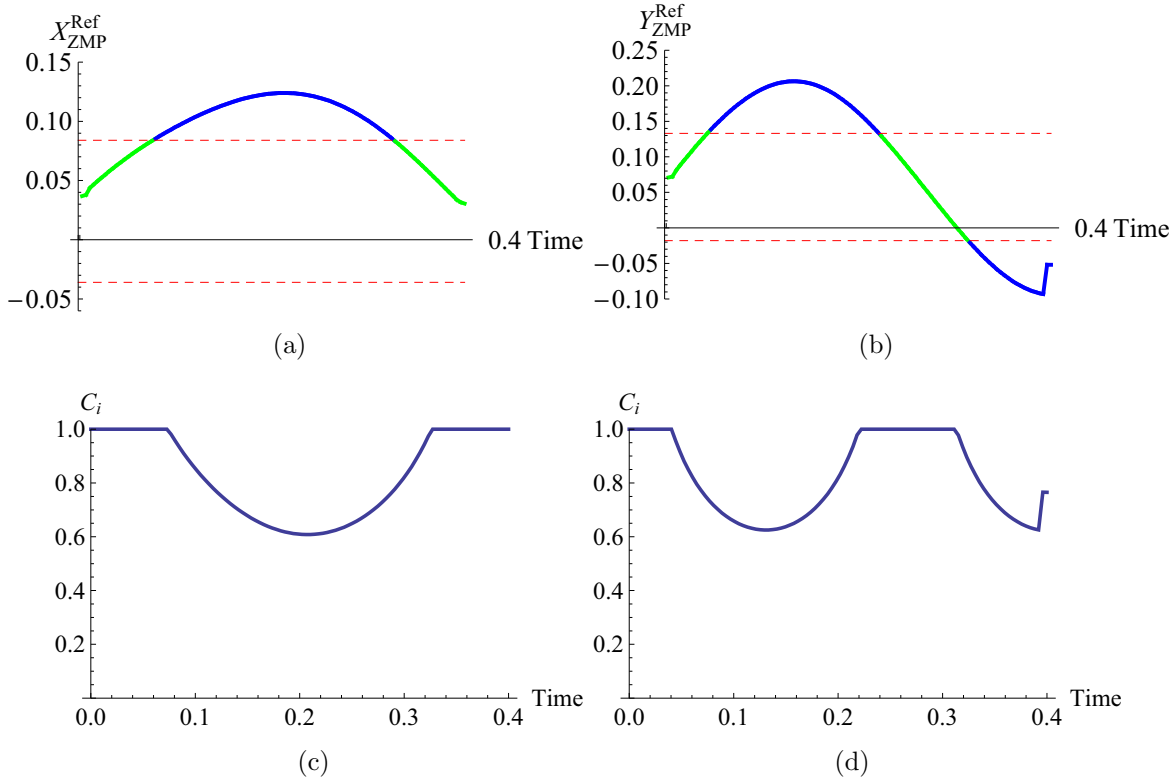


Figure 4.5: Fabricated sample reference motion. ZMP plots: $\ddot{x}^{\text{Ref}}_{\text{ZMP}}$: Categorized by instability; blue is Case 2 and green is stable. And then corresponding C_i plots.

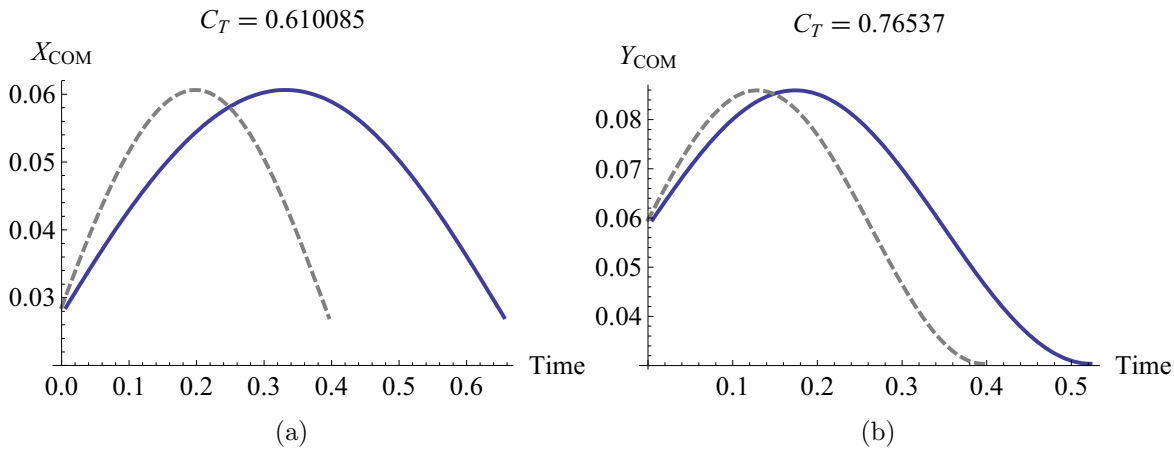


Figure 4.6: Resulting COM trajectories from slowing down the motion by the factor C_T calculated for each direction independently. $x^{\text{Ref}}_{\text{COM}}$ (Gray dashed) and $x^{\text{AMF}}_{\text{COM}}$ (Blue)

and though while difficult to see, the Y direction is over-stabilized since it is slowed more than necessary for a stable ZMP path.

4.4 Considerations for the CATF

The CATF is able to stabilize many reference motions without difficulty or special considerations by making generalizations or simplifications. Some of these may be: approximating the support polygon as a rectangle with axes perpendicular to the robot's coordinate frame; ignoring changes in the z-direction of the COM trajectory; or assuming that the resulting COM trajectory does not violate the robot's internal kinematics. This section will show how to address situations where these kinds of simplifications or generalizations are unacceptable. Additionally, this section will account for assumptions made in Sec. 4.1.

4.4.1 Addressing assumptions

Three assumptions were asserted in order to simplify the ZMP stability equations, which allowed for analytical solutions:

- Negligible moment of inertia of the robot
- The height of the COM is relatively constant
- The acceleration of the height of the COM is relatively small compared to gravity

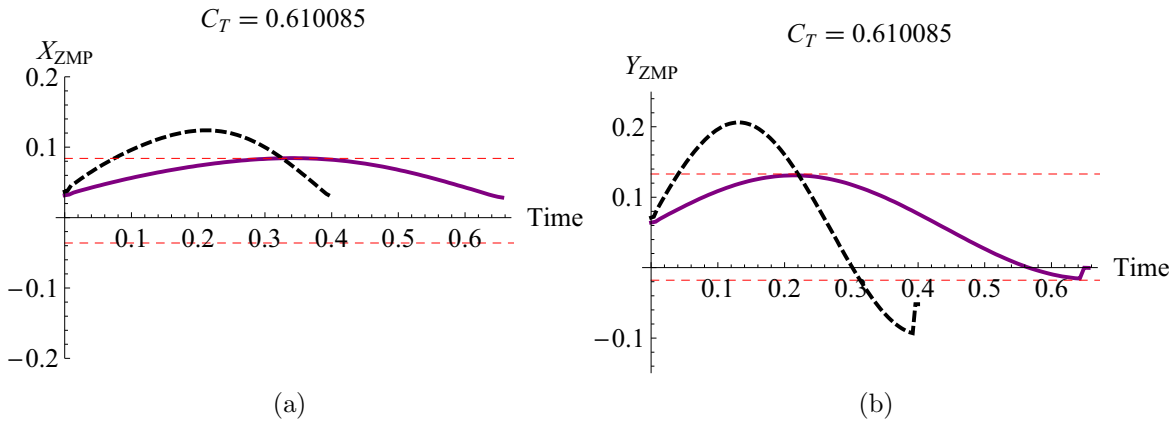


Figure 4.7: Resulting ZMP trajectories from slowing down the motion by the factor C_T. $x_{\text{ZMP}}^{\text{Ref}}$ (Black dashed) and $x_{\text{ZMP}}^{\text{AMF}}$ (Purple)

It is entirely possible that while the AMF may create a motion that is presumably stable, violations of the above assumptions would cause the resulting motion to be unstable. If there is significant acceleration of the height of the COM, which is rare, it is entirely possible that the resulting motion is unstable. For most robot motions, the height of the COM does not change much. To account for small changes in height, small accelerations, and small angular inertia effects, a factor of safety is used for creating a stable motion. Instead of using the robot's exact support polygon, using a reduced area (80%-65%) of the support polygon creates a safety factor that allows for some error in the assumptions made for the filter. Unless the assumptions are grossly violated, the safety factory will ensure that the filtered motions predicted to be stable actually are.

4.4.2 Addressing kinematic constraints

The robot's kinematic constraints can be very important when considering how a motion can be changed so it is stable. Kinematic constraints prevent links, and consequently the COM, from existing past certain regions of space. Additionally, kinematic constraints cause singularities where instantaneous link velocities can only be zero. Fortunately, spatial kinematic constraints have easily definable limits as to how far the COM of the robot can exist in the X/Y/Z direction while maintaining contact with the ground. For example, if the robot's height is to remain unchanged, there is only so far the robot can lean forward and extend its limbs without picking up a foot from the ground. If a configuration of the robot exists that defines the maximum displacement of the COM in a given direction, determining that configuration is only dependent on the ground contact. These limits must be taken into consideration so that the CATF does not generate motions that violate these limits. Additionally, the allowable space for the COM to exist may be further constrained by con-

straints on the entire robot's motion. For example, the arm motions may be constrained to remain unchanged throughout the AMF, which further limits the range of the COM in space. Additional constraints such as physical joint limits or actuator range limitations may manifest themselves in this form of a kinematic constraint.

Assuming the limits of the COM can be defined as a function of the reference motion, the limits can then be defined as functions of time; i.e. $x_{\mathrm{COM}}^{\mathrm{Limit}}(t)$. The limits can be worked into the spatial scaling portion of the CATF by comparing the result of 4.8 with

$$C_s = \frac{x_{\mathrm{COM}}^{\mathrm{err}}}{x_{\mathrm{ZMP}}^{\mathrm{nom}}\left(t_{\mathrm{m2}}\right) - x_{\mathrm{COM}}^{\mathrm{Ref}}\left(t_{\mathrm{m2}}\right)}, \tag{4.15}$$

where $x_{\mathrm{COM}}^{\mathrm{err}}$ is the maximum difference between the $x_{\mathrm{COM}}^{\mathrm{Ref}}$ and the $x_{\mathrm{COM}}^{\mathrm{Limit}}$ that occurs at time t_{m2}. The motion can be stabilized while satisfying kinematic constraints by using the value of C_s from Eq. 4.15 and 4.8for Eq. 4.7 that is closer to 1. If a condition exists where the nominal ZMP location is outside of the COM limit, the nominal ZMP should be redefined so that it is between the COM limit and closest ZMP limit (suggested to be halfway between).

Kinematic constraints are not an issue for time scaling since the joint trajectories are not modified. Therefore, the original reference motion trajectory, which is assumed to be kinematically feasible, is used for the final AMF joint trajectories.

4.4.3 Non-rectangular support polygons

The majority of instabilities for humanoid robots occur when the robot has only one foot in contact with the ground (SSP), which makes sense because the support polygon is at its smallest and typically large accelerations and displacements of the COM are present in order to move the robot. The majority of humanoid robots also have flat rectangular feet, making a rectangular support polygon assumption reasonable. Additionally, it may be possible to conservatively assume a rectangular support polygon for non-rectangular feet or for when two feet are in contact with the ground. This assumption may not be possible in all cases, which means that the shape of the support polygon has a large role in the CATF. If the support polygon is non-rectangular, then the X and Y directions cannot be easily decoupled. Two different approaches can be taken to incorporate the support polygon's non-rectangular shape into the CATF. The approaches depend on whether the motion is being filtered using space or time scaling.

Time scaling

Incorporating the support polygon's shape into the time scaling algorithm of the CATF is fairly straight forward. The value of C_i in Eq. 4.10 directly affects both $x_{\mathrm{ZMP}}^{\mathrm{AMF}}$ and $y_{\mathrm{ZMP}}^{\mathrm{AMF}}$. Therefore, by starting with a C_T value of 1 and working towards both 0 and ∞, whatever

value of C_T closest to 1 that results in both the X and Y ZMP values within the support polygon for the entire motion can be used. Though not completely an analytical solution to address non-rectangular support polygons, a one variable search is the easiest numeric solution to implement. Again time scaling is only applicable for certain cases (See 4.3).

The range of C_T values searched can be reduced by incorporating an absolute minimum and maximum ZMP value for each direction. For motions where the COM stays entirely within the support polygon, the minimum C_T value is 0 since the motion could slow down to a standstill and remain stable. Additionally, if

- $\ddot{x}^{\text{Ref}}_{\text{COM}} > 0$, use the absolute lower ZMP limit in Eq. 4.14 to determine the high end of the C_T range. The same rule applies for the Y-direction.
- else $\ddot{x}^{\text{Ref}}_{\text{COM}} < 0$, use the absolute upper ZMP limit in Eq. 4.14 to determine the high end of the C_T range. The same rule applies for the Y-direction.

Otherwise, for motions where the COM leaves the support polygon, both the maximum and minimum values of C_T can be found from Eq. 4.14 by using the absolute upper and lower ZMP limits. This is possible because if the motion can be classified as stable, Case 1 or Case 2 from Sec. 4.3, then there must exist a scaled COM acceleration that pushes the ZMP from the COM location into the ZMP range. Therefore, the acceleration, if greatly increased, would send the ZMP out of one ZMP limit and decreasing the acceleration would eventually pull the resulting ZMP outside of the other ZMP limit as it converges towards the COM location, which resides outside of the ZMP limit.

Two ranges of C_T values are calculated; one for the x and one for the y directions. The range of C_T values searched must be within both sets of ranges.

Spatial scaling

Incorporating the support polygon's shape in the spatial scaling algorithms of the CATF is a similar search as with the time scaling algorithm. By starting with a set of arbitrary C_s values for Eq. 4.7 for one direction (let's say the X-direction), the proper corresponding C_s value for the Y-direction can be calculated. Since a range of C_s value combinations are possible, including combinations that may "over-stabilize" the motion, the one that results with the smallest sum of squares of C_s could be used as the "best" combination. The set of possible C_s values for the X-direction can be determined by first answering the question: Is the entire $x^{\text{Ref}}_{\text{ZMP}}$ trajectory within the ZMP limits (where the limits are defined by the furthest most points in the x-direction that make up the support polygon)?

- Yes. The lower limit of the C_s value is 0 since the motion is within the limits of that direction.

- No. The lower limit of the C_s value is found from Eq. 4.8

The upper limit of the C_s value is 1, which is a reference COM trajectory that remains in the nominal ZMP location. By using the result from above and then varying the nominal ZMP value in each direction, a 3D map is created of the best combinations of C_s values for every $x_{\text{ZMP}}^{\text{Nom}}$ and $y_{\text{ZMP}}^{\text{Nom}}$ values. The smallest sum of squares of C_s values of the resulting map is the best combination that least changes the reference motion when filtering. Although not an analytical solution, the proposed numeric search method above does provide a way to minimize changes in the COM trajectory when using Eq. 4.7.

The process essentially scales the motion first in the X-direction from the minimally stable scale to a complete scale towards the nominal ZMP value. See Fig. 4.2 for an example of this scaling. By then varying the nominal ZMP value to all of the viable nominal and stable ZMP values, every single scaled version of the reference motion is discovered for the X-direction. Consequently, every single scaled stable version of the reference motion in the X-direction has one C_s value for scaling the Y-direction for each $y_{\text{ZMP}}^{\text{Nom}}$ value. This process then finds every stable scaled version of the reference motion with corresponding C_s pairs. The pair that results in the smallest sum of squares of C_s is considered to change the reference motion the least, and is therefore the best choice for stabilizing the motion.

4.4.4 Changing support polygons

The most common case of a changing support polygon for humanoid robots occurs when the robot takes steps, changing the ground contact. Other cases exist for deformable feet or addition/subtraction of other ground contacts. For both spatial and time scaling algorithms, a changing upper limit and lower limit ZMP value does not invalidate or complicate the analytical solutions. Both are based on using the the instant of the motion that is most unstable and do not need to consider what the changing ZMP limits may be. The nominal ZMP value does not factor in to time scaling and therefore does not change its process. However, for spatial scaling, a time varying nominal ZMP changes the calculation of C_s. The resulting equation to solve for C_s would be expressed as

$$C_s = \frac{x_{\text{ZMP}}^{\text{err}}}{x_{\text{ZMP}}^{\text{nom}} - H\ddot{x}_{\text{ZMP}}^{\text{nom}} - x_{\text{ZMP}}^{\text{Ref}}} \tag{4.16}$$

where $\ddot{x}_{\text{ZMP}}^{\text{nom}}$ is the acceleration of the nominal ZMP value. Therefore Eq. 4.16 can be used for solving for C_s when considering a changing nominal ZMP value.

Using a changing support polygon could improve the approximation of a non-rectangular support polygon as a rectangular support polygon. By having a moving rectangular approximation that corresponds to the reference ZMP trajectory or the reference COM trajectory, this approximation could have more validity. For example, as a robot is about to step forward with its right foot and shifts weight from right to left and back to front, the rectangular

approximation can move correspondingly, following and centering around an acceptable and common ZMP path for this kind of motion. Additionally, if the nominal ZMP values move at a constant velocity, then Eq. 4.8 can still be used to solve for the spatial scaling factor.

4.4.5 Changing height

Adjusting the height of the COM is generally not considered for the AMF since it is assumed constant for many of the derived solutions. However, the entire height of the robot can be raised or lowered according to Eq. 4.3 rearranged as

$$z_{\text{COM}} = \frac{g\left(x_{\text{COM}} - x_{\text{ZMP}}\right)}{\ddot{x}_{\text{COM}}} \tag{4.17}$$

where $\ddot{z}_{\text{COM}}$ is considered to be zero or negligible. This requires changes in the COM height to occur very slowly. It is very interesting that this equation resembles Eq. 4.14 if C_i is held at 1. As seen in Eq. 4.13, H, which contains z_{COM}, affects the same term that C_i affects. This means that the same analytical solutions used for time scaling can be used for scaling the Z-direction in space. Rearranging the equation also shows

$$\omega^2 = \frac{z_{\text{COM}}}{g} = \frac{\left(x_{\text{COM}} - x_{\text{ZMP}}\right)}{\ddot{x}_{\text{COM}}} \tag{4.18}$$

where ω is the natural frequency of a pendulum. This shows the relationship between C_i and the natural frequency of the system, and consequently between time and the height of the COM. Therefore, the height of the COM can be adjusted in place or in conjunction with C_i to stabilize the motion. It is possible that adjusting only the height of the COM could stabilize the motion if the range of motion for changing the height exists and it is possible to filter the motion using C_i since the same requirements for feasible filtering apply to both methods.

Chapter 5

Combinations and Constraints for the CATF

Though the spatial and time scaling algorithms for filtering and stabilizing reference COM trajectories are relatively simple and straight forward, they provide much insight into the complex interaction between the COM trajectory and the robot's stability. Many of the interesting behaviors can be seen when the CATF is constrained. Frequently, constraints are desirable when filtering a motion such as walking. For example, for a walking reference motion, it may be very important that the robot's feet make contact with the ground and maintain their position. The filtering process should not change where the feet land. Though it is impossible to stabilize a reference motion if both the spatial or time dimensions are constrained and prevented from changing in the AMF, it is possible to manipulate both dimensions in order to reduce the overall changes in either. This section will attempt to address some of the constraints that can be enforced on the CATF and the resulting algorithms that satisfy the constraints. How the constraints are selected depends on what aspects of the reference motion are most important and should not be filtered out of the final stabilized AMF motion.

5.1 Pose constraints

Having the robot in specific poses with corresponding joint positions at specific times is a very salient characteristic of a motion. For example, during walking, the robot must pick one foot off the ground, move it forward, and then set it down. The foot on the ground, in the air, and placed forwards are all poses that happen at certain times during the motion. What happens in between is not as important as long as the motion includes those poses and transitions smoothly between them. The pose constraints can translate directly into COM position constraints. Asserting that the COM must be at a certain position at a certain

time guarantees that the joint angles are kinematically feasible at that specified point in time because the reference joint angles can also follow the same constraint and therefore do not change for the specified point in time (remember that the reference motion must be kinematically feasible). Therefore pose constraints may be used synonymously with COM position constraints in this work. The number of pose constraints dictates how the AMF filters the reference motion. The poses must exist in the reference motion–they cannot be arbitrarily chosen outside of the reference motion.

An obvious way to stabilize a motion that has pose constraints is to use the time scaling algorithm of the CATF since it does not change the joint angle trajectories. However, time scaling does not guarantee a stable motion. Additionally, it may be undesirable to use time scaling if the timing or speed of the robot's motion is important. Therefore, spatial scaling can be used while satisfying pose constraints as shown in this section. Again, for most situations, the X and Y directions of the COM trajectory can be decoupled and treated independently. Therefore only the algorithms for the X-directions are described in this section, but the same algorithms can be applied in the Y-direction.

5.1.1 Zero and one pose constraints

To satisfy one pose constraint, the CATF uses the best fit line (calculated from common statistics algorithms), $x^{\text{Fit}}_{\text{COM}}$, using Eq. 4.5 as the model with $x^{\text{Fit}}_{\text{COM}}$ replacing x_{COM} and using the following constraints:

$$x^{\text{Fit}}_{\text{ZMP}} = x^{\text{Nom}}_{\text{ZMP}} \tag{5.1a}$$

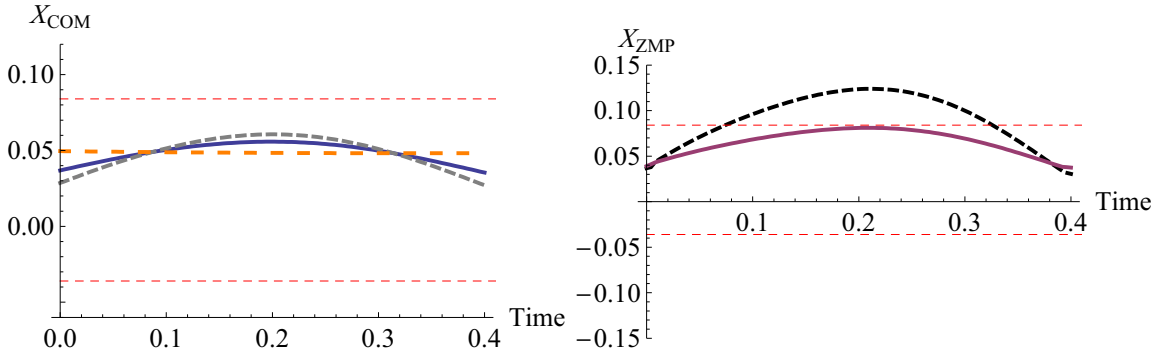
$$x^{\text{Fit}}_{\text{COM}}(t_1) = x^{\text{Ref}}_{\text{COM}}(t_1) \tag{5.1b}$$

The filtered COM trajectory results as a similar solution to Eqs. 4.7, but $x^{\text{Fit}}_{\text{COM}}(t)$ replaces $x^{\text{Nom}}_{\text{ZMP}}$, resulting in

$$x^{\text{AMF}}_{\text{COM}}(t) = x^{\text{Ref}}_{\text{COM}}(t) + C_s\left(x^{\text{Fit}}_{\text{COM}}(t) - x^{\text{Ref}}_{\text{COM}}(t)\right) \tag{5.2}$$

where C_s is defined the same way as in Eq. 4.8. This can be done because $x^{\text{Fit}}_{\text{ZMP}} = x^{\text{Nom}}_{\text{ZMP}}$ by its definition in Eq. 5.1.

An alternative to the proposed simple spatial scaling, using the above "best fit method," but with no pose constraints works equally well, but relies on statistical best fit algorithms, which is obviously slower than simple spatial scaling. Using Eq. 4.5 as a model and constraining the model to maintain a ZMP of $x^{\text{Nom}}_{\text{ZMP}}$, the resulting best fit line is used in Eq. 5.2. Figure 5.1 shows the result using this method on a sample motion. The best fit line is a constant



(a) x_{COM}^{Ref} (Gray dashed), x_{COM}^{Fit} (Orange large dashed), and x_{COM}^{AMF} (Blue)

(b) x_{ZMP}^{Ref} (Black dashed) and x_{ZMP}^{AMF} (Purple)

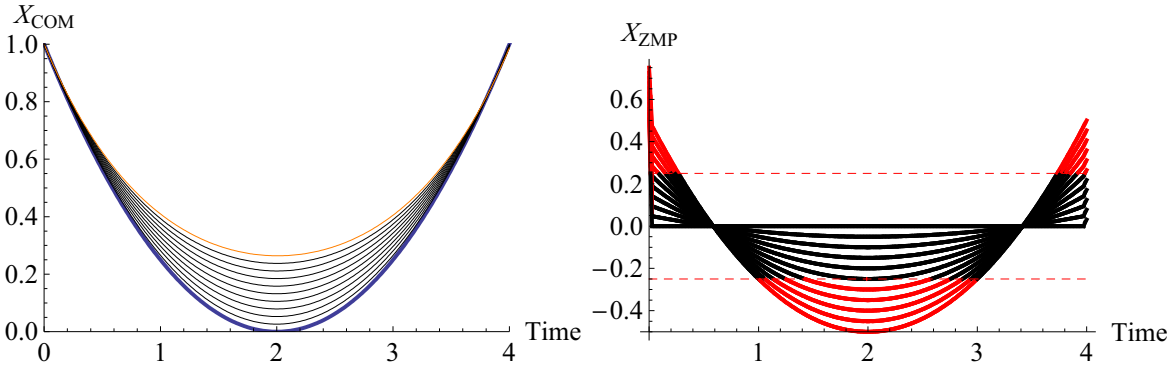
Figure 5.1: Spatial scaling using x_{COM}^{Fit} calculated using no pose constraints.

in this example, but that is because the COM trajectory is relatively symmetric and a best fit line happens to be a straight, horizontal line.

5.1.2 Two pose constraints

The solution for two pose constraints is similar to zero and one pose constraint, but adds a second constraint to 5.1. The result is no longer a stable best fit COM trajectory, but a COM trajectory that satisfies the two COM position constraints and the fixed (nominal) ZMP. It is always possible to form a stable COM trajectory with two pose constraints. Equation 4.5 has two unknowns with the ZMP fixed at x_{ZMP}^{Nom}, which means that two COM constraints can always be solved while keeping the ZMP inside of the robot's stable region. Figure 5.2 shows a stable fitted trajectory (x_{COM}^{Fit}) along with how the reference COM trajectory is attracted to the stable COM trajectory. In the example shown, the fitted trajectory is calculated using the first and last COM positions as constraints. Figure 5.3 shows the result of x_{COM}^{AMF}, which was calculated using Eq. 5.2 and applying position constraints to the beginning and end of the motion.

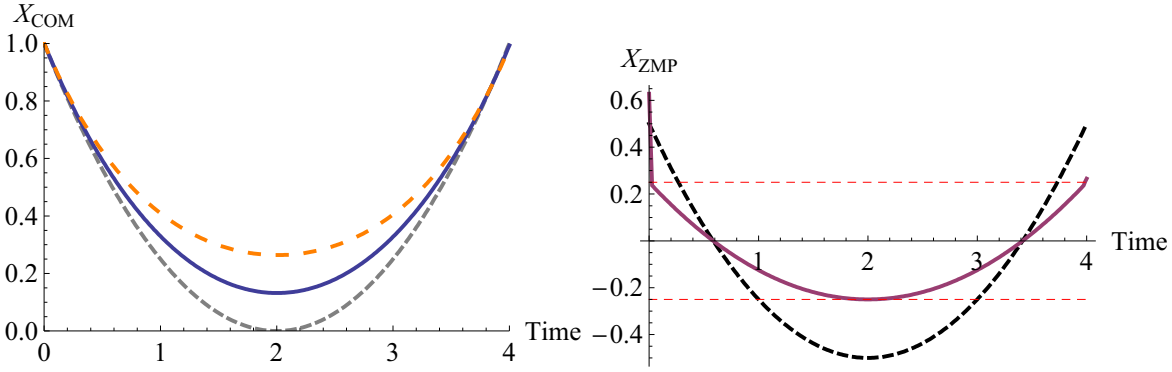
Figure 5.4 shows what different stable fitted lines would look like for constraints that pass through the final COM reference point and a second COM constraint point that increases from the beginning of the motion to almost the end of the motion. The stable solutions produced are interesting. Half of the the solutions return in the X-direction they started from, while the other half do not change direction. This is similar to pushing an inverted pendulum from a low potential energy state to higher energy state. In one case, if you push only a little, the pendulum will come back to where it started. In another case, if you push very hard, the pendulum will reach a peak and continue on, never returning to the start position. This is the behavior of the two exponentials found in Eq. 4.5.



(a) x_{COM}^{Ref} (Blue), x_{COM}^{Fit} (Orange), and various scaled trajectories in between (Black)

(b) Corresponding ZMP plot with red indicating unstable portions of a motion.

Figure 5.2: Spatial scaling using x_{COM}^{Fit} calculated using two pose constraints.



(a) x_{COM}^{Ref} (Gray dashed), x_{COM}^{Fit} (Orange large dashed), and x_{COM}^{AMF} (Blue)

(b) x_{ZMP}^{Ref} (Black dashed) and x_{ZMP}^{AMF} (Purple)

Figure 5.3: Spatial scaling using x_{COM}^{Fit} calculated using two pose constraints.

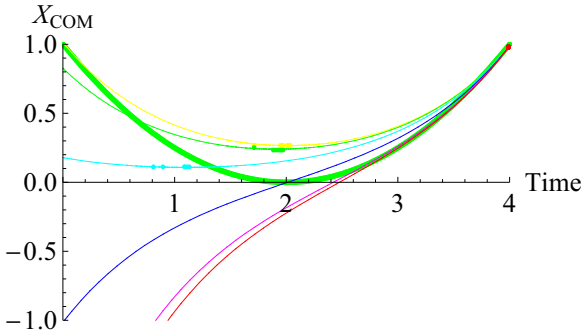


Figure 5.4: Stable COM trajectories, $x^{\text{Fit}}_{\text{COM}}$, passing through the final COM reference point and a second COM constraint position from the reference COM trajectory (green) that increases from the start of the motion to almost the end of the motion

5.1.3 Three pose constraints

A third pose constraint is not necessarily possible to satisfy while maintaining stability. Equation 4.5 can be formed to have three unknowns by allowing the ZMP to vary within the ZMP range, but since the ZMP is not permitted to leave the ZMP range, the range of a third pose constraint (that allows for a feasibly stable motion) is bounded. Therefore, it is always possible to satisfy three pose constraints if the resulting ZMP from solving Eq. 4.5 for three constraints is within the ZMP range. Figure 5.5 shows a visualization of this concept with two COM position constraints at the beginning and end of the motion and using both extremes of a ZMP range (0-4). The result is an envelope where anywhere between the two trajectories, a third COM constraint point must lie in order for three COM position constraints to be feasibly stable. As seen, the envelope is very small. No other point from the reference COM trajectory could be selected as the third COM position constraint and be feasibly stable. Figure 5.6 shows a plot with the same two COM position constraints, but a more extreme ZMP range of -60 to 4. The more extreme lower limit ZMP expands the envelope and allows for any point from the COM reference trajectory to be a third constraint point and maintain stability. If three pose constraints can be satisfied while maintaining stability, then the same method for creating the filtered motion based on the stable trajectory ($x^{\text{Fit}}_{\text{COM}}$) as used for the one and two pose constraints is used to attract the COM reference trajectory to the stable COM trajectory.

Pose constraints can also be manipulated to create pseudo velocity and acceleration constraints. Selecting two pose constraints that occur at say τ and $\tau + dt$ or $\tau + \Delta$t, is equivalent to setting a velocity constraint. After spatial scaling is applied, the two constraint points will remain unchanged and therefore the velocity for that portion of the motion remains unchanged. Additionally, using sequential pose constrains, an equivalent acceleration constraint can be enforced. Similar to the velocity constraint, by effectively creating two sequential velocity constraints, the acceleration constraint is naturally enforced as well.

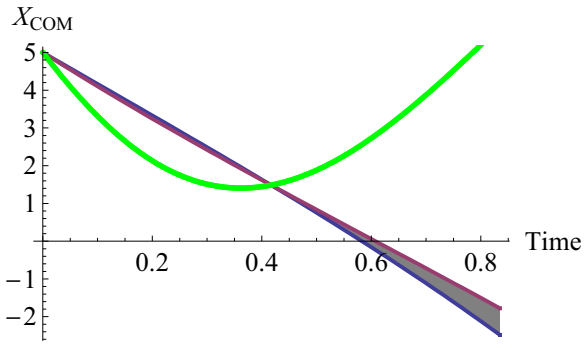


Figure 5.5: Plot showing the envelope in which a third COM constraint position could lie while satisfying the ZMP stability constraint (green is $x_{\text{COM}}^{\text{Ref}}$, blue is a stable COM trajectory using the upper ZMP limit, 4, and purple uses the lower limit ZMP, 0); ZMP range 0→4

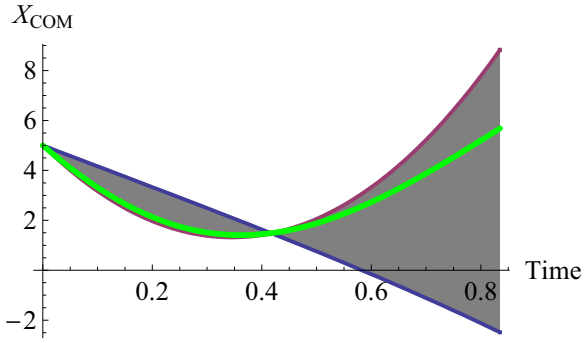


Figure 5.6: Plot showing the envelope in which a third COM constraint position could lie while satisfying the ZMP stability constraint (green is $x_{\text{COM}}^{\text{Ref}}$, blue is a stable COM trajectory using the upper ZMP limit, 4, and purple uses the lower limit ZMP, -60); ZMP range -60→4

5.1.4 Four pose constraints

If four pose constraints are needed, the process of verifying if the constraints can be satisfied while maintaining stability is an extension of previous methods. By creating two equations from Eq. 4.5, the following equations can be formed

$$x_{\text{COM}}^{\text{AMF1}}(t) = a_1 e^{\frac{t}{\sqrt{H}}} + a_2 e^{-\frac{t}{\sqrt{H}}} + x_{\text{ZMP}}^{\text{AMF1}} \tag{5.3a}$$

$$x_{\text{COM}}^{\text{AMF2}}(t) = a_3 e^{\frac{t}{\sqrt{H}}} + a_4 e^{-\frac{t}{\sqrt{H}}} + x_{\text{ZMP}}^{\text{AMF2}} \tag{5.3b}$$

where $x_{\text{COM}}^{\text{AMF1}}(t)$ is the equation to satisfy the first two COM position constraints and $x_{\text{COM}}^{\text{AMF2}}(t)$ satisfies the last two COM position constraints. Visualizing the stability envelops for both the first set of COM constraint positions and last set of COM constraint positions leads to Fig. 5.7. The trajectory that can join the two sets of solutions must stay within both stability envelops. Therefore, if a continuous area is not shared by both stability envelops in between the second and third COM constraint positions, then it is not possible to satisfy all 4 constraint COM positions while maintaining stability.

If the envelops of stability share a continuous area between the second and third COM constraint positions, then the following conditions must also be met in order for a stable trajectory to exist:

$$x_{\text{COM}}^{\text{AMF1}}(\tau) = x_{\text{COM}}^{\text{AMF2}}(\tau) \tag{5.4a}$$

$$\dot{x}_{\text{COM}}^{\text{AMF1}}(\tau) = \dot{x}_{\text{COM}}^{\text{AMF2}}(\tau) \tag{5.4b}$$

where $t_2 < \tau < t_3$ (t_2 and t_3 correspond to the second and third COM constraint positions' time respectively). Equation 5.4 requires that at some time between the second and third COM constraint point, there is a smooth transition from the stable line of the first two COM constraint points to the stable line of the last two COM constraint points. Combining Eqs. 5.3 and 5.4, the following is formed:

$$x_{\text{COM}}^{\text{AMF}}(t) = \begin{cases} a_1 e^{\frac{t}{\sqrt{H}}} + a_2 e^{-\frac{t}{\sqrt{H}}} + x_{\text{ZMP}}^{\text{AMF1}} & t < \tau \\ a_3 e^{\frac{t}{\sqrt{H}}} + a_4 e^{-\frac{t}{\sqrt{H}}} + x_{\text{ZMP}}^{\text{AMF2}} & t \geq \tau \end{cases} \tag{5.5}$$

where both $x_{\text{ZMP}}^{\text{AMF1}}$ and $x_{\text{ZMP}}^{\text{AMF2}}$ are unknowns and must be within the ZMP range. Four COM position constraints and the smooth trajectory condition give six equations. Eq. 5.5 gives seven unknowns (τ is unknown), which indicates that is possible for multiple stable trajectories to exist even when keeping the ZMP value fixed for $x_{\text{COM}}^{\text{AMF1}}(t)$ and $x_{\text{COM}}^{\text{AMF2}}(t)$. Additionally,

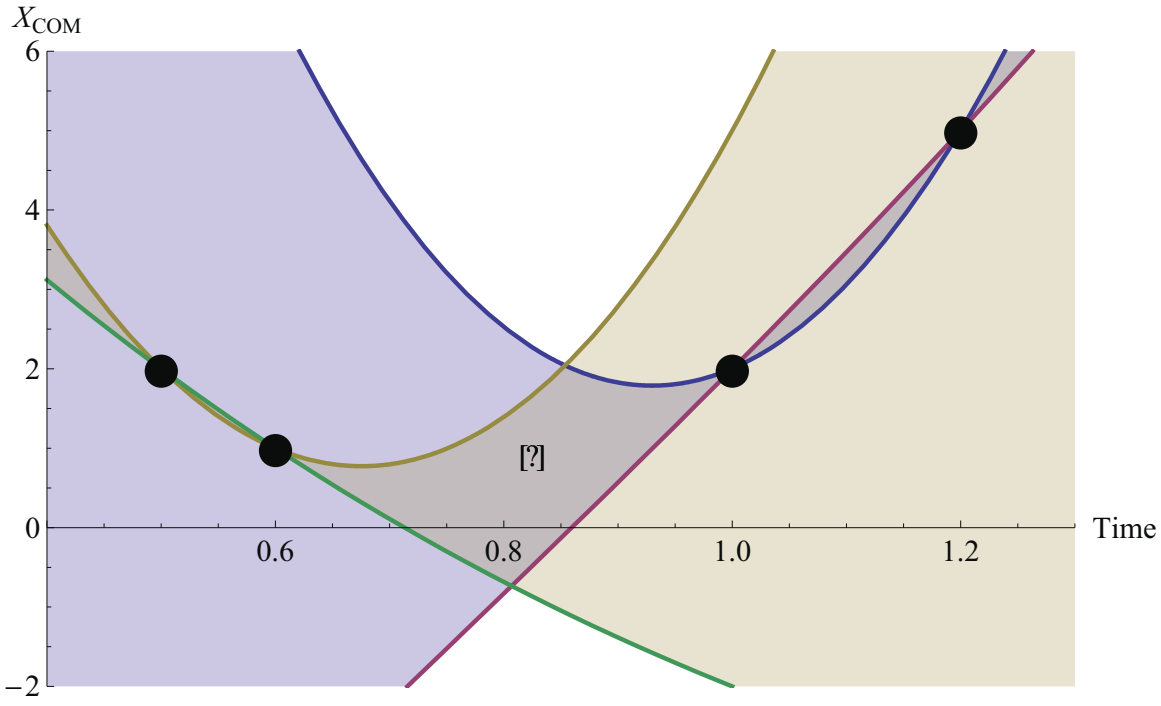


Figure 5.7: Plot showing the intersection of stability envelops for two sets of two COM constraint positions. "?" represents a possible transition point where the fitted stable ZMP trajectory could switch between the first and second stable fitted trajectory.

even if the envelops overlap, it is possible that no solutions exist due to the continuous velocity condition and the bounded range of the ZMP and τ.

Similar to section 5.1.2, the AMF calculates the best fit line of the data to fit Eq. 5.5 and the position constraints in Eq. 5.4. The resulting $x_{\text{COM}}^{\text{Fit}}$ trajectory, just as in previous sections, is then used to attract the reference COM trajectory until the entire motion is stable.

5.1.5 Five or more pose constraints

Equation 5.5 can be expanded to handle more pose constraints by adding more equations and unknowns. However, adding more and more equations relies more and more on numeric as opposed to analytical solutions. This characteristic can also be observed by further examining Eq. 4.2, where the right hand side of the equation are simply the states of the system and the left hand side of the equation can be described as a forcing function. The forcing function, or the ZMP, is what can vary to change the states. Equation 4.2 is similar to many dynamic problems, such as an inverted pendulum. Commonly throughout such types of problems, it is impossible to control the states without feedback–the system's error is naturally exponential. Similarly, it is very difficult to to create a COM trajectory that satisfies many position constraints since creating a stable trajectory depends on all of the position constraints before and after each position constraint. Small differences in initial conditions or the ZMP result in an exponentially increasing error. Therefore, some form of segmentation should be used to break up the trajectory into sub-trajectories, using pose constraints to ensure smooth transitions between fitted trajectories (See Sec. 5.3).

5.2 Combining space and time scaling

The timing and spatial information of a motion completely describe the motion. Completely constraining both is equivalent to leaving the reference motion unchanged. Neither spatial nor time scaling can be used since they both can not function under such constraints. The only way both space and time constraints can be specified simultaneously is to weight the importance of one over the other. The solution used is a sequence of time and spatial scaling–the second solution applied to the results of the first. The CATF in this usage has two stages. The second stage of the CATF must guarantee stability while the first stage only filters the reference motion closer to stability.

In the case of using the time scaling process in the first stage of the AMF, the weighting for the constraints translates to a fraction of the change that would result from completely filtering and stabilizing the reference motion using time scaling. For example, if the solution for *completely* stabilizing the reference motion is to double the length of the duration of the motion and the weighting factor was 0.5, then the time duration would see only a 150% increase as opposed to a 200% increase. Therefore, the weighting factor can be applied to

C_T from Eq. 4.14 as

$$C'_T = W \cdot (1 - C_T) + C_T \tag{5.6}$$

where W (<1) is the weighting factor and corresponds to the weighting of importance of the constraints. The result of the first stage is then filtered by the second stage of the CATF using spatial scaling until the motion is completely stable. Stability can be guaranteed since spatial scaling can always result in a stable COM trajectory.

In the case of using the spatial scaling as the first stage of the CATF, the value of C_s must guarantee that during the second stage, the CATF will be able to filter and stabilize the reference motion using time scaling. If there are any cases where it is impossible to stabilize the motion through using time scaling alone (see Case 3 in Sec. 4.3), then the C_s value must be

$$C_s > \frac{|x^{\text{Limit}}_{\text{ZMP}} - x^{\text{Ref}}_{\text{COM}}|}{x^{\text{Nom}}_{\text{ZMP}} - x^{\text{Ref}}_{\text{COM}}} \tag{5.7}$$

where $|x^{\text{Limit}}_{\text{ZMP}} - x^{\text{Ref}}_{\text{COM}}|$ is at its greatest for the motion's duration. Though Eq. 5.7 guarantees that the CATF can stabilize the motion in the second stage using time scaling, it does not consider the resulting duration of the motion after time scaling. If a portion of the reference motion is classified as Case 3, then pulling $x^{\text{Ref}}_{\text{COM}}$ just under the ZMP limit will require the COM at that moment to have almost no acceleration in order to be stable. Near zero acceleration would require a nearly infinite motion duration. Alternatively, if C_s is 1, then $x^{\text{Ref}}_{\text{COM}}$ would be pulled down so that it is constant at the nominal ZMP, which means that the motion duration could be infinitely small. Therefore, the duration or time step of the motion can actually be specified in this two stage process by adjusting the value of C_s. The time step of the filtered motion can be found simply as

$$T_{\text{Tot}} = \frac{T_s}{C_T} \tag{5.8}$$

since the entire duration/speed of the motion is scaled by C_T. If taking the approach of guessing a C_s value first, and then calculating the corresponding C_T value needed to stabilize the motion, combining Eqs. 4.8 and 4.14 yields the following equation to be used

$$C_T = \sqrt{\frac{x^{\text{Ref}}_{\text{COM}}(\tau) + C_s\left(x^{\text{Nom}}_{\text{ZMP}} - x^{\text{Ref}}_{\text{COM}}(\tau)\right) - x^{\text{Limit}}_{\text{ZMP}}\big)}{H\left(\ddot{x}^{\text{Ref}}_{\text{COM}}(\tau) + C_s\left(-\ddot{x}^{\text{Ref}}_{\text{COM}}(\tau)\right)\right)}} \tag{5.9}$$

Alternatively, as seen in Eq. 5.9, if C_T is known, then C_s can solved for. Therefore the desired duration of the motion can be set by calculating C_T from Eq. 5.8 and then C_s calculated using

$$C_s = \frac{x_{\text{ZMP}}^{\text{Limit}} - (x_{\text{COM}})_j + H\,(\ddot{x}_{\text{COM}})_j\,{C_T}^2}{x_{\text{ZMP}}^{\text{nom}} - (x_{\text{COM}})_j + H\,(\ddot{x}_{\text{COM}})_j\,{C_T}^2} \tag{5.10}$$

where j is the point where the motion is most unstable. In conclusion, the order of applying space and time scaling does not matter. Inevitably, any value of C_T can specified and a corresponding C_s value determined. However, C_T can be specified as either a weight fraction of what is needed to stabilize the motion, or as a value to dictate the motion's duration.

5.3 Segmentation

Segmentation can be employed to stabilize different parts of a motion as needed. For example, typically the double support phase of a robot's walking cycle is stable. Therefore, only the single support stage needs to be stabilized. The only clause is that the filtered version of the motion must smoothly transition to the unfiltered, yet stable, portions of the motion. Time scaling or spatial scaling with 4 point constraints could be used to accomplish this. Alternatively, the double support phase could be ignored and a spline used to join the two single support phases together.

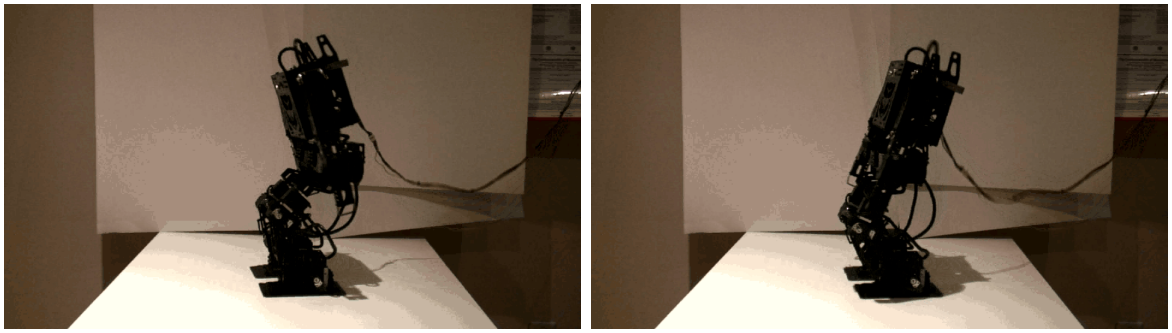
Chapter 6

COM-based Motion Adaptation (CMA)

COM-based Motion Adaptation (CMA) is the part of the AMF that uses the stable AMF COM trajectory (generated from the CATF) and the reference joint angle trajectories, to generate new joint angle trajectories that result in the AMF COM trajectory. As mentioned in previous chapters, there are an infinite number of ways to generate joint angles to satisfy a given COM position. There is also ambiguity as to what the optimum way to find such joint angles is. The CMA in this work serves a starting point and one of many ways the CMA could work. Since the AMF focuses on using analytical solutions, the CMA presented in this work will use as many analytical solutions as possible–staying away from numeric trial and error search methods. The CMA's function is still largely dependent on what similarities the filtered motion should have with the reference motion. Again, these similarities can be expressed as constraints or by altering the CMA algorithm. Presented here are a number of methods to find a solution for joint angles that satisfy the COM trajectory. Each of the methods can be weighted or ignored depending on how undesirable their impact is on the filtered motion. Additionally, any limitations in the methods' ability to the COM must be expressed as a constraints in the CATF so that the CATF does not generate a COM trajectory that the CMA can not satisfy. In this respect there is a symbiotic relationship between the CATF and CMA.

6.1 Using the waist

The current platform used for experimentation and testing of the work presented here, DARwIn, has a degree of freedom in the waist that allows the torso to pitch forwards and backwards independently of the hips and lower body. From experience, the most frequent instabilities during walking often occur in the X-direction So having a single degree of freedom



(a) (b)

Figure 6.1: Video stills of a robot leaning back and forth to maintain stability by moving the COM location in the X direction.

that can control the COM location of the robot in the X-direction independently of the Y-Direction is very desirable. Even if the hips are not perfectly level, during most motions they are close to being level, which means that pitching the torso forwards or backwards will result in mostly a change in the location of the COM in the X-direction. With this in mind, the waist joint is used to change the location of the COM in the X-direction so that the filtered COM trajectory matches the resulting COM trajectory derived from the filtered/AMF motion in the X-direction. If using the pitch in the waist is not sufficient, due to either kinematic limits or imposed range constraints, other methods can be used in addition to or in conjunction with bending at the waist. When other methods are used in addition, the waist could be moved in an extreme position to get the resulting COM as close to the AMF COM trajectory as possible and then use another method to move it the rest of the way. Alternatively, if other methods are used in conjunction with bending at the waist, the waist may move only a fraction of its limit in order to move the resulting COM while the other method is used to move the COM the rest of the way.

In order to calculate the exact waist angles needed to satisfy the filtered COM locations a very simple search method is used. Starting with the robot's reference joint angle trajectories, only the angle of the waist is changed to see the resulting COM location. Calculating the COM is a very quick, closed form solution. By scanning through the range of angles the waist can move through, the angle that correctly locates the COM can quickly be determined. Changes in the height of the COM resulting due to the pitching of the waist are generally small, but can be compensated for by adjusting the height of the hips as it will be explained in the next section.

Figure 6.1 shows video stills of the robot as it leans back and forth to stabilize the gait by moving the COM in the X direction.

6.2 Using the hips

Using the location of the hips as a tool for changing the location of the COM is useful for a number of reasons. First, moving the hips moves the majority of the robot's mass the same amount in the same direction. Though moving the hips one centimeter in the Y-direction will not necessarily mean that the COM moves one centimeter in the Y-direction, it is very close. Most of the mass is located in the torso and hips for most robots, which has a one-to-one correspondence with how the hips move. The mass in the legs is generally a lot less than the rest of the body and it moves with the hips probably at closer to a one-to-two ratio. In any case, if a support polygon is specified that is slightly smaller than the robot's actual footprint, the factor of safety should be sufficient to compensate for the error generated when assuming that the COM moves exactly with the hips. Additional searching can be used similar to the waist to find a solution that more exactly places the COM. Another aspect to consider is that moving the hip in the Y-direction may result in a change in COM location in the X-direction as well. Again, this should be relatively small. Finally, the inverse kinematic solutions for the hip location is well known and in closed form. This means that simply shifting the hip locations from the reference motion locations by the difference between the COM reference and AMF trajectories will correctly place the COM. Similar to using the waist, using the hips to place the COM will result in kinematic constraints that must be placed on the CATF. The hips can only be moved so far in the X and Y directions. The limits are also coupled to each other, meaning that while in one extreme in the X-direction, the kinematic range of the hips in the Y-direction will be different than in other configurations. Additionally, changing the hip location in the Z-direction will not only yield different ranges for the X and Y directions, but will also dramatically change the ZMP calculation. For more discussion on this intricacy, see Sec. 4.4. Generally all three directions can be assumed to be decoupled. This also allows the hips to move in the vertical direction to compensate for changes in the height of the COM caused by changes in the waist pitch.

6.3 Using the arms or legs

Using the robot's appendages to move the COM is not as straight forward as using the waist or hips. Unless the arms are carrying large masses, using the arms to move the COM will not be as effective as using the waist or hips. The arms will have to move much more in order to generate the same change in COM location that moving the waist or hips would. Therefore, using the arms to move the COM should be avoided unless using the waist/hips is insufficient or undesirable.

The arms have a natural limit of how far they can influence the location of the COM in the X, Y, or Z direction. With the arms perfectly straight, horizontal, and forwards, they will displace the COM in the positive X-direction as much as possible. Similarly, if the arms are in the same configuration, but backwards, they will displace the COM in the negative

X-direction as much as possible. If the arms can not be perfectly straight and horizontal due to joint limits or constraints, then the configuration that results in each link being as close to horizontal as possible would be the configuration that results in the maximum COM displacement.

The maximum COM displacement the arms can have in the Y-direction largely depends on the physical configuration of the robot. Obviously one of the arms can be fully extended in one of the Y-directions. However, the complimentary arm usually can not fully extend in the same direction. For example, the right arm may fully extend to the right to move the COM to in the negative Y-direction, but the left arm cannot reach through the robot's body to achieve the same result. The left arm would have to reach across the body in some form to help move the COM in the negative Y-direction. Consequently, it is entirely possible that the COM would be displaced in the X-direction as well. Similar to the hips, the limits for moving the COM in the X and Y directions are coupled.

Though not currently implemented, a possible way to incorporate the arms into the CMA is presented here. As described above, the arms have a configuration that moves the COM the most in the X or Y direction. If it is possible for the arms to correctly position the COM without violating kinematic constraints, it exists between the reference configuration and the extreme configuration. Therefore if we define the arm configuration that correctly positions the COM as some fraction between the reference configuration and the extreme configuration, the fraction can quickly be determined by searching from 0 to 1.

The same methods used to adapt the arm joint angle trajectories can be used to adapt the leg joint angle trajectories if the leg is not on the ground. However, this is not advisable if the motion is supposed to be a walking motion since the ground contact and support polygon is determined by where the swing foot touches the ground. Currently there is not a way in the AMF to incorporate a change in ground contact or support polygon due to the CMA.

Chapter 7

Practical Examples Using the AMF

In order to validate and help to illuminate the algorithms used in the AMF, this section uses some practical examples of reference motions that are filtered under different constraints using space or time scaling. Sec. B lists the physical properties of the robot that were used in the AMF algorithms. The testing platform is DARwIn, a Dynamic and Anthropomorphic Robot with Intelligence, which is described in more detail in Sec. 9. All units are standard SI unless otherwise specified. Additionally, the horizontal dashed red lines for all of the plots represent the ZMP limits unless otherwise specified.

Each example motion is organized in the following manner: The equations governing the reference motion are described, corresponding plots of the COM and ZMP are detailed, and simulation video stills are shown. Then, a select number of algorithms and constraints are used to filter the motion. For each filter application, the resulting COM and ZMP are detailed and discussed. If applicable, the corresponding CMA results are shown as well as video still shots of the filtered motion.

As detailed in Sec. 6, there are many ways to generate the joint angle trajectories of a robot from the COM trajectory. The CMA uses the reference joint angle trajectories as a starting point and then modifies them in order to achieve the desired AMF filtered COM trajectory. For the following examples, the waist is used as the primary means for adjusting the COM position in the X direction. If the waist cannot adjust the COM position in the X direction enough to satisfy the AMF COM trajectory, then the hip trajectory is adjusted to make up for any shortcomings. The hip is also used to adjust the COM position in the Y direction. For the CATF, a waist angle limit of $\pi/4$ is used to determine the kinematic limits of the COM in the X direction. However, when applying the CMA for these examples, a limit of $\pi/8$ is used in order to show how the hips can also contribute to adjusting the COM location in the X direction.

7.1 Circular sway

A simple non-walking motion such as a robot swaying in a circle (much like when using a hoola-hoop), can demonstrate the majority of the capabilities of the AMF. The hips follow a cyclic circle trajectory while the upper body is motionless.

7.1.1 Equations governing reference motion

The circular sway gait is very simple. The hips move in a circular motion and the feet do not leave the ground. The motion is described as

$$\text{Hip}_x(t) := 0.04\text{Sin}(2\pi\text{t}) + 0.02 \tag{7.1a}$$
$$\text{Hip}_y(t) := 0.04\text{Cos}(2\pi\text{t}) \tag{7.1b}$$
$$\text{Hip}_z(t) := 0.23 \tag{7.1c}$$

and $T_s = 0.5$.

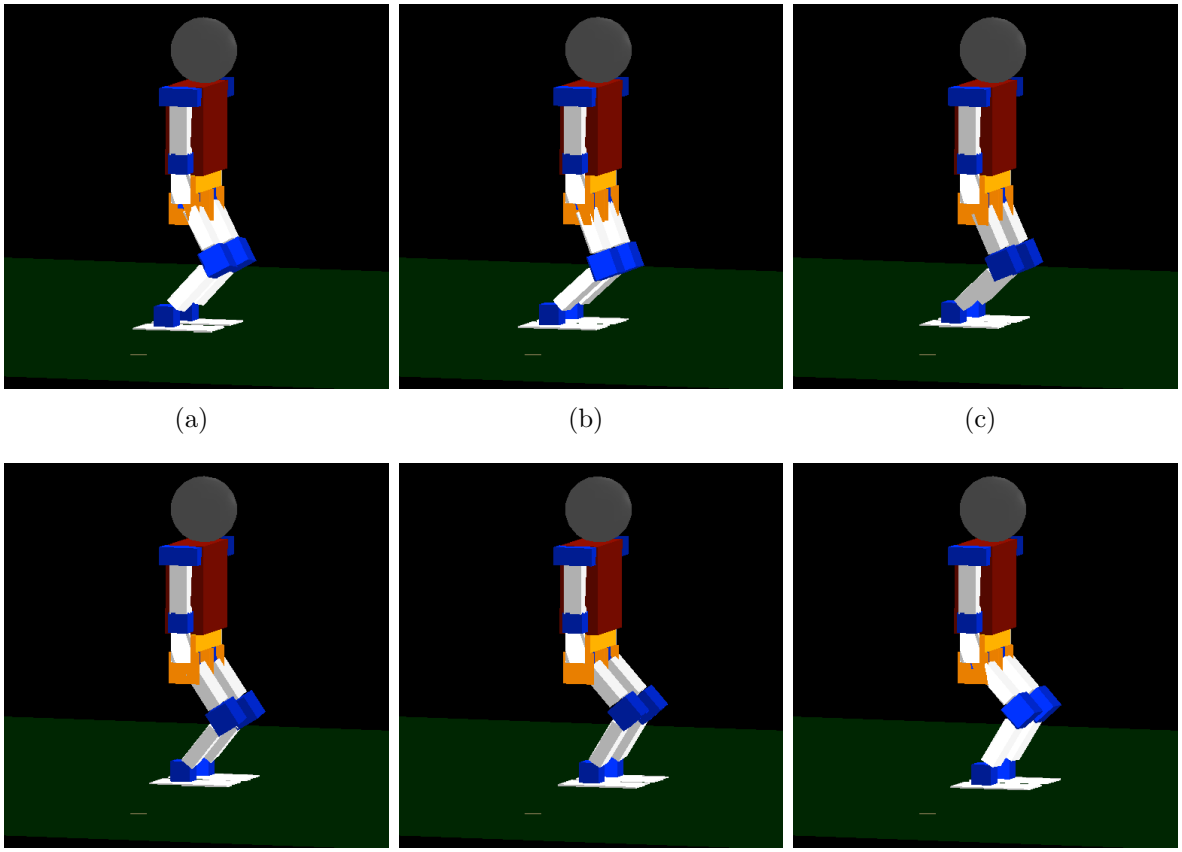
Figure 7.1 shows a sequence of still shots of the gait in a 3D visualization simulator. Figure 7.2 shows a picture of the robot attempting to perform the reference circular gait on a surface with a relatively high coefficient of friction. The instability is very clear as the robot falls over almost immediately after attempting to perform the gait. Figure 7.3 shows a picture of the robot attempting to perform the reference circular gait on a surface with a relatively low coefficient of friction. The robot does not fall over, but tips up on the edges of both its feet.

7.1.2 Resulting reference COM and ZMP trajectory

The results for the COM reference trajectory can be seen in Fig. 7.4. Also seen are the COM limits enforced by kinematic constraints. The range of the X-COM is dictated by how far forward and back the waist of the robot can move since it will be used as the primary means in the CMA when generating the filtered joint angles. The range of the Y-COM coincidentally follows the Y-COM trajectory, but is calculated based on the kinematic limitations of the robot. While maintaining the same hip height and X-COM location, the hips can only go so far in the Y-direction based on the link lengths of the legs.

The results for the ZMP reference trajectory can be seen in Fig. 7.5. As seen, both the X and Y directions of the motion have unstable portions. Figure 7.6 shows a parametric plot of both the reference COM and ZMP trajectories. [1] It is clear that circular sway motion is

[1]The ZMP points that do not line up smoothly with the rest of the motion (also seen in Fig. 7.5) are



(a) (b) (c)

(d) (e) (f)

Figure 7.1: Stills of the circular sway reference gait from simulation

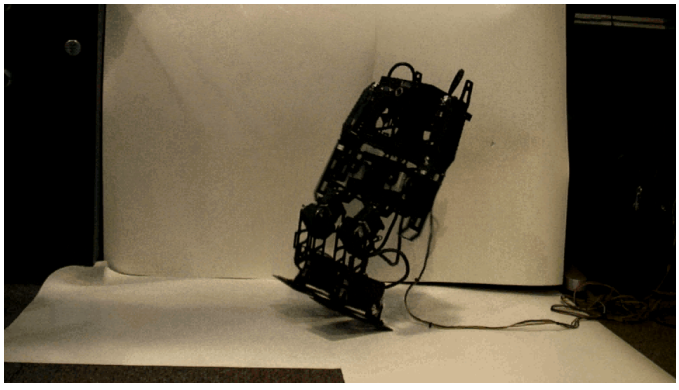


Figure 7.2: Picture of DARwIn falling over while trying to perform the reference circular gait on a high friction surface.

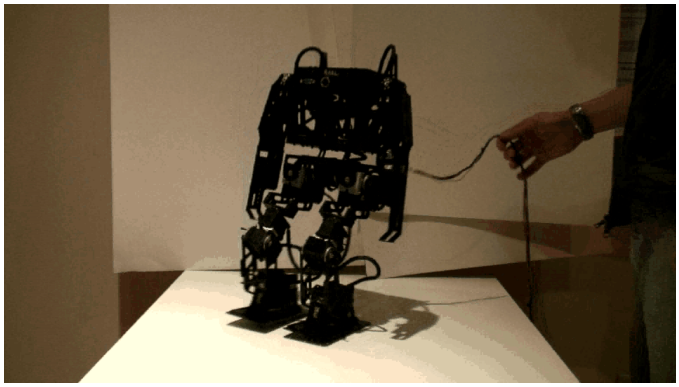


Figure 7.3: Picture of DARwIn tipping while trying to perform the reference circular gait on a low friction surface.

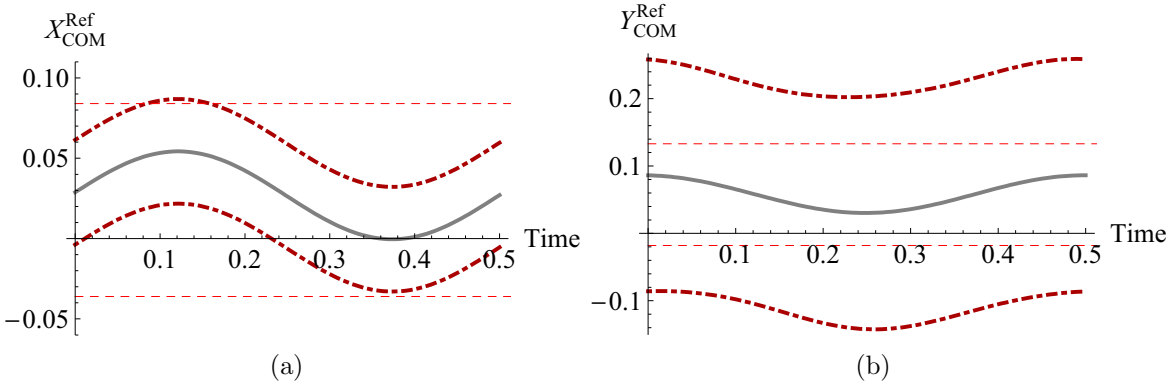


Figure 7.4: COM reference trajectories for a circular sway motion. Reference trajectory shown as solid gray and the kinematic limits of the COM are shown as dot dashed dark red.

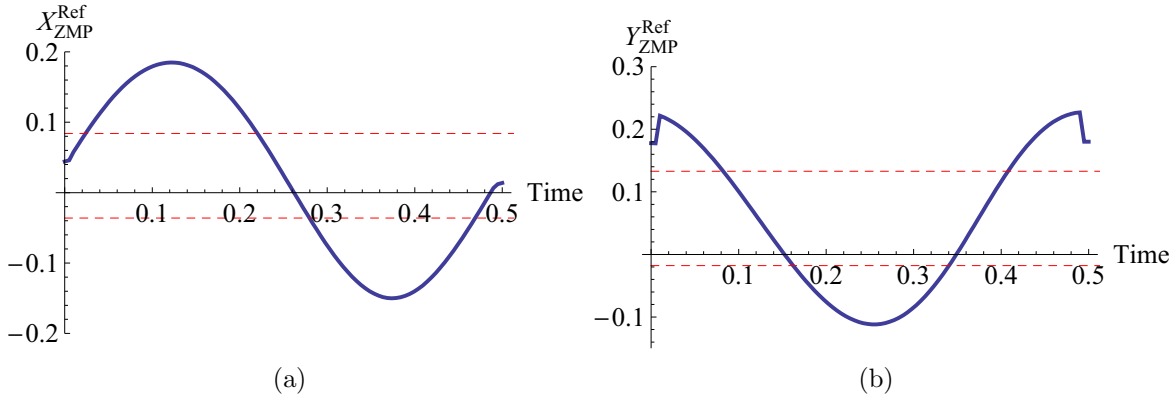


Figure 7.5: ZMP reference trajectories for a circular sway motion.

largely unstable, with the ZMP always outside of the support polygon.

7.1.3 Simple spatial scaling

The solutions from Sec. 4.2 are applied to both X and Y directions of the reference motion in order stabilize the motion. Since there is no walking or changing of ground contact, simple spatial scaling makes sense as a way to stabilize the gait while retaining characteristics of the reference gait such as shape, timing, etc. As seen in Fig. 7.7 and 7.8, the CATF scales the COM motion just enough to create a stable motion. The reference and filtered COM trajectories maintain similar characteristics of shape.

The damped out motion should result in the waist pitching in the opposite direction that the rest of the robot is moving. Additionally, the hips of the robot should not be moving as much laterally as in the reference motion. Figure 7.9 shows the resulting changes in the hip trajectory and waist angle trajectory. As seen, the waist cannot completely move the COM in the X direction to satisfy the AMF trajectory without hitting the limit of $\pi/8$. Therefore, at the clipped portions of the waist angle trajectory, the location of the hip in the X direction slightly deviates from the reference trajectory in order to locate the COM. Additionally, in the filtered version of the motion, hips do not have as large an amplitude in the Y direction; the motion damped out. Finally, there was very little change in the vertical location of the COM after changing the waist angle and hip locations. This makes sense since the waist at an angle of $\pi/8$ only changes the vertical location of the COM by about 3%. [2] Figure 7.10 shows a sequence of still shots of the filtered gait in simulation. The pitching of the torso

a result of inaccurate estimations of the acceleration of the COM at the beginning and ends of the motion. Since motion data is not available beyond those points, the accelerations calculations are a little off.

[2]Change in height of the torso's mass is $1 - Cos(\pi/8) = 7.6\%$. Additionally, the torso is only 38.6% of the total mass of the robot. Therefore, $7.6\% \cdot 38.6\% = 2.93\%$ for the total percent change in height.

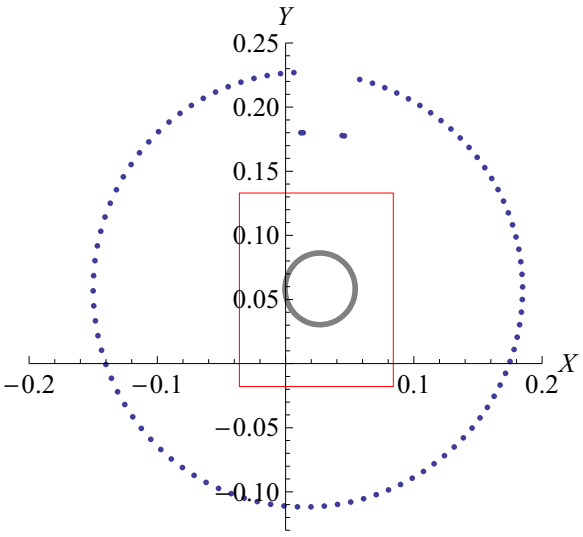
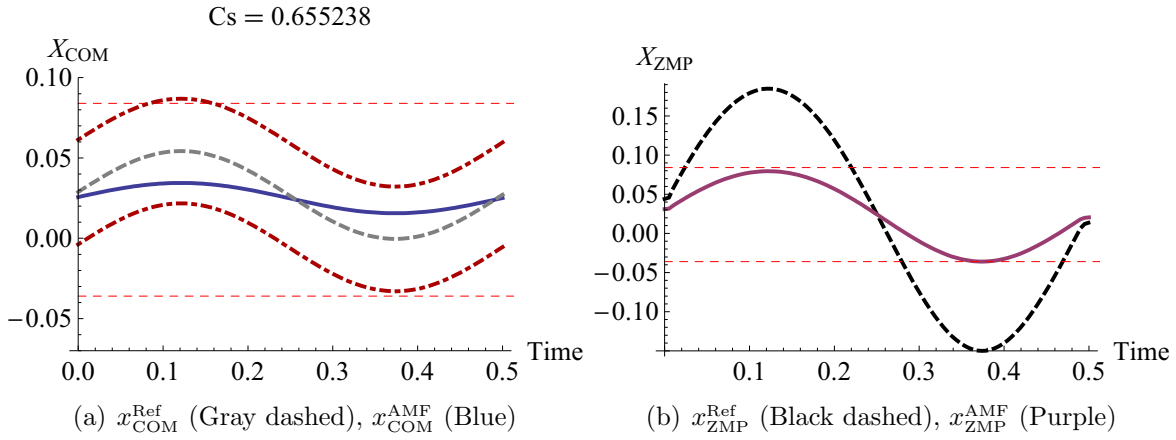
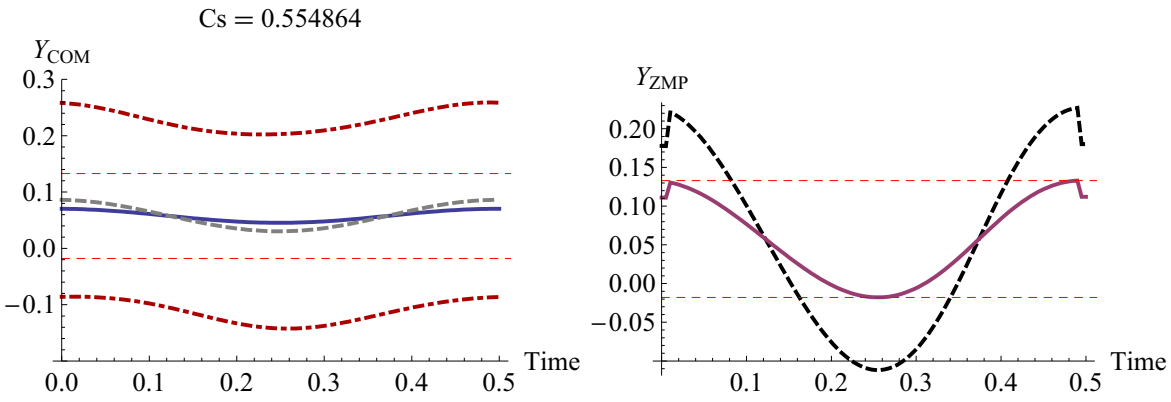


Figure 7.6: COM (Gray) and ZMP (Blue) reference trajectories for a circular sway gait. The red box represents the support polygon.



(a) $x^{\text{Ref}}_{\text{COM}}$ (Gray dashed), $x^{\text{AMF}}_{\text{COM}}$ (Blue) (b) $x^{\text{Ref}}_{\text{ZMP}}$ (Black dashed), $x^{\text{AMF}}_{\text{ZMP}}$ (Purple)

Figure 7.7: AMF X direction trajectories from spatial scaling using the circular sway gait



(a) $y^{\text{Ref}}_{\text{COM}}$ (Gray dashed), $y^{\text{AMF}}_{\text{COM}}$ (Blue) (b) $y^{\text{Ref}}_{\text{ZMP}}$ (Black dashed), $y^{\text{AMF}}_{\text{ZMP}}$ (Purple)

Figure 7.8: AMF Y direction trajectories from spatial scaling using the circular sway gait

is the most noticeable different when comparing the filtered motion against the reference motion. However, if the upper body motion is not important with respect to the importance of the lower body motion, then it is perfectly acceptable to have the upper body pitching if it means the lower body motion remains relatively unchanged.

7.1.4 Simple time scaling

In order to determine if time scaling is feasible, the CATF analyzes the motion to determine what cases of instability occur as described in Sec. 4.3. Figure 7.11 shows that both directions have instabilities, none of which are Case 3. Therefore it may be possible to stabilize the gait simply by slowing it down or speeding it up.

Figure 7.12 shows the result of using Eq. 4.13. As seen, the motion needs to slow down in both directions in order to be stable. Since the X and Y directions share the same timing, only one value of C_T can be used. The most extreme value ($C_T = 0.488$) is tested in both directions since it is smallest amount of time scaling necessary to stabilize the X direction. It is possible that in choosing the most extreme C_T value, that the Y direction AMF motion destabilizes. To check to see if this happens, Fig. 7.13 shows the resulting ZMP in both directions. As seen, both directions remain stable and the Y direction is over-stabilized since it slows down more than necessary for a stable motion in that direction.

7.1.5 Time, then spatial scaling

For specifics on the algorithms for combining spatial and time scaling, see Sec. 5. The first form of combining the space and time scaling algorithms will be to use a fraction of the

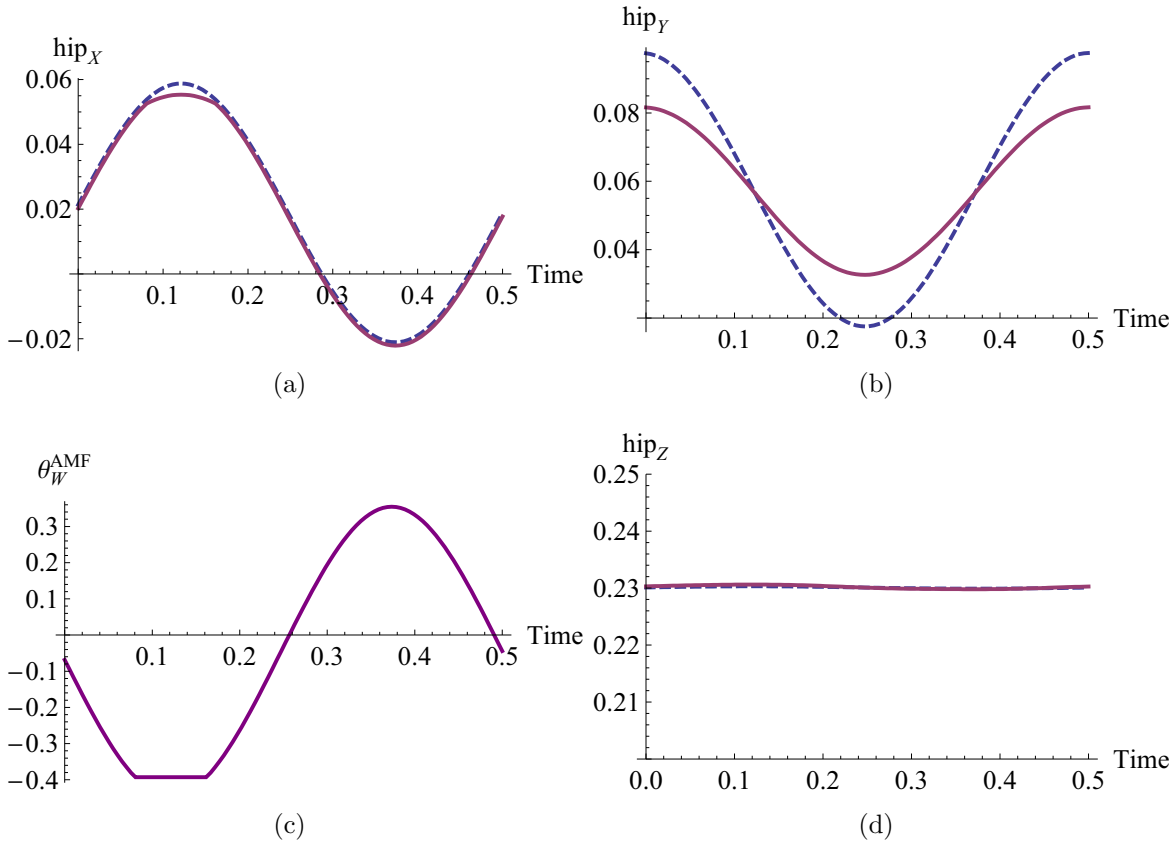


Figure 7.9: Resulting AMF motion after applying the CMA to the circular sway gait that was stabilized using simple spatial scaling by the CATF. Reference motion is dashed blue and the AMF motion is solid purple.

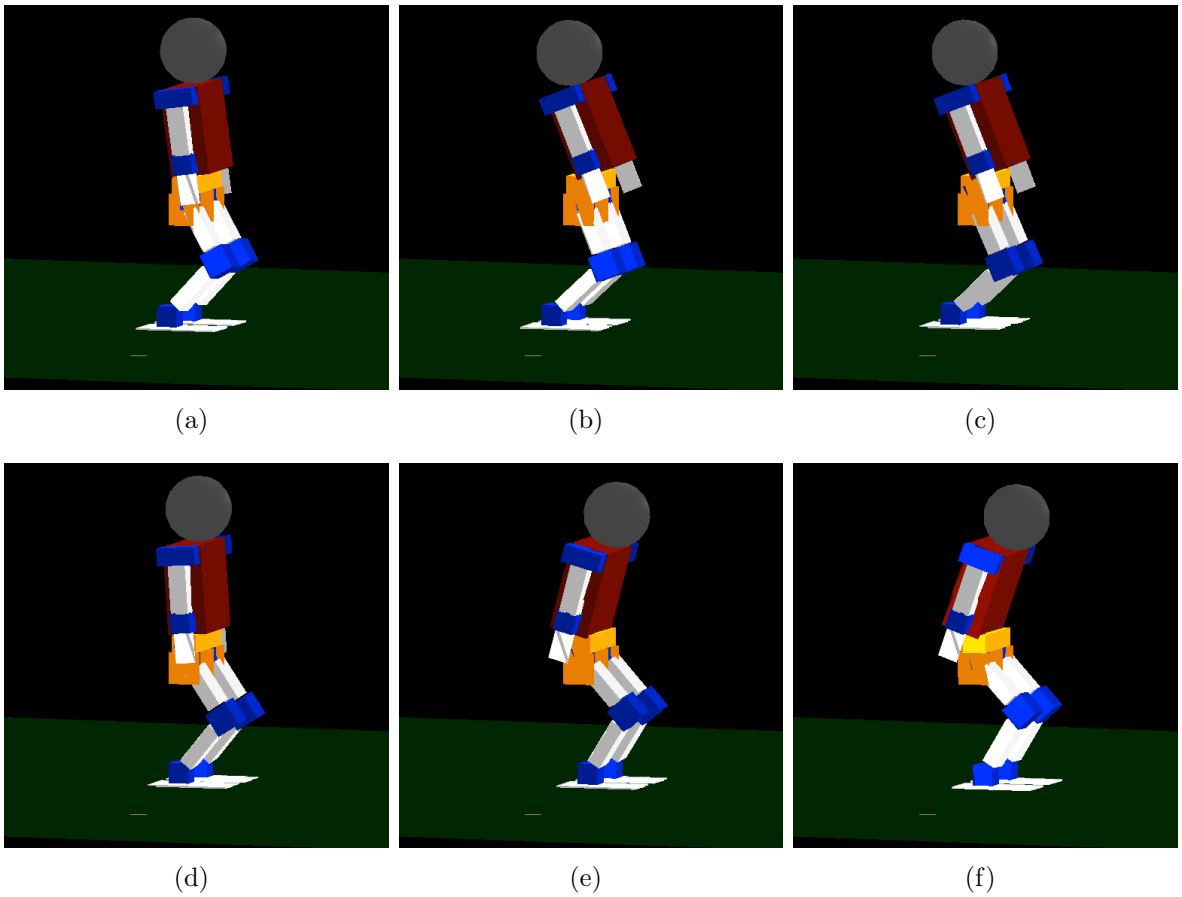


Figure 7.10: Simulation stills of the AMF solution using spatial scaling applied to the circular sway gait

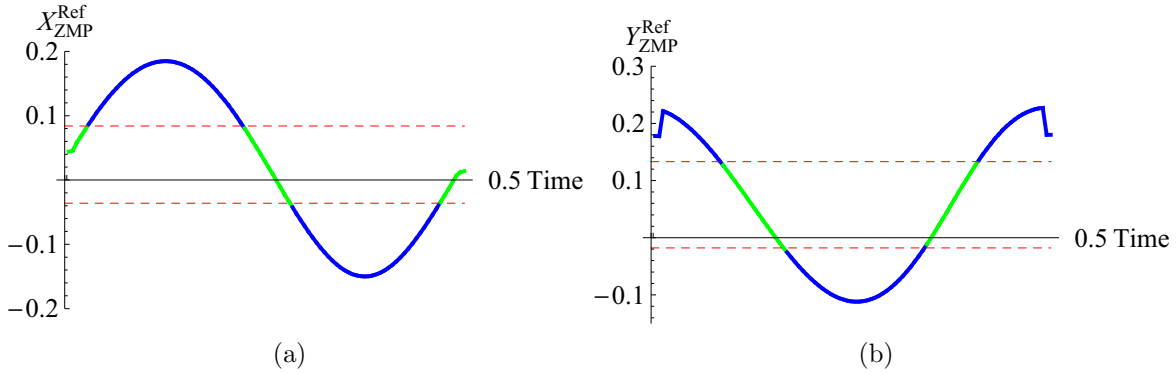


Figure 7.11: Categorized instability for the circular sway gait. Blue is Case 1, and Green is Stable.

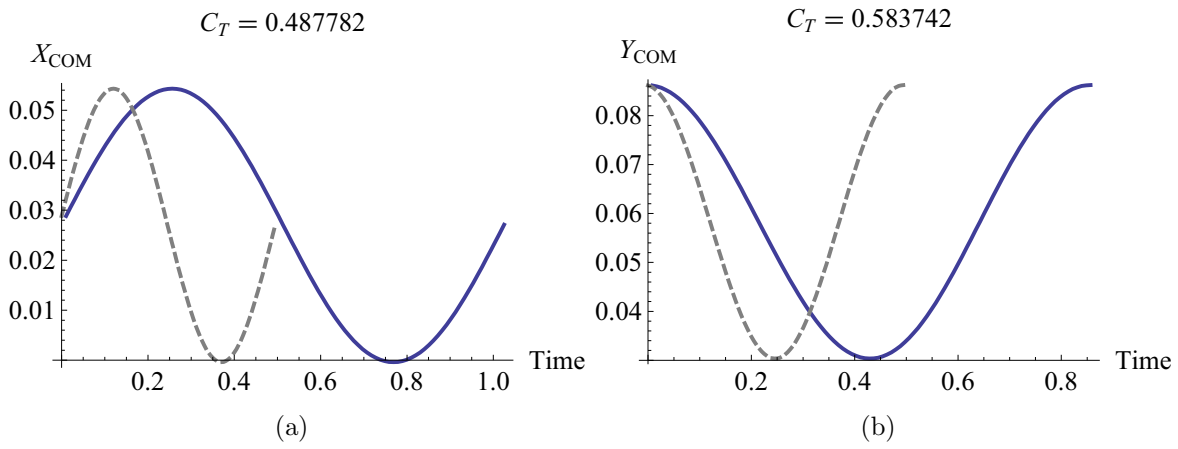


Figure 7.12: AMF COM trajectories from using time scaling on the circular sway gait. Reference COM is dashed gray and AMF COM is solid blue.

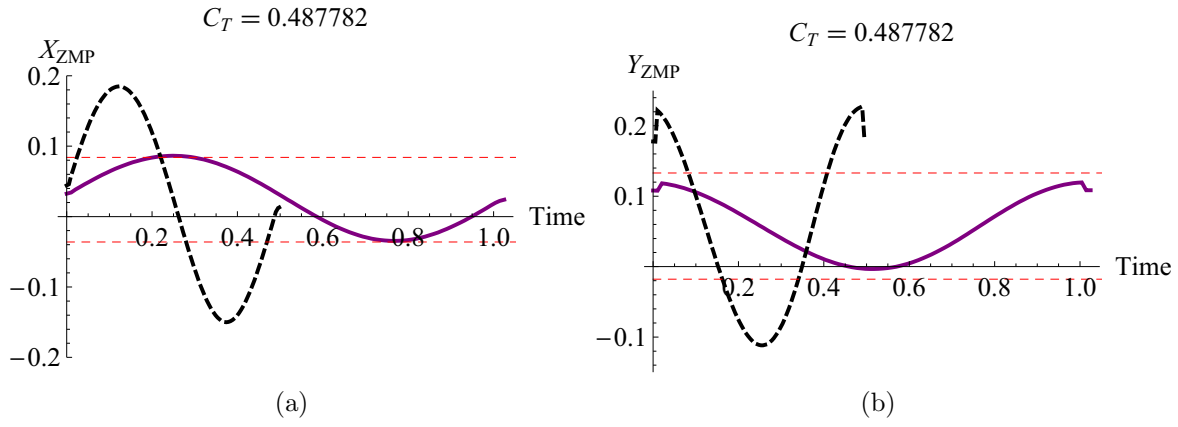


Figure 7.13: AMF ZMP trajectories from using time scaling on the circular sway gait. Reference ZMP is dashed black and AMF ZMP is solid purple.

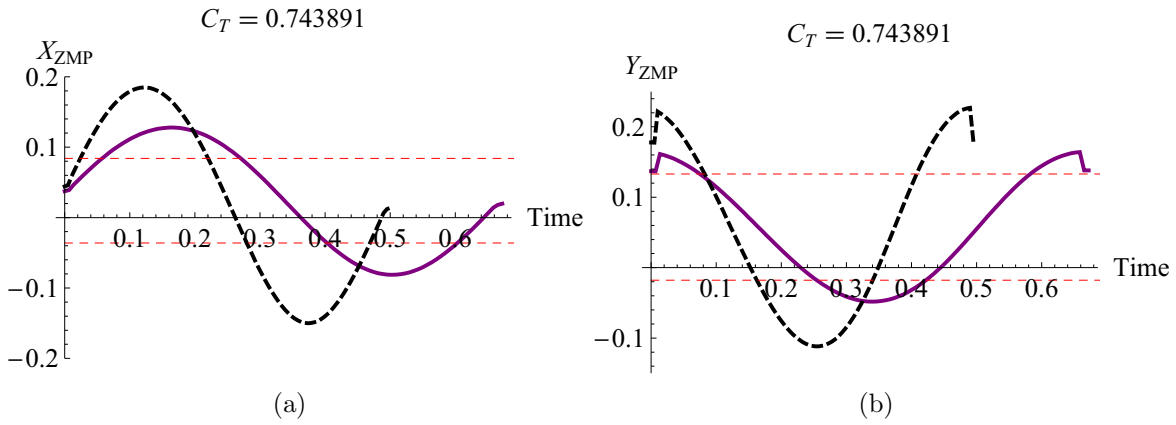
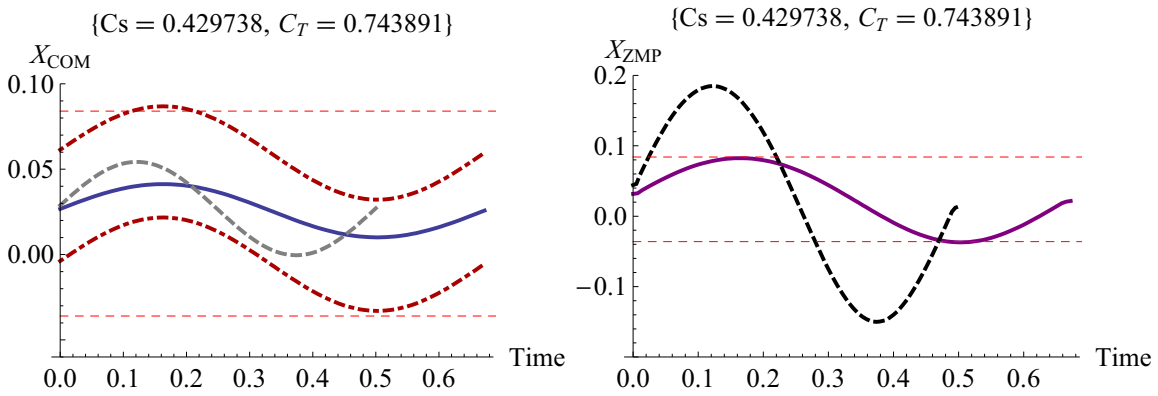


Figure 7.14: AMF ZMP trajectories from using partial time scaling on the circular sway gait. Reference ZMP is dashed black and AMF ZMP is solid purple.

time scaling necessary to stabilize the motion and then use the necessary amount of spatial scaling to stabilize the motion. Choosing an arbitrary weight of $W = 0.5$, Eq. 5.6, and the resulting C_T from the previous section, the partial, weighted C_T is 0.79. The resulting ZMP of this partial time scaling is shown in Fig. 7.14. As seen, the ZMP in both directions is still unstable and requires spatial scaling.

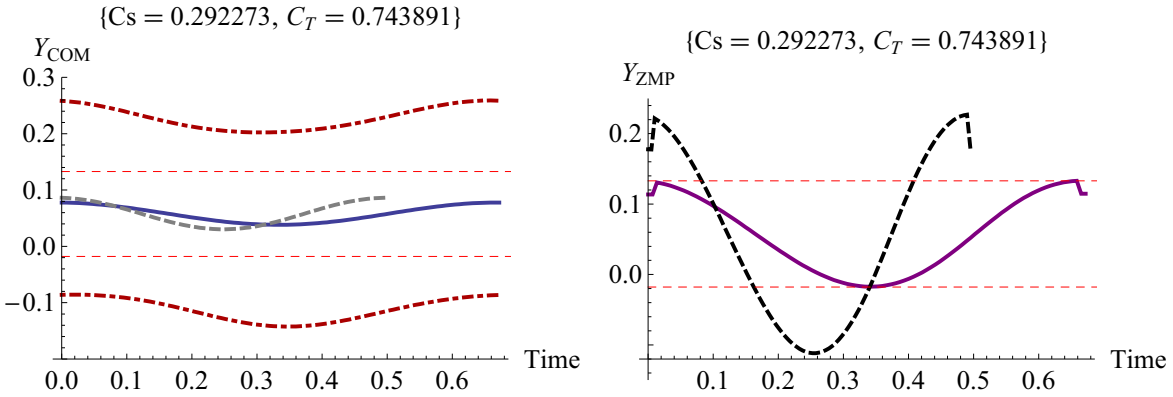
The result of the partial time scaling is used as a "reference motion" needing stabilizing from using simple spatial scaling. Figures 7.15 and 7.16 show the result of applying this second stage of the AMF. As seen, the result is a version of the reference motion that is scaled in both time and spatial dimensions. Consequently, since both algorithms were used in conjunction, less of scaling is needed in each direction than what was required independently (see previous sections). Additionally, neither motion is unnecessarily over-stabilized, which can occur when using just time scaling. Not only does combining the time and space scaling algorithms provide an excellent compromise, in some ways it changes the motion reference motion even less than space or time scaling alone.

Using only a fraction of the necessary time scaling needed to stabilize the motion and then using spatial scaling to completely stabilize the motion, led to a smaller fraction of spatial scaling needed to stabilize the motion. Therefore, we should see smaller waist angles and changes in the hip trajectory needed to satisfy the AMF COM trajectory. Figure 7.17 shows the resulting waist angle and hip trajectories. The Z direction is not shown since in the previous example, it was shown that it was negligibly impacted by the current implementation of the CMA. As seen, there is no clipping in the waist angle trajectory since it never needed to be more than $\pm\pi/8$ in order to move the COM to follow the AMF trajectory in the X direction. Additionally, the hip direction in the X direction does not change. The Y direction of the hip changes, but not nearly as much as in the previous section since the spatial scaling of the COM trajectory here is less.



(a) x^{Ref}_{COM} (Gray dashed), x^{AMF}_{COM} (Blue) (b) x^{Ref}_{ZMP} (Black dashed), x^{AMF}_{ZMP} (Purple)

Figure 7.15: AMF X direction trajectories from time then spatial scaling using the circular sway gait



(a) y^{Ref}_{COM} (Gray dashed), y^{AMF}_{COM} (Blue) (b) y^{Ref}_{ZMP} (Black dashed), y^{AMF}_{ZMP} (Purple)

Figure 7.16: AMF Y direction trajectories from time then spatial scaling using the circular sway gait

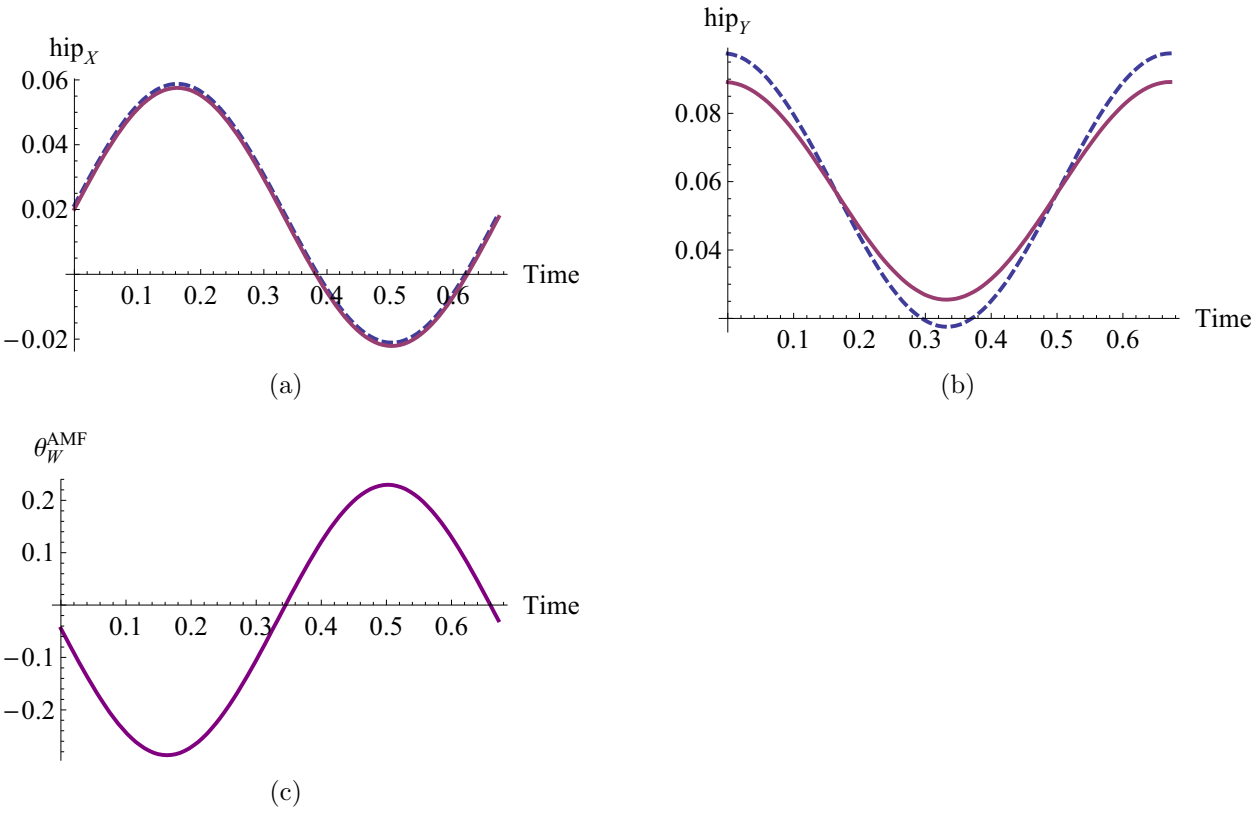
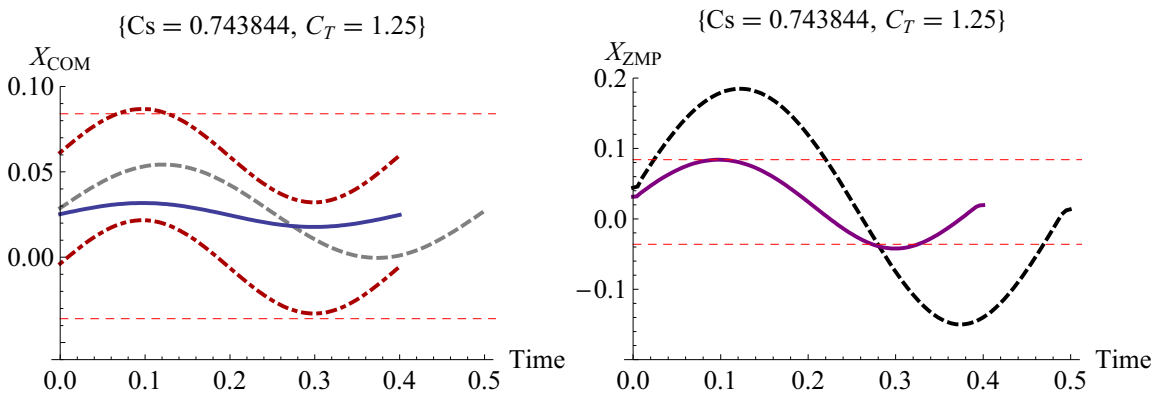


Figure 7.17: Resulting AMF motion after applying the CMA to the circular sway gait that was stabilized using time then spatial scaling by the CATF. Reference motion is dashed blue and the AMF motion is solid purple.



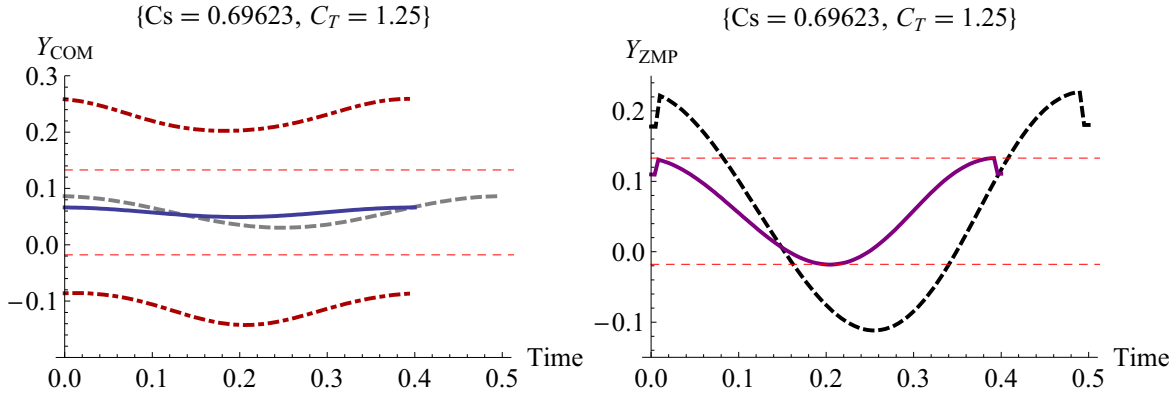
(a) $x^{\text{Ref}}_{\text{COM}}$ (Gray dashed), $x^{\text{AMF}}_{\text{COM}}$ (Blue) (b) $x^{\text{Ref}}_{\text{ZMP}}$ (Black dashed), $x^{\text{AMF}}_{\text{ZMP}}$ (Purple)

Figure 7.18: AMF X direction trajectories from specifying the step time, and then using spatial scaling to stabilize the circular sway gait

7.1.6 Setting the step time and stabilizing with spatial scaling

Section 5 explains the thought behind selecting a step time (motion duration) and then using spatial scaling to stabilize the motion. For this example, the reference step time is 0.5 seconds. If we desire the filtered motion's step time to be 0.4 seconds, C_T must be 1.25. This makes sense since a C_T value greater than one indicates that the motion is sped up. Since we saw that slowing the motion down brought stability, it follows that speeding the motion up may cause greater instabilities. Therefore we would expect a larger C_s to scale and stabilize the motion. Figures 7.18 and 7.19 show the resulting AMF motion that is only 0.4 seconds in duration and is stabilized from spatial scaling. We also see that the C_s value, as expected, is higher than if only spatial scaling were used (0.74 vs. 0.65 for the X direction and 0.69 vs. 0.55 in the Y direction).

Since the time step for this section (0.4 s) was set to be less than the reference time step (0.5 s) and, as seen from the time scaling, the motion needs to be slower in order to be stable, more spatial scaling was needed to stabilize the motion. Consequently, we should see larger changes in the hip location and larger changes in the waist angle in order to move the COM to the AMF trajectory. Figure 7.20 shows the resulting trajectories. As seen, there is more clipping than in Sec. 7.1.3, which also results in larger changes in the X trajectory of the hip. Additionally, there is also a larger change in the Y trajectory of the hip. What is important to note is that even though larger changes in the COM trajectory were needed or order to make the motion stable, the kinematics did not prevent the CMA from calculation appropriate joint angle trajectories. This is ensured by enforcing the kinematic constraints via COM trajectory limits in the CATF. Figure 7.21 shows a sequence of still shots of the filtered gait from simulation.



(a) $y^{\text{Ref}}_{\text{COM}}$ (Gray dashed), $y^{\text{AMF}}_{\text{COM}}$ (Blue) (b) $y^{\text{Ref}}_{\text{ZMP}}$ (Black dashed), $y^{\text{AMF}}_{\text{ZMP}}$ (Purple)

Figure 7.19: AMF Y direction trajectories from specifying the step time, and then using spatial scaling to stabilize the circular sway gait

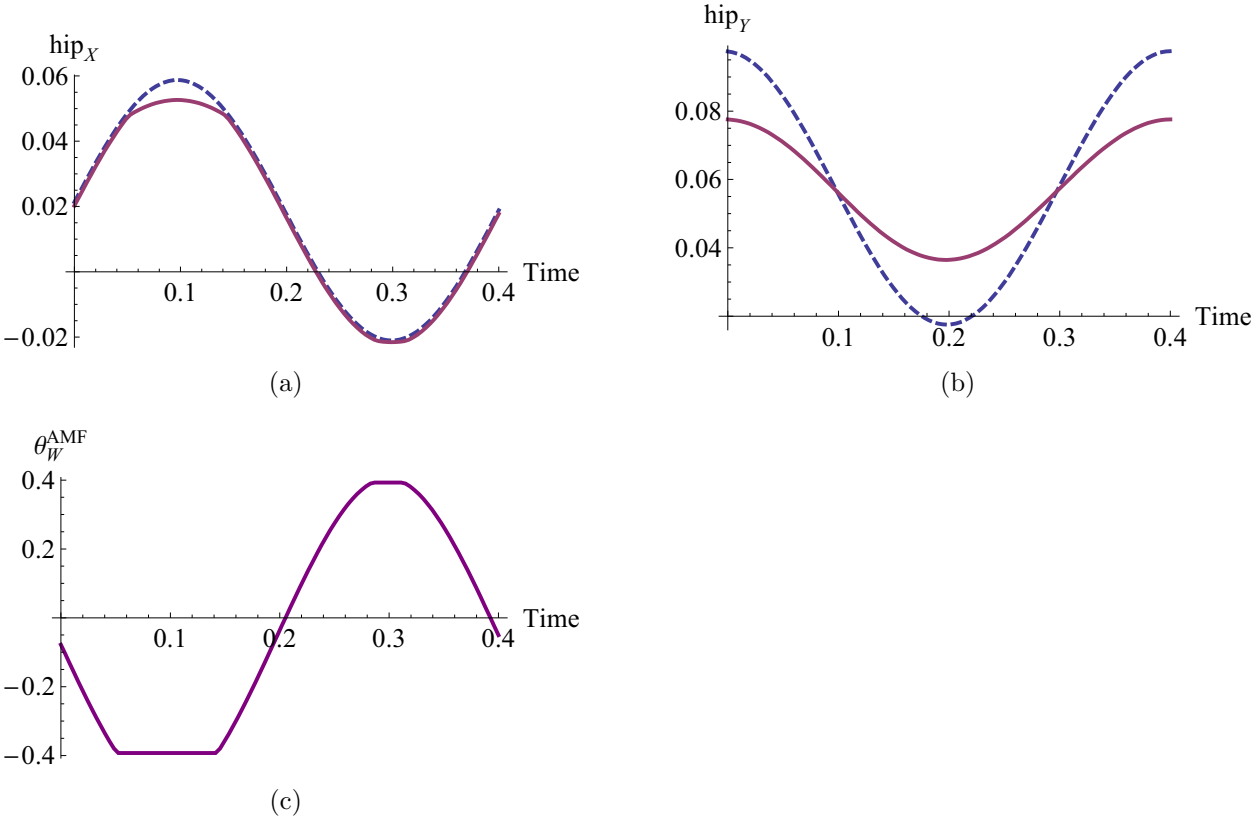
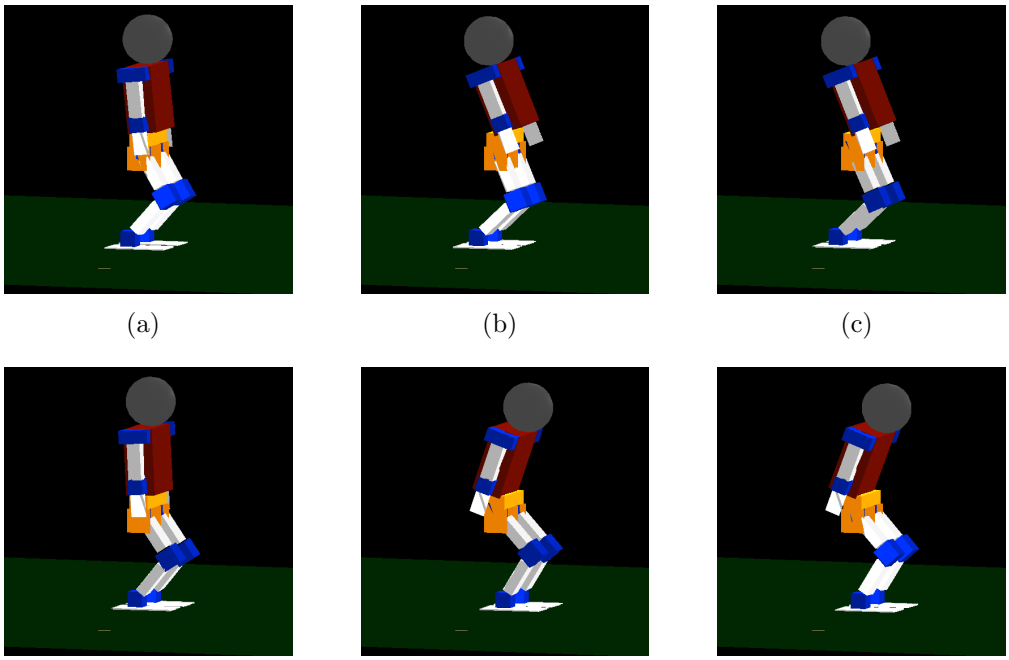


Figure 7.20: Resulting AMF motion after applying the CMA to the circular sway gait that was stabilized by first specifying the step time and then using spatial scaling in the CATF. Reference motion is dashed blue and the AMF motion is solid purple.



(a) (b) (c)

(d) (e) (f)

Figure 7.21: Simulation stills of the AMF solution using both space and time scaling applied to the circular sway gait

7.2 Walking

Walking is one of the most researched and difficult motions to successfully implement in humanoid robotics. Three gaits generated using mathematical equations and one gait generated from motion capture are used here as examples. Each gait has some form of instability so that the AMF has something to stabilize and filter. Only the single support phase of the example walking motions are filtered since they are generally the unstable portion of the motions while the DSP is generally stable. Double support walking phases offer much greater stability with the increased support polygon.

7.2.1 Piecewise gait by Qiang Huang

The piecewise gait is a very common type of walking gait used that is based on trajectory following. The gait is divided into (sometimes arbitrary) portions so that the gait is defined with different equations for each portion. Typically the hip/pelvis and ankle trajectories are the only defined trajectories from the equations. Inverse kinematics are used to derive all of the joint angles. The piecewise gait developed by Qiang was designed to be used in a numeric optimization scheme.

Equations governing reference motion

The following is a summary of the equations governing the gait described in [18]. The described gait is only a 2D planar gait–not considering the Y-direction of the robot. The Y-direction trajectories were generated using polynomial fits for this example. Qiang defines the ankle position as

$$X_a = [x_a(t), z_a(t), \theta_a(t)]^T \tag{7.2}$$

where $(x_a(t), z_a(t))$ is the ankle coordinate position and $\theta_a(t)$ is the angle of the foot with respect to the ground (x-axis). The hip trajectory is defined similarly as

$$X_h = [x_h(t), y_h(t), z_h(t)]^T \tag{7.3}$$

The time period necessary for one walking step is defined as T_c and the time of each kth step is from kT_c to $(k+1)T_c$, $k = 1, 2, 3, 4..., K$; where K is the number of steps. The kth walking step begins when the heel of the right foot leaves the ground at $t = kT_c$ and ends when the right heel hits the ground at $t = (k+1)T_c$. The left foot trajectory is the same as the right except for a T_c delay. T_d is the time of the double support phase. q_b and q_f are the angles at which the right foot leaves and lands on the ground respectively. Assuming that

the entire sole of the foot is in contact with the ground at $t = kT_c$ and $t = (k+1)T_c + T_d$, the following constraints are made

$$\theta_a(t) = \begin{array}{cc} q_{\mathrm{gs}}(k), & t = \mathrm{kT}_c \\ q_b, & t = \mathrm{kT}_c + T_d \\ -q_f, & t = (k+1)T_c \\ -q_{\mathrm{ge}}(k), & t = (k+1)T_c + T_d \end{array} \tag{7.4}$$

where $q_{\mathrm{gs}}(k)$ and $q_{\mathrm{ge}}(k)$ are the angles of the ground surface under the support foot–they are 0 on level ground.

Let $(L_{\mathrm{ao}}, H_{\mathrm{ao}})$ be the (X,Y) position of the ankle at the highest point during the swing of the foot. D_s is the length of one step and $kT_c + T_m$ is the time when the right foot is at its highest point. l_{an}, l_{af}, and l_{ab} are the length from the ankle to the ground, ankle to toe (front), and ankle to heel (back) respectively. $h_{\mathrm{gs}}(k)$ and $h_{\mathrm{ge}}(k)$ are the heights of the ground surface under the support foot and are usually 0. The swing ankle trajectory is then defined as

$$x_a(t) := \begin{cases} kD_s & kT_c \\ kD_s + (1 - \mathrm{Cos}\,(q_b))\, l_{\mathrm{af}} + \mathrm{Sin}\,(q_b)\, l_{\mathrm{an}} & kT_c + T_d \\ kD_s + L_{\mathrm{ao}} & kT_c + T_m \\ (2+k)D_s - (1 - \mathrm{Cos}\,(q_f))\, l_{\mathrm{ab}} - \mathrm{Sin}\,(q_f)\, l_{\mathrm{an}} & (1+k)T_c \\ (2+k)D_s & (1+k)T_c + T_d \end{cases} \tag{7.5}$$

$$z_a(t) := \begin{cases} h_{\mathrm{gs}}(k) + l_{\mathrm{an}} & kT_c \\ h_{\mathrm{gs}}(k) + \mathrm{Sin}\,(q_b)\, l_{\mathrm{af}} + \mathrm{Cos}\,(q_b)\, l_{\mathrm{an}} & kT_c + T_d \\ H_{\mathrm{ao}} & kT_c + T_m \\ h_{\mathrm{ge}}(k) + \mathrm{Sin}\,(q_f)\, l_{\mathrm{ab}} + \mathrm{Cos}\,(q_f)\, l_{\mathrm{an}} & (1+k)T_c \\ h_{\mathrm{ge}}(k) + l_{\mathrm{an}} & (1+k)T_c + T_d \end{cases} \tag{7.6}$$

Since the entire sole surface of the right foot is in contact with the ground at $t = \mathrm{kT}_c$ and $t = (k+1)T_c + T_d$, the following constraint must be satisfied:

$$X'_a\,[\mathrm{kT}_c] = X'_a\,[(k+1)T_c + T_d] = 0 \tag{7.7}$$

The hip trajectory in the X-direction is defined as

$$x_h(t) := \begin{cases} \mathrm{kD}_s + x_{\mathrm{ed}} & t = \mathrm{kT}_C \\ (k+1)D_s - x_{\mathrm{sd}} & t = \mathrm{kT}_c + T_d \\ (k+1)D_s + x_{\mathrm{ed}} & t = (k+1)T_C \end{cases} \tag{7.8}$$

Table 7.1: Input gait parameters used for Qiang's piecewise gait

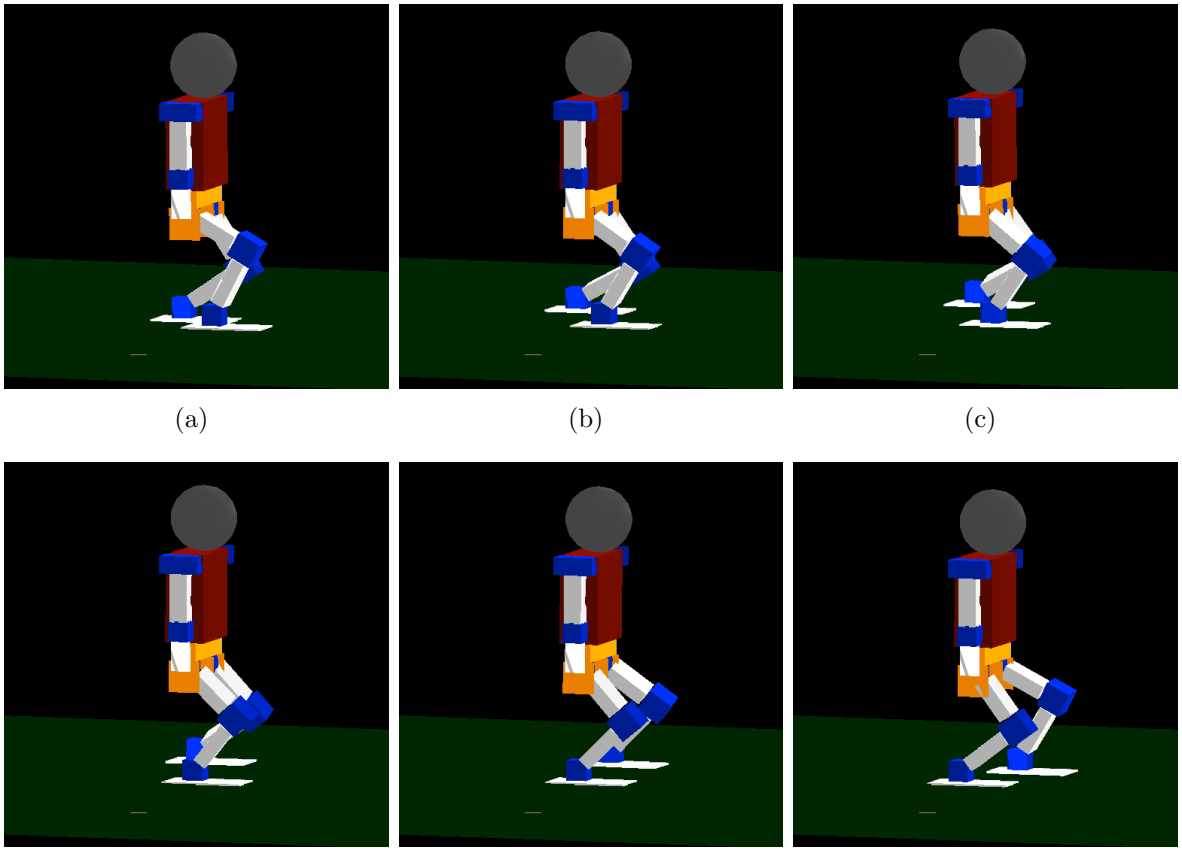
Description	Variable	Value (metric)
Step Time	T_c	1 (s)
Double Support Time	T_D	0 (s)
Time at step height	T_m	0.5 (s)
Step Length	D_s	0.10 (m)
Angle of ankle	q_*	0 (rad)
Midway step length	L_{ao}	0 (m)
Step height	H_{ao}	0.025 (m)
Minimum hip height	H_{hmin}	0.20 (m)
Maximum hip height	H_{hmin}	0.21 (m)

where x_{sd} and x_{ed} are the distances along the X-axis from the hip to the ankle of the support foot at the start and end of the single support phase, respectively [18]. A third-order periodic spline interpolation is calculated to satisfy the above constraints and continuous velocity constraints. Similarly, constraints for the Z-direction can be defined as

$$z_h(t) := \begin{cases} H_{\text{hmin}} & t = \text{kT}_C + 0.5T_d \\ H_{\text{hmax}} & t = \text{kT}_c + 0.5\,(T_C - T_d) \\ H_{\text{hmin}} & t = (k+1)T_C + 0.5T_d \end{cases} \tag{7.9}$$

where H_{hmin} and H_{hmax} are the minimum and maximum hip heights respectively. These constraints along with continuous velocity constraints are used with a third order spline interpolation to define the trajectory. Table 7.1 shows the values used for generating the gait for this example case. Figure 7.22 shows a sequence of still shots of the gait in simulation.

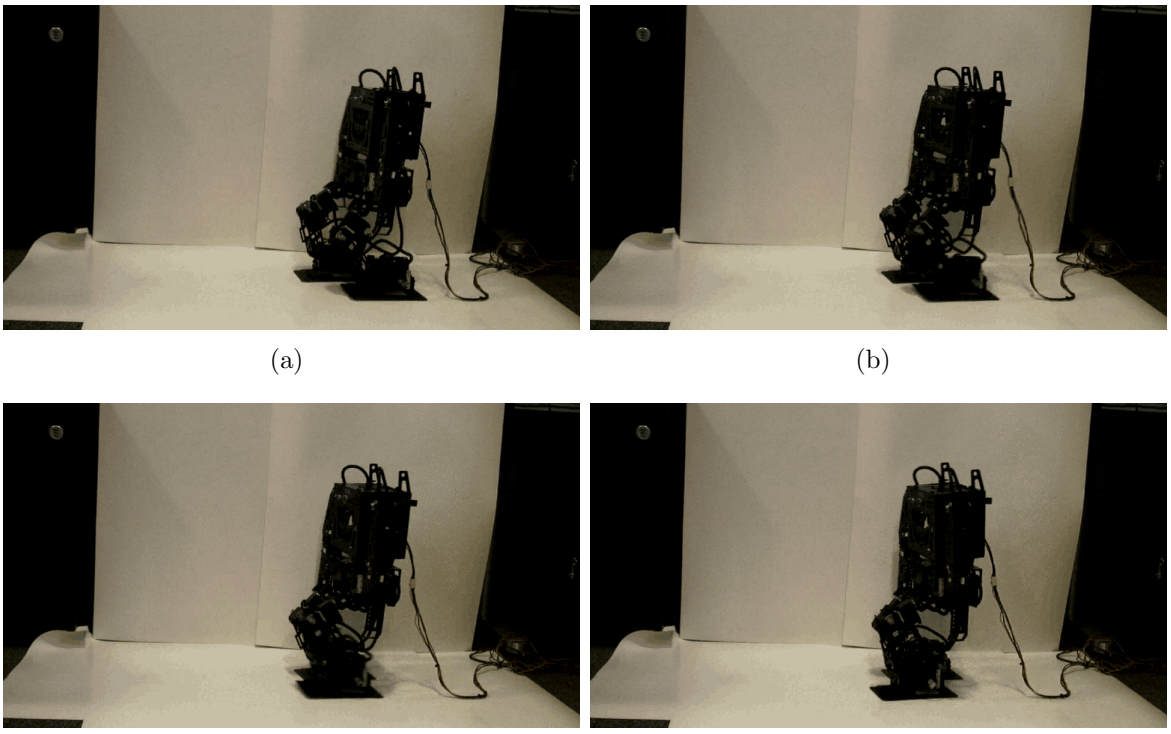
Figure 7.23 shows a sequence of pictures of the Qiang's piecewise gait. Even though the gait was modified to be unstable, the robot still does not fall over while walking. Figure 7.24 shows two pictures of Qiang's piecewise gait from the rear view. The gait is designed such that the upper body and hips are perfectly level throughout the entire motion. As seen, the robot leans significantly to each side as it steps. This leaning occurs because the motors used in the test platform, DARwIn, use a proportional position controller. Therefore, high loads on a motor cause large errors. During SSP, very large torques are need from the motors to keep the robot from leaning. As seen, the proportional controller is not enough and the robot is able to lean, which actually keeps it from falling over. Though for this gait, the robot's leaning kept it stable and from falling over, in all of the other walking gaits, this error proves to destabilize even a stable motion. The error is not as prevalent in the circular sway gait since the torques required from the motors are not as high as when in SSP during walking.



(a) (b) (c)

(d) (e) (f)

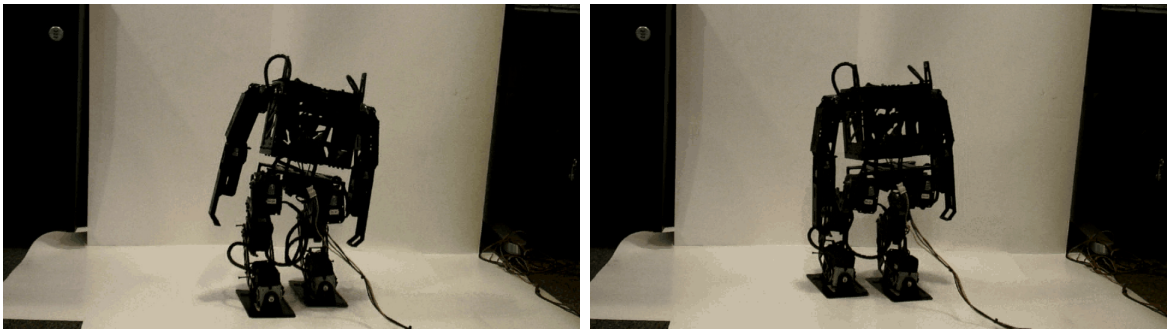
Figure 7.22: Video stills of Qiang's piecewise reference gait in simulation



(a) (b)

(c) (d)

Figure 7.23: Video stills of the Qiang's piecewise reference gait



(a) (b)

Figure 7.24: Video stills of the Qiang's piecewise reference gait that show the robot leaning

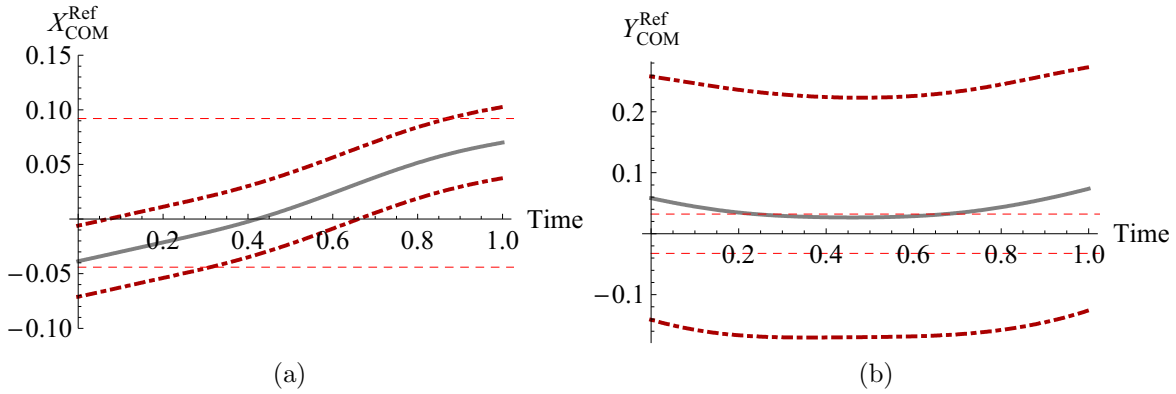


Figure 7.25: COM reference trajectories for Qiang's piecewise gait. Reference trajectory shown as solid gray and the kinematic limits of the COM are shown as dot dashed dark red.

Resulting reference COM and ZMP trajectory

The results for the COM reference trajectory can be seen in Fig. 7.25. Also seen are the COM limits enforced by kinematic constraints. The range of the COM in the X direction is dictated by how far forward and back the waist of the robot can move since it will be used as the primary means of modifying the COM position in the CMA when generating the filtered joint angles. The range of the COM in the Y direction, which is dictated by kinematic constraints, is very large because the height of the robot is relatively low, allowing for a large range of travel of the hip.

The results for the ZMP reference trajectory can be seen in Fig. 7.26. As seen, only the Y direction of the motion has unstable portions. Figure 7.27 shows a parametric plot of both the reference COM and ZMP. [3] It is clear that the Y direction of the motion is largely unstable at the beginning and end of the motion.

Simple time scaling

In order to determine if time scaling is feasible, the CATF analyzes the motion to determine what cases of instability occur as described in Sec. 4.3. Figure 7.28 shows that the Y direction has no Case 3 instabilities. Therefore it may be possible to stabilize the gait simply by slowing it down or speeding it up. However, notice that the ZMP in the X direction is very close the the edge of the support polygon at the beginning of the motion. This area could potentially destabilize if the motion's speed is changed.

[3] The ZMP points that do not line up smoothly with the rest of the motion (also seen in Fig. 7.26(b)), are a result of inaccurate estimations of the acceleration of the COM at the beginning and ends of the motion. Since motion data is not available beyond those points, the accelerations calculations are a little off.

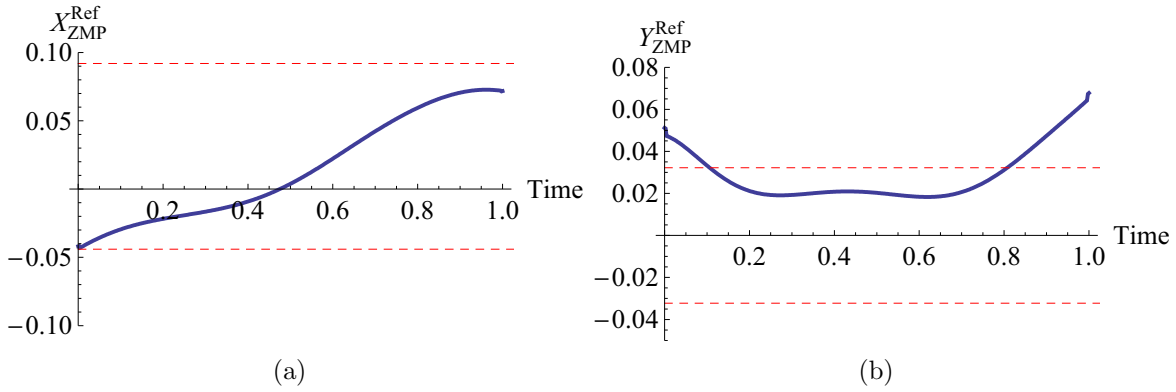


Figure 7.26: ZMP reference trajectories for Qiang's piecewise gait.

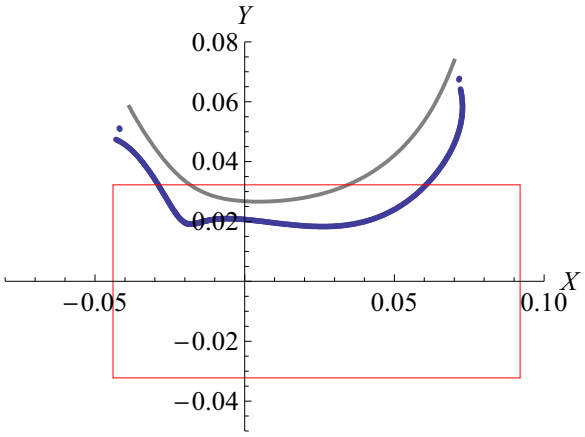


Figure 7.27: COM (Gray) and ZMP (Blue) reference trajectories for Qiang's piecewise gait. The red box represents the support polygon.

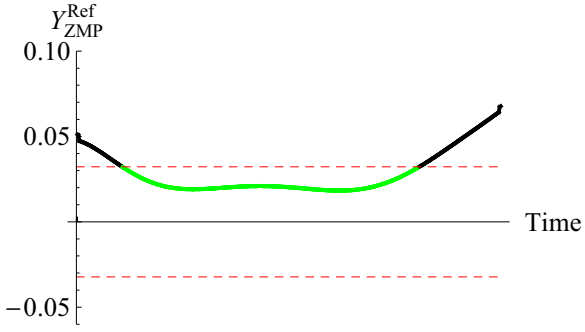


Figure 7.28: Categorized instability for Qiang's piecewise gait. Black is Case 2, and Green is Stable.

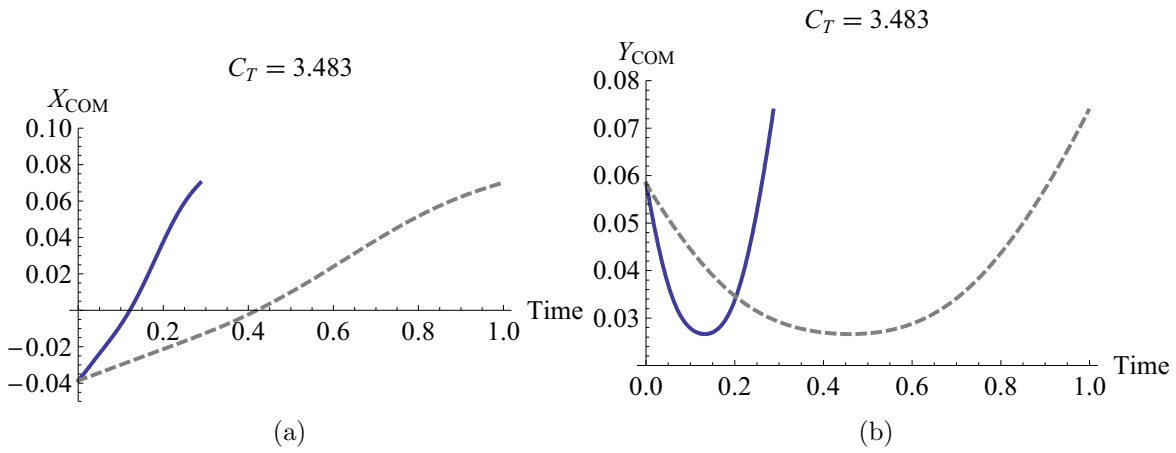


Figure 7.29: AMF COM trajectories from using time scaling on Qiang's piecewise gait. Reference COM is dashed gray and AMF COM is solid blue.

Figure 7.29 shows the result of using Eq. 4.13. As seen, the motion is sped up; a duration of 0.29 seconds compared to 1.0 seconds for the reference motion. Since it is possible that in speeding up the motion and stabilizing the most unstable portion of the reference motion in the Y direction, other parts of the motion were destabilized, it is important to check the resulting ZMP. Figure 7.30 shows the resulting ZMP in both directions. As seen, though speeding up the entire motion may have stabilized the very beginning and end of the motion in the Y direction, many other parts of the motions in the Y direction were destabilized. Not only is the Y direction unstable, but the X direction as well. Therefore, simple time scaling cannot be used to stabilize this motion.

Two and three pose constraints

Since the reference motion filtered here using Qiang's piecewise gait is only the single support phase of the robot's walking, it is useful to apply a constraint dictating that the beginning and end of the motion must remain the same. Applying this constraint can guarantee that the robot's foot placement when making contact with the ground will be the same in the filtered motion as in the reference motion. This can be guaranteed because the constraint also constrains the CMA when it generates joint angles for the filtered motion; the beginning and end poses will be exactly the same as the reference motion. Therefore, the solution in Sec. 5.1.2 is applied with constraints at the beginning and end of this reference motion. Since the reference motion is stable in the X direction, only the Y direction needs to undergo spatial scaling using the pose constraints. Figure 7.31 shows the result after applying the spatial scaling. As seen, the motion is stabilized in the Y direction except for the very end, which, as mentioned earlier, is due to inaccurate estimation of acceleration at the beginning

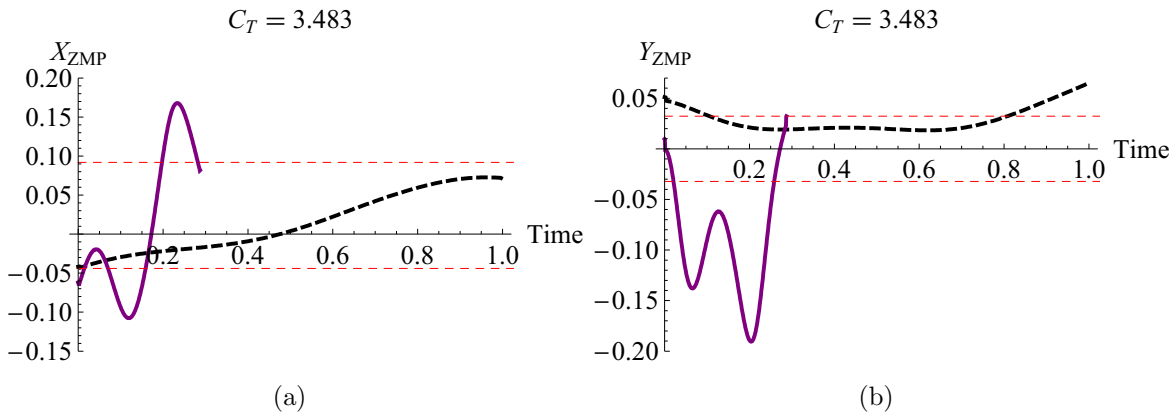


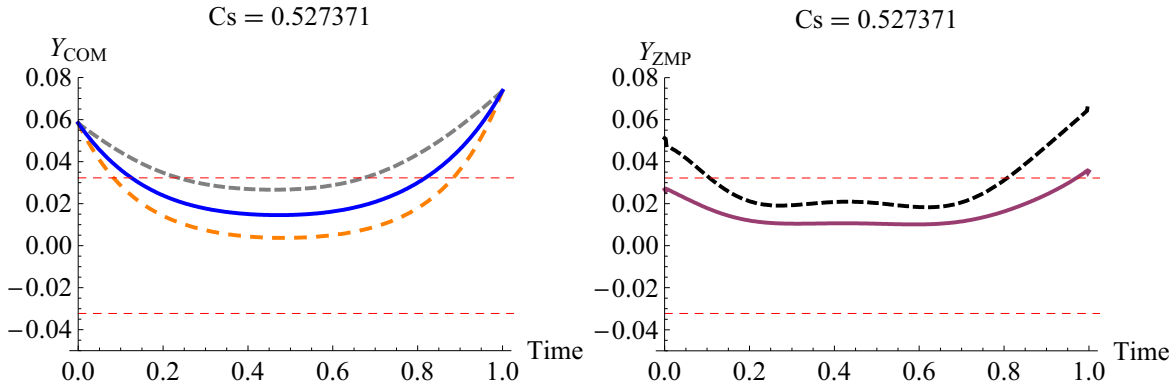
Figure 7.30: AMF ZMP trajectories from using time scaling on Qiang's piecewise gait. Reference ZMP is dashed black and AMF ZMP is solid purple.

and end of the motion. Figure 7.32 shows plots of the resulting AMF angles from the CMA. As seen, the waist does not need to move in the X direction to follow the AMF stable COM trajectory. The hips only needs to modifying its trajectory in the Y direction. Figure 7.33 shows video still shots of the AMF motion in simulation. It is difficult to seen in simulation, but the trajectory of the hips in the Y direction is slightly different than in the reference motion. The fact that it is difficult to notice speak well for the AMF goal to minimize difference between the reference and filtered motion.

Additionally, it may be useful to apply a third pose constraint to Qiang's piecewise gait. Since it is not always possible to satisfy three pose constraints (Sec. 5.1.3) it is useful to visualize where a third constraint point could lie while allowing a stable motion to be generated; Fig. 7.34 shows the corresponding envelop. As seen, it does not appear that a velocity constraint can be enforced at the beginning or end of the motion since the envelop does not include points immediately after or before the respective beginning and end points. However, if needed, other points along the reference motion could be used as additional pose constraints.

7.2.2 Cycloid gait by Jung-Yup Kim

Jung-Yup Kim's cycloid gait is the basis for the walking motion for the HUBO humanoid robot at KAIST University in Korea [4]. The cycloid gait was developed mostly as a parametric gait with the parameters tweaked and chosen from experimental values. The AMF allows the same gait generation methods to be applied to different robot designs without additional experiments since the AMF will stabilize the gait.



(a) $y^{\text{Ref}}_{\text{COM}}$ is dashed gray, $y^{\text{Fit}}_{\text{COM}}$ is long dashed orange, and $y^{\text{AMF}}_{\text{COM}}$ is blue.

(b) $y^{\text{Ref}}_{\text{ZMP}}$ is dashed black and $y^{\text{AMF}}_{\text{ZMP}}$ is purple

Figure 7.31: The CATF solution to a two point constraint on Qiang's piecewise gait in the Y-direction.

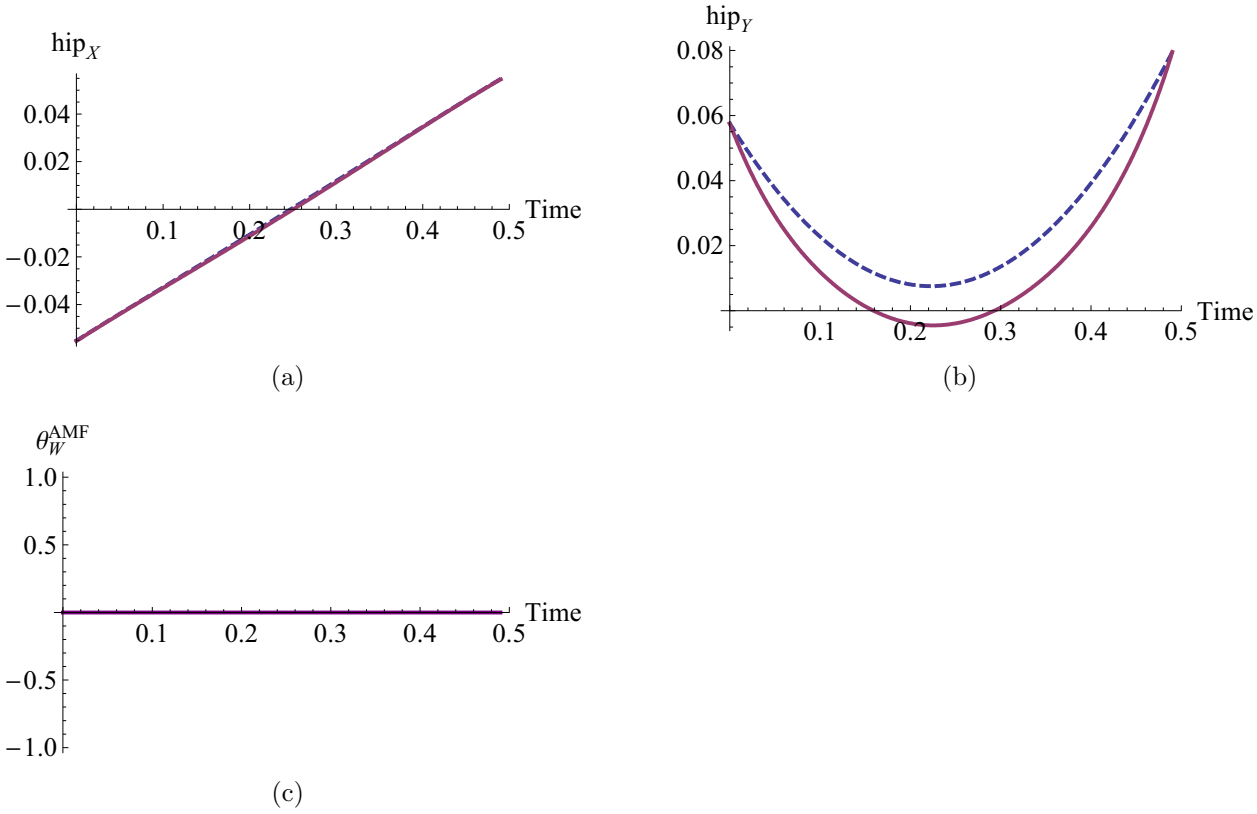
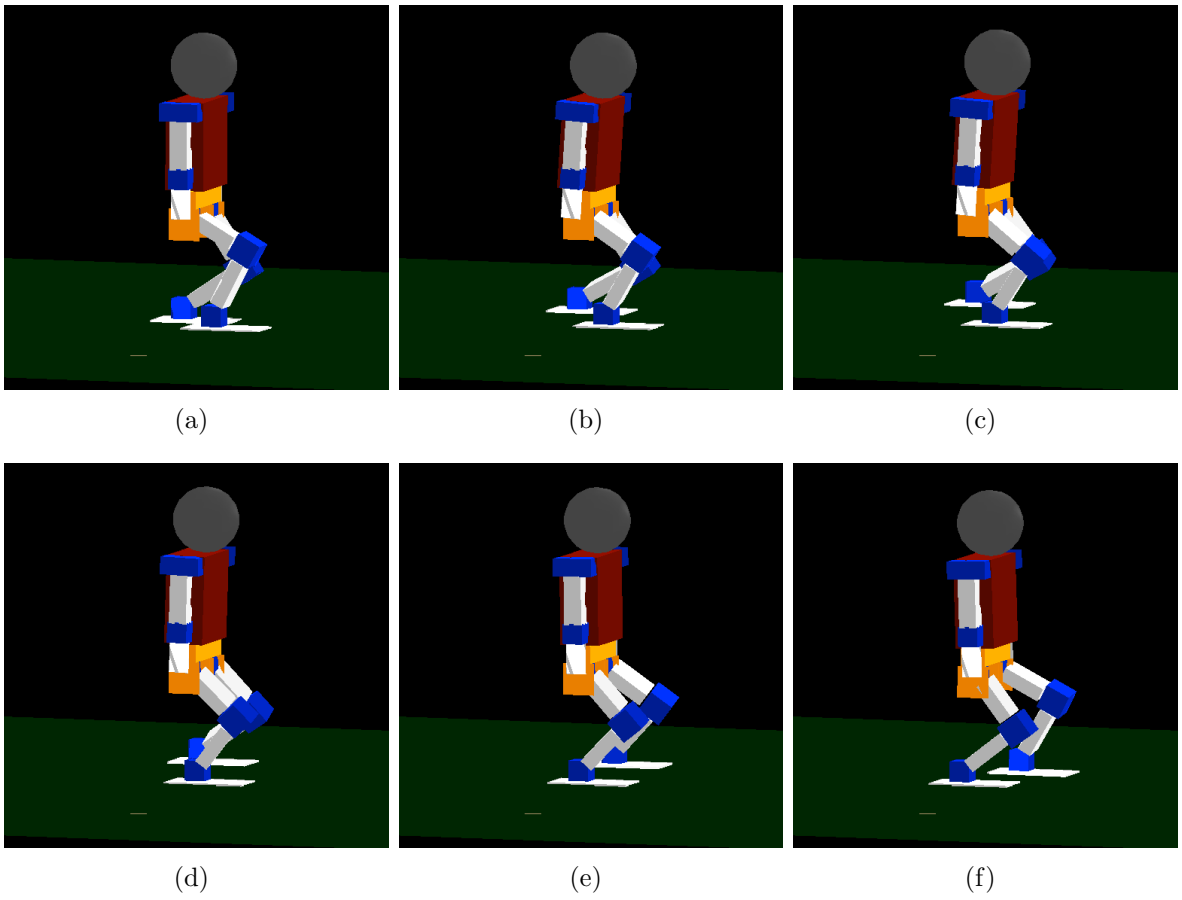


Figure 7.32: Resulting AMF motion after applying the CMA to Qiang's piecewise gait that was stabilized using two point constraints and spatial scaling by the CATF. Reference motion is dashed blue and the AMF motion is solid purple.



(a) (b) (c)

(d) (e) (f)

Figure 7.33: Simulation stills of the AMF solution with two point constraints applied to Qiang's piecewise gait.

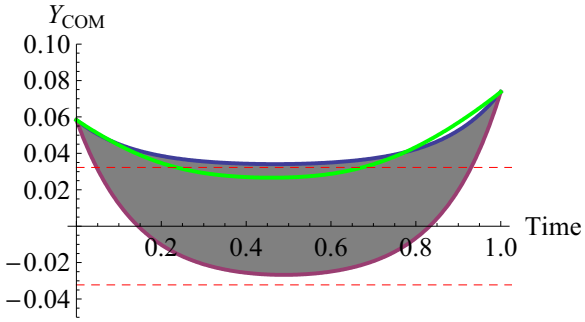


Figure 7.34: For Qiang's piecewise gait in the Y-direction, the envelop that defines where a third constraint could lie if constraint points are set for the beginning and end of the motion

Equations governing reference motion

Kim describes the gait generation with the following equations in [4].

A pendulum model is used to determine the lateral (y-direction) motion of the hip of the robot. The natural frequency is described as

$$f = \frac{1}{2\pi}\sqrt{\frac{g}{l}} \tag{7.10}$$

where l is the height of the center of mass from the ground. The pendulum model not only dictates the lateral motion of the hips of the robot, but also the vertical motion as it also follows the inverted pendulum model. The amplitude of the pendulum motion was determined experimentally in [4]. In this example, the amplitude was found using the natural frequency of the robot based on its COM height and solving for the amplitude that would create the same frequency as an inverted pendulum. The hip motion of the robot in the X direction was created as a combination of a linear and cosine function. The constants of the function have been modified for the robot platform used in this work as

$$\text{Hip}_x(t) := (1/2)\left(\frac{6}{10}\left(L_s * (1/2) * \left(1 - \text{Cos}\left(\pi\frac{t}{T_s}\right)\right)\right) + \frac{4}{10}\left(\frac{L_s}{T_s}t\right) - L_s/2\right) \tag{7.11}$$

where L_s is the step length and T_s is the step time. L_s is 0.10 m and T_s is 0.57 s for this example. The trajectory for the X-direction of the foot is described using a cycloid function. Again, the constants of the function were modified for the robot platform used in this work as

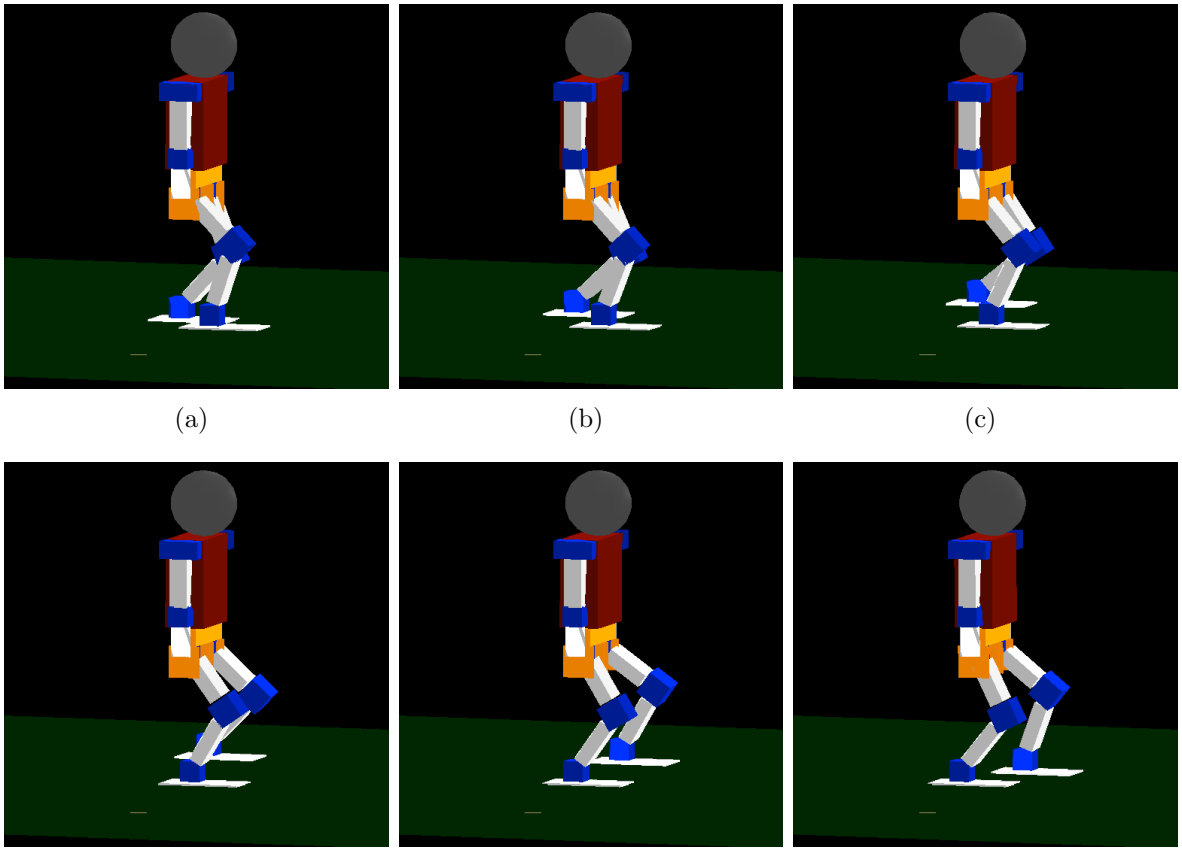
$$\text{Ankle}_x(t) := L_s\left(\frac{t}{T_s} - \frac{1}{2\pi}\text{Sin}\left(\pi\frac{2t}{T_s}\right)\right) - (L_s/2) \tag{7.12}$$

Figure 7.35 shows a sequence of video stills of the gait in simulation. Figure 7.36 shows a sequence of video still of the Kim's cycloid gait. Since the gait is unstable, additional support is added to the upper body to prevent the robot from falling over.

Resulting reference COM and ZMP trajectory

The results for the COM reference trajectory can be seen in Fig. 7.37. Also seen are the COM limits enforced by kinematic constraints.

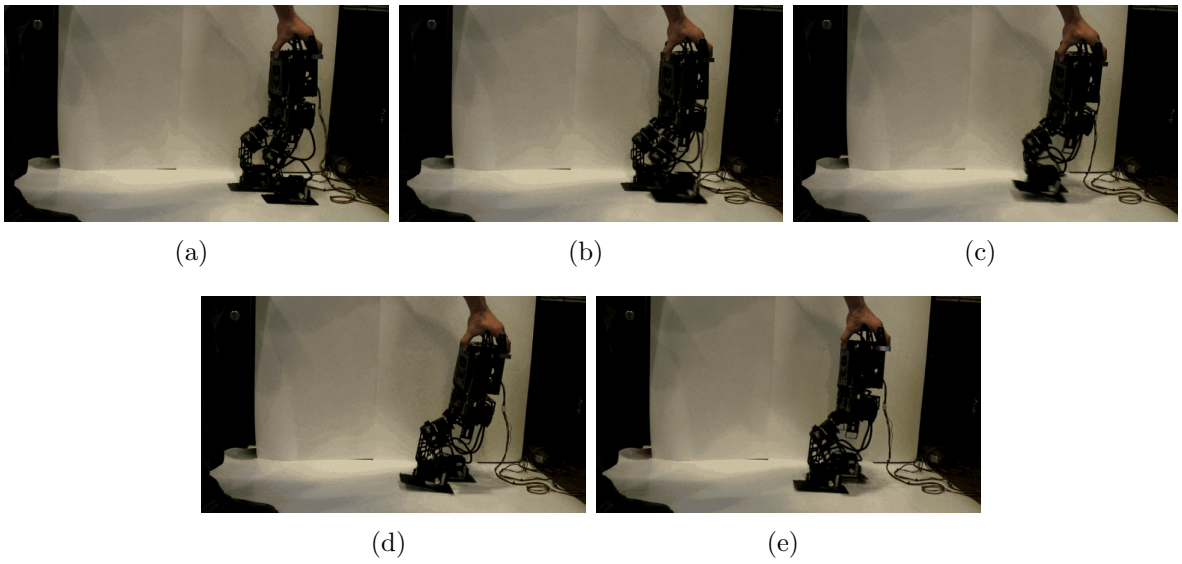
The results for the ZMP reference trajectory can be seen in Fig. 7.38. As seen only the X direction of the motion has unstable portions. Figure 7.39 shows a parametric plot of both



(a) (b) (c)

(d) (e) (f)

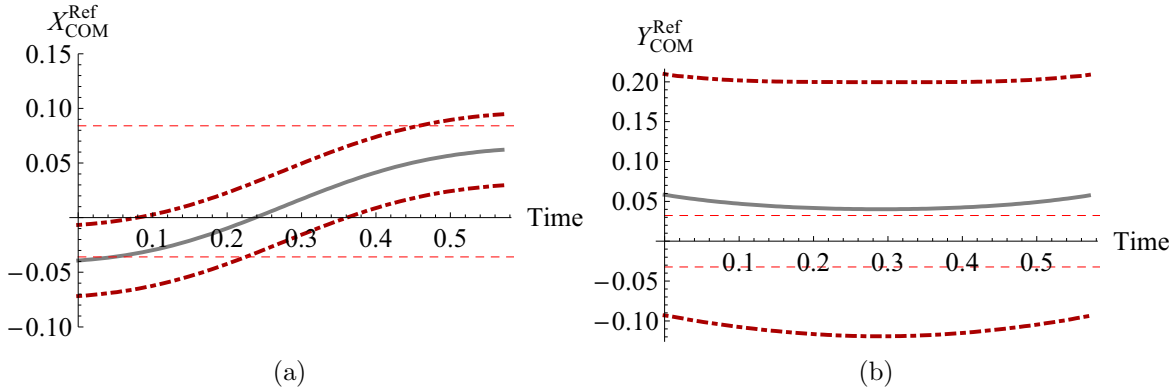
Figure 7.35: Simulation stills of Kim's cycloid reference gait



(a) (b) (c)

(d) (e)

Figure 7.36: Video sills of the Kim's cycloid reference gait



(a) (b)

Figure 7.37: COM reference trajectories for Kim's cycloid gait. Reference trajectory shown as solid gray and the kinematic limits of the COM are shown as dot dashed dark red.

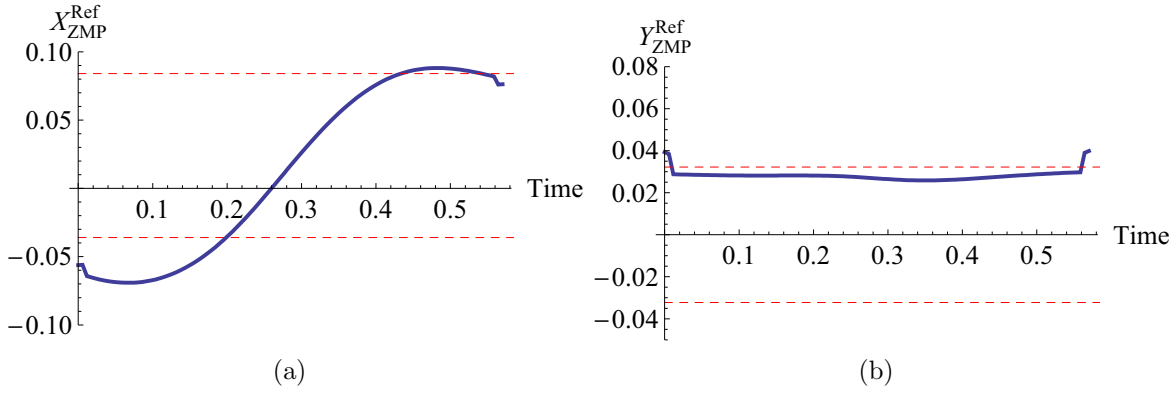


Figure 7.38: ZMP reference trajectories for Kim's cycloid gait.

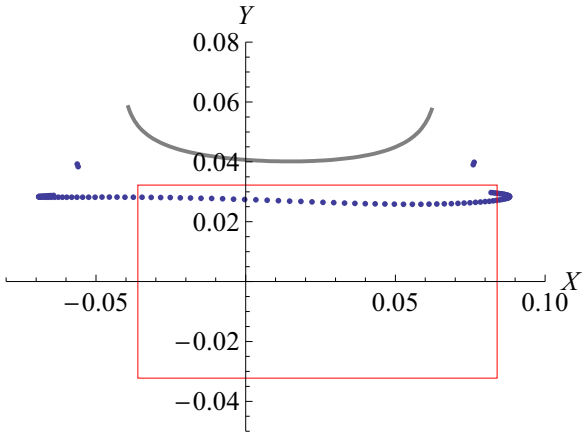


Figure 7.39: COM (Gray) and ZMP (Blue) reference trajectories for Kim's cycloid gait. The red box represents the support polygon.

the reference COM and ZMP. [4] It is clear that the X direction of the motion is unstable with the ZMP leaving the support polygon. However, in the Y direction, the motion appears to be only marginally stable–the ZMP hovering very close to the edge of the support polygon.

Simple time scaling

In order to determine if time scaling is feasible, the CATF analyzes the motion to determine what cases of instability occur as described in Sec. 4.3. Figure 7.40 shows that the X

[4]The ZMP points that do not line up smoothly with the rest of the motion (also seen in Fig. 7.38), are a result of inaccurate estimations of the acceleration of the COM at the beginning and ends of the motion. Since motion data is not available beyond those points, the accelerations calculations are a little off.

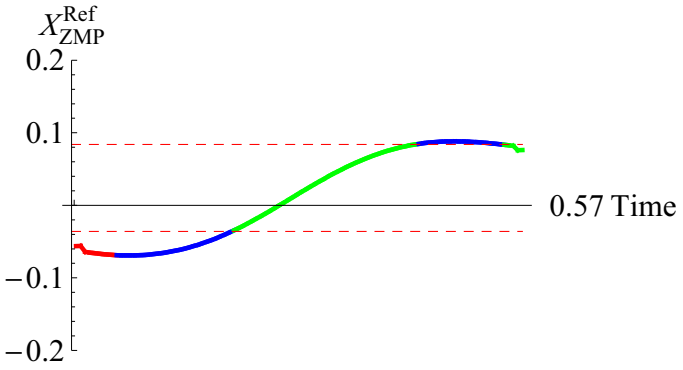


Figure 7.40: Categorized instability for Kim's cycloid gait in the X direction. Blue is Case 1, red is Case 3, and Green is Stable.

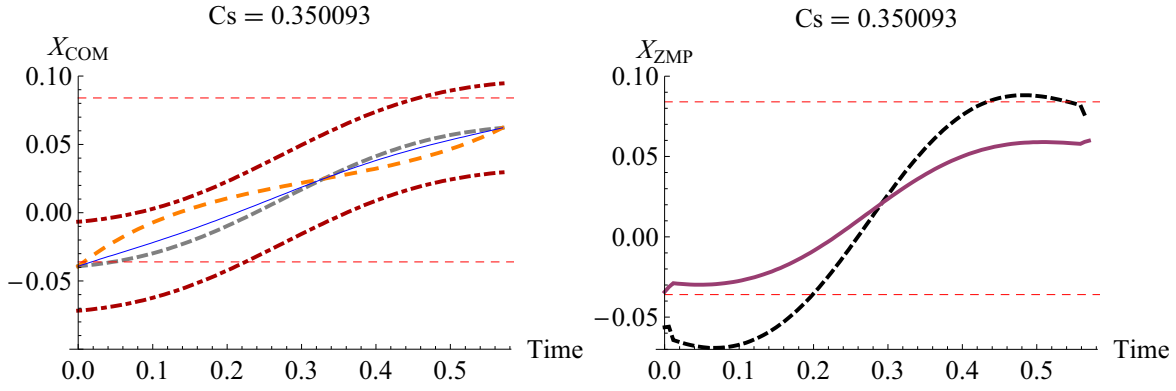
direction has Case 3 instabilities. Therefore it impossible to stabilize the gait simply by slowing it down or speeding it up.

Two and three pose constraint

As with the previous walking gait, it is useful to apply a constraint dictating that the beginning and end of the motion must remain the same. Applying this constraint can guarantee that the robot's foot placement when making contact with the ground will be the same in the filtered motion as in the reference motion. This can be guaranteed because the constraint also constrains the CMA when it generates joint angles for the filtered motion; the beginning and end poses will be exactly the same as the reference motion. Therefore, the solution in Sec. 5.1.2 is applied with constraints at the beginning and end of this reference motion. The X direction of the motion is unstable and requires spatial scaling. Since the Y direction of the motion is only marginally stable, we will apply some spatial scaling to that direction to increase stability. Figure 7.41 shows the result after applying the spatial scaling. As seen, the motion is stabilized in the X direction. Stability is slightly increased for most of the motion in the Y direction as seen in Fig 7.42. Figure 7.43 shows the resulting CMA angles. As seen, the pitch of the waist is capable of changing the location of the COM in the X direction without the hips moving in the X direction. The hip trajectory in the Y direction changes only slightly. Figure 7.44 shows video stills of the filtered motion in simulation. As seen, the upper body leans slightly forward and then back during the step in order to change the COM path so that the robot's motion is stable.

.

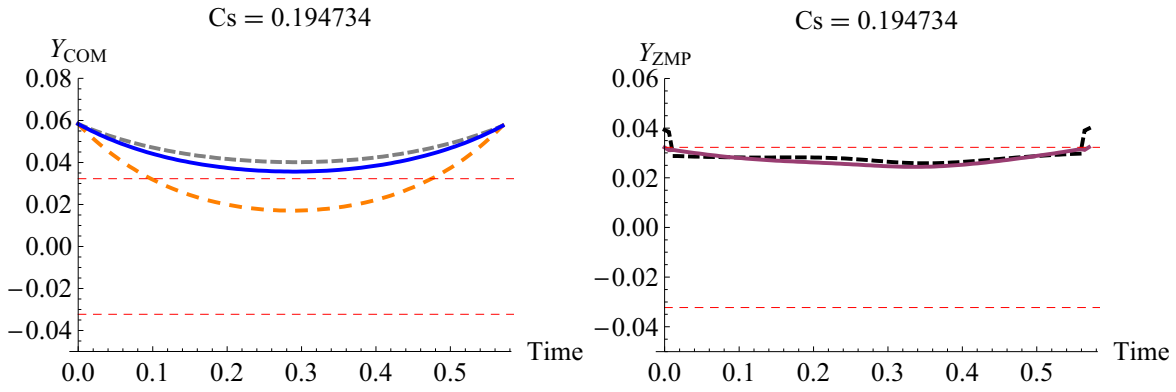
Additionally, it may be useful to apply a third pose constraint. Since it is not always possible to satisfy three pose constraints (Sec. 5.1.3) it is useful to visualize where a third constraint point could lie while allowing a stable motion to be generated; Fig. 7.45 shows the



(a) $x^{\text{Ref}}_{\text{COM}}$ is dashed gray, $x^{\text{Fit}}_{\text{COM}}$ is long dashed orange, and $x^{\text{AMF}}_{\text{COM}}$ is blue.

(b) $x^{\text{Ref}}_{\text{ZMP}}$ is dashed black and $x^{\text{AMF}}_{\text{ZMP}}$ is purple

Figure 7.41: The CATF solution to a two point constraint on Kim's cycloid gait in the X direction.



(a) $y^{\text{Ref}}_{\text{COM}}$ is dashed gray, $y^{\text{Fit}}_{\text{COM}}$ is long dashed orange, and $y^{\text{AMF}}_{\text{COM}}$ is blue.

(b) $y^{\text{Ref}}_{\text{ZMP}}$ is dashed black and $y^{\text{AMF}}_{\text{ZMP}}$ is purple

Figure 7.42: The CATF solution to a two point constraint on Kim's cycloid gait in the Y direction.

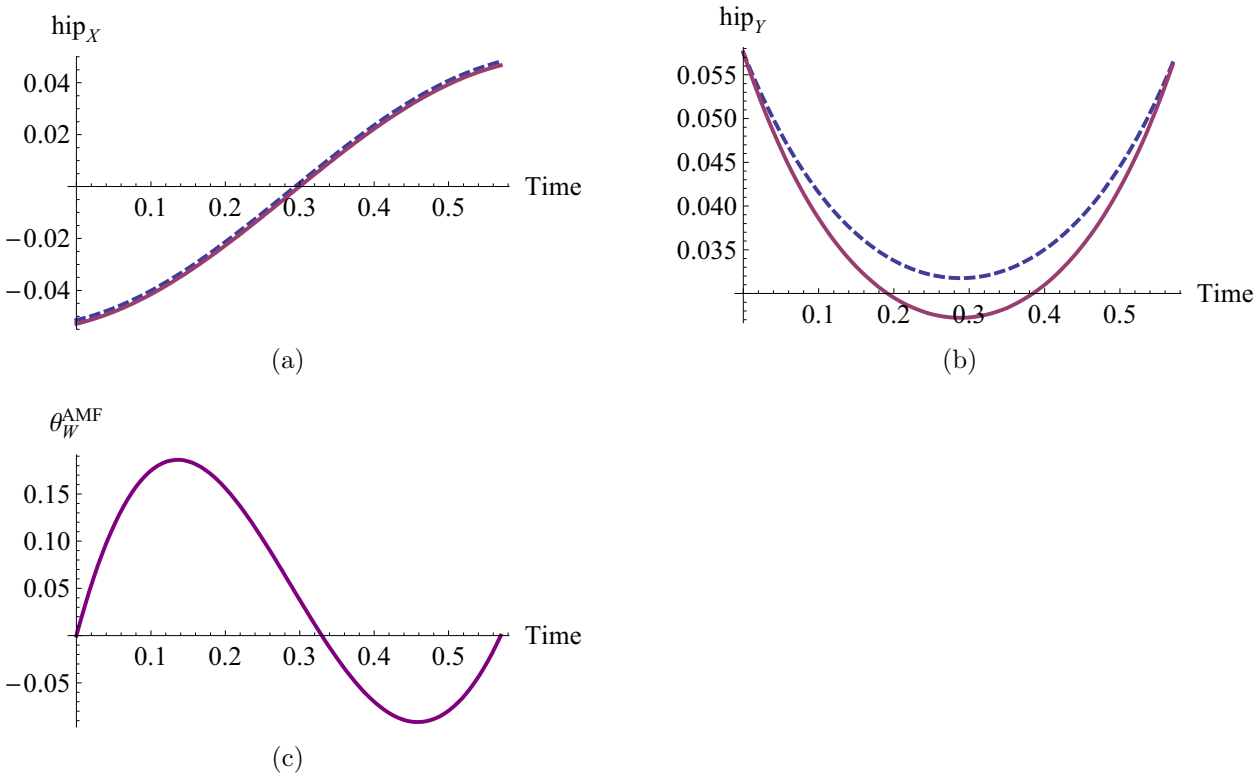
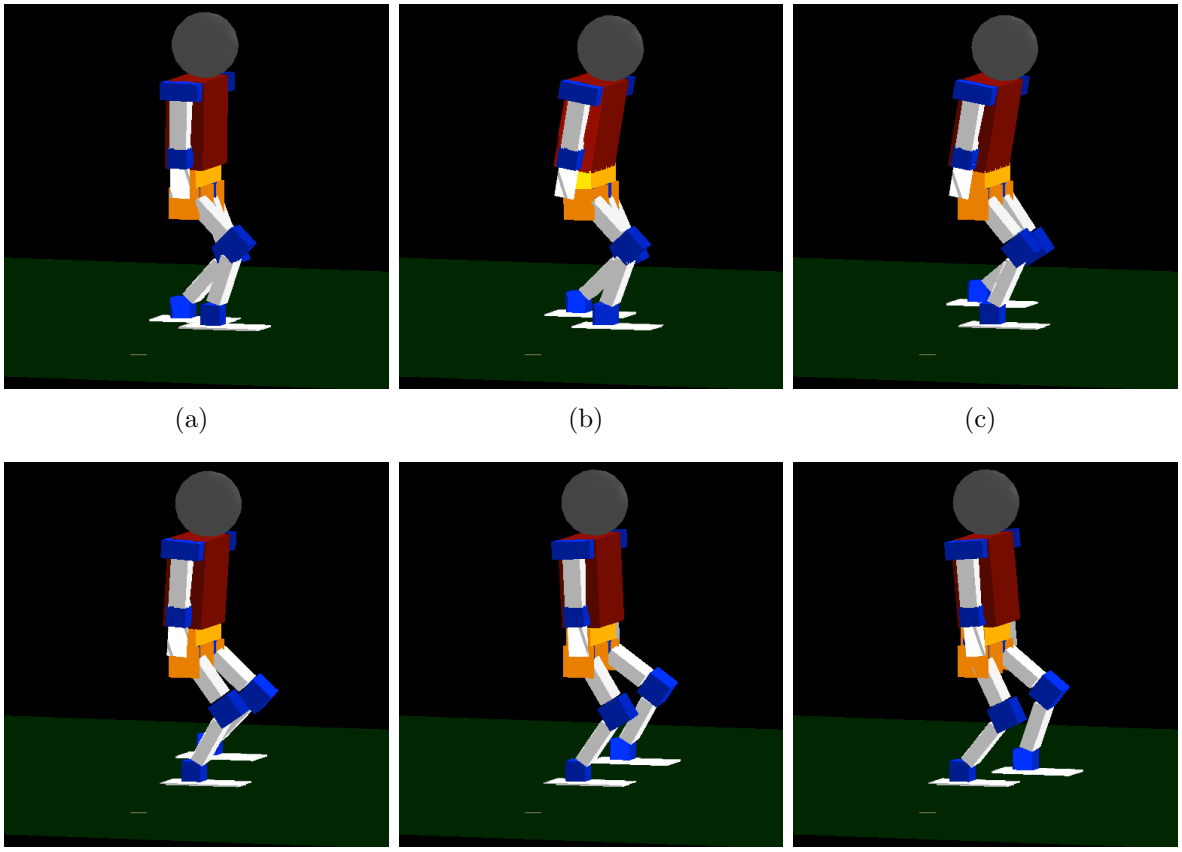


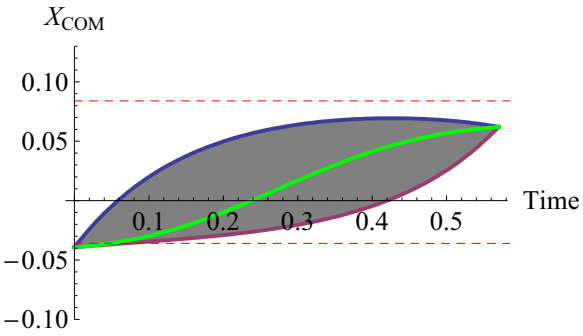
Figure 7.43: Resulting AMF motion after applying the CMA to Kim's cycloid gait that was stabilized using two point constraints and spatial scaling by the CATF. Reference motion is dashed blue and the AMF motion is solid purple.



(a) (b) (c)

(d) (e) (f)

Figure 7.44: Simulation stills of the AMF solution with two point constraints applied to Kim's cycloid gait.



(a)

Figure 7.45: For Kim's cycloid gait, the envelops that defines where a third constraint could lie in the X direction

corresponding envelop. It appears that it may be possible to enforce a velocity constraint at the beginning or end of the motion since the envelop includes all of the reference motion points in the X direction.

7.2.3 Hyperbolic gait by Lim

The hyperbolic gait developed by Lim [38] is designed to follow a very precise ZMP trajectory. The gait has stability in its design. In order to ensure that the gait is unstable (so that the AMF has something to stabilize and filter), values of the parameters for the gait were changed to ensure that some portion of the gait is unstable.

Equations governing reference motion

Lim's hyperbolic gait is based on hyperbolic cosines and sines, which, consequently, take the same form as Eq. 4.5. So it follows that using hyperbolic functions could potentially design a walking gait around a stable ZMP path/trajectory.

The following is a summary of the equations governing the walking gait developed by Lim in [38]. Let t_1, t_2, and t_3 denote the time intervals in which there is negative, zero, and positive acceleration respectively during the SSP in the X direction. Let t_4 denote the constant-speed time interval during the DSP along both x and y-axes [38]. Additionally, let t_5 be the sum of t_1, t_2, and t_3. x_{ZMP1} and x_{ZMP2} are the respective ZMP positions along the x-axis at the beginning and end of the SSP. x_s is the stride length and x' is the torso position at the end of the SSP. Therefore the body position for $(0 \leq \tau \leq t_1)$ can be defined as

$$x_b(\tau) = (x_0 - x_{\text{ZMP1}})\cosh(\text{a}\tau) + \frac{v_{\text{xD}}}{a}\sinh(\text{a}\tau) + x_{\text{ZMP1}} \tag{7.13}$$

subject to the constraints

$$x_b(t_1) = x_{\text{ZMP1}} \tag{7.14a}$$
$$\dot{x}_b(t_1) = v_{\text{xS}} \tag{7.14b}$$

where v_{xS} is the torso's velocity in the x-direction during the zero acceleration portion of the SSP. Similarly, for $(0 \leq \tau \leq t_3)$,

$$x_b(\tau) = \frac{v_{\text{xS}}}{a}\sinh(\text{a}\tau) + x_{\text{ZMP2}} \tag{7.15a}$$
$$x_b(t_3) = x' \tag{7.15b}$$
$$\dot{x}_b(t_3) = v_{\text{xD}} \tag{7.15c}$$

where v_{xD} is the torso's velocity in the x-direction during the DSP. A similar set of equations governs the y-direction of the torso's motion over the entire SSP for $(0 \leq \tau \leq t_5)$

$$y_b(\tau) = (y_0 - y_{\text{ZMP}})\cosh(\text{a}\tau) + \frac{v_y}{a}\sinh(\text{a}\tau) + y_{\text{ZMP}} \tag{7.16a}$$
$$y_b(t_5) = y_0 \tag{7.16b}$$
$$\dot{y}_b(t_5) = -v_y \tag{7.16c}$$

Therefore by using gait parameters x_{ZMP1}, x_{ZMP2}, y_{ZMP}, v_{xD}, a, x_0, and x_s, the above equations can be solved for their unknowns as found in Sec. C. The transient gait to have the robot start walking from a stopped position is defined for $(0 \leq t \leq t^*)$ as

$$x_b(t) = -x_{\text{ZMP0D}}\cosh(\text{at}) + x_{\text{ZMP0D}} \tag{7.17}$$

where x_{ZMP0D} is a necessary shift in the ZMP to shift the robot's body. As the DSP ends, the swing leg comes forward $x_s/2$ and another constant ZMP is specified at x_{ZMP0S}. During the SSP $(0 \leq \tau \leq \tau^*)$, the motion is defined as

$$\begin{aligned} x_b(\tau) &= (-x_{\text{ZMP0D}}\sinh(\text{at}^*))\sinh(\text{a}\tau) \\ &\quad + (-x_{\text{ZMP0D}}\cosh(\text{at}^*) + x_{\text{ZMP0D}} - x_{\text{ZMP0S}})\cosh(\text{a}\tau) + x_{\text{ZMP0S}} \end{aligned} \tag{7.18a}$$

Table 7.2: Input gait parameters used for hyperbolic gait generation

Description	Parameter	Value	
Frequency	a	4.98	(1/s)
DSP velocity	v_{xD}	0.1	(m/s)
Initial X ZMP during SSP	x_{ZMP1}	0.0135	(m)
Final X ZMP during SSP	x_{ZMP2}	0.03	(m)
Y ZMP during SSP	y_{ZMP}	0.08	(m)
Initial X position	x_0	0	(m)
Step length	x_S	0.15	(m)
Transient DSP duration	t^*	0.2	(s)
Transient SSP duration	τ^*	0.3	(s)

$$x_b(\tau^*) = x_s \tag{7.18b}$$

$$\dot{x}_b(\tau^*) = v_{xD} \tag{7.18c}$$

By using τ^* and t^* as inputs, the unknowns can be solved for as

$$x_{\mathrm{ZMP0D}} = \frac{-\mathrm{a}x_s \sinh(\mathrm{a}\tau^*) - \dot{x}_S(1 - \cosh(\mathrm{a}\tau^*))}{a(-\sinh(\mathrm{at}^*) + \sinh(a(t^* + \tau^*)) - \sinh(\mathrm{a}\tau^*))} \tag{7.19a}$$

$$x_{\mathrm{ZMP0S}} = \frac{-\mathrm{a}x_s(\sinh(\mathrm{a}\tau^*) - \sinh(a(t^* + \tau^*))) + \dot{x}_D(\cosh(\mathrm{a}\tau^*) - \cosh(a(t^* + \tau^*)))}{a(-\sinh(\mathrm{at}^*) + \sinh(a(t^* + \tau^*)) - \sinh(\mathrm{a}\tau^*))} \tag{7.19b}$$

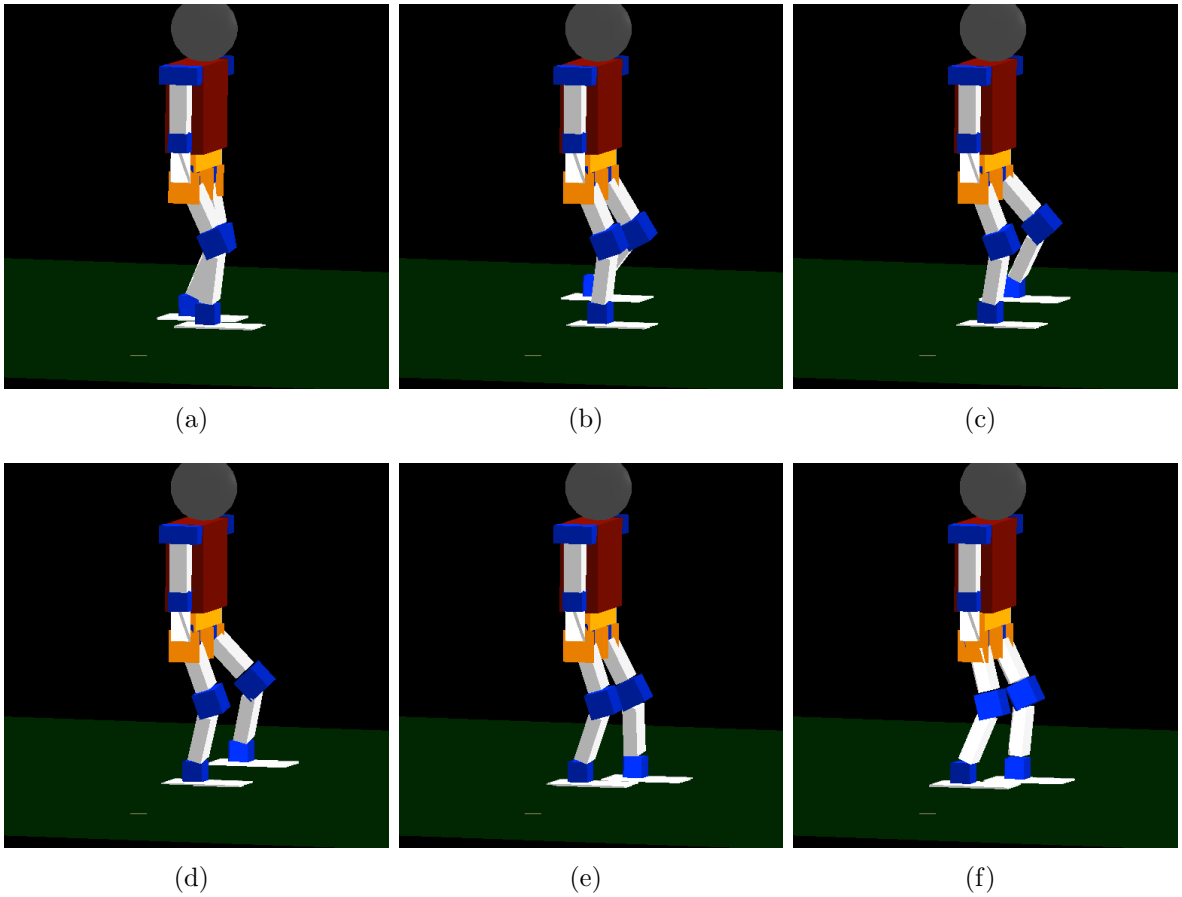
The equations above can also be similarly applied to the transient motion in the Y direction. The equations governing the swing foot trajectory are simply piecewise cosine functions that ensure a zero velocity start and end for the foot motion in all directions. The set of input gait parameters used for this experiment are found in Table 7.2. Table 7.3 shows some of the resulting dependent parameters. Figure 7.46 shows a sequence of still shots of Lim's gait in simulation. Figure 7.47 shows a sequence of video stills Lim's hyperbolic gait. As mentioned before and with the previous walking gait, support was added to the upper body of the robot to ensure that it did not fall over since the gait is inherently unstable.

Resulting reference COM and ZMP trajectory

The results for the COM reference trajectory can be seen in Fig. 7.48. Also seen are the COM limits enforced by kinematic constraints. As seen for the Y direction of the COM, the kinematic enforced limits are at first mostly dictated by the Y position of the robot. However, towards the middle and end of the motion, there is a sharp change in the limit.

Table 7.3: Resulting gait parameters

Description	Parameter	Value (s)
Step time	T_s	0.865
SSP duration of negative X COM acceleration	t_1	0.164
SSP duration of zero X COM acceleration	t_2	0.223
SSP duration of positive X COM acceleration	t_3	0.164
DSP duration	t_4	0.315
SSP duration	t_5	0.550



(a) (b) (c)

(d) (e) (f)

Figure 7.46: Simulation stills of Lim's hyperbolic reference gait

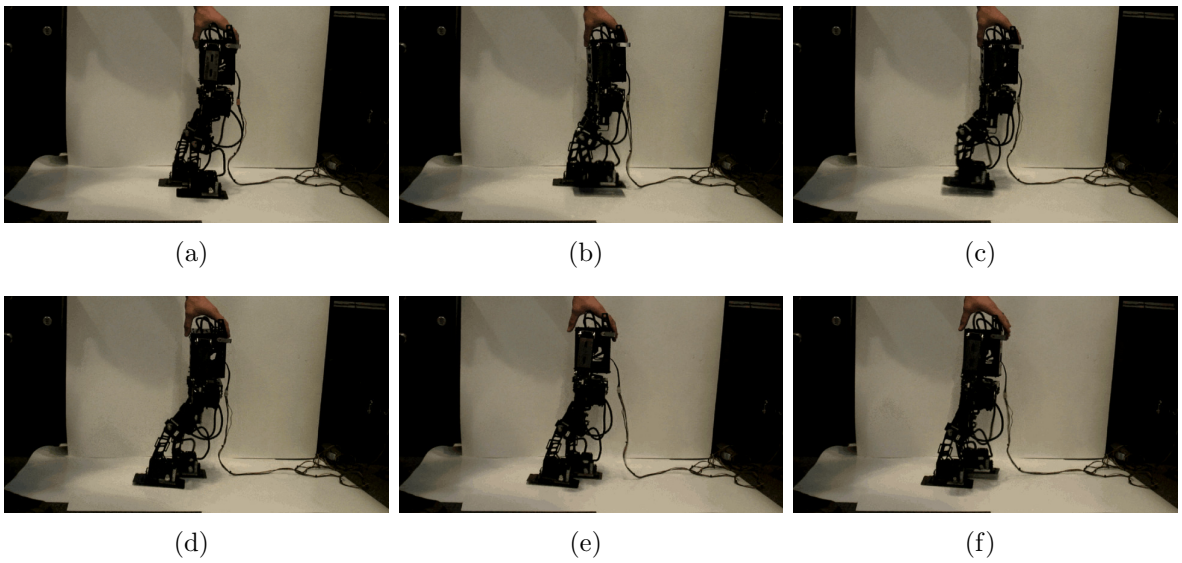


Figure 7.47: Video stills of the Lim's hyperbolic reference gait

This is because the swing foot goes forward to where it will contact the ground for the step before the hip of the robot moves very much in the X direction. Initially, the range of motion of the hip in the Y direction is dictated by the grounded foot. However, towards the end of the motion, the swing leg dictates the range of motion since it is more extended than the ground foot. The kinematic limits of the COM in the X direction is not affected the same way because the waist of the robot is used as the primary means of changing the location of the COM in the X direction. Therefore, the kinematic limit of the COM in the X direction in this instance is not dependent on the kinematic constraints of the lower body. In truth, the absolute limit for the X direction *is* dependent on the kinematic constraints of the lower body since the hips can move in the X direction to change the COM location. However, for this example and experiment, the COM limits in each direction are decoupled.

The results for the ZMP reference trajectory can be seen in Fig. 7.49. From just a glance at the ZMP trajectory, it is obvious that this gait has a significantly different ZMP trajectory from other gaits. The ZMP trajectory in the Y direction clearly shows the portion of the motion that has a constant Y ZMP value. Figure 7.50 shows a parametric plot of both the reference COM and ZMP. The ZMP jumps around because the accelerations of the motion jump around as specified in the design of the gait.

Simple time scaling

In order to determine if time scaling is feasible, the CATF analyzes the motion to determine what cases of instability occur as described in Sec. 4.3. Figure 7.51(a) shows that while the

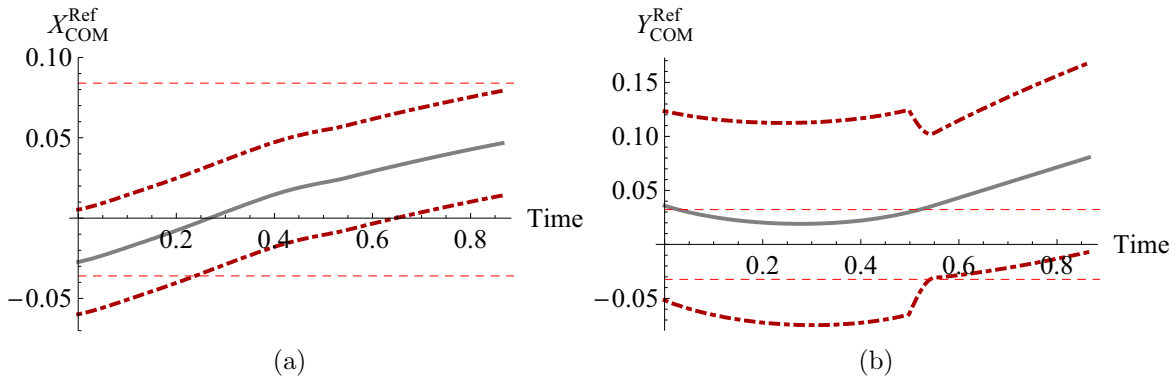


Figure 7.48: COM reference trajectories for Lim's hyperbolic gait. Reference trajectory shown as solid gray and the kinematic limits of the COM are shown as dot dashed dark red.

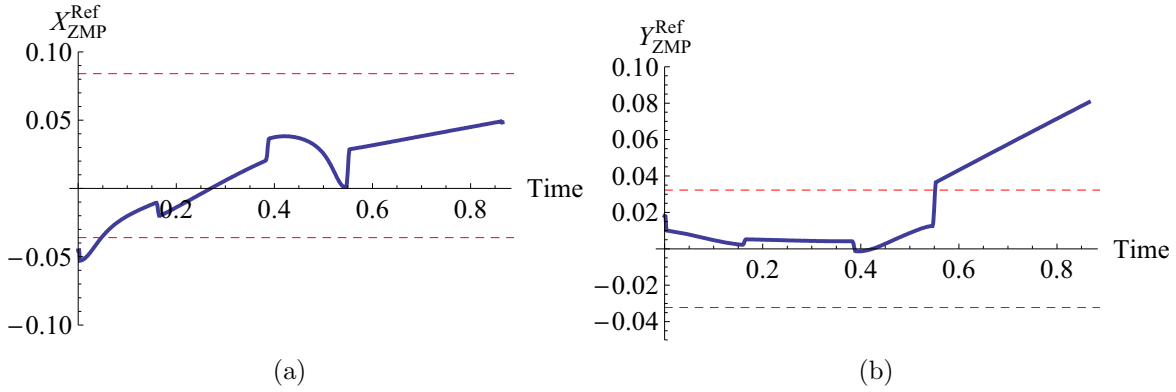


Figure 7.49: ZMP reference trajectories for Lim's hyperbolic gait.

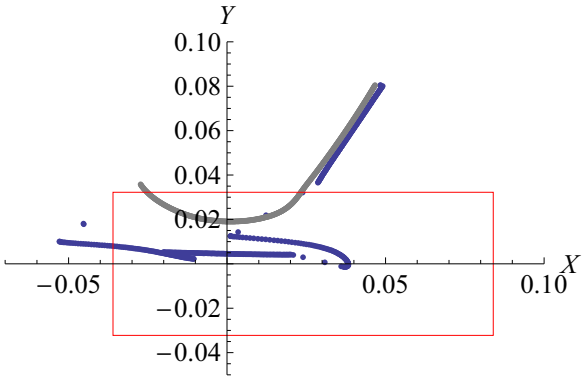


Figure 7.50: COM (Gray) and ZMP (Blue) reference trajectories for Lim's hyperbolic gait. The red box represents the support polygon.

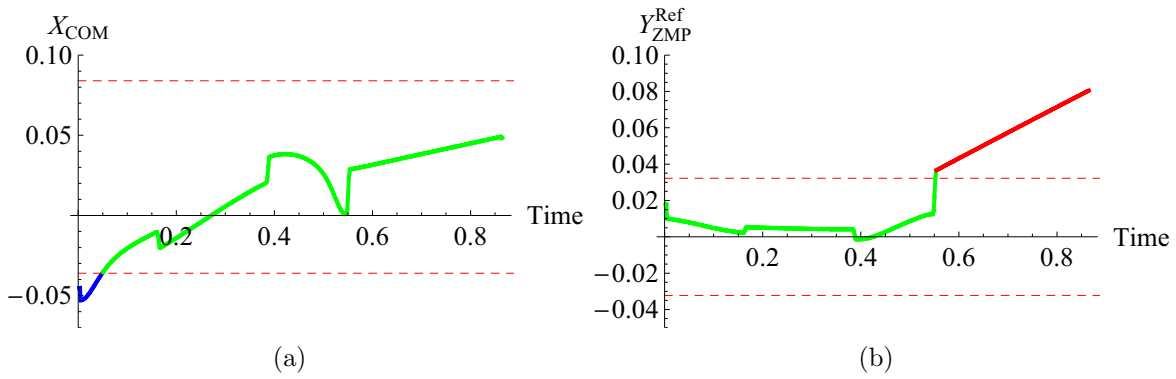


Figure 7.51: Categorized instability for Lim's hyperbolic gait. Blue is Case 1, Red is Case 3, and Green is Stable.

X direction only has Case 2 instabilities, the Y direction has Case 3 instabilities.. Therefore it impossible to stabilize the gait simply by slowing it down or speeding it up. Never-theless, to further demonstrate the nature of a Case 3 instability, we will apply time scaling to attempt to stabilize the gait in the X direction.

Figure 7.52, shows the resulting time scaling factor applied to both directions of the motion that is needed to stabilize the X direction of the motion. Figure 7.53 shows the resulting ZMP trajectories after slowing the motion down. As seen and as expected, while the X direction of the motion is stabilized, the Y direction of the motion is not stabilized at all.

Two and three pose constraints

Again, as with previous walking motions, it is useful to apply a constraint that the beginning and end of the motion must remain the same. Applying this constraint can guarantee that the robot's foot placement when making contact with the ground will be the same in the filtered motion as in the reference motion. This is can be guaranteed because the constraint also constrains the CMA when it generates joint angles for the filtered motion; the beginning and end poses will be exactly the same as the reference motion. Therefore, the solution in Sec. 5.1.2 is applied at the beginning and end of this reference motion. Figures 7.55 and 7.54 show the results after applying the spatial scaling. As seen, the motion is stabilized in both the X and Y directions. Figure 7.56 shows the resulting motion after applying the CMA and calculating the joint angle trajectories of the robot for the filtered stable motion. The waist does not have to pitch very much and the hips do not have to move at all in the X direction to move the COM to the correct X trajectory. The hips have to move someone substantially in the Y direction in order to follow the COM trajectory. Figure 7.57 shows video still of the filtered motion in simulation. The pitch of the waist is very slight and almost unnoticeable. It is almost impossible to notice the change in the Y trajectory of the hips because it is so

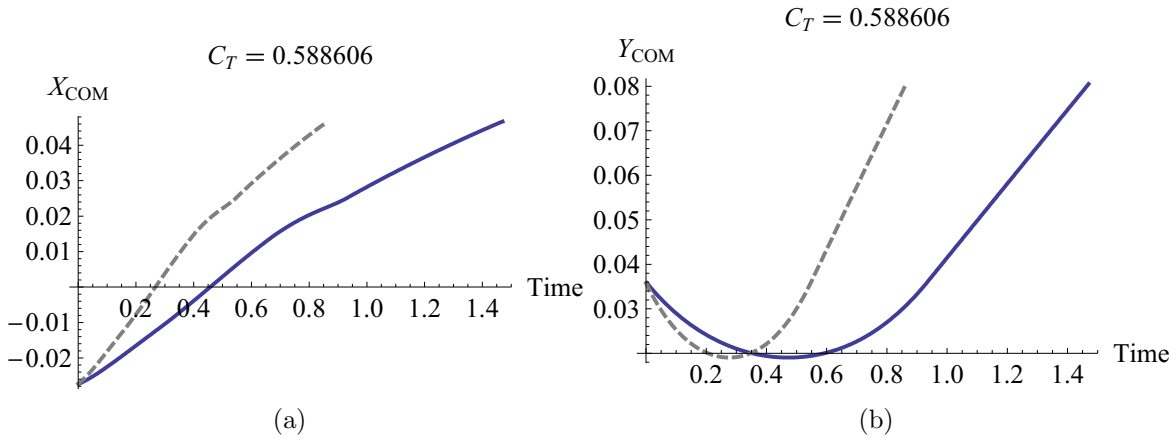


Figure 7.52: AMF COM trajectories from using time scaling on Lim's hyperbolic gait. Reference COM is dashed gray and AMF COM is solid blue.

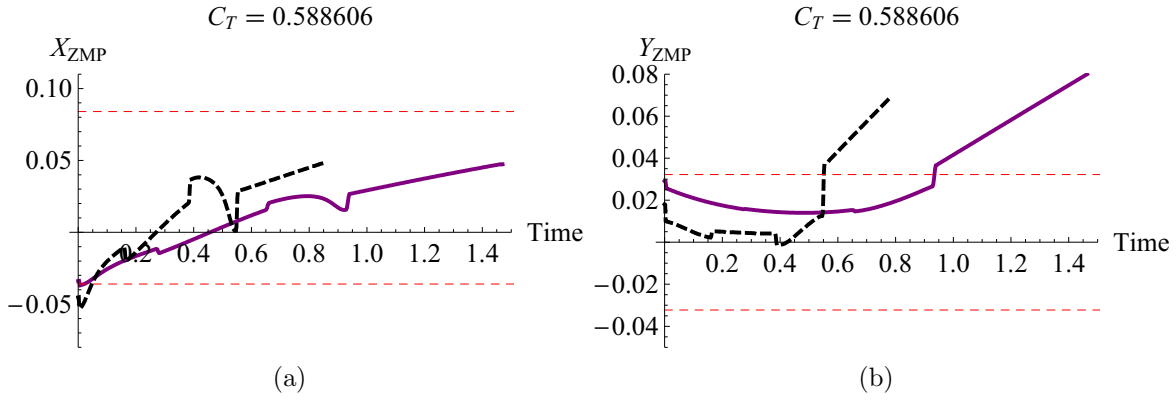
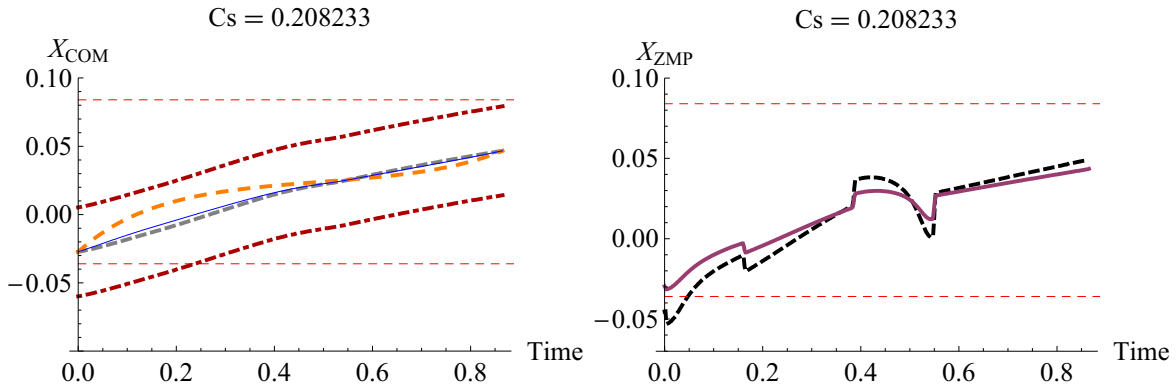


Figure 7.53: AMF ZMP trajectories from using time scaling on Lim's hyperbolic gait. Reference ZMP is dashed black and AMF ZMP is solid purple.



(a) $x^{\text{Ref}}_{\text{COM}}$ is dashed gray, $x^{\text{Fit}}_{\text{COM}}$ is long dashed orange, and $x^{\text{AMF}}_{\text{COM}}$ is blue.

(b) $x^{\text{Ref}}_{\text{ZMP}}$ is dashed black and $x^{\text{AMF}}_{\text{ZMP}}$ is purple

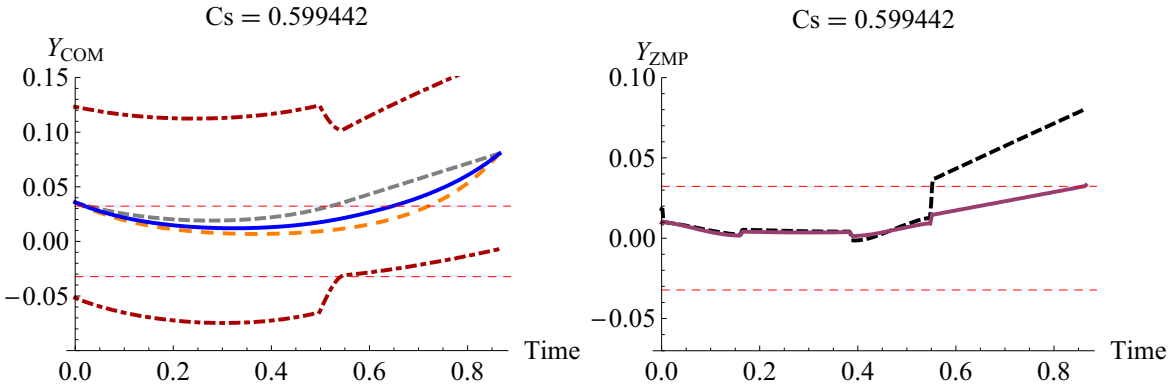
Figure 7.54: The CATF solution to a two point constraint on Lim's hyperbolic gait in the X-direction.

small relative to the size of the robot.

Additionally, it may be useful to apply a third pose constraint. Since it is not always possible to satisfy three pose constraints (Sec. 5.1.3) it is useful to visualize where a third constraint point could lie while allowing a stable motion to be generated; Fig. 7.58 shows the corresponding envelop. As seen, the envelop completely covers the X direction of the motion, allowing for a velocity constraint at the beginning or end of the motion. In the Y direction, only a velocity constraint at the beginning of the motion would be feasible.

7.2.4 Gait from motion capture

The following gait is based on data from a motion capture system monitoring a human walking normally. The motion capture data was recorded at Dr. Lockhart's Locomotion Research Laboratory (LRL) at Virginia Tech. Probably one of the more interesting gaits investigated in this work, the motion capture gait, is based on a human's walk and a human's dynamics. Therefore, translating a motion capture gait directly into a gait for a robot may not remain stable. The human walk is also not definable by any kind of sets of equations or algorithms–it is completely natural. Not only is it interesting to implement the motion, but also to see the resulting stability after translating the gait to a small humanoid robot. The motion capture data was scaled down proportionately according to the height of the human test subject and the height of the robot. The equations used for the inverse kinematics to determine the corresponding angles of the joints can be found in A.3. Figure 7.59 shows a sequence of video stills of the motion capture gait in simulation. Figure 7.60 shows a sequence of video stills of the motion capture gait implemented on DARwIn. One of the



(a) $y^{\text{Ref}}_{\text{COM}}$ is dashed gray, $y^{\text{Fit}}_{\text{COM}}$ is long dashed orange, and $y^{\text{AMF}}_{\text{COM}}$ is blue.

(b) $y^{\text{Ref}}_{\text{ZMP}}$ is dashed black and $y^{\text{AMF}}_{\text{ZMP}}$ is purple

Figure 7.55: The CATF solution to a two point constraint on Lim's hyperbolic gait in the Y-direction.

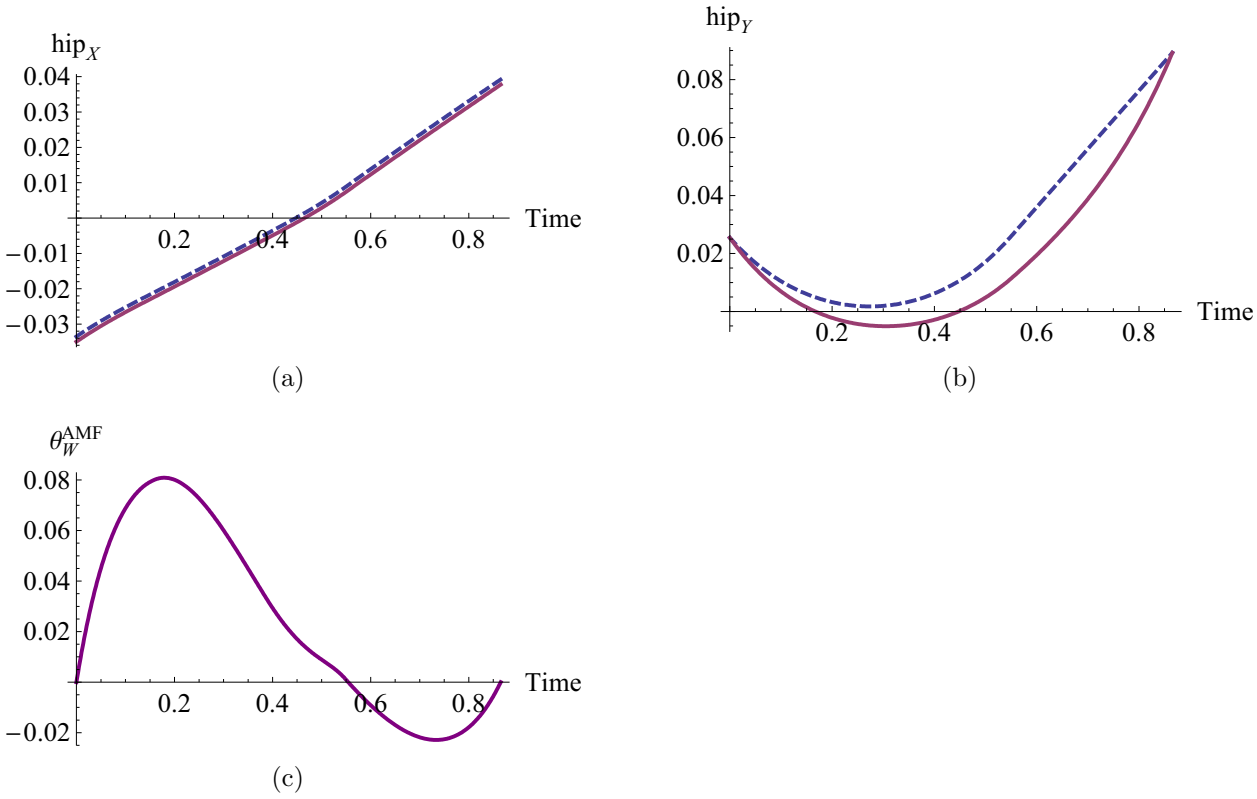
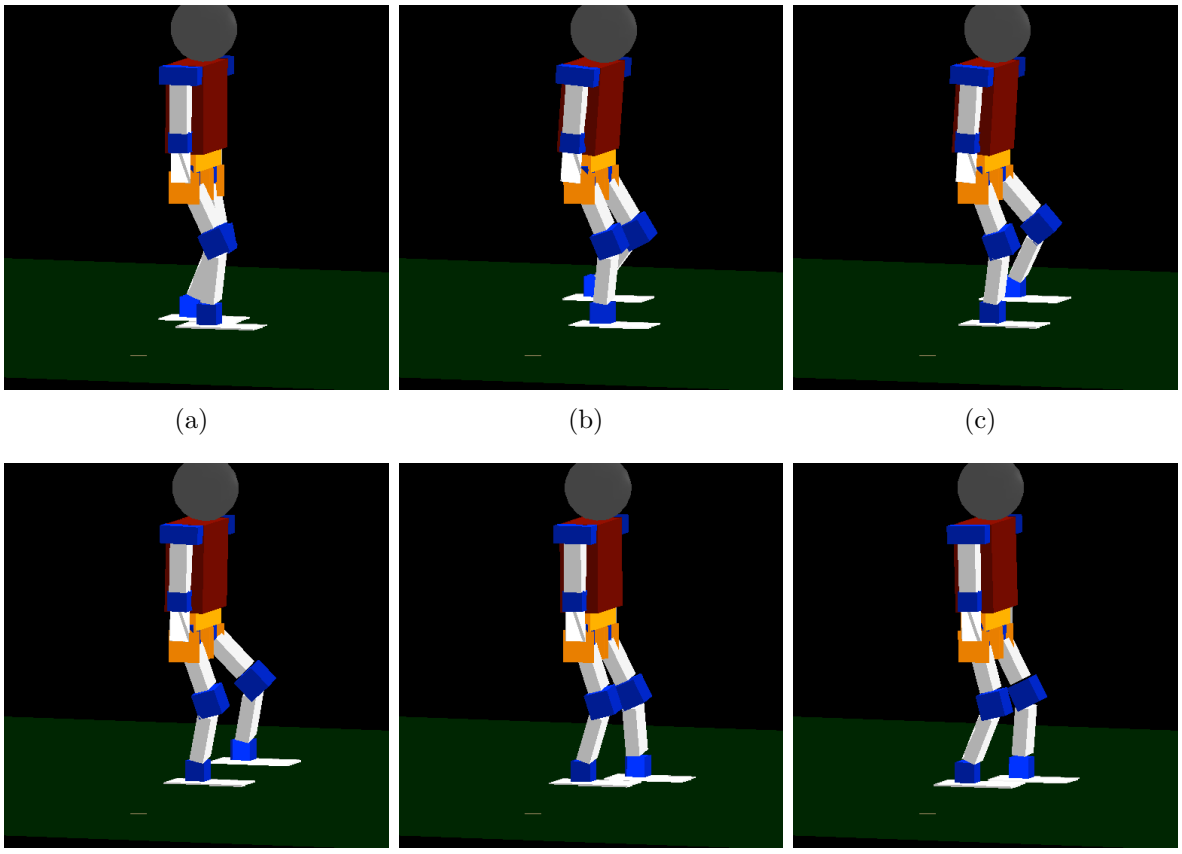


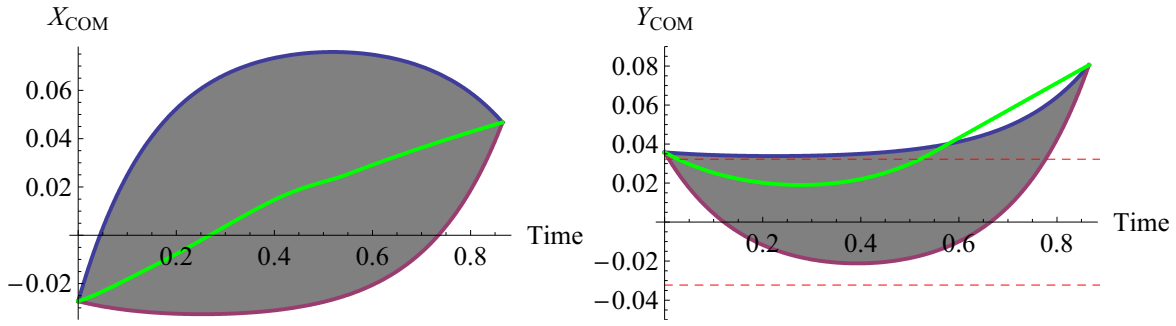
Figure 7.56: Resulting AMF motion after applying the CMA to Lim's hyperbolic gait that was stabilized using two point constraints and spatial scaling by the CATF. Reference motion is dashed blue and the AMF motion is solid purple.



(a) (b) (c)

(d) (e) (f)

Figure 7.57: Simulation stills of the AMF solution to a two point constraint on Lim's hyperbolic gait



(a) (b)

Figure 7.58: For Lim's hyperbolic gait, the envelops that defines where a third constraint could lie in both X and Y directions

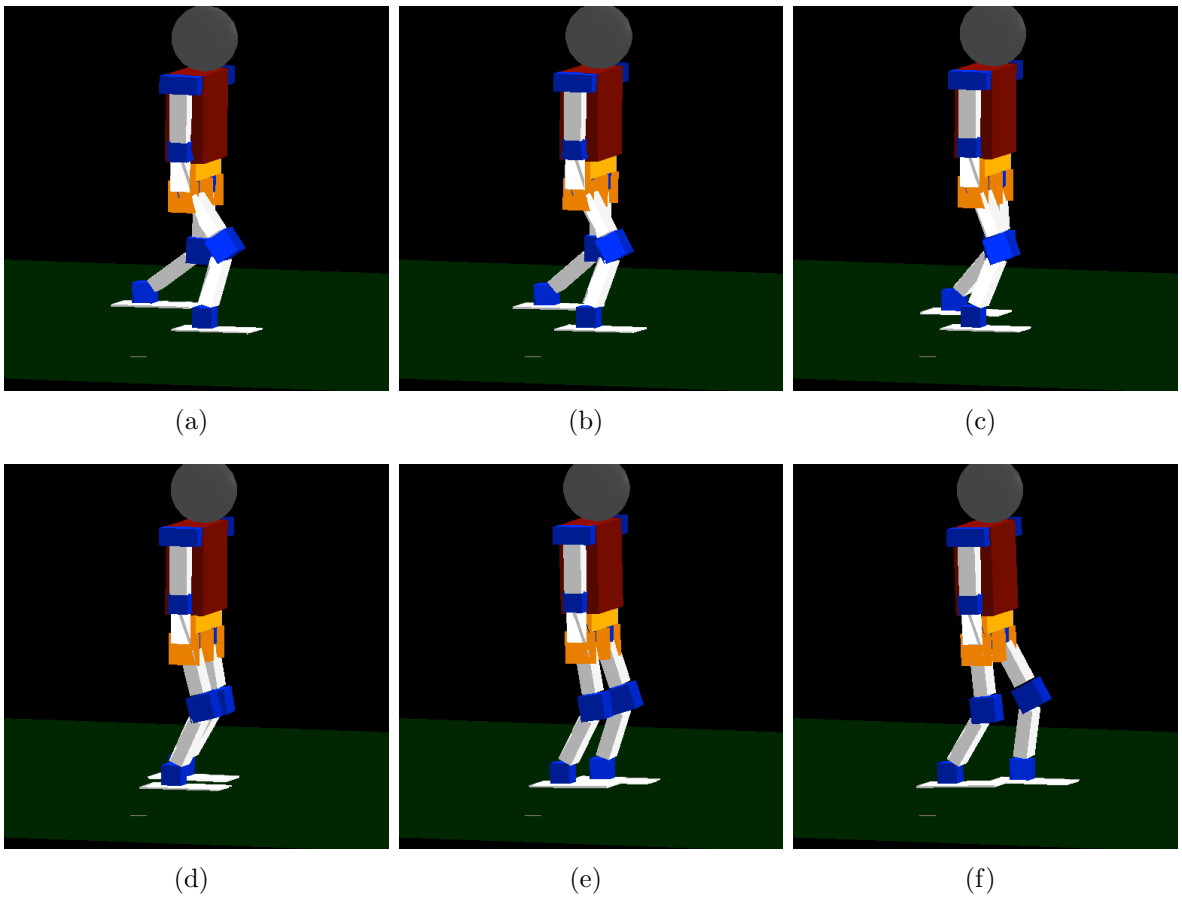
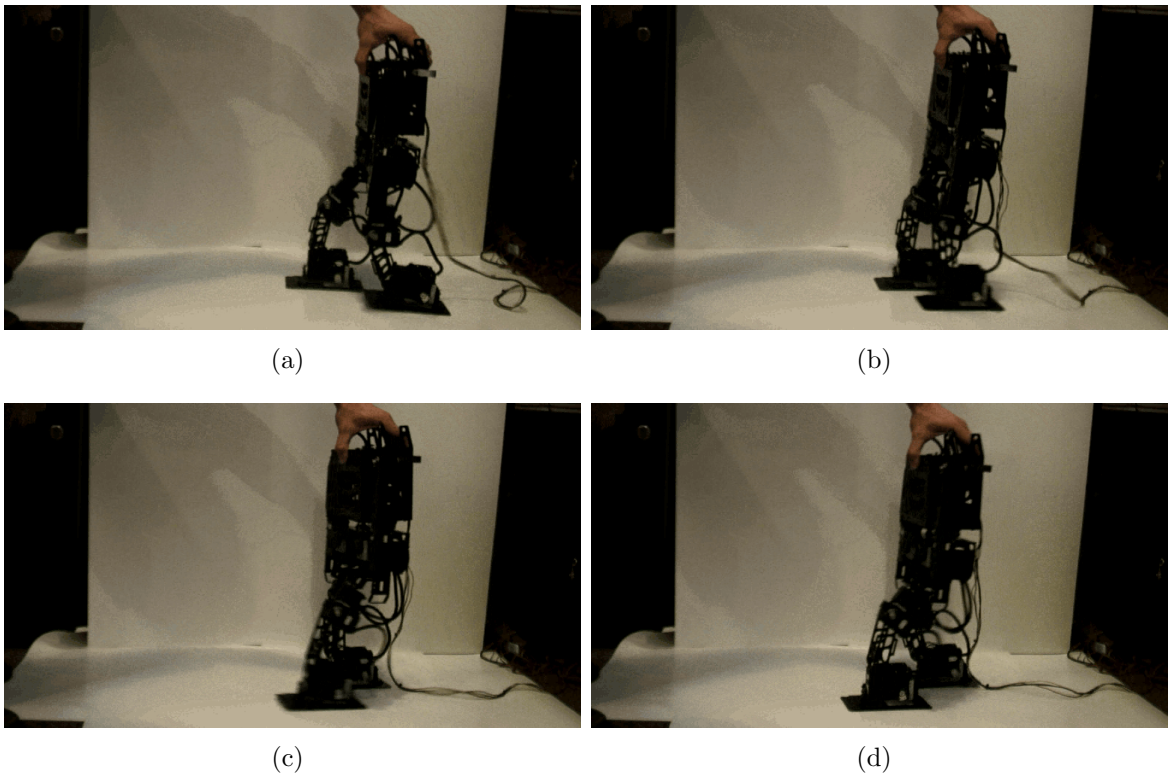


Figure 7.59: Simulation stills of the gait from motion capture

noticeably different characteristics of the gait is the small step height and relatively large step length. Most gaits generated from robots have relatively small step sizes to allow the robot's height to remain the same and to allow the knees to stay bent. Additionally, robots have someone large step heights to overcompensate for error that may present itself in the motor positions that would otherwise lead to the foot prematurely striking or dragging into the ground. When we walk as humans, we tend to take large steps (so we get where we are going faster) and lift our foots up only a little bit so as to expend as little energy as possible when walking.

Resulting reference COM and ZMP trajectory

The COM reference trajectory can be seen in Fig. 7.61. Also seen are the COM limits enforced by kinematic constraints. As seen, the kinematic limits in the Y direction grow increasingly small as the step progresses. This is because of the very large step taken by the



(a) (b)

(c) (d)

Figure 7.60: Video stills of the reference gait from motion capture.

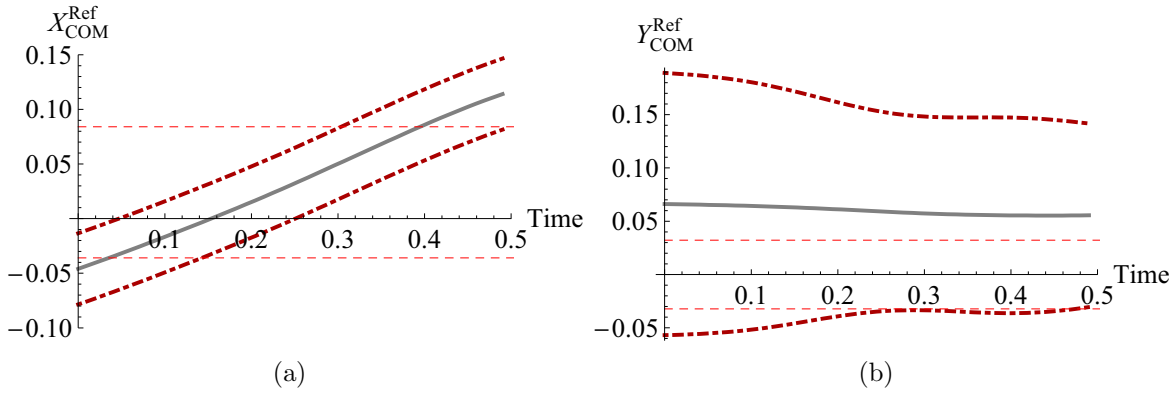


Figure 7.61: COM reference trajectories for a gait from motion capture. Reference trajectory shown as solid gray and the kinematic limits of the COM are shown as dot dashed dark red.

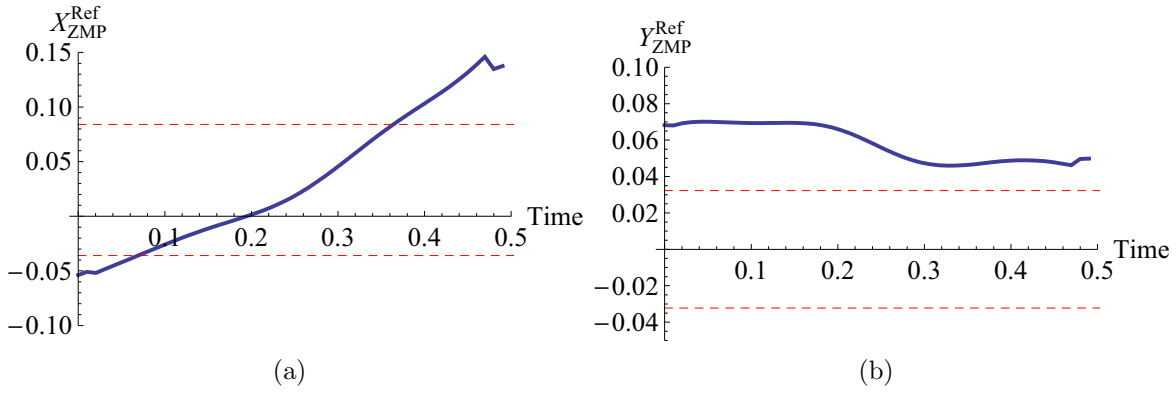


Figure 7.62: ZMP reference trajectories for a gait from motion capture.

robot, which ends up constraining the amount of lateral movement available to the hips the more and more the hips move away from the planted support foot.

The results for the ZMP reference trajectory can be seen in Fig. 7.62. Figure 7.63 shows a parametric plot of both the reference COM and ZMP. [5] The instability in the X direction is very pronounced and significant at the end of the motion. This could be because the step length is almost twice as big as previous walking gaits at 20 cm. Additionally, the entire motion is unstable in the Y direction. This may be from the slimmer profile of a human. Additionally, because the human from the motion capture is much taller than the 60cm tall robot, less sway is needed from side to side to maintain stability [4].

[5]The ZMP points that do not line up smoothly with the rest of the motion (also seen in Fig. 7.62), are a result of inaccurate estimations of the acceleration of the COM at the beginning and ends of the motion. Since motion data is not available beyond those points, the accelerations calculations are a little off.

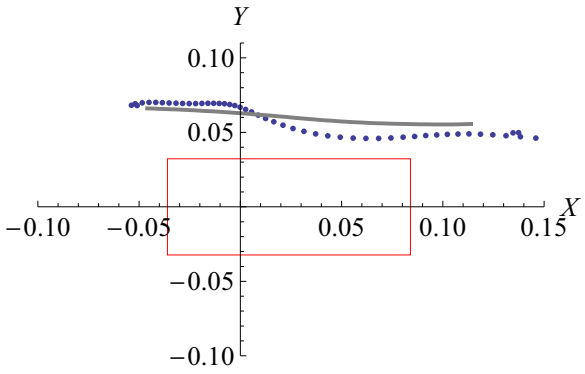


Figure 7.63: COM (Gray) and ZMP (Blue) reference trajectories for a gait from motion capture. The red box represents the support polygon.

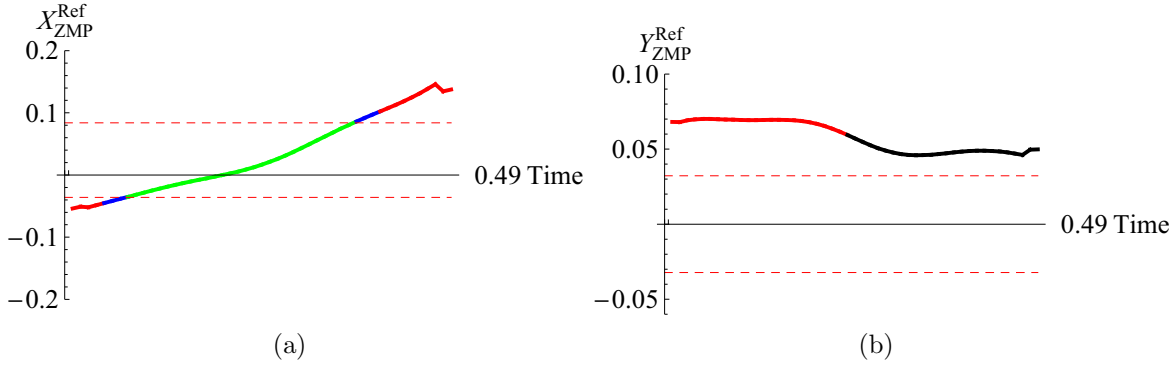


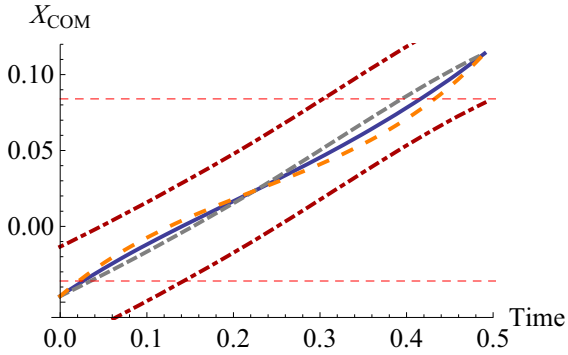
Figure 7.64: Categorized instability for a gait from motion capture. Blue is Case 1, Black is Case 2, Red is Case 3, and Green is Stable.

Simple time scaling

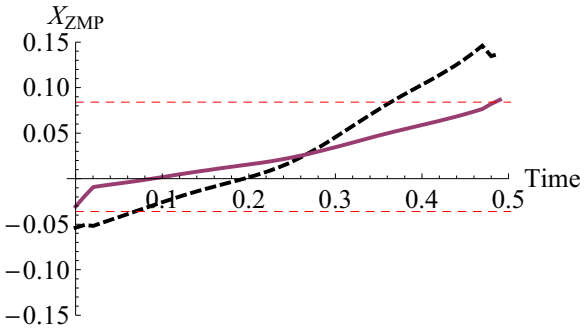
In order to determine if time scaling is feasible, the CATF analyzes the motion to determine what cases of instability occur as described in Sec. 4.3. Figure 7.64 shows that the both the X and Y direction of the COM trajectory have Case 3 stabilities. This means that the motion can not be filtered and stabilized using simple time scaling.

Two and three pose constraints

As in previous walking gaits, the solution in Sec. 5.1.2 with constraints at the beginning and end of the motion is applied. Figures 7.65 and 7.66 show the results of applying the spatial scaling. As seen, both directions of the motion are stabilized while maintaining initial and final pose constraints. Figure 7.67 shows the resulting motion of the AMF filtered motion



(a) x^{Ref}_{COM} is dashed gray, x^{Fit}_{COM} is long dashed orange, and x^{AMF}_{COM} is blue.

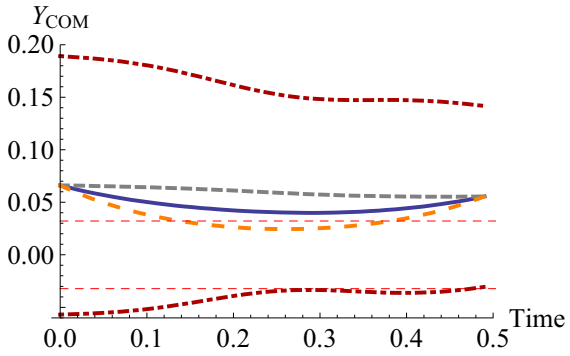


(b) x^{Ref}_{ZMP} is dashed black and x^{AMF}_{ZMP} is purple

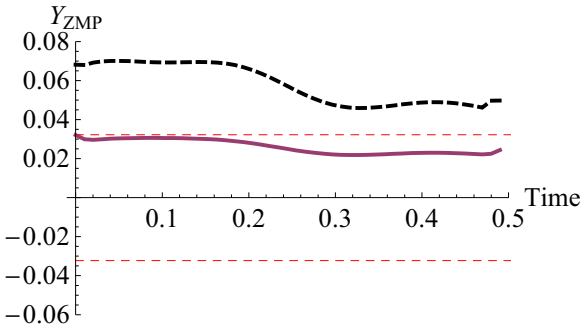
Figure 7.65: The CATF solution to a two point constraint on a gait from motion capture in the X-direction.

after applying the CMA. As seen, the waist pitches slightly in order to appropriately move the COM in the X direction. The hips move significantly in the Y direction in order to relocate the COM to follow the stable COM trajectory. Figure 7.68 shows a sequence of simulation stills of the filtered AMF motion. As seen, the waist motion is not very noticeable. What is not clearly visible in the simulation stills is that the sway of the hips in the Y direction is very different in the AMF motion when compared to the reference motion. The hips have to sway much more in order to maintain a stable ZMP trajectory.

Additionally, it may be useful to apply a third pose constraint. Since it is not always possible to satisfy three pose constraints (Sec. 5.1.3) it is useful to visualize where a third constraint point could lie while allowing a stable motion to be generated; Fig. 7.69 shows the corresponding envelop. As seen, no other pose constraints can be applied in the Y direction. In the X direction, the envelop includes a little too much. Not only are all of the points of the reference motion included in the envelop, but the envelop also includes areas that are outside of the COM kinematic limits. This is not much of an issue since none of the reference motion comes close to the COM kinematic limit. Consequently, velocity constraints could be enforced in the X direction at either the beginning or end of the motion.

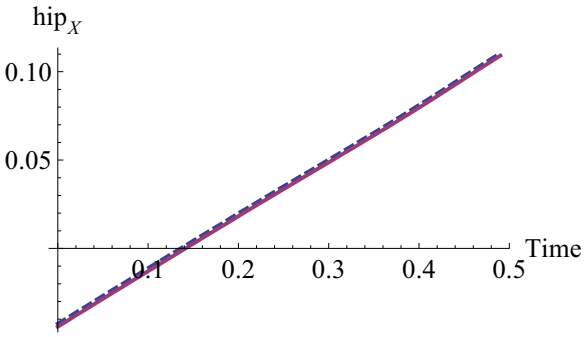


(a) y^{Ref}_{COM} is dashed gray, y^{Fit}_{COM} is long dashed orange, and y^{AMF}_{COM} is blue.

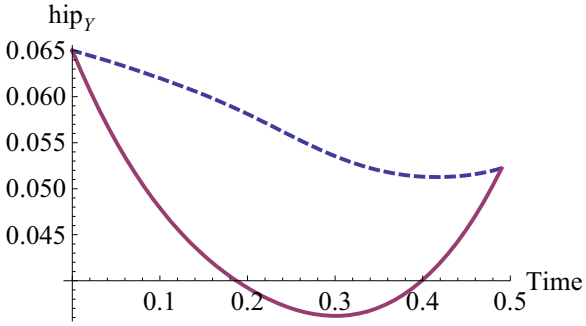


(b) y^{Ref}_{ZMP} is dashed black and y^{AMF}_{ZMP} is purple

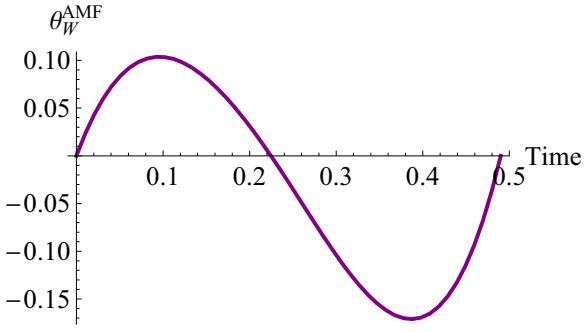
Figure 7.66: The CATF solution to a two point constraint on a gait from motion capture in the Y-direction.



(a)

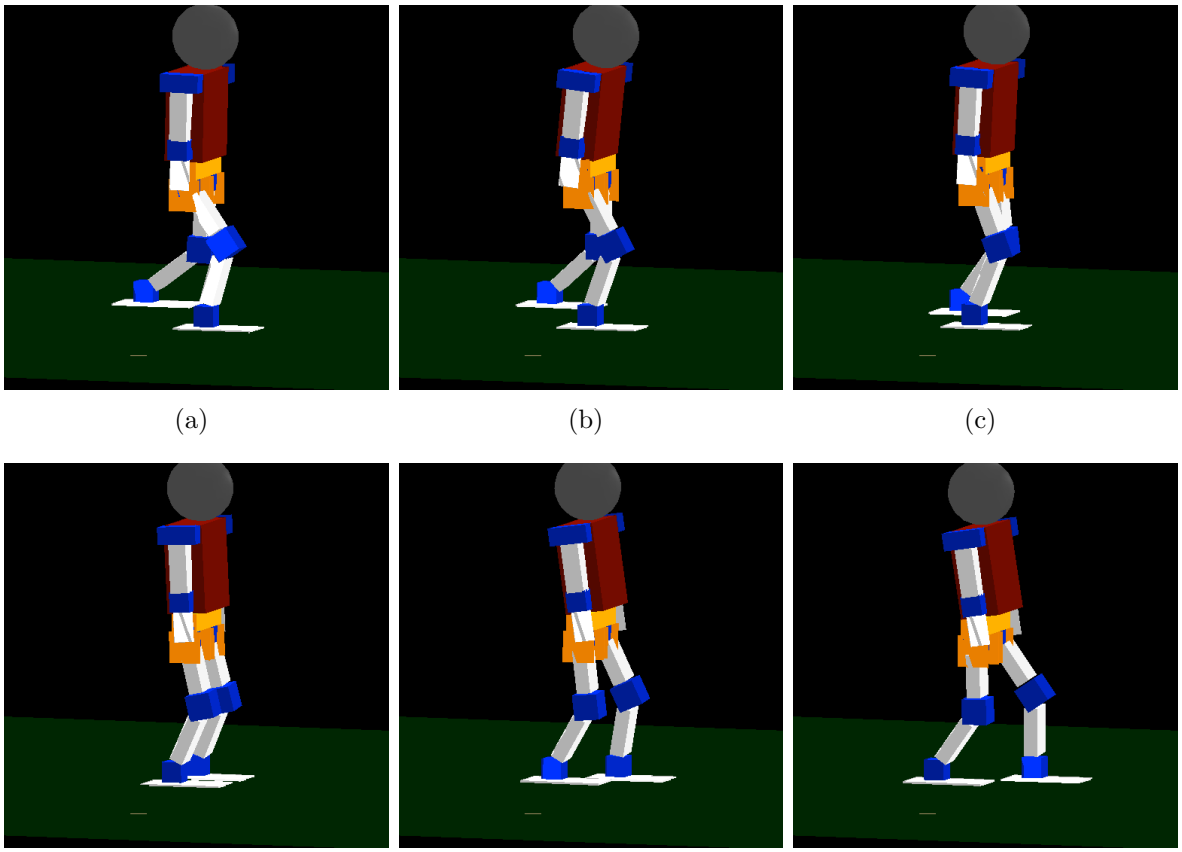


(b)



(c)

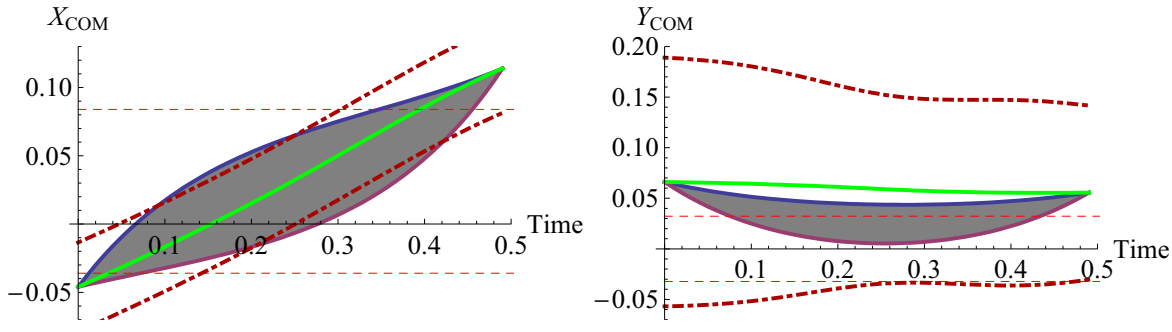
Figure 7.67: Resulting AMF motion after applying the CMA to a gait from motion capture that was stabilized using two point constraints and spatial scaling by the CATF. Reference motion is dashed blue and the AMF motion is solid purple.



(a) (b) (c)

(d) (e) (f)

Figure 7.68: Simulation stills of the the AMF solution to two point constraints to the gait from motion capture



(a) (b)

Figure 7.69: For a gait from a gait from motion capture, the envelops that defines where a third constraint could lie in both X and Y directions

Chapter 8

Conclusions

The goal of this work was to develop an analytical approach to filtering (or creating) a stable motion from a reference motion for humanoid robots. Additionally, another goal was to provide insights into the stability of the system that would otherwise be difficult to identify in a numeric optimization scheme. The filtered motion should be as similar to the reference motion as possible while satisfying applied constraints. The AMF developed simplified the problem by separating it into two parts; constrained analytical trajectory filtering and COM motion adaptation (CMA). The first part, the Constrained Analytical Trajectory Filtering (Sec. 4), used the resulting COM trajectory from the reference motion as a representation of the motion as a whole. Using the COM trajectory, the Zero Moment Point (ZMP) trajectory, which measures the stability of the robot (Sec. 4.1), indicates which portions of the motion are unstable. Additionally, during investigation of these instabilities, it was discovered there were actually different kinds of instabilities (Sec. 4.3). The different cases of instability gave more insight as to why a particular motion was unstable: the motion was too fast, too slow, or inherently unstable. Two primary means of stabilizing a gait were utilized: time and spatial scaling. Spatial scaling scaled the COM trajectory down towards a known stable trajectory. Time scaling worked similarly by changing the speed of the motion, but was limited in effectiveness based on the types of instabilities in the motion and the coupling of the X and Y directions. Additional constraints were enforced and modified solutions developed to handle those constraints. One of the most common constraints is an initial and final condition, which created different variations on spatial scaling. Finally, both time and spatial scaling were used in conjunction/combination to either reduce the amount of scaling needed to stabilize the motion or achieve a specified motion duration.

During investigation of spatial scaling, it was soon realized that the kinematic limitation of the robot had to be taken into consideration since an arbitrary COM trajectory cannot be created and followed by the robot. Therefore, the reference motion's joint angle trajectories in combination with the methods used in the CMA dictated the limitations on where the COM could exist after finishing filtering in the CATF. Generally, the kinematic constraints

are not a problem since the motion is scaled to a stable trajectory, which is typically further from any kinematic limit than the reference motion. However, by considering kinematic limits, the nominal ZMP or amount of spatial scaling could be adjusted such that kinematic constraints were not violated in the CMA when satisfying the filtered COM trajectory.

When applying pose constraints to spatial scaling, the results provided more insight to the stability of the robot's motion. Applying three and four pose constraints (Sec. 5.1.3 and 5.1.4) showed the limits on what areas of the motion could be constrained. The envelops of stability identified where additional constraints could lie and if velocity or acceleration constraints were feasible.

Both time and spatial scaling have limitation or constraints to satisfy that are inherent in the system. Conversely, they both have extremely attractive advantages over the other. Time scaling does not have to consider kinematic constraints or limitations since the COM path through space is unchanged and therefore the joint angle trajectories unchanged. Since the joint angles of the filtered motion are the same as the reference motion, and the reference motion is assumed to be kinematically feasible, the resulting joint angles for time scaling are naturally feasible. However, time scaling is ineffective for certain portions of motion that are inherently unstable, which is where the strength of spatial scaling is realized. While spatial scaling can potentially stabilize any reference motion, it must consider and follow kinematic constraints since the resulting COM path must be kinematically feasible.

Even more powerful was combining time and spatial scaling (Sec. 5.2)–making use of their strengths to make up for the compliment's weaknesses. In one instance, only a fraction of the time scaling or spatial scaling was needed when the other was used in conjunction for stabilizing the motion. In a sense, by changing the speed of the motion just a little bit, the motion could follow a path closer to the original reference path after spatial scaling. Additionally, any desired time scaling could be used to control the step time of the motion since spatial scaling could completely stabilize any added instabilities caused by time scaling.

No aspect of the AMF currently considers loading on the robot's joints or collision between robot links. Collision between robot links could be expressed as kinematic constraints, but could prove cumbersome. Torques, speeds, and accelerations of joints are not analyzed in this work. Therefore the AMF, as it is now, could potentially generate a motion that requires unreasonably high speeds or torques from the joint actuators. Even though the motion may be theoretically stable, the robot could still not be able to perform the motion. A similar situation occurs if the AMF motion has joints colliding.

In this work, time scaling was simplified such that the time scaling was constant in order to simplify and generate a solution in Sec. 4.3. However, it would be very interesting to see in future work the inclusion of a time varying time scaling function as shown in Eq. 4.12. By using a time varying function, time scaling could stabilize many more gaits–including those with Class 3 instabilities. Additionally, a varying time scale function can allow additional constraints on the motions, such as a time step constraint. With a constant time scaling function, the duration of the motion is changed, however, a changing function could allow for

time lost or gained in stabilizing unstable portions of the motion to be "made up" by doing the opposite, slowing down or speeding up, in the stable portions of the motion. Finally, a time varying function may also allow for a more direct way to determine the optimal speeds of the entire motion in order to stably complete the motion as quickly as possible. An obvious benefactor would be robots in automation tasks in industry. If industry decides to use humanoid robots or robots that use the ZMP as stability criteria, it would be very beneficial to have a closed form solution to determine the fastest way to follow a potentially unstable path through space.

A few considerations for the CATF were addressed in Sec. 4.4. An arbitrary non-rectangular support polygons proved to be difficult to account for while maintaining analytical solutions. A non-rectangular support polygon couples the X and Y directions and therefore exponentially increases the difficulty of the problem. It is possible that some predefined support polygons have closed formed solutions (such as two rectangular feet in contact with the ground) and should be investigated in future work. As a suggestion for approximating the support polygon, a time varying ZMP nominal value and limit values could be used relatively easily. It was also interesting to note that changing the height of the robot, which is not a focus of this work, has similar effects on stability as time scaling. This makes some sense since changing the height of the robot in essence changes its natural frequency and therefore should correspond to time.

The inverse of the simplification of stabilizing the COM trajectory based on the joint angles is considered when generating the joint angle trajectories for the robot based on the stable COM trajectory. Since most humanoid robots are redundant systems (especially with DOF in the upper body), it is useful to use the reference motion as a starting point for generating the filtered joint angle trajectories. A few different methods were suggested for creating the joint angle trajectories from the stable COM trajectory. For the examples in this work, the waist was used as the primary means to move the COM in the X direction. In order to make the motion look more natural, the pitch of the waist was limited to $\pm\pi/8$. When pitching the waist was not enough to move the COM in the X direction, the hips were used to move the COM the rest of the way. The hips were also used to move the COM in the Y direction.

The results of applying the AMF to the various reference gaits provided additional insight into some of the algorithms used to stabilize the reference gaits. It was interesting to see the different kinds of instabilities manifest in actual reference gait data and then the corresponding stable gaits created. The gait from motion capture was very interesting since it was much different than other walking gaits. The motion capture gait was always unstable in the Y direction, which is indicative of the difference in dynamics and physical structure between humans and the test robot platform. Standing taller with a high center of mass, slim feet, and a slim body, humans walk with very little lateral motion. On the other hand, a small robot with wide feet and a wide body needs more lateral motion to remain stable as Kim confirmed in [4].

During the experiments with the physical hardware, it was obvious that the robot platform

was not sufficient for some gaits. The servo motors used to control the joints of the robot use a simple proportional position controller, which means that there will always be error in the joint angles–even more error for higher loads. Therefore, especially for the SSP of the walking gaits where loads were very high on some of the joints and actuators, the robot did not follow the joint angle trajectories closely. Even if the generated joint angle trajectories produce a stable walking motion in simulations, the motors do not precisely follow the trajectories to implement the gait. Future work should include either a reevaluation of the robot hardware, or work on a control to compensate for the joint angle errors created at the joints.

In addition to compensating for the joints' error, future work should also investigate integrating force and torque requirements into the AMF. It would be interesting to see how joint torque, velocity, or acceleration limits manifest themselves in the AMF algorithms. In order to verify the robot's motion, it would also be interesting to include a motion capture system for the humanoid robot. By capturing the robot's motion, the resulting filtered motion can be closely analyzed and verified. This could serve as a way to verify incorporating a joint angle control law into the AMF.

Moving the COM by changing the joint angle trajectories in order to stabilize a gait can be effective, however it could generally be inefficient. If a robot is constantly bending forward or leaning in one direction in order to create a stable gait, it may be more worthwhile to redesign the robot–or at least the mass distribution of the robot. Incorporating mass distribution and the robot's physical design into the motion filter could be very valuable in understanding the relationship between gait stability and walking platforms. Determining the physical design of a robot necessary to precisely and stably mimic a human's walking motion would not only be a fantastic advancement for robotics, but also for understanding the dynamic relationship between humans and robots.

The analytical motion filter presented in this work lays out some foundations for further investigating the complex interactions between the COM trajectory and gait stability of a humanoid robot. The algorithms used in this work could also be expanded to legged robots or entirely different platforms that depend on stability and can use the ZMP as a stability criterion. One of the primary contributions of this work was showing that an entire reference motion could be stabilized using a single set of closed form solutions and equations. Previous work by others considered optimization functions and numeric schemes to stabilize all or a portion of a gait. Instead, the Analytical Motion Filter gives a direct relationship between the input reference motion and the resulting filtered output motion.

Chapter 9

The DARwIn Robot

The Dynamic Anthropomorphic Robot with Intelligence (DARwIn), a humanoid robot developed at the Robotics and Mechanisms Laboratory (RoMeLa), is a sophisticated hardware platform used for studying bipedal gaits that has evolved over time. Many versions of DARwIn have been developed, each an improvement on its predecessor. The platform is primarily used as a research tool for studying bipedal and humanoid locomotion. Additionally, DARwIn has been tailored for the international autonomous robot soccer competition, Robocup. Robocup, aside from being a competition, is a venue for advancing research in humanoid locomotion and robot intelligence.

9.1 Actuator description

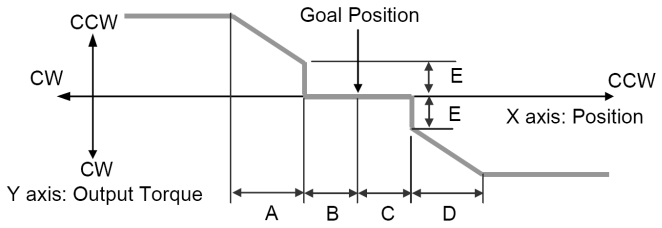
All of versions of the DARwIn robot up through version III use Dynamixel motors from Robotis, which simplified developing the robot platform [56]. The Dynamixel motor includes the motor (Swiss coreless DC motor: Maxon RE-MAX), metal gear reduction, controller, driver, and networking all in one package. The sophisticated servo motors use RS485 serial communication protocol, which allows for the motors to be daisy chained together. The most powerful motor, the RX-64 as seen from 9.1 has a torque of 64 kg-cm (55.5 in-lb). All of the motors have a range of 300^o and resolution of 0.29^o. The motors can make complete rotations, but then do not have position control.

The controller specifies the compliance of the motor according to Fig. 9.1. The compliance is also the position controller. As seen, the controller is effectively just a changing proportional controller, which means that if there is a load on the motor, there will always be some position error. This is very problematic if the motors are needed to follow an exact motion.

The Dynamixel motors communicate on an RS485 network at a default of 57600 Baud. The baud rate can be set up to 1,000,000. The motors have some EEPROM and RAM to keep

Table 9.1: Specifications of the Dynamixel motors from Robotis

	DX117	RX-28	RX-64
Weight (g)	66	72	125
Gear Reduction Ratio	1/193	1/193	1/200
Recommended Voltage (V)	16	16	18
Holding Torque (kgf.cm)	38.52	37.7	77.2
Speed (Sec/60 deg.)	0.129	0.126	0.157



A : CW Compliance Slope (Address 28)
B : CW Compliance Margin (Address 26)
C : CCW Compliance Margin (Address 27)
D : CCW Compliance Slope (Address 29)
E : Punch (Address 48, 49)

Figure 9.1: Diagram of how the torque of the Dynamixel motor is a function of the position error

Figure 9.2: The cycloid robot made by Robotis

track of various states of the motor (these are described in more detail in Sec. 10.2.1). The values are accessed using basic commands such as read and write sent on the RS485 network in the form of hexadecimal strings. Some of the values allow the user to specify safety options for the motor: voltage limits, temperature limits, range limits, load limits, etc.

9.2 DARwIn 0

To investigate the feasibility of controlling a 21-DOF, the software in development for DARwIn was first tested on a predesigned/fabricated robot from Robotis (Fig. 9.2). Since the Cycloid robot was not the first physical iteration of DARwIn, but the platform for testing the first version of software, the testing iteration is called DARwIn 0. The motors used for controlling the robot's motion were the Dynamixel DX117. DARwIn 0 proved to be a success, demonstrating that the core software developed for the robot worked for making the robot stand up and walk. The motions and gaits were developed similarly to DARwIn I's motions and gaits and are discussed in more detail in 9.3.

9.3 DARwIn I

DARwIn I was the first humanoid robot created as a senior design project at the Robotics and Mechanisms Laboratory (RoMeLa). The motions created were "recorded" by logging all of the angles of the joint motors for numerous poses, which when seen in succession and "played" back look like a walking gait or standing up. No sensor information was used for stability control and the gaits were not generated from mathematical functions. Therefore the robot would not walk successfully unless a good enough sequence of stances was recorded. Additionally, the robot would fall over in the presence of any external disturbances.

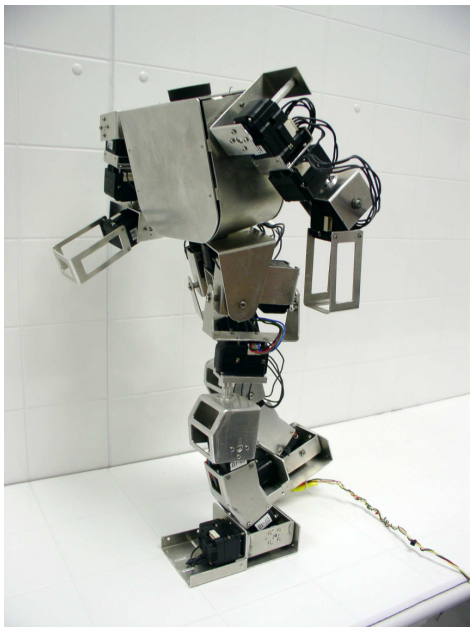


Figure 9.3: Photo of DARwIn I

DARwIn I has 21 degrees of freedom (6 in each leg, 4 in each arm, one in the waist), 4 force sensors on each feet, a 3 axis rate gyro, a 3 axis accelerometer, and space to house a computer and batteries for powering the motors, sensors, and computing equipment. DARwIn I's links are fabricated out of bent sheet aluminum. The robot uses Robotis' Dynamixel DX-117 motors for the joints and Flexiforce force sensors in the feet (though the sensors were never used).

The development of DARwIn I focused on the design for anthropomorphization (Fig. 9.3). Since the results of testing and experimentation using DARwIn would be compared with actual human data, it was necessary to design the robot to physically mimic a human as closely as possible. Using human proportion data from [57], the undergraduates designed DARwIn I's links to be in proportion to its height and its joints to follow the range of motion of an average male human. Many humanoid robots being developed at research labs today or marketed as hobbyist toys are often made just to "look" like a human. However, the senior design team took great care to design DARwIn I's proportions to be nearly identical to that of a human's. Not only is DARwIn I scaled in dimensions similarly, its primary joints are kinematically equivalent to those of humans'. Humans have a ball and socket joint at the shoulders and hips, allowing three axes of rotation about a single point (Fig. 9.4). Though DARwIn does not have a ball and socket joint, it achieves the identical kinematics with three motors' axes of rotation intersecting at a single point–making it equivalent to a ball and socket joint. Not only does this make the kinematic configuration closer to a human's, it also simplifies the mathematics involved in controlling and creating the motion of the robot.

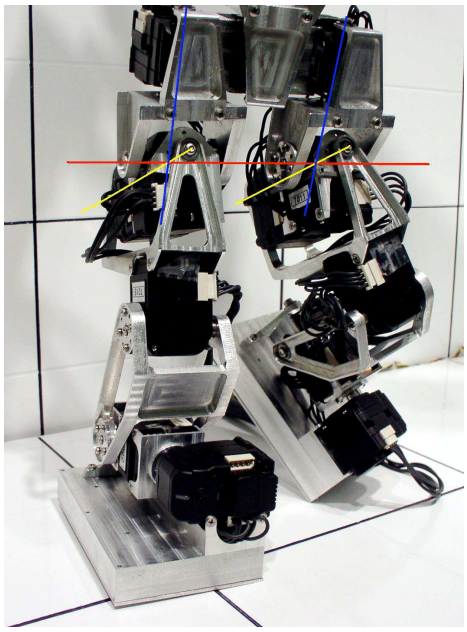


Figure 9.4: Closeup of DARwIn's hips showing kinematically spherical joints

It took many iterations to get to the final design of DARwIn I. Figure 9.5 shows the kinematic diagram that shows the planned motor arrangement of the robot. The initial design of the robot was to use rapid prototyping techniques to create an "exoskeleton" for the robot as seen in Fig. 9.6. Many hip designs (Fig. 9.7 and 9.8 were considered before deciding on a final design since the hips are very difficult and cumbersome to design. Additionally, many torso designs were considered for housing the motors appropriately for the arms (Fig. 9.9).

Eventually, the design team came to the realization that rapid prototyping, while attractive, was extremely expensive and would not be a feasible option. Therefore, the design had to be reworked to use inexpensive materials and machining. With little to no access to shop equipment, the entire robot was made of sheet aluminum. Figures 9.10 and 9.11 show the resulting designs for the upper body and hips using sheet aluminum. Figure 9.12 shows the design for the feet of DARwIn I using sheet aluminum as well as the design for the placement and use of the force sensors on the feet.

The design team also considered the range of motion of the robot in order to ensure that it was similar to that of a human's. Figures 9.13 and 9.14 show the range of motion studies performed in CAD.

Finally, two parallel software approaches were pursued to come up with a way to control the robot's motors. Both LabVIEW and C++ using QT were used as development languages. Figure 9.15 shows a screen shot of the main panel of each software tool.

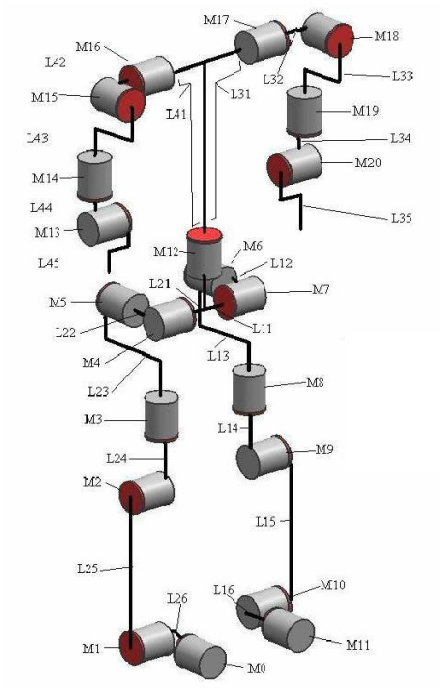


Figure 9.5: Kinematic diagram of DARwIn I

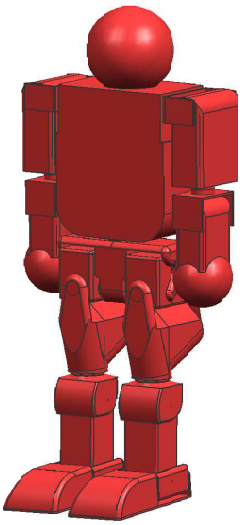


Figure 9.6: CAD rendering of a rapid prototype DARwIn I

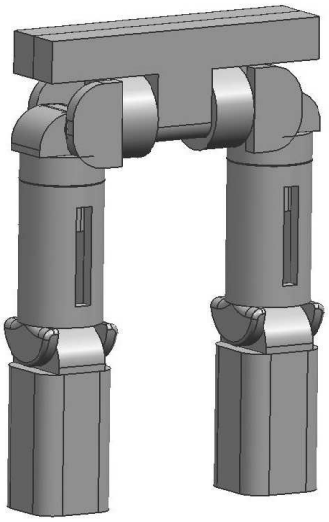
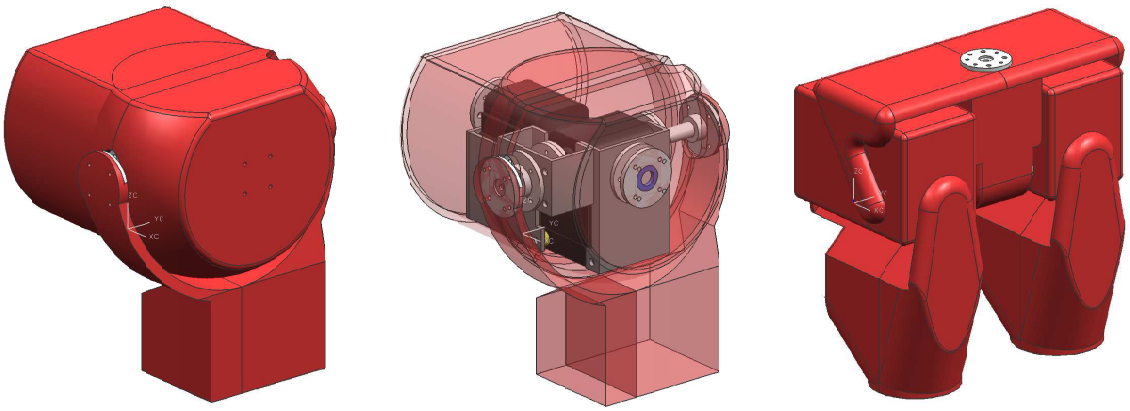
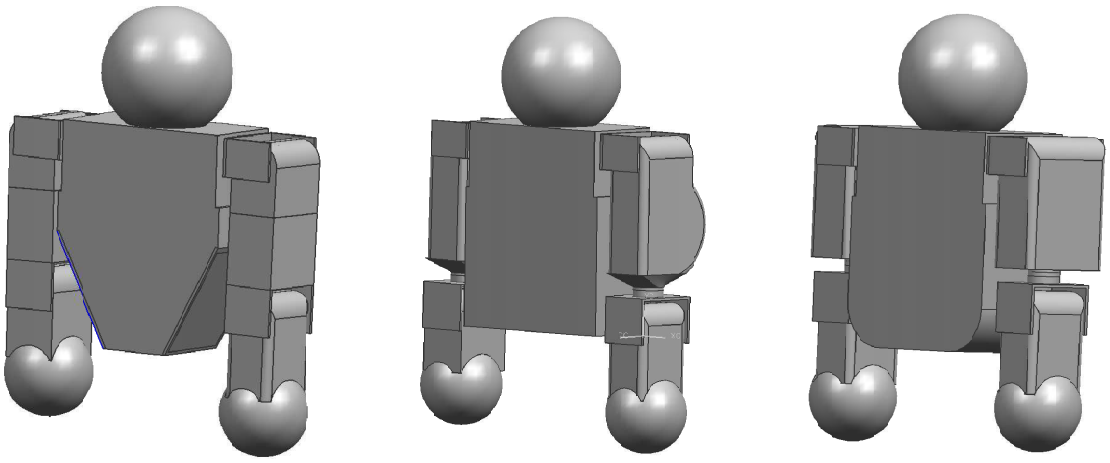


Figure 9.7: Design idea for the lower body of DARwIn I



(a) Closeup outside view (b) Closeup translucent view (c) View of hip assembly

Figure 9.8: Hip design for a possible prototype of DARwIn I



(a) Version 1 (b) Version 2 (c) Version 3

Figure 9.9: Different design ideas for DARwIn I's upper body

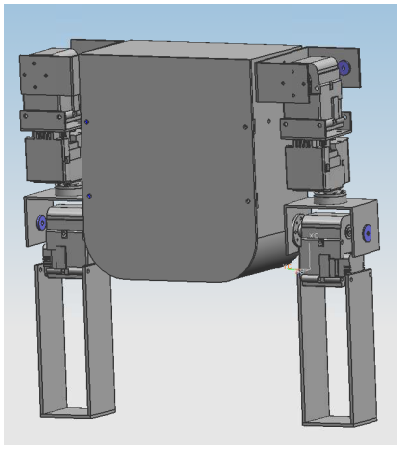
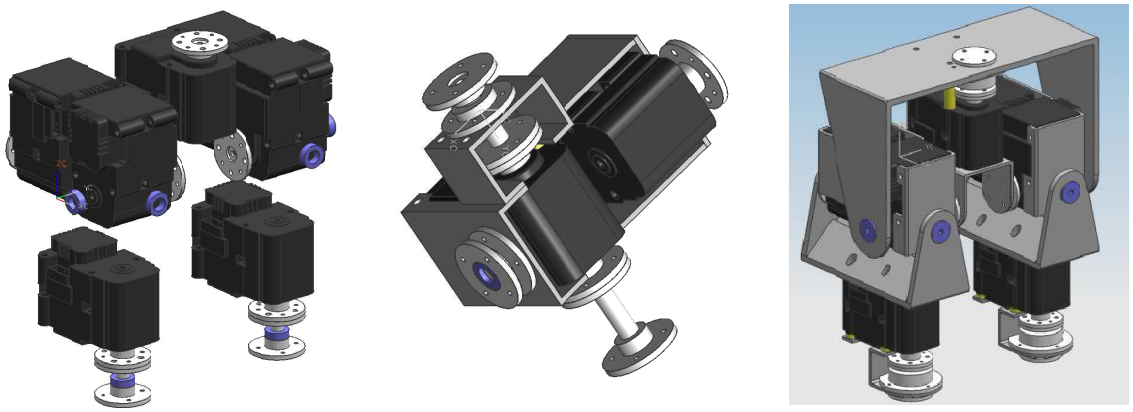


Figure 9.10: CAD drawing of the final version of DARwIn I's upper body



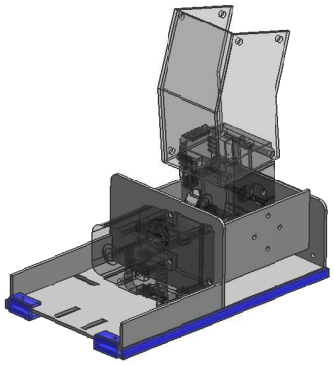
(a) Motor view (b) Hip block assembly (c) Entire hip assembly

Figure 9.11: CAD models of the final version of DARwIn I's hips

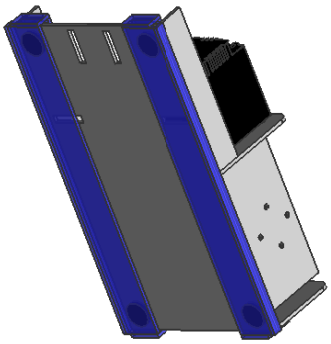
9.4 DARwIn II

DARwIn II was designed and fabricated by the 2006-2007 senior design team. Two version were created, DARwIn IIa in the Fall and DARwIn IIb in the Spring. DARwIn IIa (Fig. 9.16) builds on its predecessor with improved mechanical design, more sensors, and added intelligence. Control of the robot's motion for stability, especially for bipedal walking, often requires precise knowledge of link locations and movement. By making the robot's links as stiff as possible, there is less error in the system. If a link in ankle were to flex just 1 or 2 degrees, the upper body would sway as much as 30 millimeters. Analyzing the design of the links using finite element analysis and using a CNC machine to mill out the links from solid blocks of aluminum, the stiffness of DARwIn's links were maximized and weight minimized.

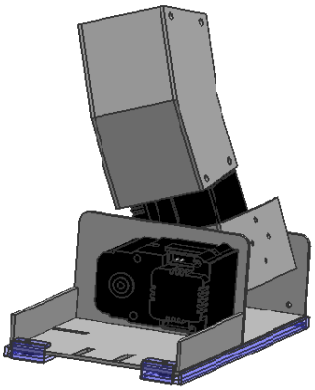
DARwIn IIb is based on the design of DARwIn IIa, but with improvements in all categories (Fig. 9.17). The motors used for articulating DARwIn's joints were replaced with a motor with twice the torque. DARwIn's link design was further refined to create even lighter weight parts. The entire computer, sensors, electronics package, and computer ports were mounted to a custom designed heat sink as a single module. This module is attached to the robot body using shock mounts, which allows easy access and removal while protecting the equipment from shock when falling.



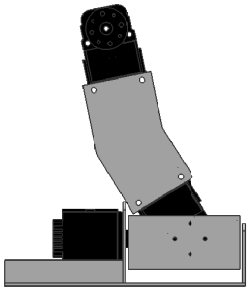
(a) Translucent view



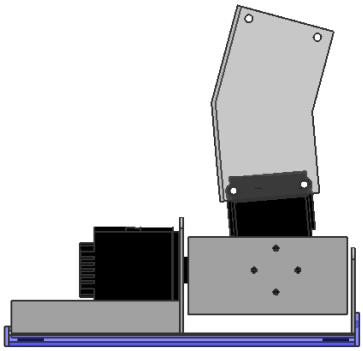
(b) View of sole of foot



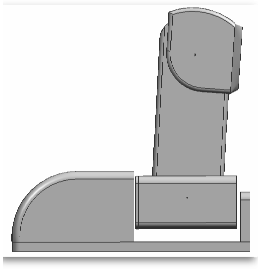
(c) Front view showing range of motion



(d) Side view showing range of motion

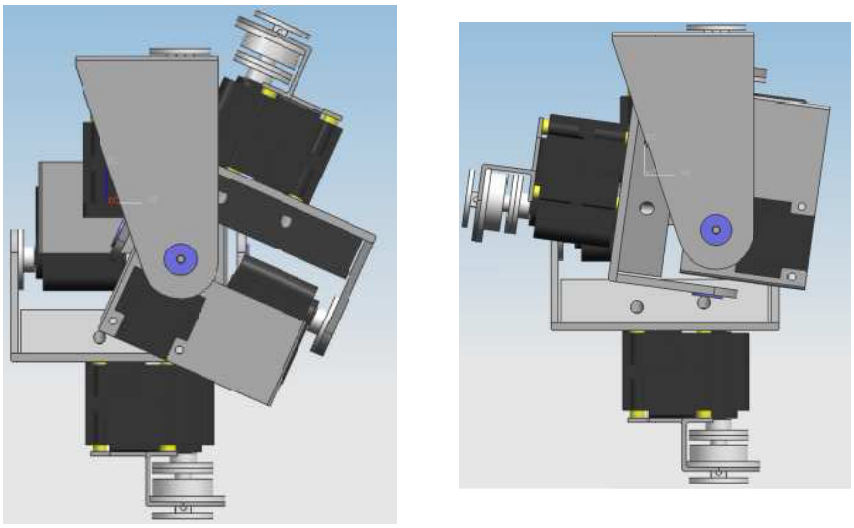


(e) Side view showing nominal position



(f) Side view showing foot with potential coverings

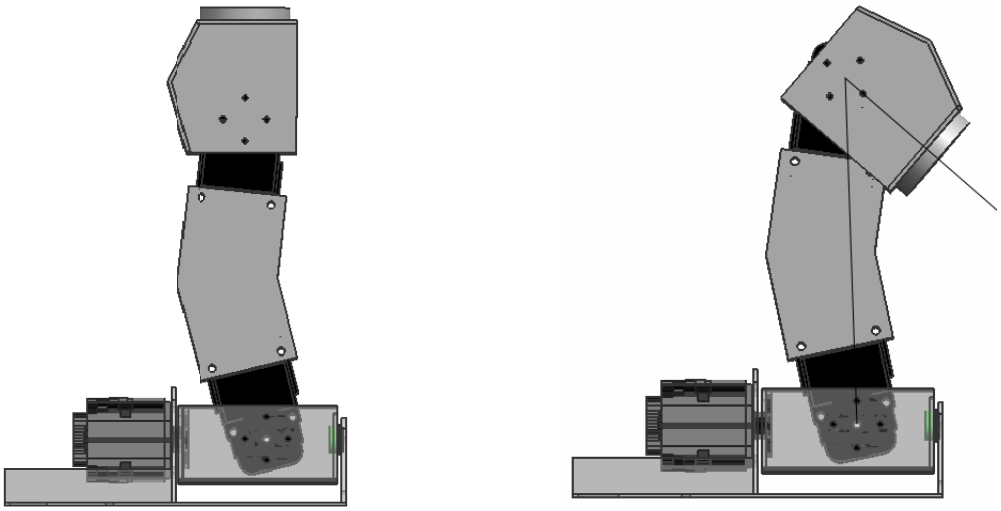
Figure 9.12: CAD models of DARwIn I's feet



(a) Right leg extended up towards chest

(b) Right leg extended back toward rear

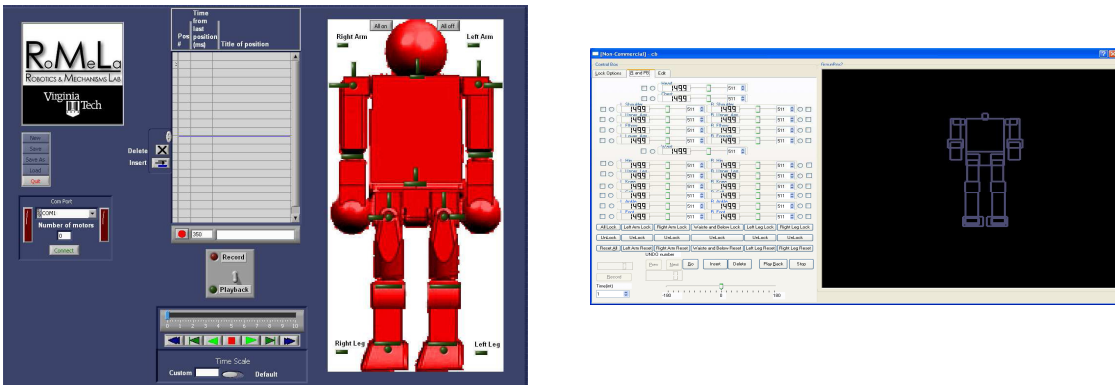
Figure 9.13: CAD model showing DARwIn I's range of motion for the hips



(a) Nominal knee position

(b) Knee fully bent

Figure 9.14: CAD models showing DARwIn I's range of motion for the knee



(a) LabVIEW interface

(b) C++ QT interface

Figure 9.15: Software approaches for controlling DARwIn I's motors

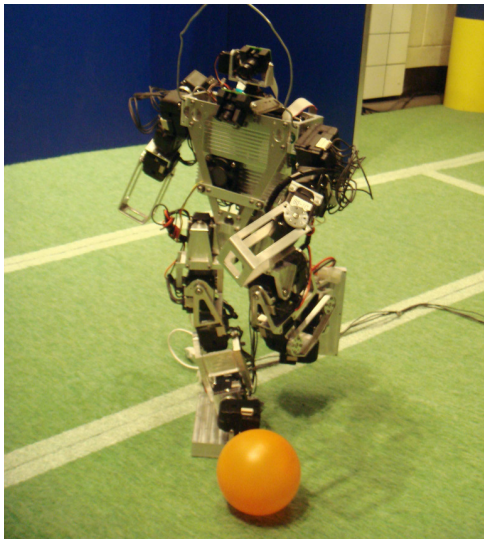


Figure 9.16: Photo of DARwIn IIA

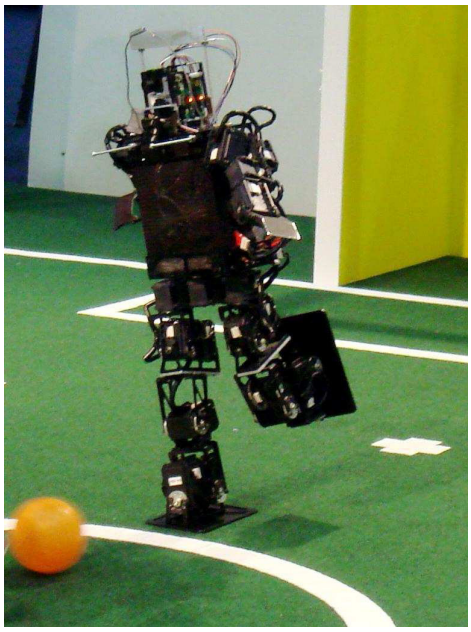


Figure 9.17: Photo of DARwIn IIB at RoboCup 2007, Atlanta

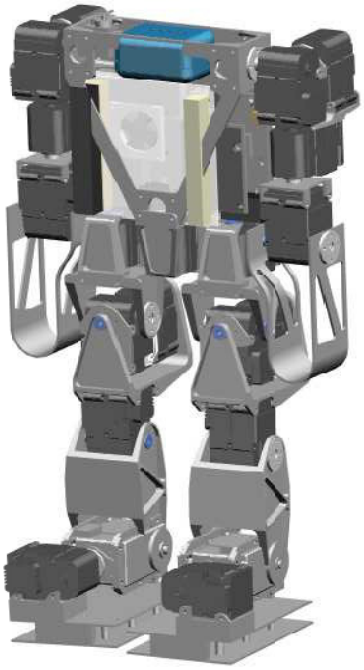
9.4.1 Mechanical Design

Figures 9.18 and 9.19 show a collection of CAD models of DARwIn IIa that detail its different parts. Some notable features are:

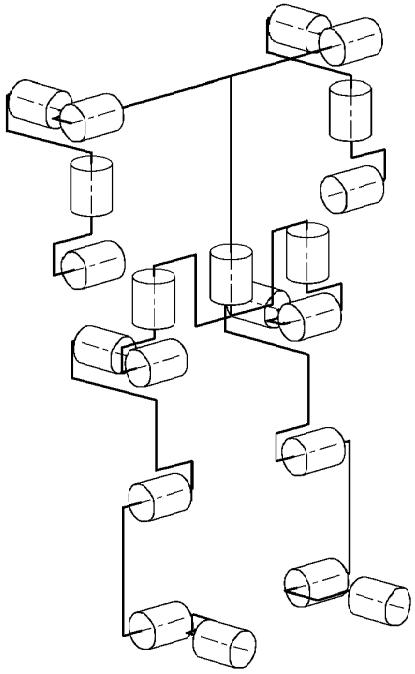
- Lithium polymer batteries housed feet
- Yawing waist motor
- 4 DOF in the amrs
- 2 DOF in the head, using Futaba servo motors
- All DX117 servo motors except for the head

9.4.2 Electronics

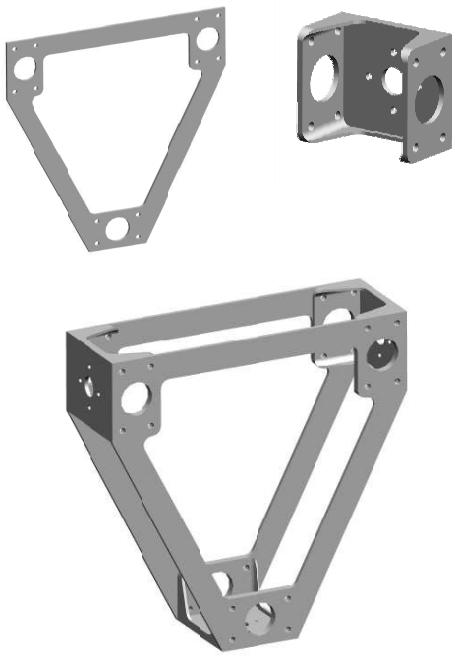
In addition to the CAD design of the robot, the added electronics to DARwIn IIa not only gave autonomy, but also added physical complexity. Figure 9.20 shows some of the diagrams of DARwIn IIa's electronics. DARwIn IIa used a 1.4 Ghz processor on a PC104+ stack from ArborUSA with an additional IEEE 1394 firewire expansion card. The IMU used was



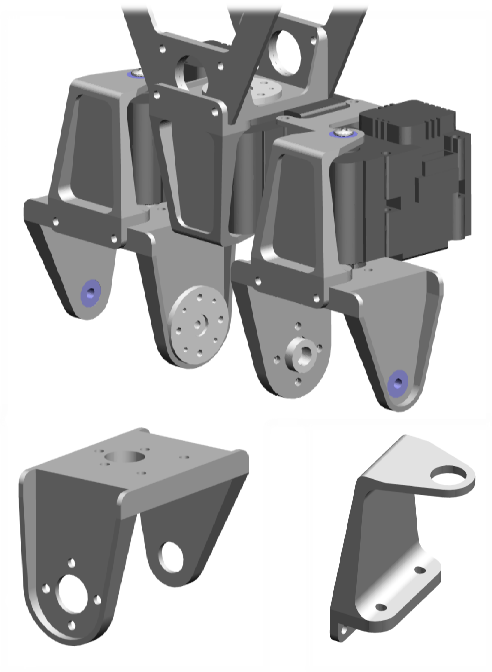
(a) Final CAD model of DARwIn IIa



(b) Final kinematic diagram of DARwIn IIa

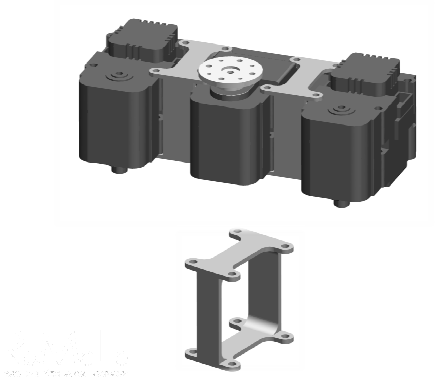


(c) Explosion of DARwIn IIa's chest

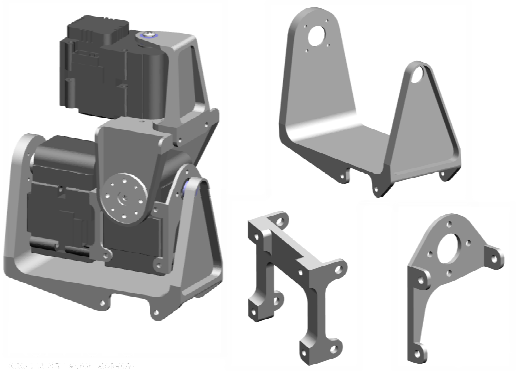


(d) Close up of DARwin IIa's hips

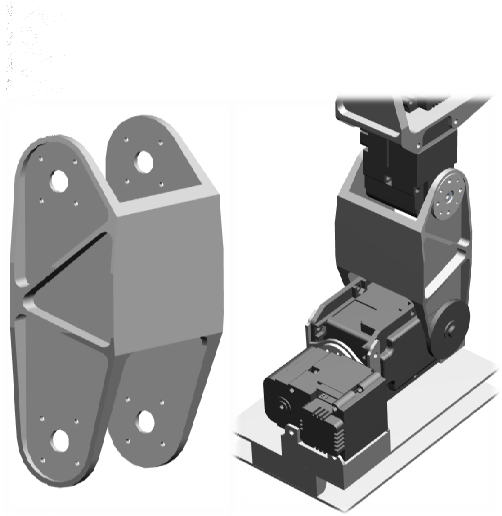
Figure 9.18: Collection of CAD models of DARwIn IIa



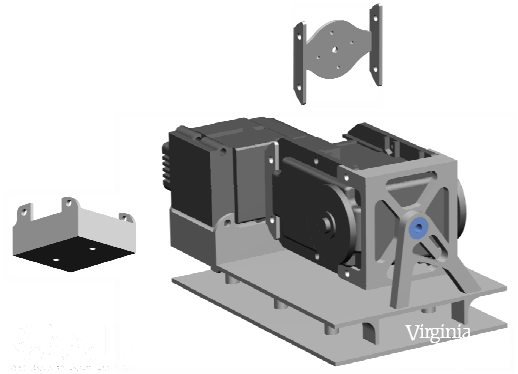
(a) Closeup of DARwIn IIa's waist/hip block



(b) Closeup of DARwIn IIa's hip assembly



(c) Closeup of DARwIn IIa's knee and shin



(d) Closeup of DARwIn IIa's foot

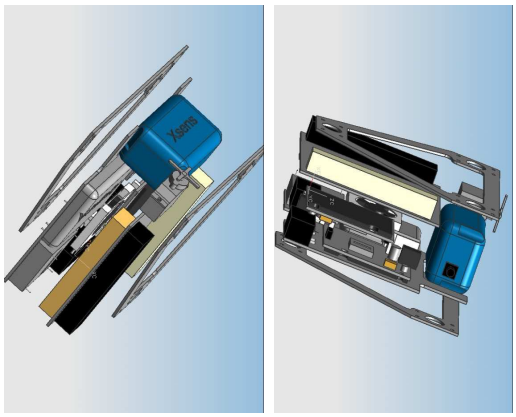
Figure 9.19: Collection of CAD models of DARwIn IIa's lower body

MicroStrain's 3D-GX. The robot also was used 802.11 Wi-Fi, was capable of battery or external power, used 2 Lithium Polymer batteries, and also used a MicroPSU for powering the computer. The XSens rate gyro and accelerometer did not work with the robot because the software used to control the robot was LabVIEW RT, which does not run the Active X controls needed. The Unibrain Fire-i cameras had seperate CCD boards, which allowed for some liberty in placing the cameras in the robot's head.

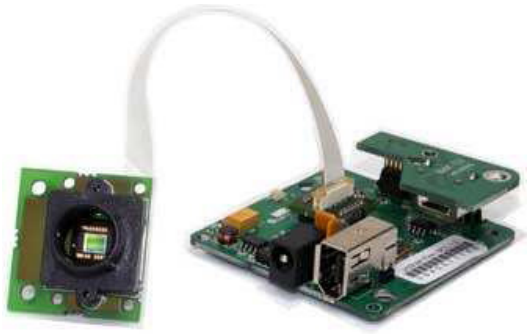
In addition to its improved mechanical design, DARwIn II a/b also has added intelligence to meet the research demands and to allow it to perform higher level tasks, like playing autonomous soccer. DARwIn II a/b's electronics provide power management, a computing architecture, and a sensing scheme aimed at providing information on salient environmental features. DARwIn's power is provided by two 8.2V (nominal) lithium polymer batteries, usually attached to the lower body (legs or feet) to keep the robot's center of gravity below its waist. These batteries provide 2.1 Ah, which gives DARwIn a little over 15 minutes of run time. The power circuit provides 3.3V, 5V, and 12V for the various digital electronics within DARwIn. However, the joint actuators, Robotis Dynamixel motors, are run directly off battery power, which drops from 16.4V to 14.8V during runtime. In addition to providing power to DARwIn's main systems, the power electronics allow for an external power connection and a seamless switch between power sources. Additionally, this circuit prevents reverse polarity, overvoltage, over-current, and under-voltage conditions from damaging the computing, sensing, and actuation components. DARwIn's computing architecture is setup to use a centralized control scheme, which is run on a PC104+ computer with a 1.4GHz Pentium M processor, 1 GB of RAM, compact flash drive for storage, IEEE 1394 card, serial communication, USB, Ethernet, and IEEE802.11 for wireless communication. The operating system is LabVIEW Real-Time [58]. DARwIn also has two IEEE 1394 (Firewire) cameras and a 6 axis rate gyro/accelerometer (IMU) for vision and localization. The cameras capture 15 frames per second at 640 x 480 resolution and 30 frames per second at 320 x 240 resolution RGB. The cameras are attached to a pan and tilt unit, which allows the robot to better look at its surroundings. Two lithium polymer batteries in the feet allow the robot to be powered autonomously.

9.4.3 Software

For high level behaviors, such as playing autonomous soccer, both versions of DARwIn II use a similar software architecture, which utilizes a behavior-based control scheme [59]. Reactive based control has the distinct advantage of being simple and robust. Figure 9.21 shows the simplified software flow diagram used for RoboCup 2007. Raw sensor data is processed into meaningful information, which gives the robot ball position, goal position, opponent positions, and orientation [60]. This information is used by the individual behaviors to dictate their respective actions. The necessary behaviors for any given situation are determined using a Hierarchical State Machine. If in any given situation, two competing behaviors are chosen, then the integrator is used for arbitration. The motion control module

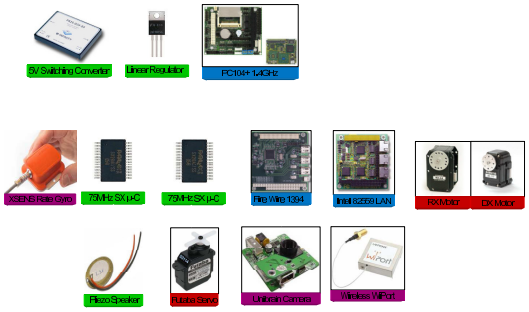


(a) Electronics layout in chest of robot



(b) Unibrain Fire-i IEEE 1394 camera used for vision

(c) PC104+ from ArborUSA used for onboard computing



(d) Electronic components used for power management and sensors

Figure 9.20: Figures of DARwIn IIa's electronics

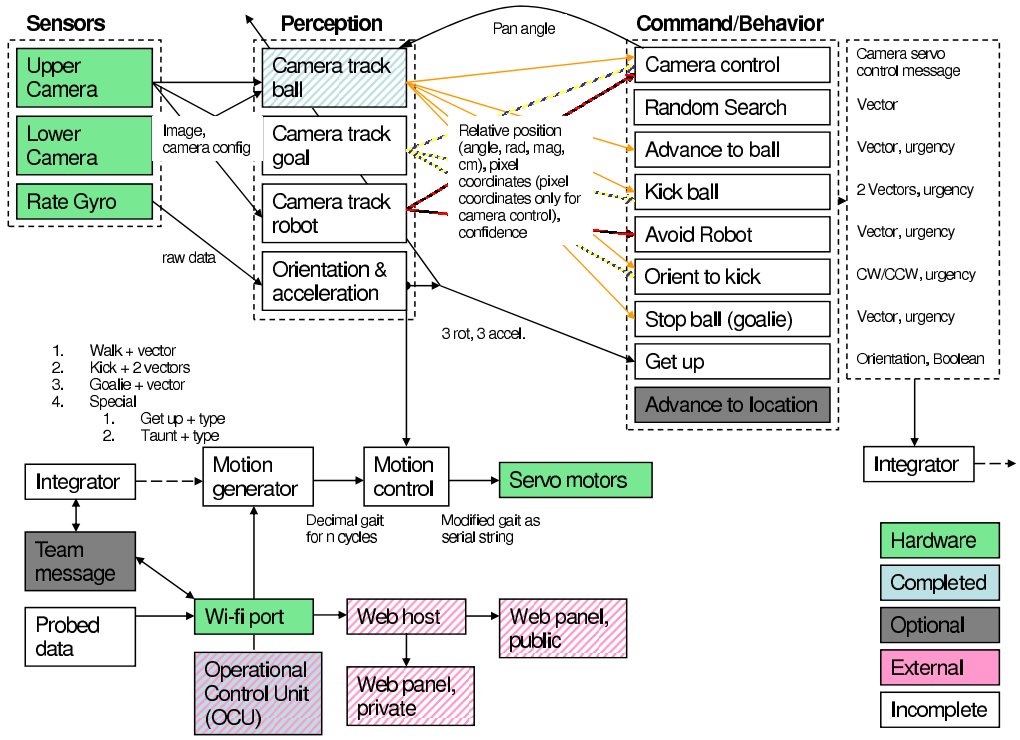


Figure 9.21: DARwIn II software diagram

receives higher level walking commands, head motion commands, and special action kicking or diving commands. The motion generator creates the necessary motion to perform these commands while using orientation information to correct and stabilize the bipedal walking gait. For inter-process communication, DARwIn's software components comply with the SAE AS-4 JAUS (Joint Architecture for Unmanned Systems) protocol. The individual modules of Perception, Behavior Control, Motion Control, and Game Control are now implemented as JAUS components with all interactions between modules occurring via JAUS messages. This peer-to-peer, modular, implementation for inter-process communications allows for automated dynamic configuration and the ability for each software component to run on any computing node on the network.

9.4.4 Additional research

Two additional projects pursued for DARwIn II were collaborating with the industrial design team to come up with a covering for DARwIn (Fig. 9.22) and investigating methods for

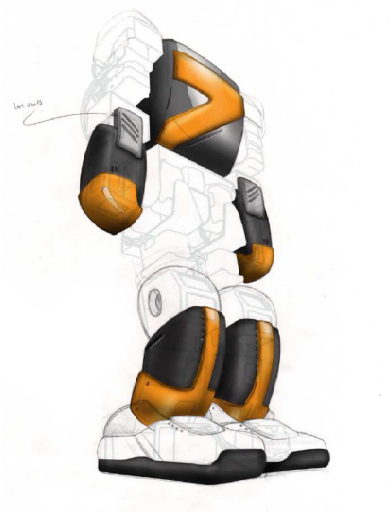


Figure 9.22: Industrial Design concept for coverings for DARwIn II

measuring the robot's moment of inertia. One of the senior design students constructed a device to measure the moment of inertia of an object as seen in Fig. 9.23. The principle of the device was to measure the change in the period of oscillation of the supported platform and use the value to calculate the object's moment of inertia. The device did not have exceptional repeatability or accuracy, but still proved to be an interesting avenue to investigate. Since the devices design, CAD has been used to subsequently supplement estimates of the moment of inertia. Additionally, the moment of inertia does not really come in to play in the dynamics of the robot when walking or performing most actions.

9.5 DARwIn III

DARwIn III was an attempt to perfect the designs before it. Successful in many ways, DARwIn III improved in a few key areas.

9.5.1 Mechanical Design

The number of DOF in each arm was reduced from 4 to 3 in order to reduce weight. The head of the robot was reduced to one DOF in order to comply with new RoboCup rules and regulations. RX-64 motors were used for almost the entire lower body in order to gain more power. RX-28 motors were used for the yawing motor in the hip to reduce the height of the robot. An RX-64 motor was used in the waist to add strength. Additional, the axis of rotation was changed from yawing to pitching in order to allow the robot to make use of

Figure 9.23: Picture of device used to measure the moment of inertia

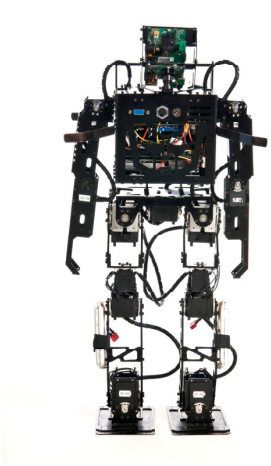
the waist for getting up and stabilizing walking motion. Figure 9.24 shows some views of DARwIn III in its completed state.

9.5.2 Electronics

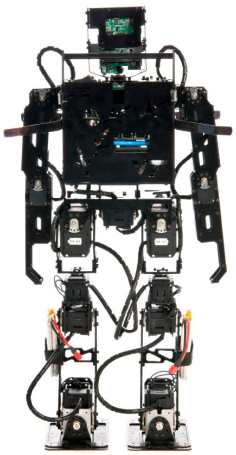
As a quick summary for some of the changes made in the electronics for DARwIn III

- Separate power and computer switches added
- Monitor and keyboard input on front of robot
- Custom circuit board created for integrating electronics and power for gyro, wireless, computer power, motor power, external power, battery power, over-voltage protection, under-voltage protection, and over current protection
- New IMU used from US Digital
- The CCD of the Unibrain Fire-i camera were attached to the board to make mounting to the robot head easier

DARwIn III experienced some power problems. Two batteries seemed to be to low a voltage because of the current draw. Though the nominal voltage may have been 14.8 volts, in actuality, the voltage dropped down to about 12 under load. Using three batteries caused motors to die and burn up. Additionally, using three batteries caused some heating problems for the RX-28 motors. Future versions may want to consider using a voltage regulator or 5 lithium polymer cells.

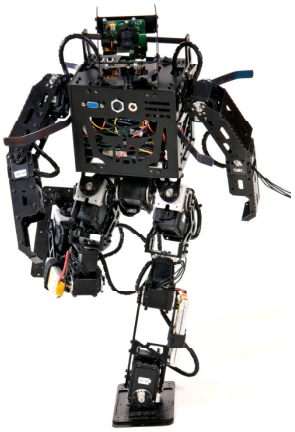


(a) Front View

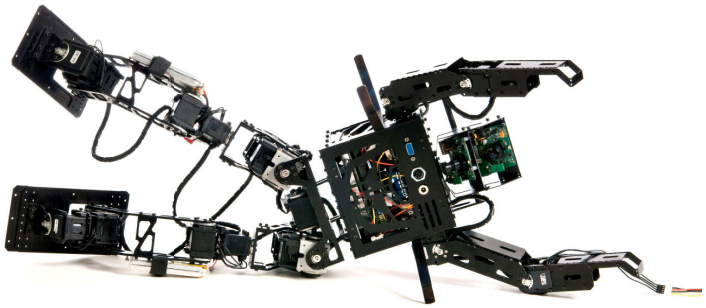


(b) Back View

(c) Left View



(d) Kicking



(e) Diving

Figure 9.24: Orthogonal views of DARwIn III

9.5.3 Software

The software created was modular, using JAUS messaging protocol to communicate between loop types. Additionally, the University of Darmstadt, Germany supplied RoMeLa with a RoboCup simulator used for simulating autonomous soccer game play. The simulator does not take into account true dynamics, but will allow the robot to walk around the field and kick the ball. Additionally, the simulator provides visual feedback of what the robot sees on the field while playing.

9.6 Senior Design and RoboCup

The undergraduate senior design project is a course required at the Virginia Tech Mechanical Engineering Department that lasts for two semesters and builds on all prior education while adding real-life experience. Among the many different senior design projects, this paper will present the Dynamic Anthropomorphic Robot with Intelligence (DARwIn) project; participating are approximately 8-10 seniors, 2-3 additional undergraduates, 2-3 graduate students, and 1-2 faculty. Currently in its third year, the DARwIn project has evolved in organization, management, and educational value along with evolving with DARwIn's design [61].

In its third year, the undergraduate design team was still coming up with innovative designs for DARwIn. Much of this innovation was fostered by an excellent management structure that was adopted from a commercial product development project manager. Overall, the process takes more paper work and bureaucracy, but it improves efficiency by ensuring proper design, thought, and consideration up front. After a couple weeks of bringing the undergraduates up to speed on the current status of the project, the undergraduates, graduates, and faculty came up with a list of items the project should produce. Each of the seniors created a "Specification" that details each what the problem is for each item, its solution, its alternative solutions, and how the item impacts other people on the team. By writing a "Specification", the students learn more about the problem they are solving and have a better idea of how to solve it before they begin design work. The group is still divided into the same sub-teams-each with a leader. The overall undergraduate group also elects an overall leader. In addition to the undergraduates' organizational structure, the graduates have also naturally organized. Now with three graduate students, there is an overall project leader and then "experts in their field," who handle items pertaining to the project that may be too complex for the undergraduates or pertain to the graduate's research; the fields being software architecture, electronics architecture, and walking behaviors. Overall, the tiered organization divides work fairly among the undergraduates and graduate students working on the project. With the help of a new organizational structure, the project looks to further improve on the successful designs of the previous versions with DARwIn III. Improvements are being made in computing power, software architecture, vision routines, walking gaits, stability control, and mechanical design.

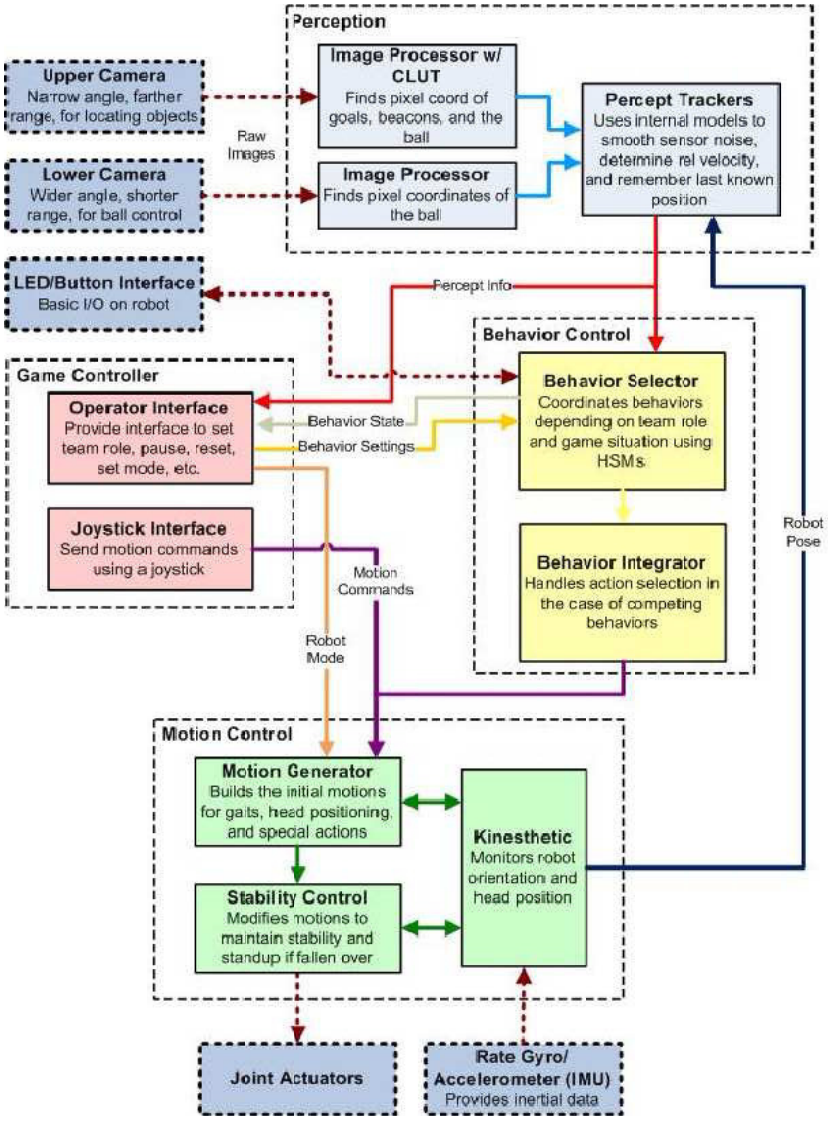


Figure 9.25: DARwIn II software diagram

9.6.1 Conclusions, observations, and lessons learned

Combining graduate research with an undergraduate senior design project on humanoid robotics seemed to work well to produce a sophisticated humanoid robot research platform and to foster a good research environment. Having the graduate student's research partially depend on the success of the senior design project gives the graduate an obvious motivation to help the seniors as much as possible to ensure their success. The graduate student also benefits greatly by saving time on designing and fabricating a sophisticated research platform. Having an international competition for the project-especially one that is internationally popular-helps to motivate the undergraduate students and also brings public attention and media, which can lead to additional funding for the project. An organizational structure is extremely desirable to have in place to make the design process efficient. Having an experienced member (either faculty or graduate student) heavily involved in the project also seems to be very important in achieving excellent results in design. The international competition is a fantastic motivator, but too much stress on the competition tends to distract the team from the real goal of the project, which is to create a research platform. Too much stress on competition can also be very disappointing if the competition results are not as expected. In any case, the competitions give valuable experience and design ideas, which can be used for future projects.

9.7 RoboCup-description

In addition to serving as research platform, DARwIn also served as the first and only US humanoid entry to qualify for the international autonomous soccer competition, RoboCup 2007 [62]. RoboCup is a soccer competition between autonomous robots. The program's goal is by 2050 to have fostered a team of humanoid robots capable of defeating the human World Cup champions in soccer. Started in 1997, RoboCup did not introduce the humanoid league until 2001. The humanoid competition requires that the robots be fully autonomous-all computing, sensing, and power must be onboard the robot. RoboCup is a challenging and exciting arena for humanoid robotics. RoboCup brought a lot of attention and excitement to the senior design project. The idea of going to competition served as a motivator in getting the students to work harder. The competition also served as a benchmark to compare DARwIn against. Using ideas from other robot designs and software, DARwIn's design continues to improve. Being the only US team in the humanoid division brought a lot of attention to DARwIn, resulting in a lot of publicity and press, which is good for fund raising [59, 63–67]. DARwIn won many other awards including:NI Week Best Application of Virtual Instrumentation overall (2007), AAAI Technical Innovation award (2007), and 2nd Place in the ASME international Student Mechanism Design Competition (2006).

Chapter 10

Software Contributions

10.1 Vision software

The vision software is critical to adding autonomy to DARwIn, allowing the robot to perceive its environment. The results of the vision software include information like object locations, object sizes, object classification, etc. For soccer playing, the vision information includes ball location, goal location, beacon locations, obstacle locations, etc. The ability for the robot be able to recognize the various objects and then accurately locate them is essential for being able to successfully play soccer autonomously. Other applications of the vision software include interaction with people. The ability to read signs and symbols allows for people to interact with the robot as well as providing an additional channel of communication. This section will detail the various pieces of the vision algorithms used in the DARwIn robot that allow it to play autonomous soccer and interact with people. All of the software used for vision processing was created using LabVIEW.

10.1.1 Cameras

DARwIn's cameras have thus far been two IEEE 1394 Unibrain Fire-i cameras. One fixed camera looks at the feet of the robot while the other camera is able to pan and/or tilt for looking around. In DARwIn III, both cameras were fixed to a pan unit so as to follow RoboCup 2008 rules and regulations. Figure 10.1 shows a diagram of the field of view the two cameras provided on DARwIn III.

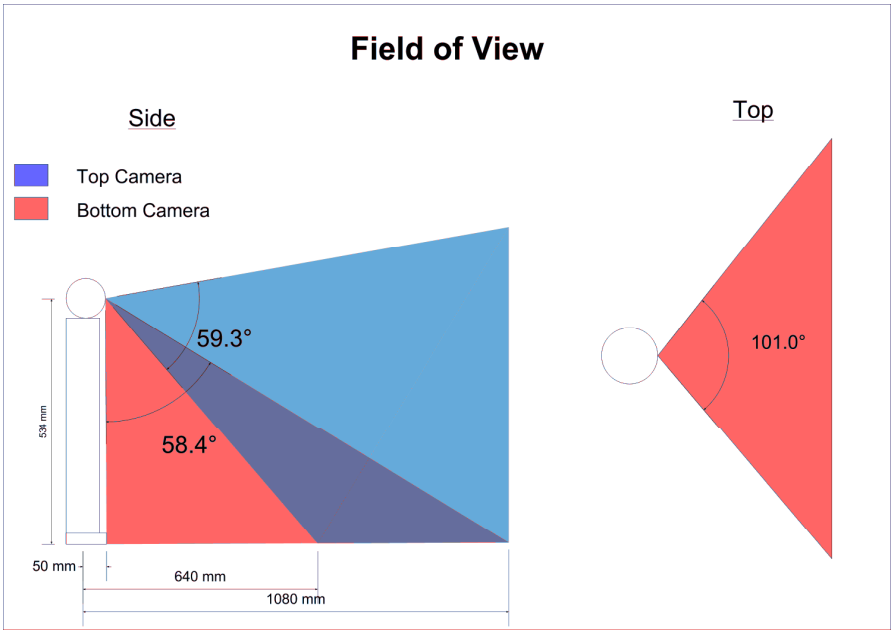
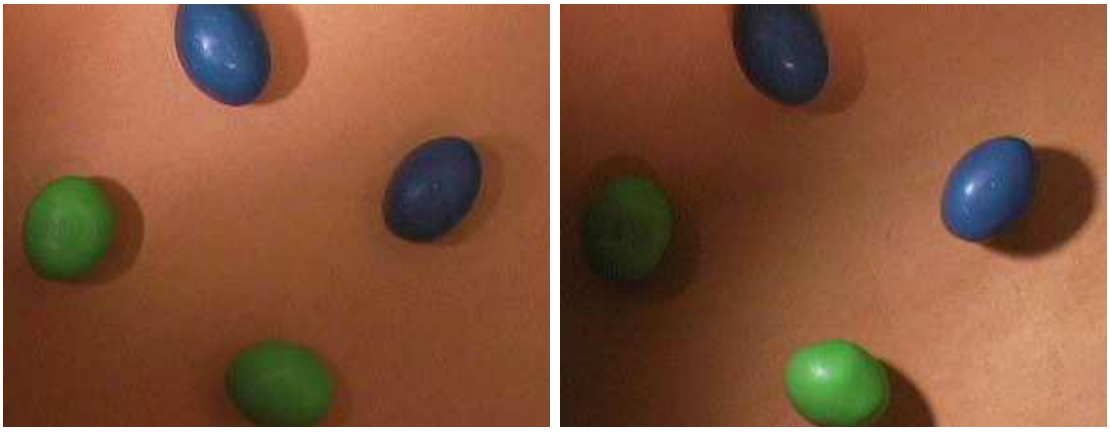


Figure 10.1: Field of View (FOV) Diagram



(a) Lighting condition 1 (b) Lighting condition 2

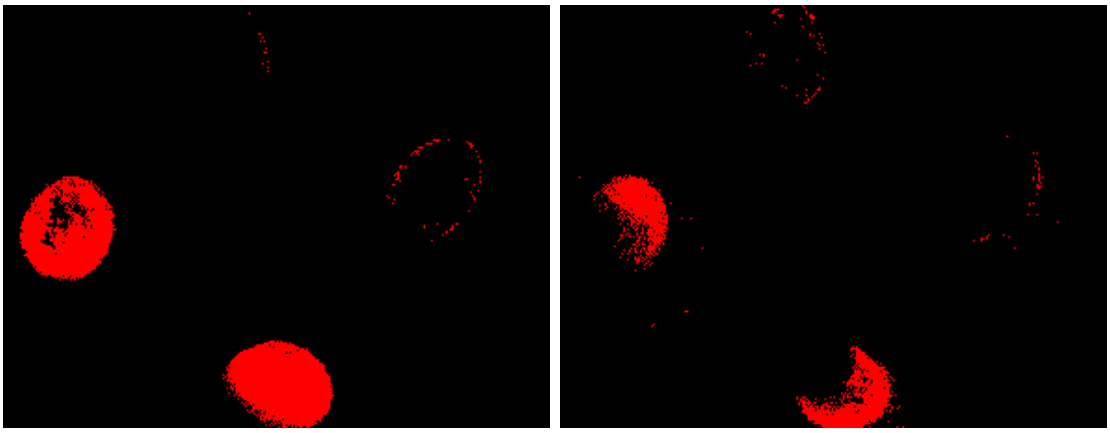
Figure 10.2: Original images of eggs to be processed

10.1.2 Thresholding

Simple techniques are sometimes the most useful and effective ways of processing images. Thresholding is a process of examining each color plane and determining if the pixel values fall within a certain range. For example, Figure 10.2(a) shows an image of colored eggs to be processed. Only the pixels that fall within a range defined as "green" pass through the filter as seen in Fig. 10.3(a).

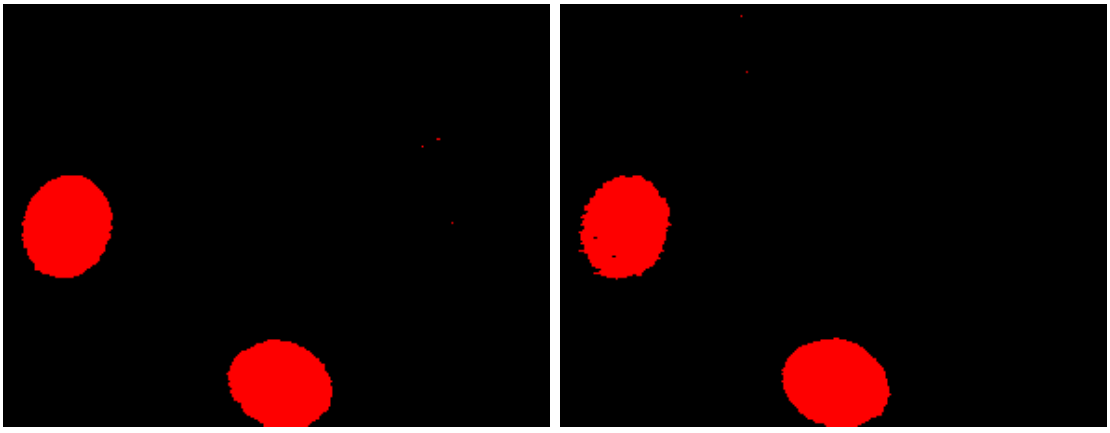
Using the RGB color planes is not very robust against changes in lighting conditions. Figure 10.2 shows two different lighting conditions for the same objects. Using the same thresholding values for filtering each image, Fig. 10.3 shows how significant information can be lost. Both green eggs in Fig. 10.3(b) do not appear as completely as in Fig. 10.3(a). Thresholding using the Hue, Saturation, and Luminescence (HSL) color spectrum is more robust to lighting conditions. The hue specificizes the color's wave length; the saturation is a measure of how much of the color is present; and the luminescence is a measure of how bright the color is. Typically, the saturation and, even more so, the hue are not sensitive to changes in lighting conditions. The luminescence is the most sensitive to lighting conditions since the luminescence is a measure of brightness. Figure 10.4 shows the result of HSL thresholding using the same images used as an example for RGB color thresholding. As seen, for both lighting conditions, the green eggs not only pass through the filter, but almost the entire egg passes through whereas portions of the eggs do not pass through using the RGB threshold.

Although the HSL color planes are much better for image thresholding techniques, converting non-native HSL images, such as RGB,to HSL is relatively processor intensive. This conversion takes 46 times longer than the threshold process itself (threshold duration of 0.9ms and



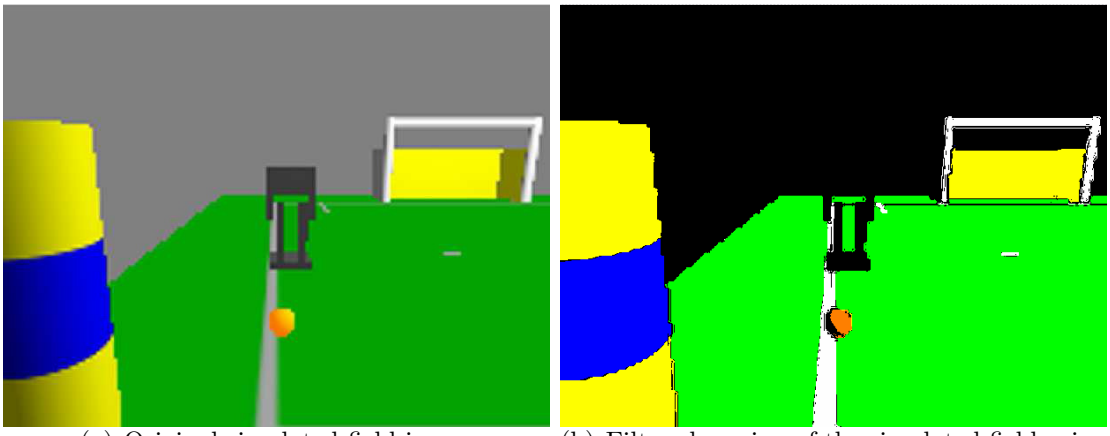
(a) Lighting condition 1 (b) Lighting condition 2

Figure 10.3: The result of thresholding 10.2 in the RGB color spectrum



(a) Lighting condition 1 (b) Lighting condition 2

Figure 10.4: The result of thresholding 10.2 in the HSL color spectrum



(a) Original simulated field image

(b) Filtered version of the simulated field using a CLUT

Figure 10.5: Applying a CLUT to a simulated field

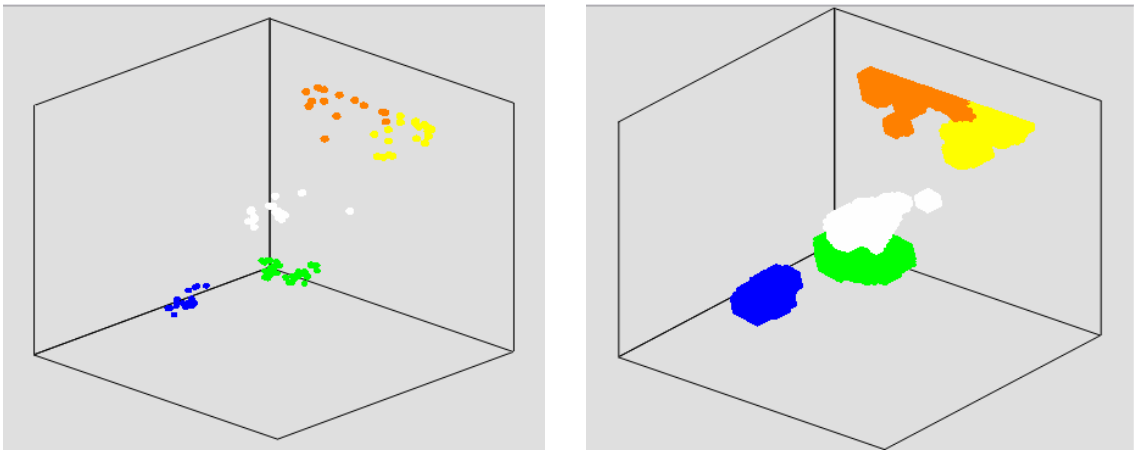
HSL conversion of 41.5ms). Because of the long conversion time, the HSL color thresholding should not be used in situations when processor time is limited and the native color spectrum of the images are not HSL.

10.1.3 Color Look-up Table

Applying a Color Look-up Table (CLUT) is a simple and computationally fast way to process color images. The CLUT has a value (either RGB, grayscale or binary) for every color spectrum value. Typically for RGB images, three 8 bit numbers (one byte for each color plane) make up a pixel. 256 different values for each color plane of a pixel creates a total of 16,777,216 combinations, each requiring a corresponding conversion value. The the CLUT is applied to an image, each pixel's RGB value is found in the CLUT and the corresponding conversion value replaces the pixel. For example, the RGB pixel value of (200,200,200) may have a conversion value of (0) for a grayscale image. All other RGB pixel values may have a conversion value of (255). Whenever the pixel value of (200,200,200) occurs, it is replaced with a black pixel; all other colors are replaced with white.

Converting the color image is very useful for color classification, which then makes object identification much simpler. Figure 10.5 shows the original and filtered image of a simulated field. The background and the opposing robot are both filtered out and ignored, but all the other objects are clearly identified and have a uniform shade of color.

Applying the CLUT takes a relatively short amount of processing time (16ms versus 30ms for converting from RGB to HSL). However, creating the CLUT takes a much longer amount



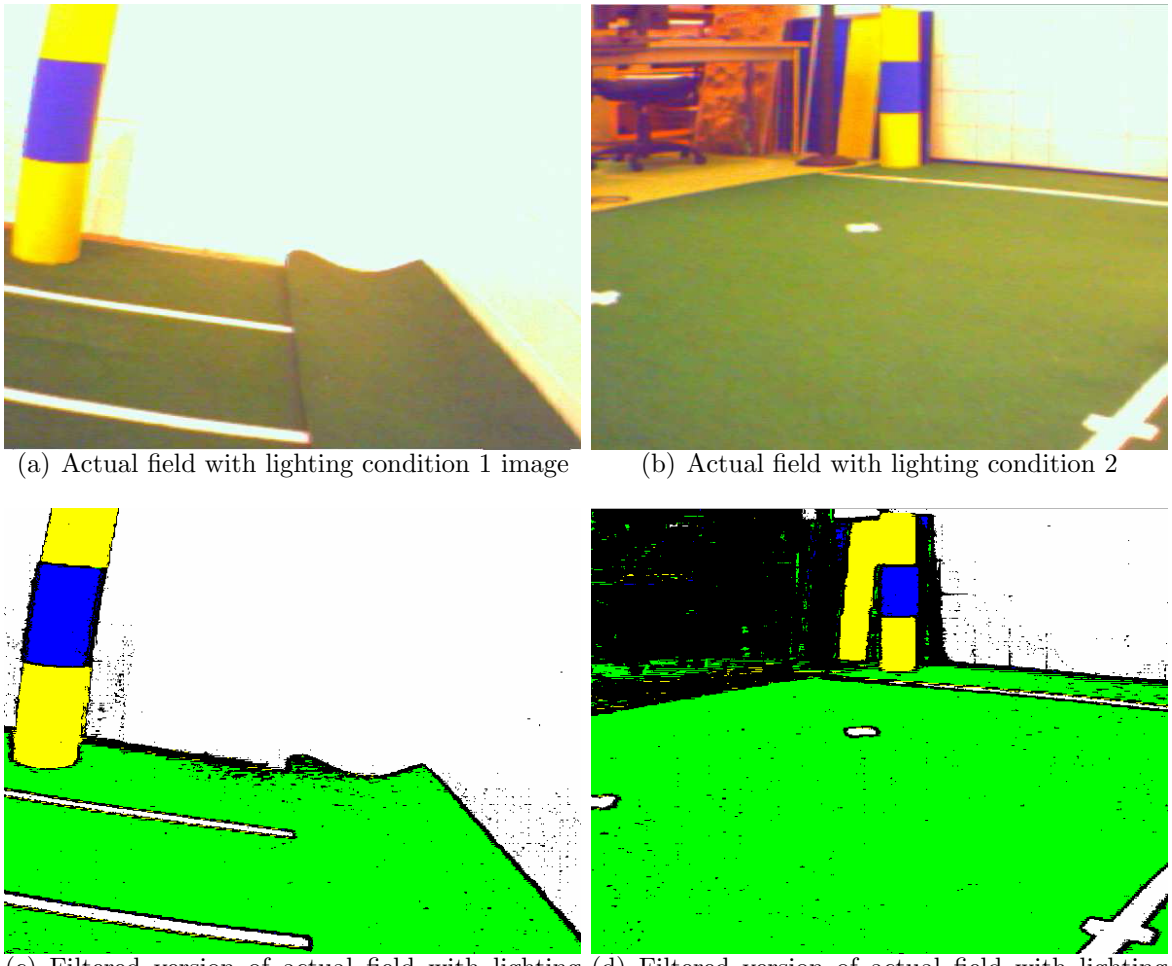
(a) CLUT formed by just a handful of points in color space

(b) CLUT generalized

Figure 10.6: Plots showing how generalizing a CLUT using exponential functions results in a more inclusive color space classificatoin

of time (anywhere from a minute to hours). In theory, every color could be mapped on the CLUT individually with a corresponding conversion. Mapping 17 million colors by hand in impractical. Instead, by picking and classifying a small set of colors (on the order of 100), and then generating a CLUT based on the small set, the CLUT can be practically implemented by individual selection of colors. The RGB color spectrum can be represented as a 3D color space, where the Red, Green, and Blue values are (x,y,z) coordinates. Using exponential functions, the CLUT can be generalized starting with only 100 points to create a CLUT made of equivalently 100,000 points [68]. Each point of the small subset of points radiates its color type as an influence on the entire colorspace. Each point in color space is then classified as its new color type based on which colortype has the most influence for that particular point in color space. Since the influence every point in the small subset has to be calculated for all other 16.7 million points, generalizing the CLUT can take a long time depending on how large the subset of points is. Even though generalizing the CLUT can take anywhere from 5 seconds to 2 minutes, applying the CLUT takes the same amount of time (around 16ms). The process of generalizing the CLUT is taken from [68] and can be found in D.1.

The CLUT although slower in application than thresholding, yields much more robust and informative results. A single thresholding operation results in only a single piece of color classification, while the CLUT results in up to 16.8 million classifications in a single operation. The CLUT almost always outperforms thresholding in the RGB color spectrum when subjected to varying lighting conditions. Figure 10.7 shows two different images with different lighting conditions that are filtered using the same CLUT. The field is much darker in Fig. 10.7(a) than in Fig. 10.7(b), yet the filtering using the CLUT shows very similar



(a) Actual field with lighting condition 1 image

(b) Actual field with lighting condition 2

(c) Filtered version of actual field with lighting condition 1 using a CLUT

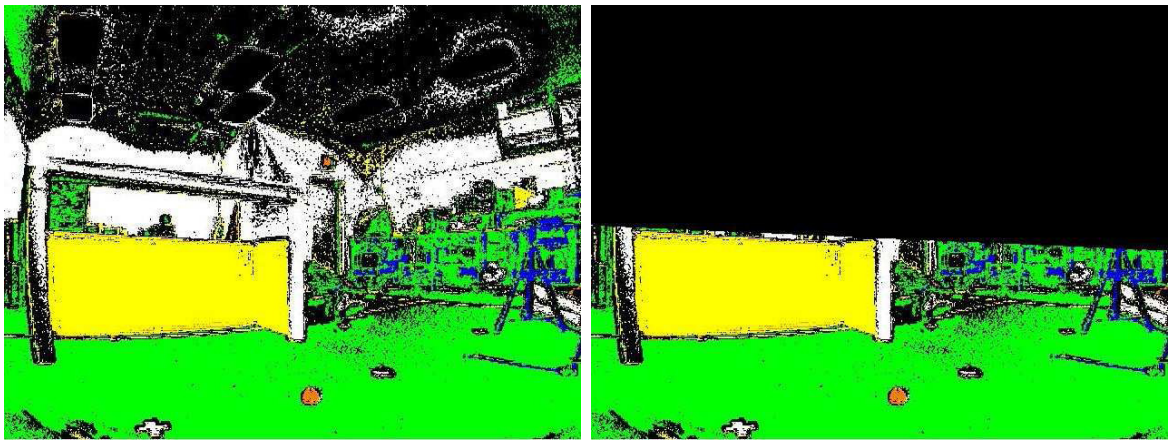
(d) Filtered version of actual field with lighting condition 2 using a CLUT

Figure 10.7: Applying a CLUT to an image of a real field

results.

10.1.4 Horizon calculation and masking

Finding the horizon is very useful in image processing. The slope of the horizon can give information about what objects are perpendicular to the horizon and therefore, are vertical. The horizon is also very useful for ignoring information that could throw off other image processing. For example, Fig. 10.8(a) shows an abstracted image (after going through a CLUT application) that has a lot of useless information (that pertains to the soccer field) in the upper half of the image. Additionally, an exit sign shows up as an orange blob in the upper half of the camera. By using the calculated horizon, all of the erroneous information



(a) Sample filtered image of a soccer field

(b) Image of field with a mask applied above the horizon

Figure 10.8: Example of how a mark was applied to an image to reduce computation based on the horizon.

is ignored and the exit sign will not throw off any image processing designed to locate the ball (Fig. 10.8(b).

Calculating the horizon requires knowledge of how the camera is mounted to the robot and the orientation of the robot's body, which can be calculated from kinematics, a rate gyro, or inertial measurement unit. To find the horizon in the image, we first represent the four corners of the image in space as

$$p_1 = \begin{pmatrix} f/\text{pix} \\ \text{Res}_x/\,2 \\ \text{Res}_y/\,2 \end{pmatrix}, p_2 = \begin{pmatrix} f/\text{pix} \\ -\text{Res}_x/\,2 \\ \text{Res}_y/\,2 \end{pmatrix}, p_3 = \begin{pmatrix} f/\text{pix} \\ \text{Res}_x/\,2 \\ -\text{Res}_y/\,2 \end{pmatrix}, p_4 = \begin{pmatrix} f/\text{pix} \\ -\text{Res}_x/\,2 \\ -\text{Res}_y/\,2 \end{pmatrix} \tag{10.1}$$

where Res_x and Res_y are the x and y resolutions respectively, f is the focal length of the camera, and pix is the size of each pixel. The locations are relative to the focal point of the camera. The entire camera goes through a rotation described by a roll and pitch of the body, a yaw from the panning of the head, and then another y-rotation for the camera's pitch relative to the head. Combining these rotations forms:

$$^{C}_{0}R = R_x(\alpha) \times R_y(\beta) \times R_z(\psi) \times R_y(\phi) \tag{10.2}$$

where α and β are the roll and pitch of the body respectively, ψ is the camera's pan angle, and ϕ is the camera's pitch angle relative to the head. Each corner of the image is then transformed by

$$p' = {}^{C}_{0}R \times p \tag{10.3}$$

where p' is position of the corner in the global coordinate frame. The horizon can be found by the intersection of the horizon plane ($z = 0$) with the lines formed by connecting the top pair to the bottom pair of the corners of the image. The three dimensional lines can be described by a set of parametric equations:

$$x = x_0 + a \cdot t \tag{10.4a}$$
$$y = y_0 + b \cdot t \tag{10.4b}$$
$$z = z_0 + c \cdot t \tag{10.4c}$$

where the coefficients are the change in values for the line. Therefore, using the following notation,

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \boldsymbol{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{10.5}$$

the intersection of the horizon with one of the lines can be found (by setting $z = 0$) as the pixel along the edge of the image that coincides with the horizon, H_{pix} (left edge), can be calculated as

$$h_x = (p'_1\boldsymbol{x}) + \frac{(\boldsymbol{p}'_2\boldsymbol{x} - p'_1\boldsymbol{x})\, p'_1\boldsymbol{z}}{p'_1\boldsymbol{z} - p'_2\boldsymbol{z}} \tag{10.6a}$$
$$h_y = (p'_1\boldsymbol{y}) + \frac{(\boldsymbol{p}'_2\boldsymbol{y} - p'_1\boldsymbol{y})\, p'_1\boldsymbol{z}}{p'_1\boldsymbol{z} - p'_2\boldsymbol{z}} \tag{10.6b}$$
$$H_{\text{pix}} = \sqrt{((p'_1\boldsymbol{y}) - h_y)^{\,2} + ((p'_1\boldsymbol{x}) - h_x)^{\,2} + (p'_1\boldsymbol{z})^{\,2}} \tag{10.6c}$$

where h_x and h_y represent the location of the intersecting point with the horizon and H_{pix} is the pixel location of the intersection along the image edge relative to p'_1. The same equation is used for finding the right pixel location, but uses p'_3 and p'_4 instead of p'_1 and p'_2. The derivation of these equations can be found in Sec. D.2.

The relative yaw, θ, of the camera to the global position of the robot can be calculated as

$$\theta = \sin^{-1}\left(\text{xy} \Big/ \sqrt{1 - \text{xz}^2}\right) \tag{10.7}$$

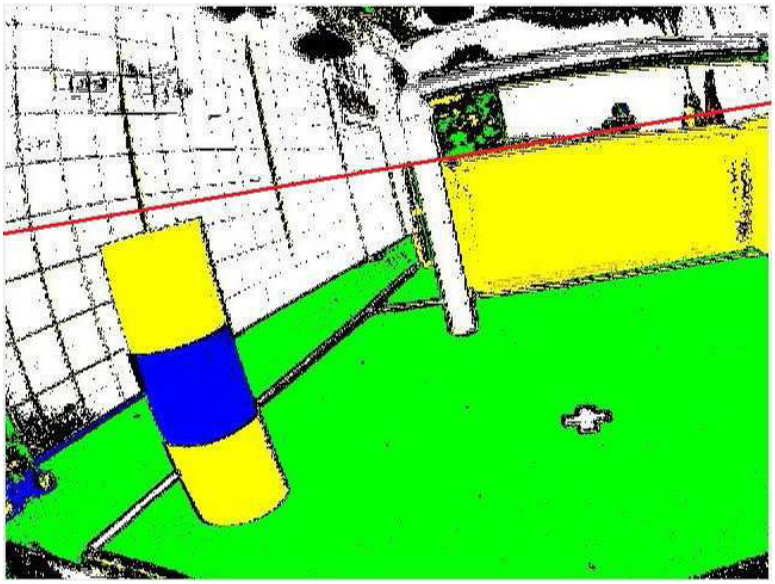


Figure 10.9: Image of field with a line overlay showing the calculated horizon

where xy is the projection of the x-axis of the camera's coordinate frame onto the global y-axis and xz is the projection of the camera's x-axis onto the global z-axis.

Figure 10.9 shows an image the a playing field with the calculated horizon overlaying.

10.1.5 Finding a ball

While competing in RoboCup soccer, one of the most important factors in performance is the robot's ability to locate the ball. The lighting conditions and allowable colors is relatively controlled. The competition has a specific set of colors for the field, field lines, goals, beacons, and balls. Additionally, individual robots must be mostly black/gray and wear a cyan/magenta marker. The ball is the only orange object typically present on the field of play. In RoboCup 2007, the ball was a 8.4cm diameter orange plastic ball, which was very difficult to find and expensive to purchase. In RoboCup 2008, the ball was changed to an orange tennis ball to make it easier for competitors to acquire balls for practicing and vision testing.

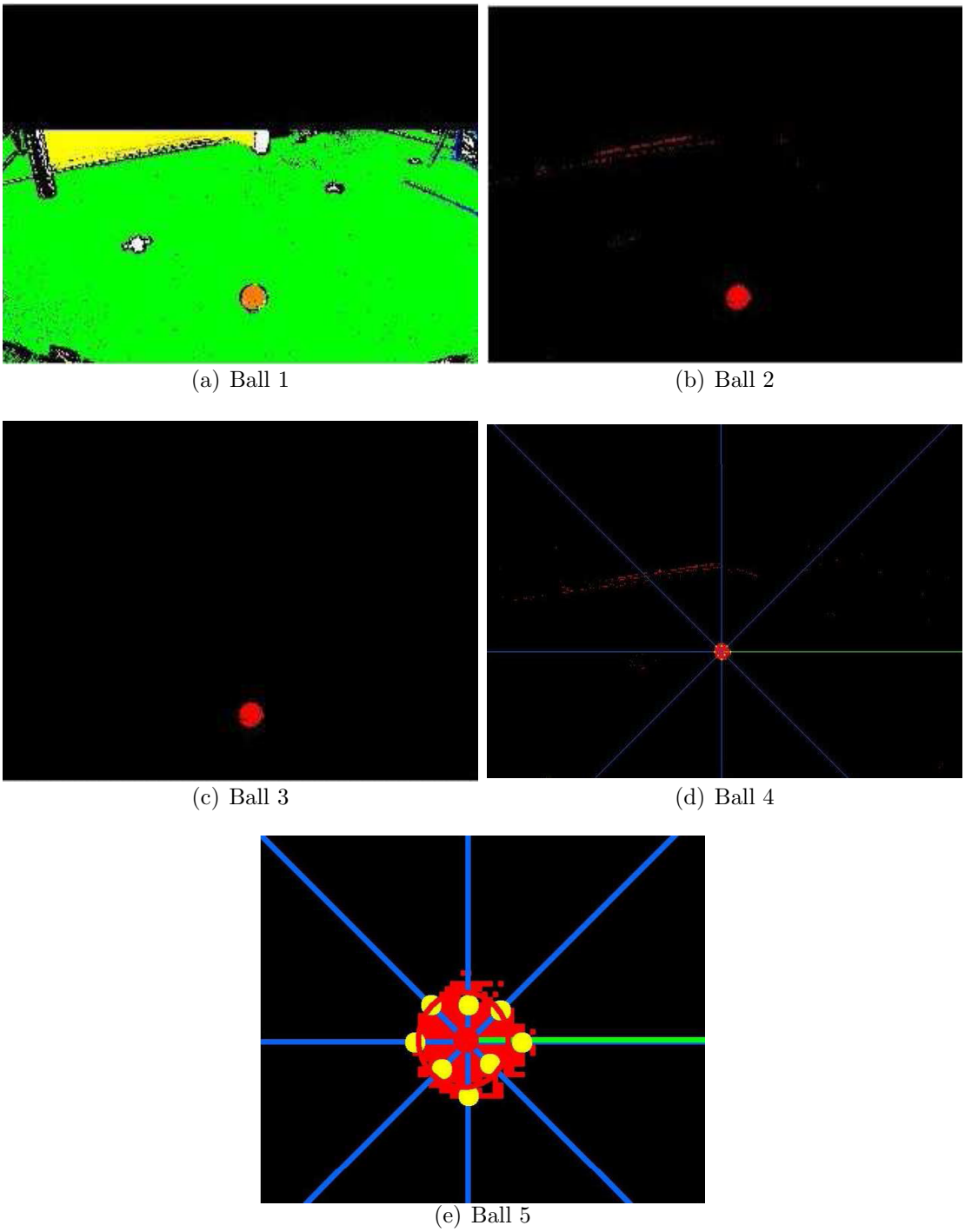
Using either the CLUT or thresholding, everything "orange" in the image is used for finding the ball (Fig. 10.10(a)). Oftentimes, noise or random light effects cause bits of orange to show up in the image even when they do not really exist on the field (Fig. 10.10(b)). Running the image through a particle filter cleans up the image to leave only large areas of "orange" (Fig. 10.10(c)). Typically, the only large area of orange left the image is the ball. Calculating the centroid of the large orange area should lead to the center of the

ball. In some cases, the ball's centroid is adequate for identifying the object's location. In other cases, the ball can be obstructed by other robots, obstacles, or the cameras field of view. When the ball cannot be completely seen, the centroid is used as a starting point for identifying a circle (Fig. 10.10(d)). Lines radiating out from the centroid detect the edges of the ball (Fig. 10.10(e)). The resulting edge points are used to calculate the circle's radius and center. Using this method, even if the ball is obstructed, its center can be accurately calculated.

10.1.6 Finding beacons

In RoboCup 2007, four beacons, 90cm high and 20cm in diameter were used at the corners of the field to aide in robots' ability to localize. RoboCup 2008 reduced the number of beacons to two and the height to 60 cm, and moved the beacons to side lines centering the field's length. The beacons have three bands of color, alternating yellow-blue-yellow or blue-yellow-blue depending on the beacon's position on the field (Fig. 10.11). The beacons serve as a very useful landmark for robot's using a world model, which use landmarks to localize the robot in the world model.

Finding the beacons in an image can be successful by exploiting an unique property–every beacon has two vertical transitions of blue to yellow or visa versa. Nowhere else on the field of play does is this feature seen. Since the blue to yellow and yellow to blue transitions are the most important, and other transitions and colors are less important, the image is converted to a grayscale equivalent where everything blue is represented as white, everything yellow is black, and all other colors are gray. This conversion creates the steepest contrast (black against white) where yellow meets blue (Fig. 10.12). All other edges are white to gray or black to gray. Since every beacon has some portion as yellow, all the yellow objects in the image are identified (Fig. 10.13(a)). Using scan lines perpendicular to the horizon (Fig. 10.13(b)), the type of edge transitions of every yellow beacon band is classified as a beacon edge or an edge to space (not a beacon edge). If the top of a yellow beacon band has beacon edges, while the bottom of a yellow beacon band has edges to space, the yellow band is the lower portion of a yellow-blue-yellow beacon. If the situation is reversed, the yellow band is the upper portion of a yellow-blue-yellow beacon. If both the top and bottom of the band are beacon edges, then the yellow band is the middle portion of a blue-yellow-blue beacon. In this manner, each beacon can be identified and classified. The beacon's height can be measured by starting "far" above or below beacon, and then, while moving closer to the beacon, detecting the gray to black/white transition, which indicates a beacon band (Fig. 10.13(c)).



(a) Ball 1

(b) Ball 2

(c) Ball 3

(d) Ball 4

(e) Ball 5

Figure 10.10: Sequence of image processing to identify an orange ball

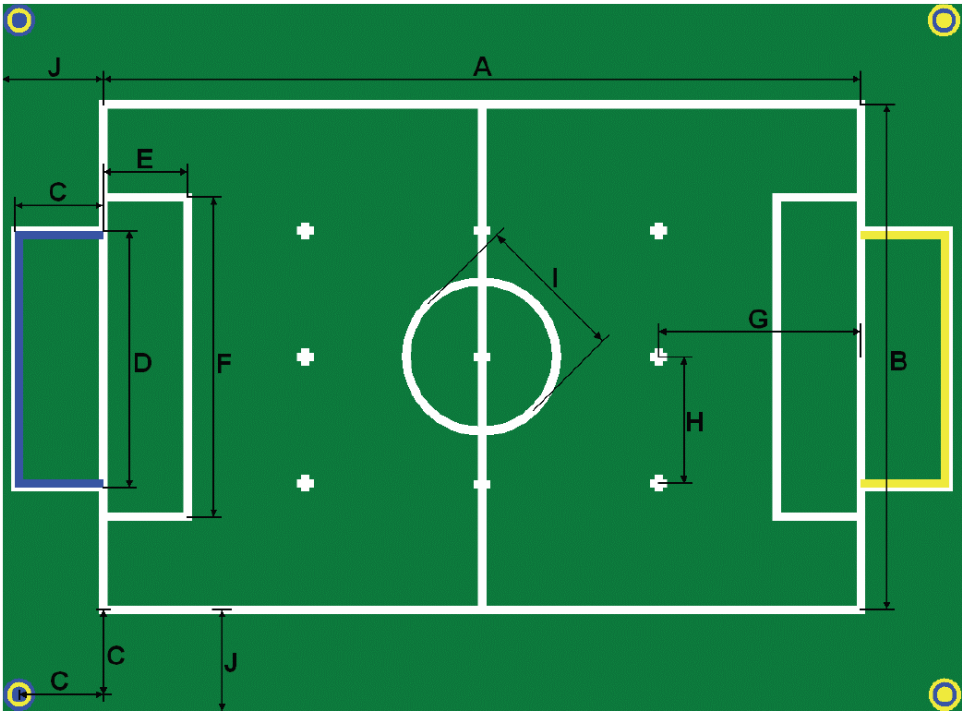


Figure 10.11: Field diagram for the 2007 RoboCup humanoid competition

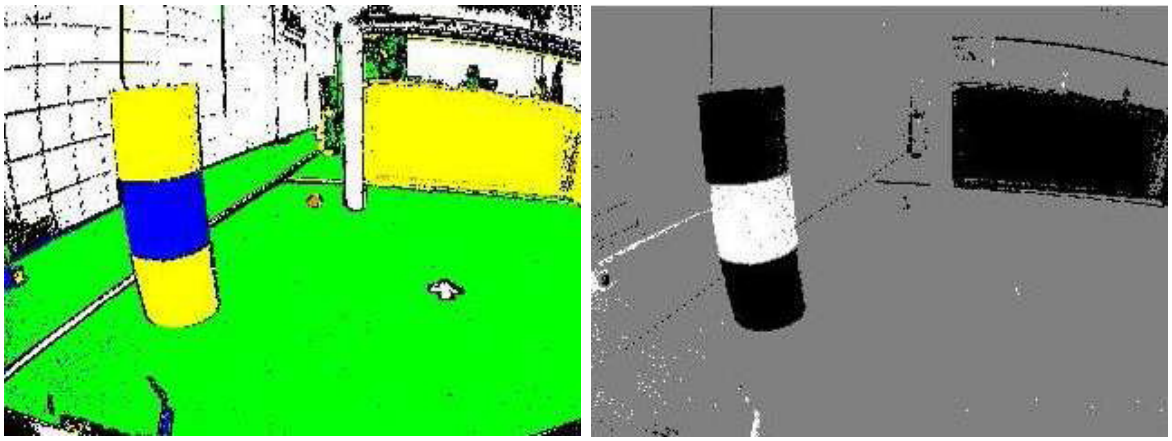
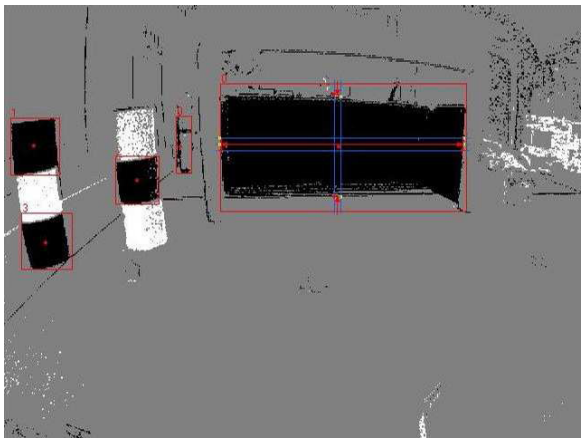
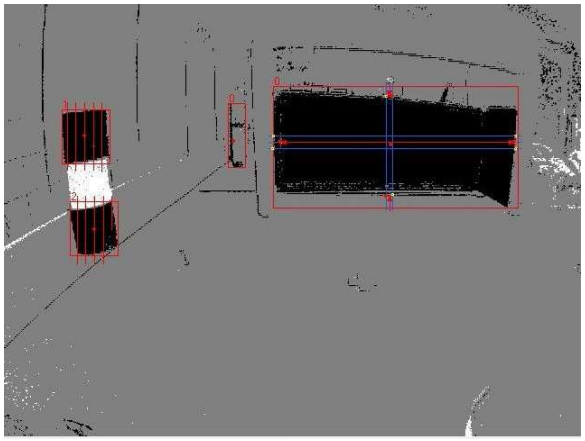


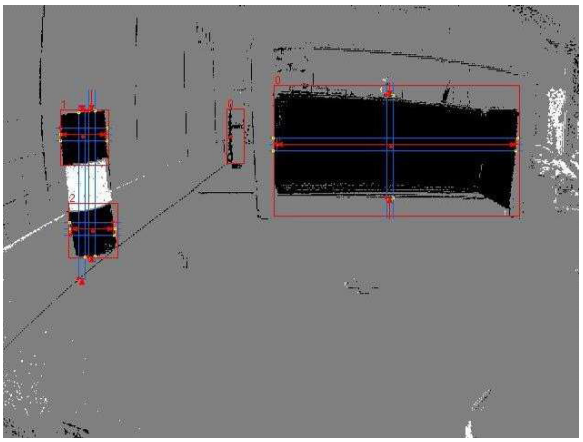
Figure 10.12: Example of blue and yellow contrasting



(a) Example locating "yellow" objects, which are represented as black



(b) Example of scan lines on the left beacon



(c) Example of measuring a beacon

Figure 10.13: Example of how goals and beacons were located and measured

10.1.7 Finding goals

The goals in soccer are also another very important object to identify, since it is the ball has to be kicked into the goal in order to score points and win. The goals can also be used to aide in localization, providing another landmark feature. Finding the goals in an image is much simpler process than finding the beacons of in an image. The goals are the largest area of blue/yellow that is not a beacon. Therefore, to identify the goals, all the objects that are yellow or blue with a minimum size are classified as potential goals. If the robot is very close to a beacon, it is possible that the bands of the beacon could be identified as a goal. In order to avoid this confusion, if both the yellow and blue goal are identified in the same image or are located in same direction along the horizon, the potential goals are ignored. Any remaining goals are classified as goals. The height of the goal is measured at its center. The height of the goal is the most reliable measurement since it is not distorted by the angle at which it is viewed or if it is partially obstructed.

10.1.8 Finding obstacles and other robots

Obstacles are important to identify since they can impede walking or be a potential opponent that would take away a possessed soccer ball. Obstacles are detected by identifying breaks in the field of play. Anything not green or white is a potential obstacle. The obstacles pixel that is lowest in the image should correlate to where the obstacle meets the ground. Where the obstacle meets is the ground is what is used for identifying its location.

Each robot wears a magenta/cyan team marker. Detecting these markers is relatively simple using the CLUT. If the heading of an marker matches the heading of an obstacle, it can be assumed the obstacle and robot are one and the same. There are not many cases where an obstacle would be detected and a marker present if the obstacle were not another robot.

10.1.9 Estimating object distances

An object's location is very useful when planning behaviors. For example, it is important to know where the ball is so the robot can decide if it should kick the ball or get closer. Once an object is identified through image processing, the object's relative location to the robot is determined using two different methods. The first uses the object's size in the image to determine its distance, and geometry to determine its heading; its size in the image decreases the further away an object is. The second method for calculating the relative location of an object is to assume the cameras' height to be known and use geometry to calculate an object's location.

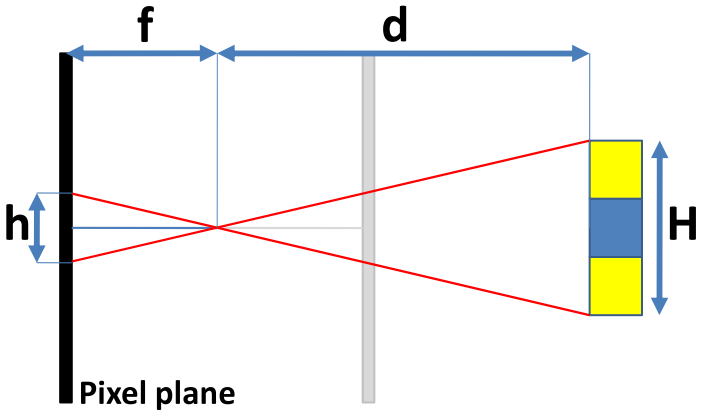


Figure 10.14: Diagram of a pinhole camera model and an object seen

Using object size

Using a pinhole camera model, the size of an object in a camera image is proportional to its distance from the camera. Figure 10.14 is a diagram showing how an object is seen on the pixel plane of a camera. Two similar triangles are formed that apexes at the focal point of the camera located f away from the pixel plane. Using the camera's focal length and the object's height, which are known in advance, the height of the object projected onto the camera is proportional to its distance from the camera:

$$d = \frac{f}{h}H \tag{10.8}$$

where d is the distance of the object from the camera, f is the camera focal length, h is the projected height of the object onto the pixel plane, and H is the known height of the object.

Using geometry

Once again, assuming the pinhole camera model and assuming a known height of the robot, the pixel locations in the image are used to map the objects relative position.

10.1.10 Shot Detection

The ability to detect a shot is important for autonomous soccer competition. Though a robot goalie may be able to identify and locate a ball, determining if a shot is taken is a more challenging task. Noise from the image may cause the ball's size and location to change from frame to frame. The movement of the ball could be interpreted as a shot, which would then trigger the goalie to dive to try and stop the shot. Falsely identifying a shot and diving makes the rest of the goal vulnerable to a real shot while the goalie recovers from diving. In order to avoid this, a linear correlation between the x-position and time is used to calculate the correlation coefficient. If the coefficient is close to 1, then the ball is most likely moving in the x-direction. Otherwise, a coefficient closer to 0 indicates that the changing x-position does not have a linear pattern and the change is probably due to noise. One it is determined that the ball is moving, its velocity and heading are also calculated to determine if the ball is moving towards the goal, and if so, to which side of the goalie the ball will go by. The velocity and direction of the ball is calculated using the results of the linear regression between the x-position of the ball and time. The trajectory is calculated using a linear regression between the x and y values of the locations of the ball. The sign of the y-intercept of the fitted line indicates to which side of the robot the ball will pass.

10.1.11 Optical Character Recognition

The ability for to read text in images not only adds another level of perceived intelligence to a robot, but it also acts as a practical method of giving commands to the robot. Optical Character Recognition (OCR) is found many office software packages as well as in programming languages. Built-in functions in LabVIEW are able to recognize alpha numeric characters within a box bounding the text. The OCR module is able to recognize text after training, which involves acquiring an image of a single character and training the OCR module what the character is. Since the software associates a specific image with a corresponding character, the OCR module is trained for one specific font at a time. Since LabVIEW already offered an OCR package, the only challenge left was to identify a block of text in an image. A red box was drawn around the text to act as a marker for identifying the text block.

10.1.12 Handshake and Dice game

Frequent demonstrations and public outreaches are very successful when they involve a lot of people interaction. Having a robot shake hands or play an interactive game, even a simple game, adds personality and piques interest. Using the OCR software, a sign saying "Handshake" shown to the robot triggers a "Handshake" routine. The routine start with the robot simply extending its hand as an invitation. By polling the motor torques in the extended hand, the software can detect when someone grabs the robot's hand. Once someone

grabs the robot's hand, the robot proceeds to shake the person's hand.

In order to play a game of dice with the robot, a sign showing "Dice" shown to the robot triggers the "Dice" routine. The routine start with the robot signaling that it is ready play a game of dice. Two soft dice colored maroon and orange are used as props. The two different colors allows the robot to easily distinguish between two dice thrown. Using simple thresholding techniques on the HSL color spectrum, it is relatively simple to identify each dice in an image EXAMPLE. One the robot has signaled that it is ready to play, it waits until only one dice is seen in the lower camera (presumably the dice meant for the robot). Once only one dice is seen, the robot pick up the dice and throws it. Afterwards the robot signals to whoever is interacting with the robot that they can throw their dice. Once the robot sees both dice, the number of dots (also identified using a threshold on the HSL spectrum) are counted on each dice and compared. Since the robot knows which color dice belongs to whom, the number of dots counted on each dice determines who wins. Based on the the result the robot responds appropriately–cheering if it wins, sulking it if loses, and shrugging if it ties.

10.2 Interface software

To interface with the motors of the robot and to record/play back motions, LabVIEW was used as a tool for building a capable software solution. The sophisticated servo motors from Robotis have 50 different pieces of accessible information, which can be important for successfully implementing motion on the robot. Additionally, the LabVIEW software provides an easy and intuitive way to create motions; providing abilities such as copy, paste, mirror, etc.

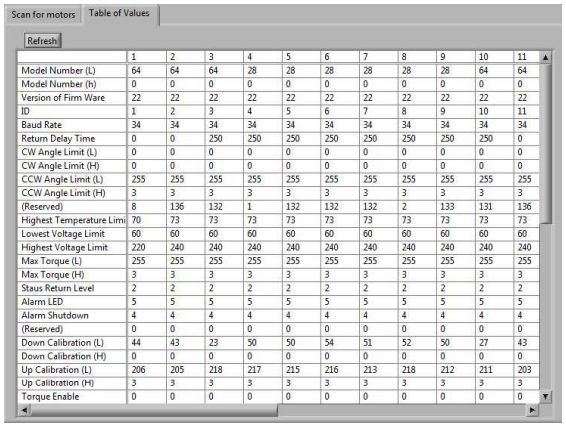
10.2.1 Address access

The dynamical motors as described in Sec. 9.1 have 50 different addresses that describe the state of the motor. Table 10.1 shows the different addresses available and their meaning.

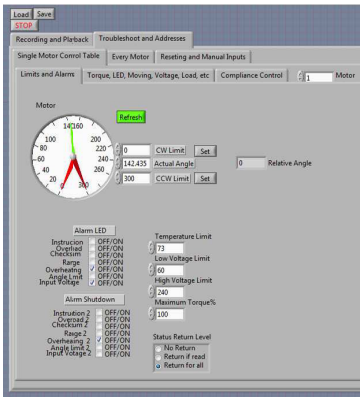
In order to interface easily with the many different addresses of the robot, multiple LabVIEW panels were used to filter through the information. All of the information for each motor could be viewed at once in a table of values as seen in Fig. 10.15(a). The position of the motor as well as safety parameters could be set and viewed for each motor individual as seen in Fig. 10.15(b). The position, speed, and other miscellaneous address data could be viewed in the position and speed front panel for a motor as seen in Fig. 10.15(c). Finally, the compliance of the motor, which equates to the motors controller, could be viewed and changed in the compliance front panel as seen in Fig. 10.15(d).

Table 10.1: List of addresses and their descriptions of the Dynamixel motor from Robotis

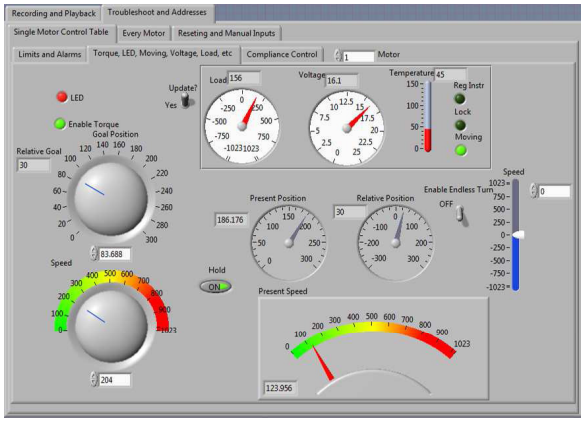
Address	Hexadecimal Address	Address Description
0	0X00	Model Number (L)
1	0X01	Model Number (H)
2	0X02	Version Firmware
3	0X03	ID
4	0X04	Baud Rate
5	0X05	Return Delay Time
6	0X06	CW Angle Limit (L)
7	0X07	CW Angle Limit (H)
8	0X08	CCW Angle Limit (L)
9	0X09	CCW Angle Limit (H)
11	0X11	High Limit Temperature
12	0X12	Low Limit Voltage
13	0X13	High Limit Voltage
14	0X14	Max Torque (L)
15	0X15	Max Torque (H)
16	0X16	Status Return Level
17	0X17	Alarm LED
18	0X18	Alarm Shutdown
24	0X24	Torque Enable
25	0X25	LED
26	0X26	CW Compliance Margin
27	0X27	CCW Compliance Margin
28	0X28	CW Compliance Slope
29	0X29	CCW Compliance Slop
30	0X30	Goal Position (L)
31	0X31	Goal Position (H)
32	0X32	Moving Speed (L)
33	0X33	Moving Speed (H)
34	0X34	Torque Limit (L)
35	0X35	Torque Limit (H)
36	0X36	Present Position (L)
37	0X37	Present Position (H)
38	0X38	Present Speed (L)
39	0X39	Present Speed (H)
40	0X40	Present Load (L)
41	0X41	Present Load (H)
42	0X42	Present Voltage
43	0X43	Present Temperature
44	0X44	Registered Instruction
46	0X46	Moving
47	0X47	Lock
48	0X48	Punch (L)
49	0X49	Punch (H)



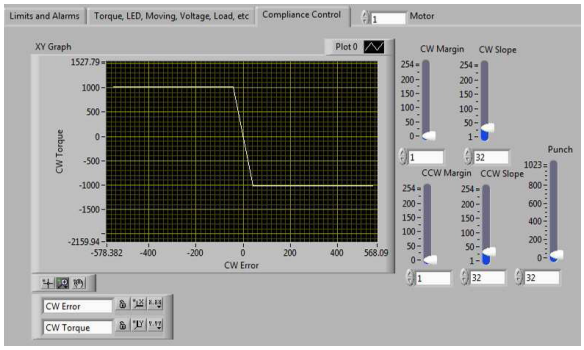
(a) Screen shot of the front panel used to display the values of all of the addressees of all of the motors



(b) Screen shot of the front panel used to display a motor's safety settings and its real-time position



(c) Screen shot of the front panel used to display information about and control the position and speed of motors



(d) Screen shot of the front panel used to modify and visualize the compliance of the motors

Figure 10.15: Screen shots of motor interface software

10.2.2 Recording and playback

Recording a robot's motion and then playing back the motion is a very common practice among robotisists. By capturing individual poses of the robot, and then having the robot go through the captured poses smoothly, you can create almost any motion. This can be done for walking, kicking, dancing, interacting, etc. The software detailed in this section is used to effectively capture the robot's poses and then smoothly play the motions back.

Robots are commonly disassembled for repair or modification, and then reassembled. Often times it is difficult to reassemble a robot the exact same way it was put together. Servo motors often have a "zero" position which is a position that the motor's encoders considers "zero." If the motor is attached to the robot in a slightly different orientation that originally, then the "zero" position of the motor changes with respect to the joint's orientation. For this reason, when motions are recorded, they are recorded relative to a "zero position." The zero position is a predefined known position that the robot can always start from. For the humanoid robot, this position is when the robot stands perfectly straight up, with all joints aligned and the legs at a 90 degree angle to the ground. The arms are perfectly straight down at the robot's side. Other zero position configurations can include putting the robot in a configuration where the range of motion limits the joint's motion, such as bending the forearm until it contacts the upper arm. Even if the robot is reassembled slightly differently than it was originally assembled, as long as the zero position is recorded, all the motions that were previously recorded should still work.

Figure 10.16 shows the front panel of the LabVIEW software used to manage, record, and playback motions for the robot. Additionally, it provides the functionality to turn on or off every motor individually or as a whole. The table on the front panel lists the different positions recorded, which can be copied, pasted, and also mirrored. The mirroring function allows for a copied motion to be duplicated, but mirrored about the XZ plane of the robot. Therefore if a copied gait consisted of the robot lifting up the right foot and moving it forwards, taking a step, the mirrored version would look the same but with the left foot lifting up and moving forwards to take another step. Additional functionality is included to tweak the timing of the gait. The time between each recorded pose and the overall time scale can all be controlled and dictated with this front panel. Additional delays or minimum speeds could be added between poses.

The LabVIEW program is capable of reading and exporting many different file formats for the gaits. The native format is the LabVIEW gait, which saves information including the time steps, time scaling, etc. Additionally, LabVIEW can read and export ".txt" files that are tab delimited tables, which can be created and read by Excel or most spreadsheet programs. When a number of gaits were created, in order to compile them together in an easy to use manner, the LabVIEW assembly and RoboCup assembly collected selected gaits, named them, and associated a zero position file for easy playback. The zero position file defines the positions of the servo motors of the robot when the robot stands straight up and down. All of the recorded gaits are then measured relative to the zero file so they can be

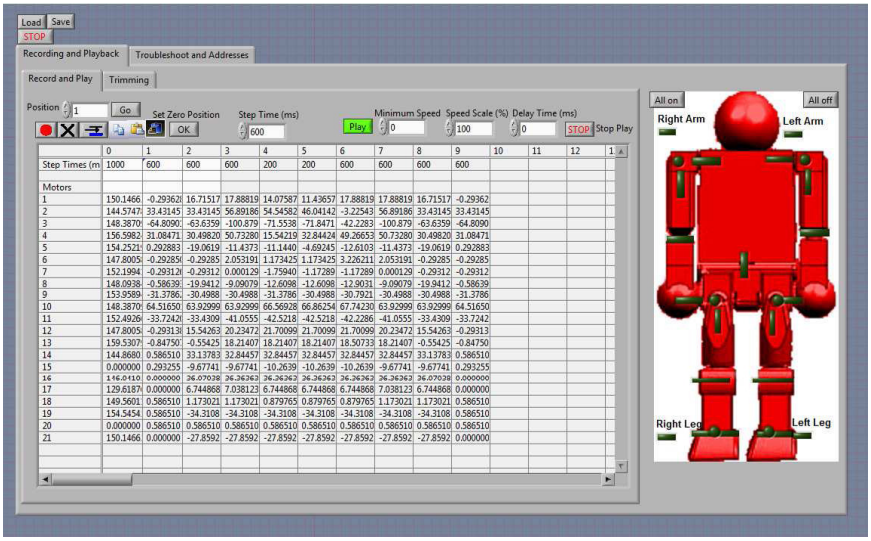


Figure 10.16: Screen shot of the recording table used to display recorded poses of the robot

used for other robots that may have slightly different zero positions.

10.3 Motion software

All of the gaits described in this work were implemented in parameterized forms in LabVIEW. In addition to gait generation, a kinematic simulator for visualization the gaits without physical hardware and a state machine for selecting appropriate gaits for walking were developed.

10.3.1 Kinematic simulator

The kinematic simulator was a very important tool for previewing gaits before actually attempting to implement them. If there was an obvious error in creating a gait, such as a leg running into the body or the robot walking backwards, it could be seen first in the kinematic simulator. The kinematic simulator was created to mimic the robot system. It only accepted the same serial strings that were sent to the motors for control. Figure 10.17 shows a sequence of pictures from the 3D kinematic simulator showing a kicking motion.

The model used in the simulator has adjustable link lengths and values, making it easy

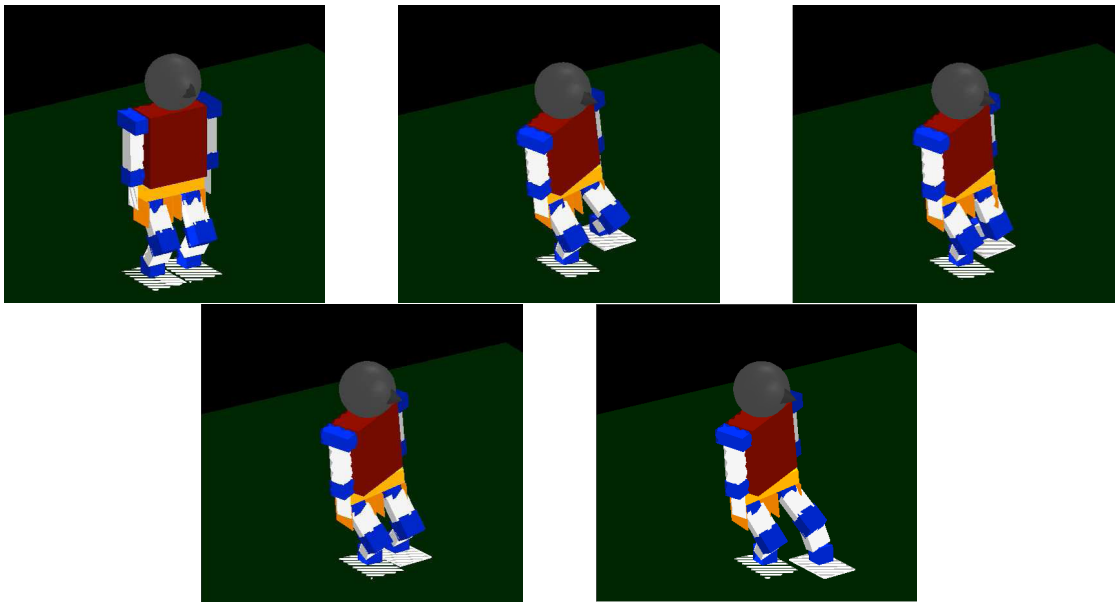


Figure 10.17: Sequence of screen shots of the kinematic simulator. Displaying a kicking motion.

to use for many other DARwIn version. However, the model does need different inverse kinematics and programs to handle different kinematic configurations between robots. This is relatively easy to do. In fact, the simulator was modified to visualize another robot in RoMeLa, STriDER, a three legged robot.

10.3.2 State machine

During RoboCup competition, different gaits were required to accomplish the various tasks of approaching a ball, repositioning, and kicking. In order to address this, a state machine using LabVIEW's state machine toolkit was used as seen in Fig. 10.18. The state chart kept track of what state the robot was currently in. Then, based on the desired action, would step through the necessary states to perform the action. For example, if the robot were mid-step and wanted to kick the ball, it first needs to complete the step and then come to a stop before attempting to kick the ball.

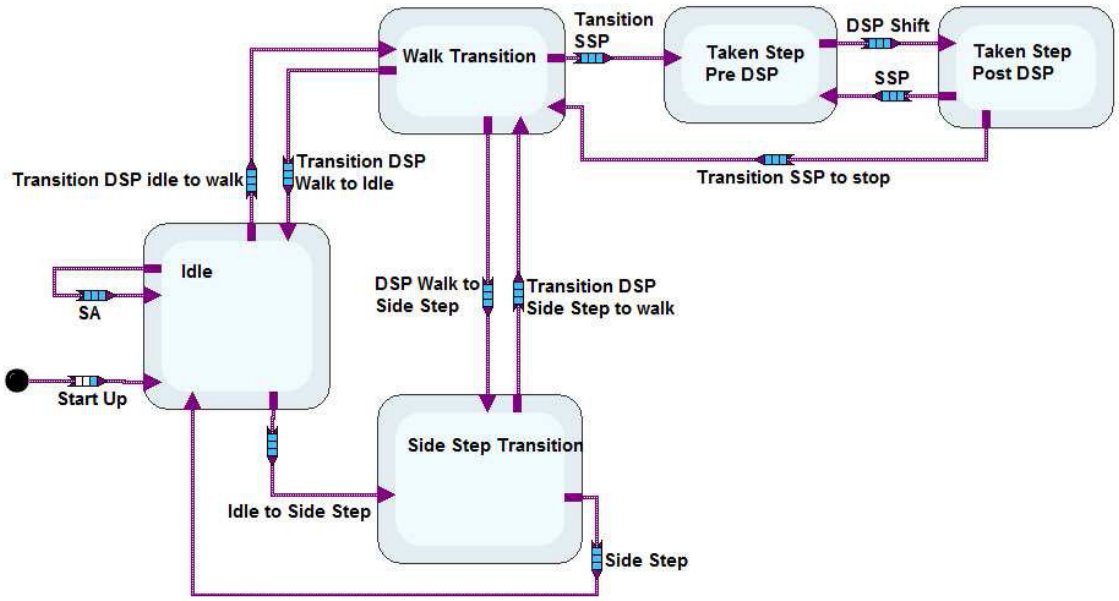


Figure 10.18: Screen shot of state chart in LabVIEW for walking

Bibliography

[1] M. Raibert. Walking robots, 2009.
http://www.ai.mit.edu/projects/leglab/robots/robots.html.

[2] K. Hirai. The development of honda humanoid robot. *IEEE Int. Conf. on Robotics nd Automation*, pages 1321–1326, 1998.

[3] T. Ishida. A small biped entertainment robot sdr-4x ii. *Proc. IEEE Symposium on Comp. Intelligence in Robotics and Automation*, pages 1046–1051, July 2003.

[4] J. Kim. *On the Stable Dynamic Walking of Biped Humanoid Robots*. PhD Thesis, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, 2006.

[5] Tad McGeer. Passive dynamic walking. *Int. J. Rob. Res.*, 9(2):62–82, 1990.

[6] Daan Hobbelen. Denise, 2009.
http://www.3me.tudelft.nl/.

[7] S. Collins. Walking robots, February 2005.
http://www-personal.umich.edu/∼shc/robots.html.

[8] M. Raibert and J. Hodgins. Adjusting step length for rough terrain locomotion. In *IEEE: Transactions on Robotics and Automation*, volume 7, pages 289–298, June 1991.

[9] Marc H. Raibert. *Legged robots that balance*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[10] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *SIGGRAPH Comput. Graph.*, 25(4):349–358, 1991.

[11] Boston Dynamics. Big dog, 2009.
http://www.bostondynamics.com/content/sec.php?section=BigDog.

[12] Frank C. Anderson and Marcus G. Pandy. Dynamic optimization of human walking. *Journal of Biomechanical Engineering*, 123(5):381–390, 2001.

[13] Sven Behnke. Online trajectory generation for omnidirectional biped walking. In *ICRA* [69], pages 1597–1603.

[14] Guy Bessonnet, Stéphane Chessé, and Philippe Sardain. Optimal gait synthesis of a seven-link planar biped. *I. J. Robotic Res.*, 23(10-11):1059–1073, 2004.

[15] G. Bessonnet, P. Seguin, and P. Sardain. A parametric optimization approach to walking pattern synthesis. *Int. Journal of Robotics Research*, 24(7):523–536, July 2005.

[16] C. Chevallereau and Y. Aoustin. Optimal reference trajectories for walking and running of a biped robot. *Robotica*, 19(5):557–569, 2001.

[17] Qiang Huang, Shuuji Kajita, Noriho Koyachi, Kenji Kaneko, Kazuhito Yokoi, Hirohiko Arai, Kiyoshi Komoriya, and Kazuo Tanie. A high stability, smooth walking pattern for a biped robot. In *ICRA*, pages 65–, 1999.

[18] Qiang Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie. Planning walking patterns for a biped robot. *Robotics and Automation, IEEE Transactions on*, 17(3):280–289, Jun 2001.

[19] Lingyun Hu, Changjiu Zhou, and Zengqi Sun. Biped gait optimization using spline function based probability model. In *ICRA* [69], pages 830–835.

[20] Genci Capi, Yasuo Nasu, Leonard Barolli, and Kazuhisa Mitobe. Real time gait generation for autonomous humanoid robots: A case study for walking. *Robotics and Autonomous Systems*, 42(2):107–116, 2003.

[21] Genci Capi, Yasuo Nasu, Leonard Barolli, Kazuhisa Mitobe, and Kenro Takeda. Application of genetic algorithms for biped robot gait synthesis optimization during walking and going up-stairs. *Advanced Robotics*, 15(6):675–694, 2001.

[22] D. Kim, N. Kim, S. Seo, and G. Park. Use of fuzzy system in modeling of biped walking robot. *Int. J. Know.-Based Intell. Eng. Syst.*, 9(4):341–349, 2005.

[23] D. Kim, S.-J. Seo, and G.-T. Park. Zero-moment point trajectory modelling of a biped walking robot using an adaptive neuro-fuzzy system. *Control Theory and Applications, IEE Proceedings* -, 152(4):411–426, July 2005.

[24] A. Dasgupta and Y. Nakamura. Making feasible walking motion of humanoid robots from human motion capture data. *Proc. of Int. Conf. on Robotics and Automation*, pages 1044–1049, May 1999.

[25] J. Kim, I. Park, and J. Oh. Experimental realization of dynamic walking of the biped humanoid robot khr-2 using zero moment point feedback and intertial measurement. *Advanced Robotics*, 20(6):707–736, June 2006.

[26] J. Kim and J. Oh. Realization of dynamic walking for the humanoid robot platform khr-1. *Advanced Robotics*, 18(7):749–768, August 2004.

[27] Jungho Lee Ill-Woo Park, Jung-Yup Kim and Jun-Ho Oh. Online free walking trajectory generation for biped humanoid robot khr-3 (hubo). In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1231 – 1236, May 2006.

[28] F. Buczek, K. Cooney, M. Walker, M. Rainbow, M. Concha, and J. Sanders. Performance of an inverted pendulum model directly applied to normal human gait. *Clinical Biomechanics*, 21:288–296, October 2005.

[29] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by a simple three-dimensional inverted pendulum model. *Advanced Robotics*, 17(2), 2003.

[30] T. Sugihara and Y. Nakamura. Contact phase invariant control for humanoid robot based on variable impedant inverted pendulum model. *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 1:51–56 vol.1, Sept. 2003.

[31] C. Zhu, Y. Tomizawa, X. Luo, and A. Kawamura. Biped walking with variable zmp, frictional constraint, and inverted pendulum model. *Proc. of Int. Conf. on Robotics and Biomimetics*, pages 425–430, April 2004.

[32] C. Zhu and A. Kawamura. Walking principle analysis for biped robot with zmp concept, friction constraint, and inverted pendulum. *Proc. of Int. Conf. on Intelligent Robots and Systems*, pages 364–369, October 2003.

[33] M. Morisawa, S. Kajita, K. Kaneko, K. Harada, F. Kanehiro, K. Fujiwara, and H. Hirukawa. Pattern generation of biped walking constrained on parametric surface. *Proc. of Int. Conf. on Robotics and Automation*, pages 2405–2410, April 2005.

[34] S. Kajita, K. Kaneko, K. Harada, F. Kanehiro, K. Fujiwara, H. Hirukawa, and K. Yokoi. Biped walking pattern generation by using preview control of zero-moment point. *Proc. of Int. Conf. on Robotics and Automation*, pages 1620–1626, September 2003.

[35] A. Takanishi, Hun ok Lim, M. Tsuda, and I. Kato. Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface. In *Proceedings or IROS 1990 IEEE International Workshop on Intelligent Robots and Systems*, volume 1, pages 323–330, July 1990.

[36] S. Lim. Dynamic stair walking of biped humanoid robots. *Journal of Mechanical Science and Technology*, 21(6), 2007.

[37] S. Lim and I. Ko. Locomotions of a biped robot: Static vs. dynamic gaits. *Journal of KSME*, 30(6), 2006.

[38] S. Lim, K.I. KIM, Y.I. Son, and H.I. Kang. Walk simuations of a biped robot. *International Conference on Control, Automation, and Systems*, 2005.

[39] K. Muecke, D. Hong, and S. Lim. Precision circular walking of bipedal robots. *ASME/IDETC/CIE*, August 2008.

[40] T. McGeer. Passive walking with knees. *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1640–1645 vol.3, May 1990.

[41] Steven H. Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *I. J. Robotic Res.*, 20(7):607–615, 2001.

[42] Kyosuke Ono, Ryutaro Takahashi, and Toru Shimada. Self-excited walking of a biped mechanism. *I. J. Robotic Res.*, 20(12):953–966, 2001.

[43] Arthur D. Kuo. Stabilization of lateral motion in passive dynamic walking. *I. J. Robotic Res.*, 18(9):917–930, 1999.

[44] Zina Deretsky. Robots: An exhibition of u.s. automatons from the leading edge of research, July 2008.
`http://www.nsf.gov/news/newsmedia/robotics05/exhibitors.jsp`.

[45] T. Nagasaki, S. Kajita, K. Kaneko, K. Yokoi, and K. Tanie. A running experiment of humanoid biped. *Proc. of Int. Conf. on Intelligent Robots and Systems*, pages 136–141, September 2004.

[46] B.-J. Lee, Y.-D. Kim, and J.-H. Kim. Balance control of humanoid robot using its upper body. In *Proceedings of the 2004 FIRA Robot World Congress*, October 2004.

[47] Jong Hyeon Park. Impedance control for biped robot locomotion. *Robotics and Automation, IEEE Transactions on*, 17(6):870–882, Dec 2001.

[48] Qiang Huang and Y. Nakamura. Sensory reflex control for humanoid walking. *Robotics, IEEE Transactions on*, 21(5):977–984, Oct. 2005.

[49] Z. Popovic. Editing dynamic properties of captured human motion. *Proc. of Int. Conf. on Robotics and Automation*, 1:670–675, April 2000.

[50] V. Zordan and J. Hodgins. Motion capture-driven simulations that hit and react. *Proc. of ACM SIGGRAPH/Europgraphics Symposium on Computer Animation*, pages 89–96, 2002.

[51] Andrew Witkin and Michael Kass. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1988. ACM.

[52] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.

[53] K. Yamane and Y. Nakamura. Dynamics filter-concept and implementation of online motion generator for human figures. *Proc. of Int. Conf. on Robotics and Automation*, 19(3):421–432, June 2003.

[54] M. Vukobratovic. Zero-moment point–thirty five years of its life. *Int. Journal of Humanoid Robotics*, 1(1), 2004.

[55] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic. *Scientific Fundamentals of Robotics 7: Biped Locomotion*. Springer-Verlag, 1990.

[56] Kab Il Kim, Young I. Son, and Paul B. S. Kim. Construction of small humanoids with a new joint actuator module. In *ICRA*, pages 4510–4514. IEEE, 2004.

[57] David A. Winter. *Biomechanics and Motor Control of Human Movement*. Wiley, August 2004.

[58] Shekhar Sharad and Karl Muecke. Teaching complete embedded systems design process with graphical system design methodologies. In *ICPADS '07: Proceedings of the 13th International Conference on Parallel and Distributed Systems*, pages 1–6, Washington, DC, USA, 2007. IEEE Computer Society.

[59] K. Muecke and D. Hong. A reactive approach to behavior based control of a soccer playing humanoid robot. *Proc. UKC Conference*, August 2007.

[60] National Instruments. Ni vision, imaq vision concepts manual. January 2005.

[61] K. Muecke and D. Hong. The synergistic combination of research, education, and international robot competitions through the development of a humanoid robot. *ASME/IDETC/CIE*, August 2008.

[62] Karl Muecke and Dennis W. Hong. Darwin's evolution: development of a humanoid robot. In *IROS*, pages 2574–2575. IEEE, 2007.

[63] K. Muecke and D. Hong. Darwin's evolution: Development of a humanoid robot for research and evolution. *Industrial Embedded Systems*, December 2007.

[64] K. Muecke, P. Cox, and D. Hong. Darwin: Dynamic anthropomorphic robot with intelligence, part 1-concept and general overview. *SERVO Magazine*, 4(12), December 2006.

[65] K. Muecke, P. Cox, and D. Hong. Darwin: Dynamic anthropomorphic robot with intelligence, part 2-parts, wires and motors. *SERVO Magazine*, 5(1), January 2007.

[66] K. Muecke, R. Mayo, and D. Hong. Darwin: Dynamic anthropomorphic robot with intelligence, part 3-darwin 2.0: The next generation. *SERVO Magazine*, 5(2), February 2007.

[67] D. Hong, K. Muecke, R. Mayo, J. Hurdus, and B. Pullins. Robocup 2007. darwin's first soccer tournament: America's first entry to the humanoid division of robocup. *SERVO Magazine*, September 2007.

[68] T. Rfer, T. Laue, H.-D. Burkhard, J. Hoffmann, M. Jngel, D. Ghring, M. Ltzsch, U. Dffert, M. Spranger, B. Altmeyer, V. Goetzke, O. von Stryk, R. Brunn, M. Dassler, M. Kunz, M. Risler, M. Stelzer, D. Thomas, S. Uhrig, U. Schwiegelshohn, I. Dahm, M. Hebbe, W. Nistico, C. Schumann, and M. Wachter. Germanteam 2004, technical report of the rococup-team in the sony legged robot league. Technical report, HU-Berlin, U-Bremen, TU-Darmstadt, U-Dortmund, 2004. 299 pages.

[69] *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*. IEEE, 2006.

Appendix A

AMF Derivations

A.1 Deriving C_s

From Eq. 4.8,

$$x_{\text{COM}}^{\text{AMF}} = x_{\text{COM}}^{\text{Ref}} + C_s \left(x_{\text{ZMP}}^{\text{nom}} - x_{\text{COM}}^{\text{Ref}} \right) \tag{A.1}$$

and rewriting Eq. 4.3 as

$$x_{\text{ZMP}}^{\text{AMF}} = x_{\text{COM}}^{\text{AMF}} - \ddot{x}_{\text{COM}}^{\text{AMF}} \cdot H \tag{A.2}$$

substitutions lead to

$$x_{\text{ZMP}}^{\text{AMF}} = x_{\text{COM}}^{\text{Ref}} + C_s x_{\text{ZMP}}^{\text{nom}} - C_s x_{\text{COM}}^{\text{Ref}} - H \left(\ddot{x}_{\text{COM}}^{\text{Ref}} - C_s \ddot{x}_{\text{COM}}^{\text{Ref}} \right) \tag{A.3a}$$

$$x_{\text{ZMP}}^{\text{AMF}} = x_{\text{COM}}^{\text{Ref}} - H\ddot{x}_{\text{COM}}^{\text{Ref}} - C_s \left(x_{\text{COM}}^{\text{Ref}} - H\ddot{x}_{\text{COM}}^{\text{Ref}} \right) + C_s x_{\text{ZMP}}^{\text{nom}} \tag{A.3b}$$

$$x_{\text{ZMP}}^{\text{AMF}} = x_{\text{ZMP}}^{\text{Ref}} - C_s x_{\text{ZMP}}^{\text{Ref}} + C_s x_{\text{ZMP}}^{\text{nom}} \tag{A.3c}$$

$$x_{\text{ZMP}}^{\text{AMF}} - x_{\text{ZMP}}^{\text{Ref}} = C_s \left(x_{\text{ZMP}}^{\text{nom}} - x_{\text{ZMP}}^{\text{Ref}} \right) \tag{A.3d}$$

$$C_s = \frac{x_{\text{ZMP}}^{\text{AMF}} - x_{\text{ZMP}}^{\text{Ref}}}{x_{\text{ZMP}}^{\text{nom}} - x_{\text{ZMP}}^{\text{Ref}}} \tag{A.3e}$$

$$C_s = \frac{x_{\text{ZMP}}^{\text{Limit}} - x_{\text{ZMP}}^{\text{Ref}}}{x_{\text{ZMP}}^{\text{nom}} - x_{\text{ZMP}}^{\text{Ref}}} \tag{A.3f}$$

Then if $x_{\text{ZMP}}^{\text{Limit}} - x_{\text{ZMP}}^{\text{Ref}}$ can be replaced as $x_{\text{ZMP}}^{\text{err}}$, then

$$C_s = \frac{x_{\text{ZMP}}^{\text{err}}}{x_{\text{ZMP}}^{\text{nom}} - x_{\text{ZMP}}^{\text{Ref}}} \tag{A.4}$$

Amazingly, the same solution is derived when replacing $x_{\text{ZMP}}^{\text{nom}}$ with $x_{\text{COM}}^{\text{Fit}}$ where $x_{\text{COM}}^{\text{Fit}}$ is calculated from Eq. 4.5.

$$x_{\text{ZMP}}^{\text{AMF}} = x_{\text{COM}}^{\text{Ref}} - H\ddot{x}_{\text{COM}}^{\text{Ref}} - C_s\left(x_{\text{COM}}^{\text{Ref}} - H\ddot{x}_{\text{COM}}^{\text{Ref}}\right) + C_s\left(x_{\text{COM}}^{\text{Fit}} - H\ddot{x}_{\text{COM}}^{\text{Fit}}\right) \tag{A.5a}$$

$$x_{\text{ZMP}}^{\text{AMF}} = x_{\text{ZMP}}^{\text{Ref}} - C_s x_{\text{ZMP}}^{\text{Ref}} + C_s x_{\text{ZMP}}^{\text{Fit}} \tag{A.5b}$$

$$C_s = \frac{x_{\text{ZMP}}^{\text{err}}}{x_{\text{ZMP}}^{\text{Fit}} - x_{\text{ZMP}}^{\text{Ref}}} \tag{A.5c}$$

and when $x_{\text{ZMP}}^{\text{Fit}} = x_{\text{ZMP}}^{\text{nom}}$,

$$C_s = \frac{x_{\text{ZMP}}^{\text{err}}}{x_{\text{ZMP}}^{\text{nom}} - x_{\text{ZMP}}^{\text{Ref}}} \tag{A.6}$$

A.2 Deriving C_T

We can describe time scaling as

$$x_{\text{COM}}^{\text{AMF}}(t) = x_{\text{COM}}^{\text{Ref}}(t') \tag{A.7a}$$

$$y_{\text{COM}}^{\text{AMF}}(t) = y_{\text{COM}}^{\text{Ref}}(t') \tag{A.7b}$$

where t' is the new time information, which can be described as

$$t' = \int_0^t C(t)dt \tag{A.8}$$

where $C(t)$ describes the amount of "slowing down" or "speeding up" applied to the reference motion. We can then relate the reference and AMF motion as

$$x_{\text{COM}}^{\text{AMF}}(t) = x_{\text{COM}}^{\text{Ref}}\left(\int_0^t C(t)dt\right) \tag{A.9}$$

Subsequently, when taking the derivative, it is important to note that $C(t)$ is timing varying. Therefore,

$$\dot{x}^{\mathrm{AMF}}_{\mathrm{COM}}(t) = \dot{x}^{\mathrm{Ref}}_{\mathrm{COM}} \left(\int_0^t C(t)dt \right) C(t) \tag{A.10}$$

and following another derivative,

$$\ddot{x}^{\mathrm{AMF}}_{\mathrm{COM}}(t) = \ddot{x}^{\mathrm{Ref}}_{\mathrm{COM}} \left(\int_0^t C(t)dt \right) C^2(t) + \dot{x}^{\mathrm{Ref}}_{\mathrm{COM}} \left(\int_0^t C(t)dt \right) C'(t) \tag{A.11a}$$

$$\ddot{x}^{\mathrm{AMF}}_{\mathrm{COM}}(t) = \ddot{x}^{\mathrm{Ref}}_{\mathrm{COM}}(t')C^2(t) + \dot{x}^{\mathrm{Ref}}_{\mathrm{COM}}(t')C'(t) \tag{A.11b}$$

where $C'(t)$ is the time derivative of the time scaling function. Writing the equation for the ZMP of the AMF motion as,

$$x^{\mathrm{AMF}}_{\mathrm{ZMP}}(t) = x^{\mathrm{AMF}}_{\mathrm{COM}}(t) - H\ddot{x}^{\mathrm{AMF}}_{\mathrm{COM}}(t) \tag{A.12}$$

combining Eqs. A.11 and A.12, we form

$$x^{\mathrm{AMF}}_{\mathrm{ZMP}}(t) = x_{\mathrm{COM}}(t') - H\left(\ddot{x}_{\mathrm{COM}}(t')C^2(t) + \dot{x}_{\mathrm{COM}}(t')C'(t)\right) \tag{A.13}$$

The equation can be further simplified by treating it as a set of discrete functions;

$$\left(x^{\mathrm{AMF}}_{\mathrm{ZMP}}\right)_i = \left(x_{\mathrm{COM}}\right)_i - H\left(\left(\ddot{x}_{\mathrm{COM}}\right)_i {C_i}^2 + \left(\dot{x}_{\mathrm{COM}}\right)_i C'_i\right) \tag{A.14}$$

If we set C_i as constant and C'_i is zero, then we can say

$$\left(x^{\mathrm{AMF}}_{\mathrm{ZMP}}\right)_i = \left(x_{\mathrm{COM}}\right)_i - H\left(\ddot{x}_{\mathrm{COM}}\right)_i {C_i}^2 \tag{A.15}$$

and we can solve for C_i as C_T, a constant, by

$$x^{\mathrm{Limit}}_{\mathrm{ZMP}} = \left(x_{\mathrm{COM}}\right)_\tau - H\left(\ddot{x}_{\mathrm{COM}}\right)_\tau {C_T}^2 \tag{A.16a}$$

$$C_T = \sqrt{\frac{\left(x_{\mathrm{COM}}\right)_\tau - x^{\mathrm{Limit}}_{\mathrm{ZMP}}}{H\left(\ddot{x}_{\mathrm{COM}}\right)_\tau}} \tag{A.16b}$$

where the maximum instability occurs at τ.

A.3 Deriving the inverse kinematics

The inverse kinematics for a kinematic structure as seen in Sec. B.1 can be derived as follows. If θ_R is the rotation in the Z-direction of the grounded foot relative to the body, the following values can be used to describe the position of the hip of the robot relative to a coordinate frame that is at the ankle but has its X-axis aligned with the body's X-axis:

$$x' = (x_h - x_a)\,\mathrm{Cos}\,(\theta_R) + (y_h - y_a)\,\mathrm{Sin}\,(\theta_R) \tag{A.17a}$$
$$y' = -\,(x_h - x_a)\,\mathrm{Sin}\,(\theta_R) + (y_h - y_a)\,\mathrm{Cos}\,(\theta_R) \tag{A.17b}$$
$$z' = z_h - z_a \tag{A.17c}$$
$$\tag{A.17d}$$

where x_h, x_a, y_h, and y_a are the X hip, X ankle, Y hip, and Y ankle positions. The following angles can then be determined using common inverse kinematics

$$\theta_K = -\mathrm{ArcCos}\left(\frac{z'^2 + x'^2 + y'^2 - {L_{\mathrm{thi}}}^2 - {L_{\mathrm{shk}}}^2}{2L_{\mathrm{thi}}L_{\mathrm{shk}}}\right) \tag{A.18a}$$
$$\theta_{\mathrm{A2}} = -\mathrm{ArcTan2}\left(x', \sqrt{y'^2 + z'^2}\right) + \mathrm{ArcTan2}\left(\mathrm{Sin}\,(-\theta_K)\,L_{\mathrm{thi}}, L_{\mathrm{shk}} + L_{\mathrm{thi}}\mathrm{Cos}\,(-\theta_K)\right) \tag{A.18b}$$
$$\theta_{\mathrm{A1}} = \mathrm{ArcTan2}(y', z') \tag{A.18c}$$

where L_{thi} and L_{shk} are the link lengths of the thigh and shank of the robot, and θ_K, θ_{A2}, and θ_{A1} are the knee angle, pitching angle of the ankle, and rolling angle of the ankle respectively.

The following rotations describe rotations from the hip to the thigh where R_x, R_y, and R_z are standard coordinate transform matrices in the X, Y, and Z directions:

$$R_x\,[\theta_{\mathrm{H1}}]\,R_y\,[\theta_{\mathrm{H2}}]\,R_z\,[\theta_{\mathrm{H3}}] \tag{A.19}$$

where θ_{H1}, θ_{H2}, and θ_{H3} are the hip roll, pitch, and yaw angles respectively, which then leads to the rotation matrix

$$\begin{bmatrix} -\mathrm{S}\theta_{\mathrm{A2}}\left(\mathrm{C}\theta_R\mathrm{S}\theta_K + \mathrm{C}\theta_K\mathrm{S}\theta_{\mathrm{A1}}\mathrm{S}\theta_R\right) + \mathrm{C}\theta_{\mathrm{A2}}\left(\mathrm{C}\theta_K\mathrm{C}\theta_R - \mathrm{S}\theta_{\mathrm{A1}}\mathrm{S}\theta_K\mathrm{S}\theta_R\right) \\ \mathrm{C}\theta_K\left(\mathrm{C}\theta_R\mathrm{S}\theta_{\mathrm{A1}}\mathrm{S}\theta_{\mathrm{A2}} + \mathrm{C}\theta_{\mathrm{A2}}\mathrm{S}\theta_R\right) + \mathrm{S}\theta_K\left(\mathrm{C}\theta_{\mathrm{A2}}\mathrm{C}\theta_R\mathrm{S}\theta_{\mathrm{A1}} - \mathrm{S}\theta_{\mathrm{A2}}\mathrm{S}\theta_R\right) \\ -\mathrm{C}\theta_{\mathrm{A1}}S\left(\theta_{\mathrm{A2}} + \theta_K\right) \end{bmatrix} ... \begin{pmatrix} -\mathrm{C}\theta_{\mathrm{A1}}\mathrm{S}\theta_R \\ \mathrm{C}\theta_{\mathrm{A1}}\mathrm{C}\theta_R \\ \mathrm{S}\theta_{\mathrm{A1}} \end{pmatrix} ... \tag{A.20}$$

$$\left(\begin{array}{c} \mathrm{C}\theta_K \left(\mathrm{C}\theta_R \mathrm{S}\theta_{\mathrm{A}2} + \mathrm{C}\theta_{\mathrm{A}2} \mathrm{S}\theta_{\mathrm{A}1} \mathrm{S}\theta_R\right) + \mathrm{S}\theta_K \left(\mathrm{C}\theta_{\mathrm{A}2} \mathrm{C}\theta_R - \mathrm{S}\theta_{\mathrm{A}1} \mathrm{S}\theta_{\mathrm{A}2} \mathrm{S}\theta_R\right) \\ \mathrm{S}\theta_{\mathrm{A}2} \left(\mathrm{C}\theta_R \mathrm{S}\theta_{\mathrm{A}1} \mathrm{S}\theta_K + \mathrm{C}\theta_K \mathrm{S}\theta_R\right) + \mathrm{C}\theta_{\mathrm{A}2} \left(-\mathrm{C}\theta_K \mathrm{C}\theta_R \mathrm{S}\theta_{\mathrm{A}1} + \mathrm{S}\theta_K \mathrm{S}\theta_R\right) \\ \mathrm{C}\theta_{\mathrm{A}1} C\left(\theta_{\mathrm{A}2} + \theta_K\right) \end{array} \right]$$

The following rotation matrices describe the rotations from the foot to the thigh:

$$R_z\left[\theta_R\right] R_x\left[\theta_{\mathrm{A}1}\right] R_y\left[\theta_{\mathrm{A}2}\right] R_y\left[\theta_K\right] \tag{A.21}$$

which leads to the rotation matrix

$$\left(\begin{array}{ccc} \mathrm{C}\theta_{\mathrm{H}2} \mathrm{C}\theta_{\mathrm{H}3} & -\mathrm{C}\theta_{\mathrm{H}2} \mathrm{S}\theta_{\mathrm{H}3} & \mathrm{S}\theta_{\mathrm{H}2} \\ \mathrm{C}\theta_{\mathrm{H}3} \mathrm{S}\theta_{\mathrm{H}1} \mathrm{S}\theta_{\mathrm{H}2} + \mathrm{C}\theta_{\mathrm{H}1} \mathrm{S}\theta_{\mathrm{H}3} & \mathrm{C}\theta_{\mathrm{H}1} \mathrm{C}\theta_{\mathrm{H}3} - \mathrm{S}\theta_{\mathrm{H}1} \mathrm{S}\theta_{\mathrm{H}2} \mathrm{S}\theta_{\mathrm{H}3} & -\mathrm{C}\theta_{\mathrm{H}2} \mathrm{S}\theta_{\mathrm{H}1} \\ -\mathrm{C}\theta_{\mathrm{H}1} \mathrm{C}\theta_{\mathrm{H}3} \mathrm{S}\theta_{\mathrm{H}2} + \mathrm{S}\theta_{\mathrm{H}1} \mathrm{S}\theta_{\mathrm{H}3} & \mathrm{C}\theta_{\mathrm{H}3} \mathrm{S}\theta_{\mathrm{H}1} + \mathrm{C}\theta_{\mathrm{H}1} \mathrm{S}\theta_{\mathrm{H}2} \mathrm{S}\theta_{\mathrm{H}3} & \mathrm{C}\theta_{\mathrm{H}1} \mathrm{C}\theta_{\mathrm{H}2} \end{array} \right) \tag{A.22}$$

Then comparing elements, we can determine the hip angles as

$$\begin{aligned} \mathrm{Sin}\left(\theta_{\mathrm{H}2}\right) &= \mathrm{Cos}\left(\theta_{\mathrm{A}2}\right) \cdot \left(\mathrm{Cos}\left(\theta_K\right) \mathrm{Sin}\left(\theta_R\right) \mathrm{Sin}\left(\theta_{\mathrm{A}1}\right) + \mathrm{Cos}\left(\theta_R\right) \mathrm{Sin}\left(\theta_K\right)\right) \\ &\quad + \mathrm{Sin}\left(\theta_{\mathrm{A}2}\right) \cdot \left(\mathrm{Cos}\left(\theta_R\right) \mathrm{Cos}\left(\theta_K\right) - \mathrm{Sin}\left(\theta_R\right) \mathrm{Sin}\left(\theta_{\mathrm{A}1}\right)\right) \qquad \text{(A.23)} \\ \mathrm{Sin}\left(\theta_{\mathrm{H}1}\right) &= \frac{-\mathrm{Cos}\left(-\theta_R\right) \mathrm{Cos}\left(\theta_{\mathrm{A}2} + \theta_K\right) \mathrm{Sin}\left(\theta_{\mathrm{A}1}\right) + \mathrm{Sin}\left(-\theta_R\right) \mathrm{Sin}\left(\theta_{\mathrm{A}2} + \theta_K\right)}{-\mathrm{Cos}\left(\theta_{\mathrm{H}2}\right)} \qquad \text{(A.24)} \\ \mathrm{Sin}\left(\theta_{\mathrm{H}3}\right) &= \frac{-\mathrm{Cos}\left(\theta_{\mathrm{A}1}\right) \mathrm{Sin}\left(\theta_R\right)}{-\mathrm{Cos}\left(\theta_{\mathrm{H}2}\right)} \qquad \text{(A.25)} \end{aligned}$$

Appendix B

Physical Properties of the Robot

B.1 DH diagram for DARwIn III

Fig. B.1 shows the DH diagram and coordinate axes for DARwIn III. The dark lines indicate the Z-axis direction while the gray lines indicate the X-axis direction. The first character of the label represents the intuitive joint name: hip (h), knee (k), ankle (a), and waist (w). The second character indicates whether it is on the left (L) or right (R) side of the body. The final character indicates the nominal direction of the axis of rotation X, Y, or Z.

B.2 Tables of properties

Table B.1: DARwIn III's link lengths

Link Name	Length (m)
Shank	0.14729
Thigh	0.14085
Hip width	0.1151
Waist height	0.0064
Shoulder height	0.157
Shoulder width	0.1959
Upper arm length	0.1128

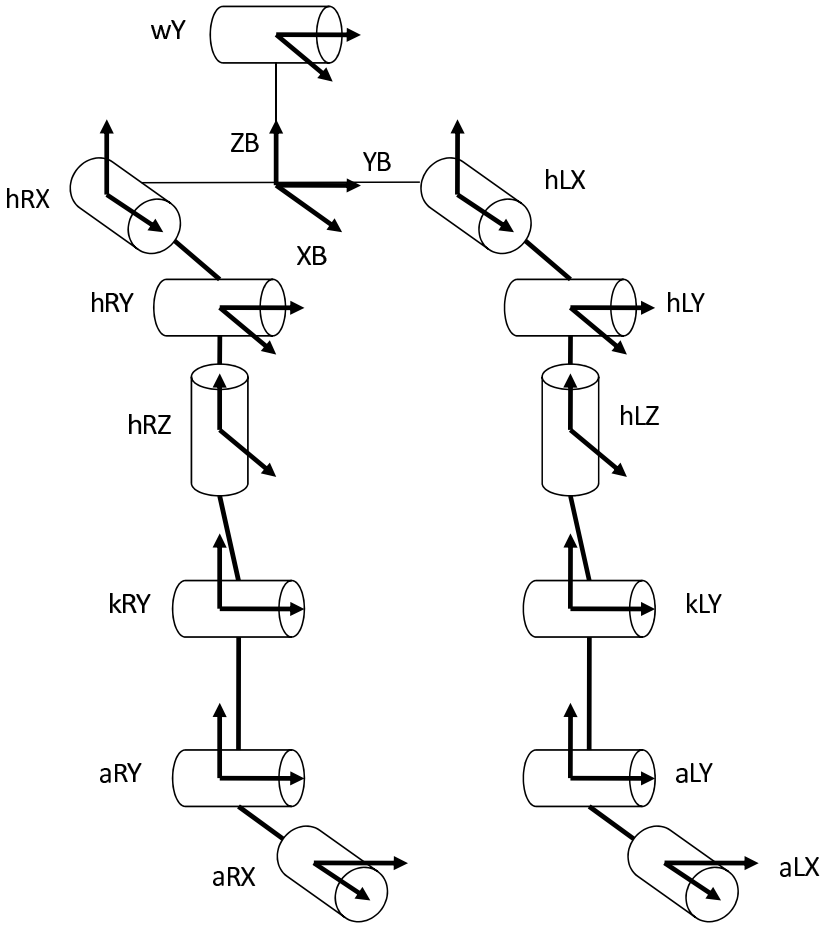


Figure B.1: Axes designation for DARwIn III

Table B.2: DARwIn III's link mass values

Link Name	Mass (kg)
Foot	0.145
Ankle	0.250
Shank	0.054
Thigh	0.197
Lower hip	0.052
Upper hip	0.250
Hip	0.262
Torso	1.357
Total	3.515

Table B.3: DARwIn III's link mass locations using coordinate axes from Sec. B.1

Link Name	X (m)	Y (m)	Z (m)
Foot	0.0229	0	0.00627
Ankle	0	0	0
Shank	0.0739	0	0
Thigh	0	0	0
Lower hip	0	0	0.051
Upper hip	0	0	0
Hip	0	0	0.0064
Torso	0	-0.119	0

Table B.4: DARwIn III's support polygon

Label	Value (m)
$x_{\text{ZMP}}^{\text{Low}}$	-0.056
$x_{\text{ZMP}}^{\text{High}}$	0.104
$x_{\text{ZMP}}^{\text{Nom}}$	0.024
$x_{\text{ZMP}}^{\text{Low}}$	-0.043
$x_{\text{ZMP}}^{\text{High}}$	-0.043
$x_{\text{ZMP}}^{\text{Nom}}$	0.000

Appendix C

Gait Equations

C.1 Dr. Lim's straight hyperbolic gait

$$a = \sqrt{\frac{g}{z_b}} \tag{C.1a}$$

$$t_1 = \frac{1}{2a} \ln \left(\frac{v_{\mathrm{xD}} - a\left(x_0 - x_{\mathrm{ZMP1}}\right)}{v_{\mathrm{xD}} + a\left(x_0 - x_{\mathrm{ZMP1}}\right)} \right) \tag{C.1b}$$

$$v_{\mathrm{xS}} = a\left(x_0 - x_{\mathrm{ZMP1}}\right) \sinh\left(a \cdot t_1\right) + v_{\mathrm{xD}} \cdot \cosh\left(a \cdot t_1\right) \tag{C.1c}$$

$$x' = x_{\mathrm{ZMP1}} + x_{\mathrm{ZMP2}} - x_0 \tag{C.1d}$$

$$t_4 = \frac{x_0 + x_s/2 - x'}{v_{\mathrm{xD}}} \tag{C.1e}$$

$$q = e^{\frac{a\left(x_{\mathrm{ZMP2}} - x_{\mathrm{ZMP1}}\right)}{v_{\mathrm{xS}}}} \cdot \frac{a\left(x_{\mathrm{ZMP1}} - x_0\right) + \dot{x}_D}{v_{\mathrm{xS}}} \cdot \sqrt{\frac{v_{\mathrm{xD}} - a\left(x_0 - x_{\mathrm{ZMP1}}\right)}{v_{\mathrm{xD}} + a\left(x_0 - x_{\mathrm{ZMP1}}\right)}} \tag{C.1f}$$

$$y_0 = \frac{a \cdot t_4 \cdot y_{\mathrm{ZMP}}(1-q)}{a \cdot t_4(1-q) - 2(1+q)} \tag{C.1g}$$

$$v_y = 2y_0/t_4 \tag{C.1h}$$

$$t_5 = \frac{1}{a} \ln \left(\frac{-v_y + a\left(y_0 - y_{\mathrm{ZMP}}\right)}{v_y + a\left(y_0 - y_{\mathrm{ZMP}}\right)} \right) \tag{C.1i}$$

$$t_2 = \left(x_{\mathrm{ZMP2}} - x_{\mathrm{ZMP1}}\right)/v_{\mathrm{xS}} \tag{C.1j}$$

$$t_3 = t_5 - t_1 - t_2 \tag{C.1k}$$

$$T_s = t_1 + t_2 + t_3 + t_4 \tag{C.1l}$$

(C.1m)

Appendix D

Software Derivations

D.1 Generalizing the color lookup table

Quoted from [68]:

• Each point, which is assigned to a color class, irradiates [its] influence to the whole color space, with an influence factor [that] decreases exponentially with the distance:

$$I_i(p_1, p_2) = \begin{cases} \lambda^{|p_1 - p_2|} & i = c(p_2) \\ 0 & \forall i \neq c(p_2) \end{cases} \tag{D.1}$$

where two arbitrary point in the color map, $\lambda < 1$ is the exponential base, $I_i(p_1, p_2)$ is the influence of p_2 on the (new) color class $i \in \{\text{red}, \text{orange}, \text{yellow}, ...\}$ of p_1, and $c(p_2)$ is the color class of p_2

• Since the cost of the influence calculation is $O(n^2)$, where n is the number of elements of the color table, instead of the [euclidean] distance, a manhattan distance is used:

$$|p_1 - p_2|_{\text{manhattan}} = |p_{1R} - p_{2R}| + |p_{1G} - p_{2G}| + |p_{1B} - p_{2B}| \tag{D.2}$$

• For each point in the new color table, the total influence for each color class is computed:

$$I_i(p_0) = B_i \cdot \sum_{p \neq p_0} I_i(p_0, p) \tag{D.3}$$

where $B_i \in (0..1]$ is a bias factor which can be used to favor the expansion of one color class over its neighbors

• Then the color class that has the highest influence for a point is chosen, if:

$$\frac{\max(I_i)}{I_{\text{bk}} + \sum_i I_i} > \tau \tag{D.4}$$

where τ is a confidence threshold, I_{bk} is a constant value assigned to the influence of the background (noColor) which prevents an unbounded growth of the colored regions to the empty areas, and $i \in \{\text{red}, \text{orange}, \text{yellow}, ...\}$ again represents the color class.

D.2 Derivation of the pixel location of the horizon

Using the equations

$$x = x_0 + a \cdot t \tag{D.5a}$$
$$y = y_0 + b \cdot t \tag{D.5b}$$
$$z = z_0 + c \cdot t \tag{D.5c}$$

and setting $t = 0$ for the initial condition, we can say that the parametric line equations are

$$x = p_1' \boldsymbol{x} \tag{D.6a}$$
$$y = p_1' \boldsymbol{y} \tag{D.6b}$$
$$z = p_1' \boldsymbol{z} \tag{D.6c}$$

where p_1' is found from Eq. 10.3. Similarly, setting $t = 1$, we can find values for the coefficients;

$$\boldsymbol{p_2'x} = p_1' \boldsymbol{x} + a \tag{D.7a}$$
$$a = \boldsymbol{p_2'x} - p_1' \boldsymbol{x} \tag{D.7b}$$

$$\boldsymbol{p_2'y} = p_1' \boldsymbol{y} + b \tag{D.8a}$$
$$b = \boldsymbol{p_2'y} - p_1' \boldsymbol{y} \tag{D.8b}$$

$$p_2' \boldsymbol{z} = p_1' \boldsymbol{z} + c \tag{D.9a}$$
$$c = p_2' \boldsymbol{z} - p_1' \boldsymbol{z} \tag{D.9b}$$

Solving for t and setting $z = 0$, which occurs at the horizon,

$$0 = z_0 + c \cdot t \tag{D.10a}$$

$$t = \frac{-z_0}{c} \tag{D.10b}$$

Substituting t in to Eq. D.5, the x and y locations of the pixel that intersects the horizon can be found as:

$$x = p'_1\boldsymbol{x} + (\boldsymbol{p'_2x} - p'_1\boldsymbol{x}) \cdot \frac{-p'_1\boldsymbol{z}}{p'_2\boldsymbol{z} - p'_1\boldsymbol{z}} \tag{D.11a}$$

$$y = p'_1\boldsymbol{y} + (\boldsymbol{p'_2y} - p'_1\boldsymbol{y}) \cdot \frac{-p'_1\boldsymbol{z}}{p'_2\boldsymbol{z} - p'_1\boldsymbol{z}} \tag{D.11b}$$