A Stability-Estimator to Unify Humanoid Locomotion: Walking, Stair-Climbing and Ladder-Climbing

A Thesis

Submitted to the Faculty

of

Drexel University

by

Robert W. Ellenberg

in partial fulfillment of the

requirements for the degree

of

PhD in Mechanical Engineering and Mechanics

June 2014

This copy is annotated.

Acknowledgements

I'd like to thank all of the supportive and helpful people over the years that have made this work possible. The long list starts with my advisor and mentor, Prof. Paul Oh, for many years of advice and guidance. Thanks also to my fellow graduates at DASL, Dr. Youngbum Jun, Dr. Kiwon Sohn, Dr. Christopher Korpela, and Dr. Daniel Lofaro, as well as my colleagues, friends and co-founders at Carbide Labs, LLC (RJ Gross, Bob Sherbert, Keith Sevcik, and Shrey Shah). We have worked together, travelled together, and grown together, and I would never have learned as much or had as much fun without them.

Thanks also to my collaborators at Purdue University, Prof. C. S. George Lee, H. Andy Park, and Manas Paldhe, as well as Prof. Kris Hauser, Yajia Zhang, and Jingru Luo from Indiana University at Bloomington, for their tireless work on humanoid ladder climbing and motion planning.

Finally, thanks to my mother, Lena, father, Bill, brother, John, and sister, Lauren, have been loving and supportive for as long as I can remember.

Dedications

This dissertation is dedicated to my fianceé, Laura Farrow Raines, whose unending love and support made this work possible.

Abstract

A Stability-Estimator to Unify Humanoid Locomotion: Walking, Stair-Climbing and Ladder-Climbing

Robert W. Ellenberg

Advisor: Paul Y. Oh, PhD

The field of Humanoid robotics research has often struggled to find a unique niche that is not better served by other forms of robot. Unlike more traditional industrials robots with a specific purpose, a humanoid robot is not necessarily optimized for any particular task, due to the complexity and balance issues of being bipedal. However, the versatility of a humanoid robot may be ideal for applications such as search and rescue. Disaster sites with chemical, biological, or radiation contamination mean that human rescue workers may face untenable risk. Using a humanoid robot in these dangerous circumstances could make emergency response faster and save human lives. Despite the many successes of existing mobile robots in search and rescue, stair and ladder climbing remains a challenging task due to their form.

To execute ladder climbing motions effectively, a humanoid robot requires a reliable estimate of stability. Traditional methods such as Zero Moment Point are not applicable to vertical climbing, and do not account for force limits imposed on end effectors. This dissertation implements a simple contact wrench space method using a linear combination of contact wrenches. Experiments in simulation showed ZMP equivalence on flat ground. Furthermore, the estimator was able to predict stability with four point contact on a vertical ladder. Finally, an extension of the presented method is proposed based on these findings to address the limitations of the linear combination.

Contents

List of Figures

List of Algorithms

1. Motivation

The field of Humanoid robotics research has often struggled to find a unique niche that is not better served by other forms of robot. Unlike more traditional industrials robots with a specific purpose, a humanoid robot is not necessarily optimized for any particular task, due to the complexity and balance issues of being bipedal. However, the versatility of a humanoid robot may be ideal for applications such as search and rescue. Disaster sites with chemical, biological, or radiation contamination mean that human rescue workers may face untenable risk. Using a humanoid robot in these dangerous circumstances could make emergency response faster and save human lives.

A humanoid shape is a natural advantage because manufacturing plants and commercial buildings are designed for humans. Everything from control stations for equipment, access doors, ladders, and catwalks are designed for safe access by human workers. Despite the many successes of small mobile robots in search and rescue, this fundamental problem of scale makes navigation a constant challenge. A humanoid robot, by contrast, can step over obstacles and rubble. Using both feet and hands, it could climb stairs and even ladders, potentially reaching difficult-to-access controls or trapped people.

The common factor in many rescue-type task is robust locomotion. To reach a control panel, for example, a humanoid may need to walk over rubble, climb a ladder, or traverse a catwalk. These tasks have in many cases been demonstrated already by research robots. Milestones such as bipedal walking [60], stair climbing [23], running [15], and push-recovery [65] have all been demonstrated on robots such as the ASIMO, Hubo, HRP2 and Petman. Motion planning for ladder climbing has also been explored in literature such as [93] and [94].

In all of these efforts, a fundamental part of motion planning and control is the ability to estimate stability. Unlike statically-stable quadrupedal and wheeled robots, bipeds have a small footprint and often require active balance control. Locomotion over flat, smooth terrain is well-researched; Research humanoids such as ASIMO, Hubo, HRP, and Wabian have been shown to perform dynamic walking under a variety of conditions. In many of these robots, balance is maintained by measuring and controlling the robot's Zero Moment Point (ZMP). Unfortunately, the limitations of ZMP methods restrict their practical use to flat terrain or stairs. For problems like ladder climbing, however, ZMP methods do not effectively incorporate the effects of hands grasping hand-holds or ladder rungs.

To progress towards a motion controller ladder climbing and general terrain, a common requirement is an improved stability estimator that incorporates contact forces from both feet and hands. The contributions of

this dissertation address this problem with the following contributions:

- A humanoid robot simulation environment for motion planning and hardware control.
- A Contact Wrench Space stability metric that accounts for grasp forces and force limits of end-effectors.

2. Locomotion and Stability Literature

2.1 Linear Dynamic Models

The starting point for any attempt to design a stability estimator for a humanoid robot is the concept of Zero Moment Point (ZMP). Originally introduced by [86], the ZMP is an imaginary point on the ground though which the equivalent reaction wrench has zero moment. In other words, if all forces do to foot contact were concentrated at this point, no torque would be required to support the robot's weight. If the ZMP is within the convex hull of the robot's feet, then it will be stable. If the ZMP lies outside of this area, then the net moment of any force the foot can apply is not zero, and the robot will fall over. ZMP control avoids this by adjusting the robot's posture, trying to keep the ZMP within the foot area.

ZMP control has been applied to many humanoids, such as ASIMO, Hubo [60], HRP2 [76], Wabian [92], and MiniHubo [37]. Because much of the robot's mass is concentrated in the torso, the total mass of the robot is often modeled as a single lumped body. If the robot's hips are constrained to a fixed height, the motion can be modeled by a Linear Inverted Pendulum Model (LIPM), originally introduced in [40]. This linear model is convenient for linear controller design, and allows forward and lateral motion to be handled separately.

In a typical LIPM-based trajectory planner, the actual trajectories of the robot's torso and feet are parameterized based on a small number of parameters. For example, forward walking requires only step length, step height, hip height, and turning angle. These open-loop trajectories are designed so that the Zero Moment Point (ZMP) is always contained within the support polygon (region of contact with the floor). On flat and level ground, a humanoid executing this trajectory can achieve dynamic stability even without feedback control.

Another advantage of ZMP and the LIPM is that open-loop trajectories can be defined using only a few parameters. In [41], only desired foot positions and the mass/height ratio were necessary to constrain the motion. The walking controller monitored ZMP using force/torque sensors, correcting the desired hip position online. The balancing problem here becomes a tracking problem in two senses. The first is that the robot's hip is tracking the desired pose relative to its feet. The second is that the ZMP of the robot itself is estimated through the net contact forces at the feet, and is corrected by small offsets to the support foot's pitch and roll axes. In [70], the Center of Pressure (CoP) of a robot's foot was shown to be identical to the ZMP when the pose is stable.

The major contributions in this fields have shown that ZMP methods are simple to apply and well-suited

to flat, even terrain. Unfortunately, to be usable for climbing problems, the controller needs to allow vertical movement, and account for forces from the hands to keep the robot from falling backwards. The original LIPM formulation was not designed to handle slopes or steps, though more recent work has shown that a modified ZMP controller can handle terrain that is piecewise-flat. A notable example is [26], which accounted for hand holds and sloped steps in the walking controller.

Other examples of stair climbing have been shown in [58] and [23]. In these cases, the ZMP model was modified to allow smooth vertical motion during stepping. The climbing motions employed in these examples are dynamic and periodic. Smooth, dynamic motion is often more energy-efficient than slow quasi-static motion, and energy savings could therefore lead to longer run times. Improvements to ZMP-based control were introduced in [99] and [35], which demonstrated stair-climbing on sloped steps.

Dynamic walking can deal effectively with certain disturbances, but becomes increasingly challenging when the desired motions cease to be periodic. The advantages of dynamic walking such as energy savings and reduced hip motion amplitudes may be offset by the difficulties of planning effective transitions between drastically different foot goals. A dynamic trajectory needs to anticipate the next foot position in advance so that the hip can move continuously. Achieving dynamic walking in conditions such as steep staircases, discontinuous or sloping ground, bumpy terrain, etc. requires additional planning and/or control.

Given the potential of stair-climbing as a simplified version of ladder climbing, the first step was to reproduce a ZMP-controlled walking gait modified for stair-climbing use. In [38], stair-climbing was successfully shown using both the MiniHubo and Hubo2 robots. By planning for static stability, open-loop trajectories were demonstrated to be stable on stairs of 30 deg. However, the fine tuning required meant that small modelling errors could cause the robot to topple. Later work dealt with improved trajectory planning [37], but these early results indicated that sensing capability was paramount in achieving robust climbing.

In [33], an alternate stability metric was introduced that allowed arbitrary poses of the robot. Known as the Contact Wrench Sum (CWS), it worked by checking if a given wrench (or applied force and torque) can be cancelled out by reaction wrenches from the robot's contacts. These ideas were similar to the Grasp Wrench Space (GWS), the standard method of evaluating grasp stability [9]. The wrench on the robot due to gravity (acting at the center of mass) is treated as the primary disturbance. Unlike ZMP methods, the Contact Wrench Sum can handle arbitrary contact normals, and accounts for the effects of friction. This concept was refined in [98], by formulating the contact wrench sum as a polyhedral cone. This method can also estimate stability margin by measuring the distance to the edge of the polyhedral cone from the current gravity wrench. More recent work [96] has refined this method to include soft-contact effects in the computation of the wrench sum.

To simplify calculations, the CWS cone method assumes that infinite normal forces are possible. How-

ever, the forces that an end effector can produce are ultimately a function of each of the joints in that limb. High contact forces may be unrealistic in some cases. For example, if the robot's knees are bent, then the normal force at the feet may be limited by the torque produced at the knees. This means that the polyhedral cone approximation could potentially indicate that a pose is stable, even if the forces required are unreasonable relative to the robot limits. A similar issue is present with robot hands, as grip strength is often small compared to a humanoid's weight. While existing CWS methods are an improvement over ZMP, these solutions do not yet address climbing problems.

2.2 Rough Terrain Walking

If stairs and ladders are considered to be extreme examples of rough terrain, then the field of rough terrain walking may hold answers to this problem. Robust locomotion is widely explored in quadrupedal robots. An important example is the Boston Dynamics Big Dog [67] can walk over rough and unmeasured terrain, without foreknowledge. Robust walking has also been demonstrated on smaller quadrupeds such as the little dog [100], in which the small quadrupedal robot was able to cross rough terrain and compensate online for misplaced feet. Unfortunately, the majority of these algorithms rely on the statically stable nature of a quadruped robot. The stability metric of the robot itself is equivalent to ZMP for humanoids.

For legged robots, rough terrain walking presents several new challenges. Rocky ground could be sloped, have fissure or gaps, and have unpredictable surface friction. The foot placement for each step will in general be at a different height and orientation than the previous one. One attempt to adapt a quadruped's periodic gait to such variations is [72]. The controller can choose a stepping foot goal so that the center of mass is always supported statically by the 3 supporting feet. For unknown terrain, compliance and landing control of each foot is critical to prevent a robot from toppling itself. In [64], the authors demonstrate a method of controlling the Little Dog quadruped over rough terrain using compliance control of the feet and a smooth, periodic motion for the center of mass, primarily based on rejecting the "disturbances" that the terrain introduces, while maintaining a flat floor assumption.

These disturbance-rejection methods have been extended to bipedal robots as well. For example, [77] demonstrates a landing controller that compensates for unpredictable floor heights, allowing an HRP-2 Humanoid to walk over small steps and bumps. In [60], the KHR2 humanoid is outfitted with foot compliance control to absorb small errors in foot placement. In both cases, however, the footstep height is relatively low. These methods would not be effective for large obstacles, since the robot's feet could impact an obstacle from the front rather than the top.

The control methods discussed previously can be considered ”blind” since the robot only requires force feedback from its legs to stabilize itself. To improve performance, some controllers have incorporated knowledge of the terrain to plan more efficient and stable trajectories. [85], for example, plans a desired trajectory offline using terrain knowledge, then adapts the joint trajectory online to best execute the overall plan. [100] demonstrated an optimization approach that searches for candidate foot poses that minimize the robot's movement and maximize the size of the support polygon. In both cases, the robots can handle obstacles and terrain features of a similar order as the robot itself.

In many ways, the problems of stair-climbing and rough-terrain walking share similar challenges as the environment becomes more complex. Stairs with varying slope, height, and surface shape present similar challenges as rough terrain with large elevation changes and bumpy surfaces. In both cases, recent literature shows great advances in planning. However, hardware implementations suffer from the same model simplifications in dynamic walking work cited previously. Therefore, a robust state estimator is still a fundamental requirement for these cases.

2.3 Footstep Planning and Navigation

Given the difficulty of measuring stability over unknown terrain, early walking planners such as [47], [14] tried instead to avoid challenging terrain. The footstep planners from these works focused on choosing reachable and stable footholds, avoiding terrain which was too rough or sloped for the robot to handle. To evaluate possible steps, a generalized step cost was developed to analyze how difficult it would be for a humanoid robot to place its foot in a given location. Measures such as bumpiness, slope, stability, and safety were combined to assign a desirability score to a given foot placement. By searching the space of foot placements, the planner produced optimal paths through the environment.

More recent work such as [25] and [91] used a camera and LiDAR to search for flat surfaces in an environment. The resulting environments are segmented as either polygonal meshes or voxel grids, and used as source data for planning algorithms. [2] uses a swept-volume approximation to ensure that the entire footstep motion does not collide with obstacles. The planner rapidly updates and re-plans in response to a changing environment by visual recognition of obstacles on a flat floor.

Footstep planning is an important contribution to a robot's stability. In environments with many choices of path, choosing the right path can bypass terrain too difficult for the robot. In search and rescue environments, however, there may be very little choice. On a shipboard environment, or in subterranean tunnels, ladders may be the only path to the goal. Therefore, footstep planning alone is not enough to provide robustness in a

climbing or unstructured scenario.

2.4 Contact Sensing

A common feature of ZMP and CWS methods is that both methods estimate a stability margin as part of the computation. To do so, however, it is vital to know where the robot is actually contacting the environment. If a foot is lifted in the air, then it cannot contribute to stability. Both ZMP and CWS methods will fail if they incorrectly assume a contact that isn't there. In a sense, these metrics depend on properly estimating *potential* contact forces, rather than any instantaneous state. This information must be available from sensors or intrinsic state knowledge. Typically, contact forces in humanoids are measured with force / torque sensors as in [61]. [34] showed that tactile feedback similarly could improve the grasp quality and reduce forces applied using a robot gripper. The fast, immediate feedback provided by force-sensing elements is therefore an excellent candidate for evaluating holds on a ladder. Tactile feedback can provide more detailed information about contact forces, but has traditionally been expensive due to the complex and delicate nature of the sensors. However, recent developments such as the Takktile sensor [78] can produce force resolutions of under .01N for a cost of about 200USD for a finger-sized array.

Computer vision provides another source of feedback about the robot's state. Given the important role vision plays in human balance in human infants [51], vision may have the potential to help measure stability in humanoid robots. Unfortunately, the biggest current hurdle of computer vision is processing speed. Identifying salient features in a complex scene in a span of milliseconds is a computational challenge. While high-speed reactions to visual data are possible, notably in [79], in which high-speed autonomous driving was demonstrated based on visual analysis of an off-road path in the DARPA Grand Challenge.

In the context of ladder climbing, vision is useful as a source of planning data. By identifying ladder rungs and walls, a vision system can prime a motion planner with possible holds to advance the robot. If rapid regrasping is necessary, then vision data could help guide the hand towards the target as well. Unfortunately, visually verifying that contact is made is challenging for a number of reasons. Besides the obvious problems of occlusion by the robot's limbs, subtle positions shifts can cause loss of foot placement or grip. While vision would be valuable to correct gross placement errors, identifying the exact pose from visual information alone is a daunting problem.

2.5 Critical Gap

Based on the current literature, the state of the art in stability estimation for humanoid robots applies both CWS and ZMP methods. These methods have been extended to address a variety of special cases, yet no single solution has emerged that fully addresses the challenges of ladder climbing.

To investigate this problem, early work focused on trajectory generation and control for dynamic walking in Chapter 3, which demonstrated the need for balance control. Investigating open loop planning for stair-climbing in Chapter 4 led to successful physics simulation and hardware testing of humanoid stair climbing, and showed the limit of ZMP-based stability estimation. Chapter 5 explored planning work in ladder climbing, demonstrating a successful ladder climbing simulation using three and four point contact. Finally, a novel Contact Wrench Space stability check is developed and evaluated in Chapter 6.

3. Preliminary Work

The path to ladder climbing and humanoid robot stability began with the unlikely field of humanoid robot dance. The hypothesis was that music and dance, due to formal structure and language, could be a more readily achievable bar for robotic creativity than spoken language. This ambitious goal started with an implementation of a beat tracker and a set of dance motion primitives in [20] and [19]. Based on the measured tempo of a music track, the humanoid performed dance motions at the same tempo by randomly selecting from these primitives. This work was conducted using the Hitec RoboNova (Figure 3.1), chosen due to its low cost and commercial availability.

Figure 3.1: Hitec RoboNova performing a choreographed dance routine.

While individual motions were manually adjusted to avoid falls, transitions between motions often led to instability. Development and testing also required custom interfaces, drivers, and low-level communication hardware. Furthermore, implementing any results on a larger humanoid would mean significant re-work to migrate these custom interfaces. This difficulty in comparing results and propagating advances has consistently been a challenge in humanoid robotics, due to differences of kinematic structure, software environments, operating systems, and actuation technologies.

3.1 Dynamic Walking Experiments

The development of walking controllers for miniature humanoids was a collaborative effort with Youngbum Jun, Robert Sherbert, and Roy Gross, and has been published in [38] and [37]. These first experiments in dynamic walking were done with the MiniHubo (Figure 3.2) humanoid robot. The MiniHubo is a scaled version of KAIST's Hubo2 humanoid, and shares the same sequence of joints for the lower body. The primary motivation for starting small was that the strength-to-weight ratio of miniature robots is higher than that of full-size humanoids. Falls and motion failures are safer and easier to recover as well. The trade-off to the smaller size is that the dynamics are faster due to the lower center of mass, which leaves a shorter time to recover if the robot starts to fall.

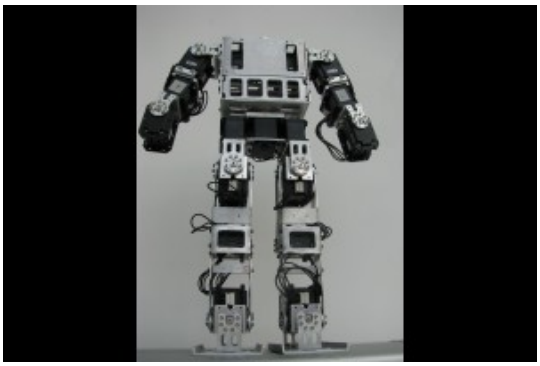

Figure 3.2: MiniHubo, a scaled-down version of KAIST's Hubo2 humanoid.

The walking controller developed for MiniHubo in [38] was inspired by [60], which applied ZMP control to the full-size Hubo2 humanoid. It first creates task-space trajectories for the feet and hips (Section 3.1.1). These trajectories are then translated into joint space trajectories using a numerical inverse kinematic solver (Appendix A). Finally, the joint space trajectories are executed on the robot and feedback control is optionally applied to stabilize the robot during motion to damp out unwanted dynamics.

3.1.1 Trajectory Planner

The trajectory planner creates task space trajectories from a small set of parameters. These parameters are extracted from the robot itself, as well as from the desired footstep locations, and used along with a template trajectory to create a walking pattern motion in task space, which is the space of the hips and feet.

One forward walking cycle consists of one step from each foot, with the hips moving simultaneously to

Table 3.1: MiniHubo trajectory generator parameters

Parameter	Symbol	Value
Mass	m	$2.1kg$
Hip Height	h	$254mm$
Foot Width	w_f	$64mm$
Foot Length	l_f	$109mm$
Foot Spacing	d_y	$10mm$

Step distance (forward)	A_x
Step height	A_z
Step width	d_y
Total time	t_s
Double Support time	$t_d s$
Single Support time	$t_s s$
Hip Height	h
Hip Sway	h_y

Table 3.2: Complete list of parameters required for a walking controller for MiniHubo

maintain balance. The robot begins in the Double Support Phase (DSP), in which both feet are on the ground. In this phase, the robot can shift its weight between feet. The next phase is the Single Support Phase (SSP), in which one foot is lifted and swung forward. Once the foot lands again, the robot returns to DSP, allowing it to shift its weight over the opposite foot. A complete cycle consists of two DSP's and two SSP's, one each for the left and right foot.

The foot placements are specified by a step distance A_x and step spacing d_y, where x and y refer to the robot's forward and lateral directions . Each foot placement also determines the desired ZMP position during each single-supporting phase. Since the ZMP is a function of the hip's position and acceleration, the hip position is derived from the foot positions and step time.

To build the LIPM model for trajectory planning, MiniHubo was parameterized in terms of its mass, leg length, and foot area. These parameters (Table 3.1) are then used to determine the ideal walking frequency in (3.1), based on the hip height h. This frequency is then inverted to find the overall step time t_s, which is divided into the single support time t_{ss} and double support time t_{ds}. The ratio of each is manually tuned, but the single support time is typically about $8x$ larger than the double support time.

$$f = \sqrt{\frac{g}{h}} \tag{3.1}$$

The motion sequence for forward walking consists of four processing steps:

1. Footstep planning
2. ZMP trajectory generation
3. Hip trajectory generation from ZMP control
4. Foot trajectory generation

At the highest level of planning, footstep goals are chosen as a sequence of $2D$ points on the ground. A ZMP trajectory is then calculated by moving the ZMP in sequence over each supporting foot. The ideal ZMP trajectory is fixed over each supporting foot during SSP, and moves at a constant speed between footstep goals during DSP. The ZMP position for a left-to-right walking cycle is given in (3.2), which depends on the durations of the SSP (t_{ss}), DSP (t_{ds}), and foot positions p_{left} and p_{right}.

$$\vec{p}_{ZMP} = \vec{p}_{left}\frac{t_{ds}-t}{t_{ds}} + \vec{p}_{right}\frac{t}{t_{ds}}, 0 < t < t_{ds}\vec{p}_{ZMP} = \vec{p}_{right}, t_{ds} < t < t_{ss} \tag{3.2}$$

The foot trajectory for the swing foot is generated by interpolation between successive foot positions. The step length A_x, the horizontal spacing between foot centers d_y and the lift height A_z are used to calculate a cycloidal interpolation for the x component of the motion (3.3), and a half-cycle sinusoidal interpolation for the z component (3.4).

$$x = A_x\left(\omega * t_{ss} - sin\left(\omega t_{ss}\right)\right) \tag{3.3}$$

$$z = A_z sin(\omega t_{ss}) \tag{3.4}$$

In general, turning while walking forward introduces a centripetal acceleration component, which in turn affects the zero moment point location. While it is possible to reformulate the ZMP equation with additional accelerations factored in, a first approximation was developed using a simple hip yaw offset.

To complete a turn, two footsteps are required. The first step leads the turn, which requires that a left turn begin with a left step, and vice-versa. In the first step, each hip yaw joint's yaw constraint is given an opposite offset, following a cycloidal amplitude. The following step inverts this path, bringing both legs' yaw angles back to zero (Fig. 3.4). These angles are specified for the solve as the y-components of the foot's x vector in the global frame.

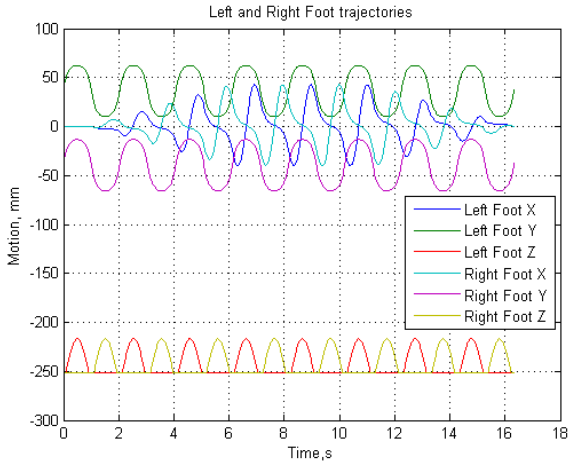

Figure 3.3: Foot trajectories relative to the hip motion, showing alternate lift & landing using sinusoidal (Z) and cycloidal (X) motion.

3.1.2 Initial Results

As shown in [37], an arbitrary obstacle-avoiding trajectory was planned using using A*. However, when executing this trajectory, the physical robot would lose its balance if it walked continuously for more than approximately 8 steps. Due to small imperfections in the ground and compliance in the robot joints, the walking pattern slowly diverged until the hips no longer followed a dynamically stable path. The short-term solution to this issue was to execute walking motions in short sequences. The postural oscillations decayed in under 2 seconds on stopping, allowing the robot take another sequence of steps.

3.2 Simulation and Planning with OpenHubo

The challenges of dynamic walking suggested that, for more complex motions, stability would increasingly become a challenge to maintain. Another challenge with the toolchain used in [37] was that the motion planning environment was entirely separate from the simulation environment. Ensuring that the planning environment and simulation environment matched was left to the user. One limitations of the trajectory planning process was the isolation between various phases of motion planning. As obstacles became more complex, it would be advantageous to have unified planning and simulation environment to allow rapid iteration of motion planning and evaluation with dynamic simulation.

Robot simulators of many forms exist to address this problem. For example, a popular open-source physics engine, Open Dynamics Engine (ODE) forms the basis of many game engines and robotics simulators. Simulators such as OpenRAVE [17], Webots [55], and Gazebo offer straightforward implementations

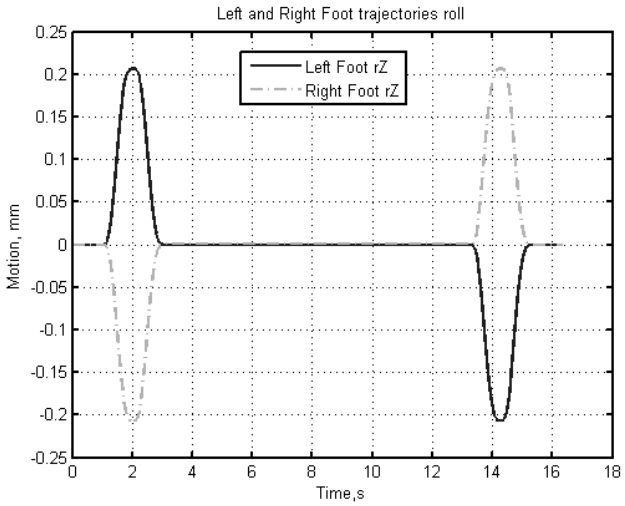

Figure 3.4: Simple hip yaw offsets for low-velocity turning

of the Open Dynamics Engine to model dynamic behavior of articulated robots. Other platforms such as OpenHRP2 [42] use similar dynamics engines based on the featherstone algorithm , and assume rigid body contact. This combination was verified and validated in [16], in which a bouncing ball in OpenHRP3 was compared to a physical example. These simulators demonstrate that ODE can produce physically plausible behavior even for complex systems.

OpenRAVE stands out as particularly desirable from a development perspective. It implements both Bullet and ODE physics engines allowing the user to choose whichever is most suited to their application. OpenRAVE also includes a variety of plugins, including sampling-based motion planning, grasping, trajectory retiming, and both closed-form and numerical inverse kinematics. The API is available in C, C++ and Python, maximizing code reuse and compatibility with common development tools in robotics. For these reasons, OpenRAVE was chosen as the basis for the OpenHubo simulator.

The OpenHubo simulation environment 3.5 was designed to provide both the planning power of OpenRAVE with the dynamic simulation of other ODE-based simulators. Its purpose is to allow motion planning, simulation, and testing / evaluation of controllers in a virtual environment. OpenHubo extends OpenRAVE with full position, torque, and passive control of robot joints during physics simulation. OpenHubo also provides sensor models of Hubo sensors, including accelerometers, Inertial Measurement Units (IMU) and Tri-axial force/torque sensors. Finally, OpenHubo presents a command-line interface based on IPython [62]. Users can quickly create scripts to control the model using an extensive Python API, and control the robot interactively. Tab completion is provided for the API through the IPython interpreter.

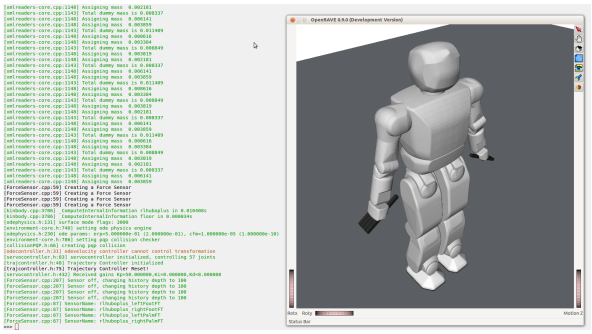

Figure 3.5: OpenHubo environment showing python scripting interface and OpenRAVE GUI

OpenHubo makes possible a simple and direct workflow from concept to implementation on physical hardware. Standard motion planning libraries are available from the OpenRAVE framework, to plan kinematic motions that are stable and collision-free. When run without a physics engine, OpenHubo can simulate motion and environmental interaction quickly for motion evaluation. With a single setting, a motion can be executed on one of several humanoid models using physics simulation. A logging system provides a complete history of the robot's state for export and analysis. Once a motion has been evaluated in this way, OpenHubo can be connected to Hubo-ACH directly control the Hubo hardware. Data is passed continuously between OpenHubo and Hubo-ACH, presenting physical sensor and encoder data transparently. This drop-in replacement of simulation with hardware control means that no software porting is required to validate motions on real hardware. The simulation loop in OpenHubo runs in soft real-time, allowing reliable control of the Hubo2 CAN bus. Figure 3.6 shows a histogram of simulation intervals when OpenHubo is run at the frequency of the Hubo2 CAN bus, $100Hz$. The standard deviation of the loop times is $0.05ms$, two orders of magnitude lower than the loop time itself.

3.2.1 Model Export

The geometry model of the Hubo2 and Hubo+ robots (Fig. 3.7) were based on 3D solid models developed in partnership with Hubo Lab at KAIST. This assembly model contained all of the massive, discrete parts used in the real robots, including simplified models of electronics and servo components. To simplify geometry processing, the coordinate systems of each robot link are aligned with the global orientation at the home position. This coordinate standard has the X axis pointing forward, the Y axis to the robot's left, and the Z axis upwards. Since the links share a common orientation, a simple sign convention was chosen so that all joint axes are positive in the counterclockwise direction about each axis.

Each robot model was exported from a solid model assembly to Unified Robot Description Format

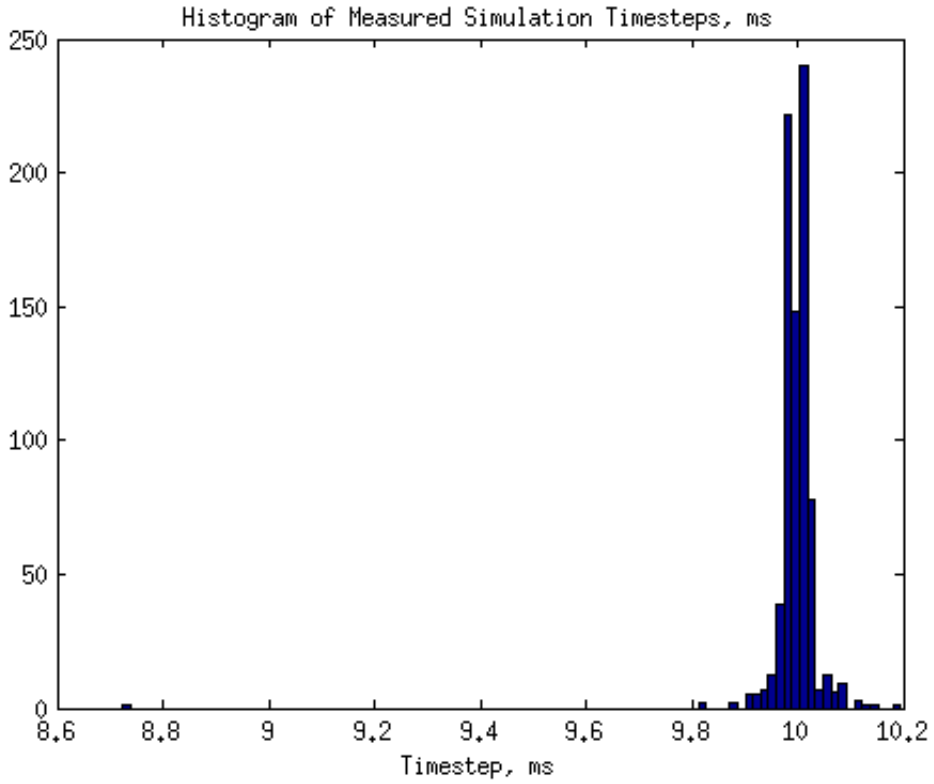

Figure 3.6: Timing Deviation of OpenHubo acting as a motion data source for Hubo-Ach

(URDF) using the SolidWorks URDF Exporter[1]. Since the models are exported with detailed features, a convex decomposition was applied to each rigid link to remove internal features and simplify tessellation at the surface of the model. The purpose-built stl-vrml toolbox for MATLAB[2] was used to manually decompose each rigid link into convex sub-bodies. The individual STL files for a given link were then combined and compressed into a single model for use in the robot model.

3.2.2 Modeling Servo Actuators

The Open Modular Robotics Platform [46] provides a simple servo implementation for OpenRAVE[3], allowing proportional control of a robot's motorized joints. At each simulation timestep, the controller calculates a desired joint velocity (3.5) from the joint error and a constant proportional gain. This desired input velocity is then passed as a command to the joint's velocity motor. The velocity motor then applies torque up

[1] http://wiki.ros.org/sw_urdf_exporter
[2] https://github.com/robEllenberg/stl-vrml-toolbox
[3] http://www.iearobotics.com/

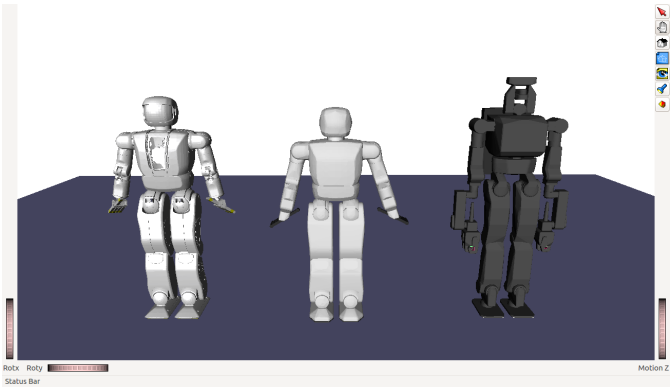

Figure 3.7: Hubo2, Hubo+, and DRC-Hubo robot models in OpenHubo.

to the joint's maximum to produce the desired velocity in the next timestep.

$$\omega_d = K_p(\theta_d - \theta_a) \tag{3.5}$$

This velocity motor is commonly used as a joint actuator in ODE-based simulations. It assumes that it is possible to achieve any desired torque up to a specified limit within one simulation timestep. Essentially, this means that the joint is very stiff to perturbations, since there is artificially high torque-control bandwidth. The authors of ODE claim that for a highly gear-reduced servo, this assumption is acceptable due to gearbox friction and other factors.

This basic assumption was tested by a step response of the elbow joint from the homes position. As shown in Figure 3.8, there significantly greater overshoot and settling time of the servo model compared to the physical servo.

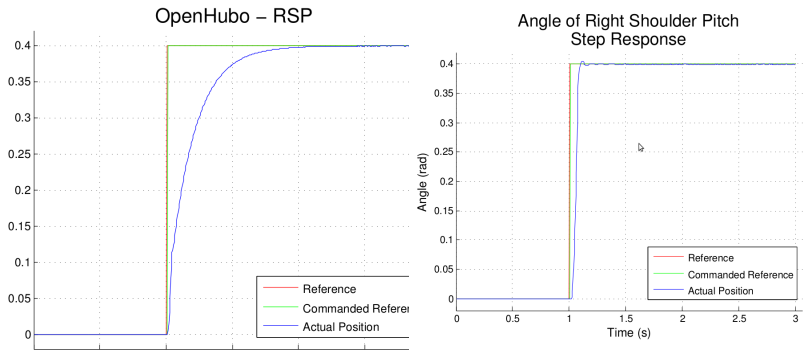

Figure 3.8: Step responses of OpenMR servo gains vs actual Hubo+ for the Shoulder Pitch Joint.

To correct the mismatch, the servo controller plugin of OpenHubo implements the following improvements over the original servo controller. Full PID control of either motor torque or velocity is implemented with user-controllable gains. To reduce the effects of past error on the integral of error, an optional decay constant K_f is added to the integral of joint error E (3.6). A first-order discrete lowpass filter is similarly implemented on the derivative term to reduce perturbations due to numerical errors. The servo gains in (3.8) were then adjusted to match response time and overshoot of the shoulder pitch joint.

$$E_{k+1} = E_k \delta t + E_k K_f \tag{3.6}$$

$$\dot{e}_{k+1} = (e_{k+1} - e_k)(1 - K_a) + \dot{e}_k K_a \tag{3.7}$$

$$K_p = 150, K_d = 0.9, K_i = 0, K_f = 0.9998, K_a = 0.1 \tag{3.8}$$

3.2.3 Force Sensing with ODE

To effectively simulate a humanoid robot during whole-body motions, contact forces are quite important to measure. Contact forces are used to determine stability using the ZMP criterion of [87]. If the humanoid is modelled as a single linearized inverted pendulum, then the zero Moment Point (ZMP) represents a fictitious point through which the sum of contact forces must pass to produce a given acceleration of the lumped mass. If this point lies within the convex hull of ground contacts, then the robot is stable.

To properly simulate a force/torque sensor, however, it is not enough to simply report the simulated forces acting on a body. On a real robot, the force/torque information will have a bandwidth determined by the sensor's construction and electronics, as well as coupling between measurement axes. The 3-axis force/torque sensors in the Hubo2+ (Figure 3.9) measure torques by the difference in strain between bars a/b and c/d, while axial force Fz is measured by the sum of these strains. A decoupling matrix is manually determined by applying a series of known loads to the sensor, and recording each raw measurement. The coupling matrix is then estimated as the least-squares error minimum between the sensed values and the known applied loads.

The ODE API provides access to the integrators of force and torque for each body used during the simulation step. During the update at each timestep, applied forces from directly applied loads, joint motors, and contact are all summed together to generate a net applied force and at the body's center of mass. Critically, the only feedback mechanism provided to the user is the function dSetJointFeedback(), which allocates a

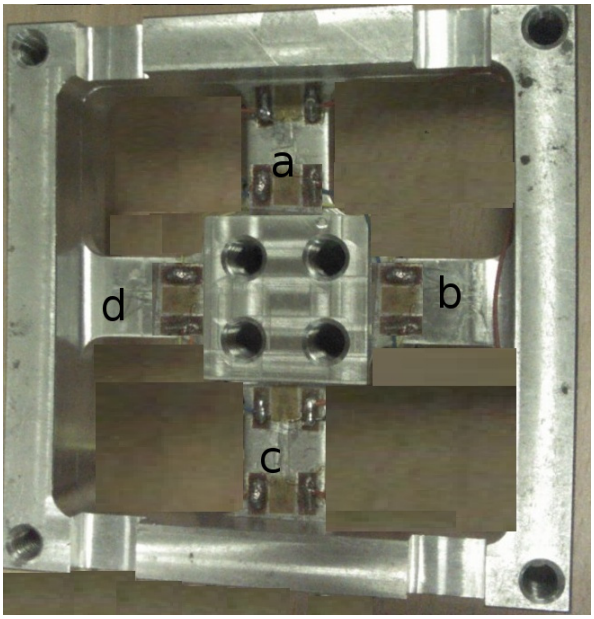

Figure 3.9: The construction of a KAIST/Rainbow force torque sensor. Four uniform beams a,b,c, and d join the outer and inner frames. The outer frame mounts to the arm/leg, while the inner frame attaches to the hand/foot.

data structure that stores forces and torques applied to each body attached to a given joint. Because contact joints are created and destroyed sometimes within a single timestep, querying this feedback from contact joints becomes expensive in terms of dynamic memory allocations. Therefore, the strategy employed by the simForceSensor plugin from OpenGrasp[4] queries only forces created by joints within the robot's structure. This solution alone is effective for determining forces between two bodies of the robot, but does not directly expose forces do to the temporary contact joints from the simulation.

To measure contact forces in OpenHubo, the structure of the robot model was modified to take advantage of this. At each end effector, the interface surface (palm of hand or sole of foot) was modeled as a distinct body from simple primitives (Figure 3.10). This body was joined to the end-effector body with a dummy joint, which constrained all motion between the end effector body and the interface body. Because this interface body represented a physical part of the robot, it was assigned the equivalent mass and inertia of those parts as determined by the CAD model. Now, any contact forces on this body are transmitted rigidly to the end-effector body through the dummy joint.

One additional step is necessary to model a discrete force/torque sensor. The joint feedback structure returns applied force and torque in the local frame of each body attached to the joint. A force/torque sensor

[4]http://opengrasp.sourceforge.net

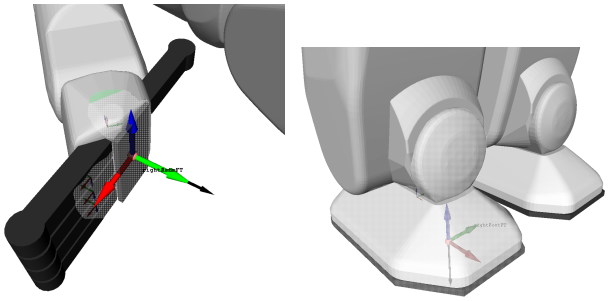

Figure 3.10: End Effectors of OpenHubo model showing end-effector bodies for hand and foot (light) and interface body (dark).

will be mounted at some offset from this joint, however. From the CAD model, the position offset of the FT sensor axis is measured compared to the inner (parent) body, which is then used as the dummy joint anchor. The equivalent wrench applied to the parent body is found at this new location in (3.9). Despite the frequent variation in contact locations, the net wrench is always transmitted rigidly through the FT sensor and dummy joint.

$$W_{FT} = [I_3 [d]_X] W_G \tag{3.9}$$

3.2.4 Scripting with Python

Included with OpenHubo are a set of python modules that further extend the analysis capabilities over the C++ plugins. Table 3.3 shows included example scripts and their purpose. Thanks to the extensive python bindings available in OpenRAVE, almost all of the robot and simulation is exposed to the scripting engine. This allows users to very quickly create and utilize novel features. The center of pressure example, for instance, shows how the robot's force sensors can be queried, converting that data into an estimate of the center of pressure at each foot. The use of both python elements for rapid prototyping and C++ plugins for time-critical code allows features to be developed by the average user. Useful functions that have been proven through practical use can then be converted to native C++ functions, allowing the developers to concentrate on implementation and testing. A minimal python example (Figure 3.11) shows how little user code is required (under 20 lines) to load the simulation and execute a simple trajectory.

3.2.5 Simulation Structure and Processes

Figure 3.12 shows the overall structure of the OpenHubo simulation environment. OpenRAVE provides the backbone of the system, providing libraries and data structures for kinematics, planning, and simulation

Table 3.3: Example scripts included with OpenHubo

servocontrol.py	Gain and position reference commands
trajectorycontrol.py	Building a waypoint-based trajectory
achread.py	Display desired and actual joint states from hubo-ach
measure_forces.py	Force/torque sensor use and tuning
ladder_grasp.py	Using CoMPS plugins for grasp planning

Figure 3.11: Minimal Python example code for OpenHubo to load and pose a robot.

```
import openhubo as oh
from numpy import pi

#Configure environment and load command-line options
(env,options)=oh.setup('qtcoin',True)

#Define robot XML file (defaults to Hubo+ if unspecified)
options.robotfile='huboplus.robot.xml'

#Define scene XML file for simulation environment
options.scenefile='simplefloor.env.xml'

#Load simulations scene from options specified
[robot,ctrl,ind,ref,recorder]=oh.load\_scene(env,options)

#Begin simulation thread
env.StartSimulation(oh.TIMESTEP)

#Define the pose class to interact with the robot
pose=oh.Pose(robot,ctrl)

#Update the desired pose using name-lookup
pose['REP']=-pi/2
pose['LEP']=-pi/2

#Send the updated pose command to the robot
pose.send()

#Read the actual pose values from the robot at any time.
pose.update()
```

execution. Dynamic simulation is provided by the ODE plugin oderave, which synchronizes the simulation environment between OpenRAVE and ODE. The collision detection library selected for OpenHubo is provided through the libpqp plugin to OpenRAVE.

For a single simulation step, a collision check is triggered, building a list of temporary contact joints at that time step. In parallel, the servo controller produces an updated velocity command for each velocity motor. Coupled with the current position and velocity of the robot, this dataset is used to set up initial conditions for the QuickStep solver in ODE. QuickStep iteratively solves the Newton-Euler equations of each bod, subjected to the kinematic constraints imposed by the body and contact joints. The result is an updated position and

velocity dataset, which is sent back to the OpenRAVE environment to update the simulation and display.

W

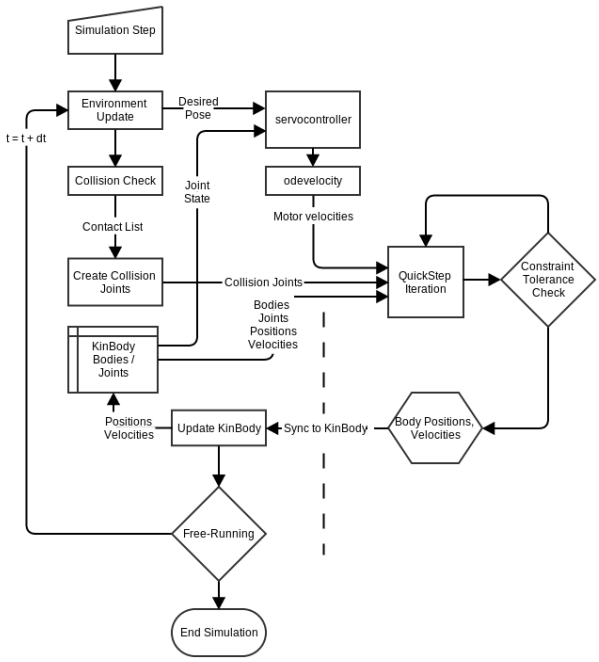

Figure 3.12: OpenHubo Block diagram showing data flow during dynamics simulation step

3.3 Validation of OpenHubo

The most valuable use of simulation is as a predictor of the hardware's performance. Therefore, the simulation's results must be specifically validated against the hardware. For a robot with n rotary DOF, major parameters affecting simulation accuracy are listed in Table 3.4. Due to the scale of the problem, it is impractical to validate all of the parameters in a single experiment. When validating mass distribution, for example, an error could be due to incorrect mass, incorrect center of mass, or a combination thereof. For a single pose, the problem is grossly under-constrained.

Table 3.4: Major parameters affecting simulation results for an exactly known environment

Parameter	Abbrev.	Dimension	Static	Dynamic
Mass	m	n	y	y
Inertia	I	$6xn$	n	y
CoM	$\vec{c}$	$3xn$	y	y
Effective Joint Stiffness	k_i	n	y	y
Effective Joint Damping	b_i	n	n	y
Effective Joint Friction	μ_i	n	n	y
Maximum Joint Torque	τ	n	y	y
Global Friction	μ_g	1	y	y
Collision Stiffness	ERP	1	y	y
Collision Damping	CFM	1	n	y

3.3.1 Isolating Servo Parameters

Within the normal operation of the Hubo's servo motors, input signals are band-limited to be under 2.5Hz due to a low-pass filter implemented within hubo-ach. While faster motions are possible by bypassing this filter, the remainder of this section assumes that this filter is in place. The majority of controllers and locomotion strategies do not implement high-bandwidth motions due to the stiffness of the system. The remainder of this section focuses on this "normal operation" region.

While a step response is typically used to identify system parameters, the high forces and potential for damage make this an impractical approach for characterizing joints such as the hips and shoulders. The high inertia seen at these joints due to swinging limbs can create potentially unsafe torques, potentially damaging equipment. To prevent stress and damage to the servos, the test signal was formulated as a randomized Gaussian signal with a cutoff frequency of $6Hz$ to satisfy the Nyquist Criterion. Signal amplitude was capped by the maximum allowed velocity, acceleration, and torque, listed in Table 3.5. Each experiment consisted of a $5sec$ cubic-smoothed ramp-up, a $60sec$ test period, and a $5sec$ ramp-down to rest.

For upper body joints, the Torso was fixed to a steady surface to reduce body sway. Data was recorded directly from the internal state (ACH channel) at $200Hz$, which captures the current reference and encoder position. A round-trip latency of to $0.005sec$ to $0.015sec$ was expected. The input delay can be seen in the difference between the commanded position (blue) and motor reference position (green) in Figure 3.13.

After collecting the response data, ARMAX models of 2nd, 3rd, and 6th orders were generated to determine the necessary complexity to approximate the system behavior. The frequency response of three of these models are shown in Figure 3.14. Each model had a nearly identical fit approximation of 95.1%, indicating that the simplest model provides a usable approximation of a single joint's behavior at low frequencies and amplitudes. The linear model is only valid in the neighborhood of the tested configuration. However, if lin-

Table 3.5: Servo System ID

Joint	Amplitude (deg)
Waist	5.7
Shoulder Pitch	5.7
Shoulder Roll	14.3
Shoulder Yaw	14.3
Elbow Pitch	7.1
Wrist Yaw	14.3
Wrist Pitch	7.1
Wrist Roll	7.1
Hip Yaw	5.7
Hip Roll	2.8
Hip Pitch	7.1
Knee Pitch	4.6
Ankle Pitch	5.7
Ankle Roll	4.5

ear approximations at known starting angles can be matched between the simulation and hardware, then the non-linearity will be modeled by the physics simulation.

3.3.2 Static Stability Bounds

To evaluate the mass distribution, the robot model was posed at the edge of stability by tilting at the ankles, knees, hips, and shoulders. By finding the angle when the robot begins to fall, isolate the effects of center of mass and mass value. Table 3.6 shows the initial comparison of the hardware and simulation at extreme poses.

Table 3.6: Static Stability Bounds

Pose	Joints	Hardware Bounds (rad)	Simulated Bounds
Ankle Tilt	LAP,RAP	+/- 0.14	+/-0.17
Knee Bend	LKP,RKP	+/-.2	+/-.21
Hip Tilt	LHP,RHP	+/-1.0	+/-1.2
Shoulder Pitch swing	LSP,RSP	+/-3.6	+/-3.6

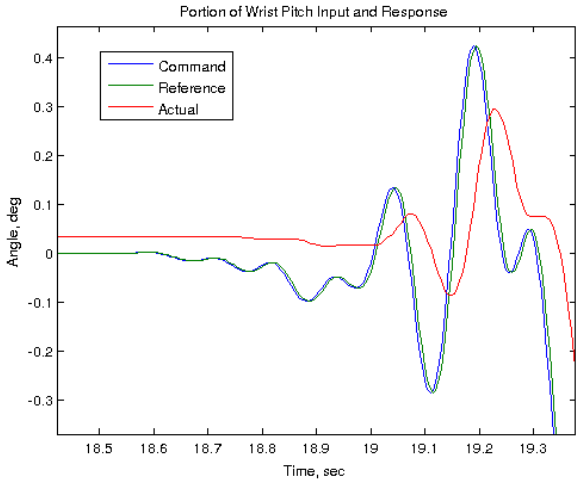

Figure 3.13: Example input and response for Wrist Pitch Joint

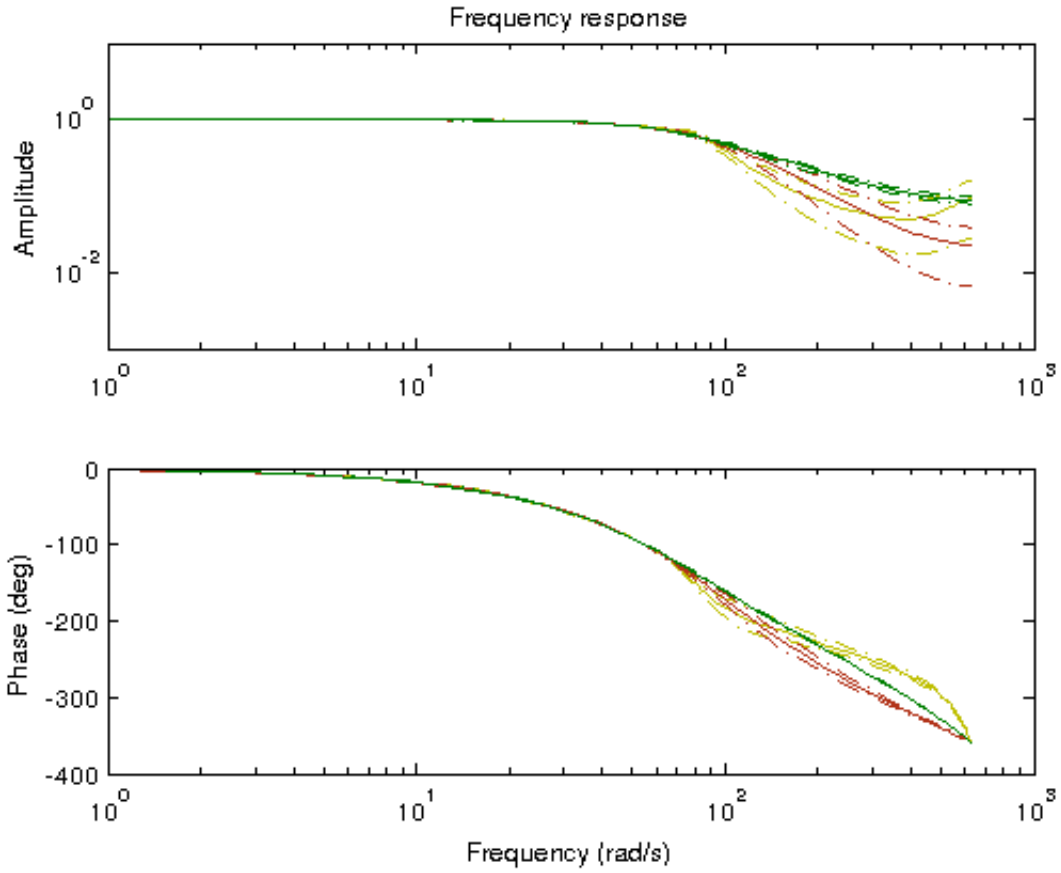

Figure 3.14: Comparison of Frequency response of 2nd-order, 3rd order noise-free, and 6th order ARMAX models.

4. Stair Climbing Experiments

Given the underrepresentation of ladder-climbing work in literature, we first started with an easier problem that was better represented: stair climbing. This seemed like a good starting point because of the overlap between stairs and ladder designs. For example, a 60° ship's ladder (Figure 4.1) has vertically-spaced flat treads and hand-rails just like a typical set of stairs. These similarities suggest that stair climbing methods could be applicable to ladder-climbing, and extend the useful range of humanoids to handle these cases.

Figure 4.1: Examples of ship-style ladders with a wheeled base (left) and fixed base (right).

In this chapter, the CBiRRT algorithm [6] was applied to the stair-climbing problem. The goal of the experiment was to plan statically stable stair-climbing trajectories using manually specified goal regions for each limb of the Hubo2 humanoid. Planning performance was evaluated in OpenHubo. A successful plan consists of a series of whole-body motions that are statically stable at every point, and do not collide with the staircase during the motion. Sections 4.1 explores the current state of the art in stair climbing and randomized motion planning. Section 4.2 explains the motion planning implementation and methods of evaluation. Finally, section 4.3 documents the performance and limitations of the CBiRRT method.

4.1 Literature Review

4.1.1 Stair-Climbing

Stair climbing has been demonstrated in many robots, from purpose built-quadrupeds [31] [22] to miniature humanoids [58], to full-size humanoids [23]. The typical research problem solved is aimed at consistent and obstacle-free stairs that can be climbed with a periodic motion that is stable and energy-efficient.

Early stair climbing work was attempted using a variety of strategies. In [71], a biped robot consisting of 7 DOF legs could walk up stairs using a statically-stable trajectory and an actuated balance mass representing net motion of upper body limbs. Notable advancements have been made in dealing with non-ideal stairs. [99] and [35] proposed a ZMP-based methods for stair-climbing with sloped steps. [26] demonstrates the additional use of a hand-rail to stabilize a humanoid against tipping due to sloped stair lands.

Dynamic walking can deal effectively with certain disturbances, but becomes increasingly challenging when the desired motions cease to be periodic. The advantages of dynamic walking such as energy savings and reduced hip motion amplitudes may be offset by the difficulties of planning effective transitions between drastically different foot goals. A dynamic trajectory needs to anticipate the next foot position in advance so that the hip can move continuously. Achieving dynamic walking in conditions such as steep staircases, discontinuous or sloping ground, bumpy terrain, etc. requires additional planning and/or control.

Given the potential of stair-climbing as a simplified version of ladder climbing, the first step was to reproduce a ZMP-controlled walking gait modified for stair-climbing use. In [38], stair-climbing was successfully shown using both the Mini-Hubo and Hubo2 robots. By planning for static stability, open-loop trajectories were demonstrated to be stable on stairs of approximately 30°. However, the fine tuning required meant that modelling errors of as little as 2° could cause the robot to topple. Later work dealt with improved trajectory planning [37], but these early results indicated that sensing capability was paramount in achieving robust climbing.

4.1.2 Rapidly-Exploring Random Trees and Motion Planning

Motion planning with parameterized trajectories was limited to known, easily parameterizable environments. However, real world environments such as stairs, curbs, and ramps are not always consistent or flat. To handle these more complex cases with parameterized trajectories drastically increases the number of parameters as the environmental variation increases. Probabilistic planners address this problem by searching an environment by randomized exploration, producing trajectories based on the results of a search. While these trajectories are often suboptimal in terms of performance criteria such as energy, joint torque, or motion

time, these methods can handle environments of arbitrary complexity.

Rapidly exploring random trees are one such method, which in general can explore any vector space efficiently. [49] introduced the concept in 1998, explaining in detail the algorithm, along with simple examples. The RRT was immediately useful for planning the trajectories of simple, low DOF robots. In [1], the authors plan motions of a skid-steering robot, taking into account limitations of both the position of the robot and its velocity. Similar kinodynamic planning was demonstrated in [48], demonstrating solutions for classic problems like the "Piano mover" problem, and motion planning for car-like robots with non-holonomic constraints. The randomized search finds a suboptimal solution while searching only a fraction of the total space. A^* and similar algorithms, by contrast, exhaustively search a potentially very large space for an optimal solution. For problems like the moon-lander and piano-mover, RRT methods produce solutions that respect bounds, such as fuel consumption or time.

A simple RRT planner has a number of limitations, however. The tree is rooted at a start configuration, and expands in all directions. If the search is totally random, the volume the tree must fill increases polynomially with the dimension of the search space. Sampling from both goal configuration and random poses in [50], the planner was able to converge to a goal while exploring a smaller volume. [3] introduces the bidirectional RRT search, which grows trees from both the goal and start configurations of a manipulator, improving convergence on "hard-to-reach" goals. To simplify the definition of a planning problem, a Jacobian Transpose method was used to grow the tree towards a task-space goal in [84], and later in [89] by directly inverse kinematic solution.

Unfortunately, the RRT methods mentioned so far all share the same limitation. If the goal configuration is difficult to reach, either by being far from the initial pose, or near a joint singularity, the solution time will drastically increase. One way to improve the robustness of the solution is to choose more than one goal configuration. In many cases, multiple goal configurations may be permissible. In [4], the authors propose the Workspace Goal Region, which identifies a volume in the 6 dimensional space of end-effector position and orientation that constitutes a valid goal region. This concept is refined and extended in [5] and [6], which formalize Task Space Regions (TSR's) as a way to specify a goal for a manipulator. A region in the space of the end-effector is identified by a goal transformation matrix, and set of allowable deviations in translation and rotation from this goal. If the end effector can be rotated about one axis, as if it was grasping a cylindrical object, then bounds of rotation about that axis could be expanded to allow more potential goal positions.

A significant percentage of computation time in RRT methods is the evaluation of each sampled pose. For every new pose found, it must be checked for world and self collision, and support. While the resulting trajectories are guaranteed to be collision-free within the accuracy of the model, this can become very

Property	Symbol	Value Range
Slope angle	θ	$30^\circ - 50^\circ$
Tread depth	d_t	$20cm - 28cm$
Tread rise	d_r	$16cm - 24cm$
Tread width	d_w	$56cm - 112cm$
Railing height	h_r	106cm

Table 4.1: Standard OSHA Industrial Stair dimensions per Standard 1910.24

computationally expensive, in some cases contributing over 90% of the computation time. Several methods have been proposed to address this limitation, including using vector processors such as GPU's [44], or lazy collision checking using approximate robot shapes [81]. Selectively reducing the set of active joints as in [83] can even allow simple tasks to be planned online. More recently, [7] showed that RRT planning could re-use and repair existing trajectories, massively speeding up planning time for previously planned motions. Therefore, RRT-methods have shown potential for real-time replanning, an important feature for a ladder climbing controller.

All of the RRT methods cited so far are only as effective as the quality of the pose check they use. While [5] introduces a support-based stability check, no RRT-based planner incorporates the more general contact wrench methods established in section 2.1. Other probabilistic planners are discussed in subsection 5.1.1 that incorporate more general stability metrics.

4.2 Motion Planning

4.2.1 Stair Parameterization

Based on the safety requirements published by the US Occupational Safety and Health Administration [69], a typical staircase in an office or industrial environment must satisfy the requirements listed in Table 4.1. These dimensions of a typical staircase, depicted in Figure 4.2 are designed to fit the range of motion of an average human. While the OSHA standards clearly specify what a typical human-scale staircase is, the Hubo2 robot is approximately $3:4$ scale. Therefore, the prototype staircase was scaled down by the same factor. The resulting staircase dimensions are given in Table 4.2. The resulting dimensions were then used to model a simple staircase of geometric primitives using OpenRAVE XML format (Figure 4.3).

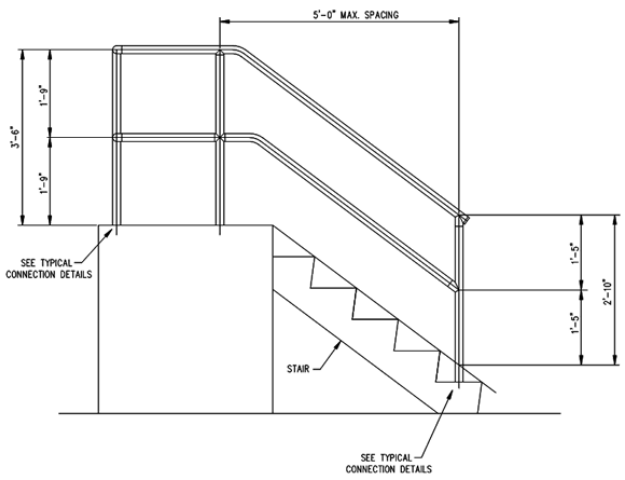

Figure 4.2: Handrail dimensions for an OSHA-compliant staircase.

Dimension	Scale Factor	Model Value
θ	1.0	30°
d_t	0.79	$22.1cm$
d_r	0.79	$19.5cm$
d_w	1.0	$80cm$
h_r	0.65	$65cm$

Table 4.2: Dimensions of simulated staircase, reduced from OSHA standards by the Hubo2 scale of approximately 3 : 4.

4.2.2 Motion Sequence

The motion planner divides the stair climbing problem into five distinct phases, as shown in the frames of figure 4.4:

1. Shift body weight over the right foot in anticipation of lifting the left foot.
2. Plan an arm motion with the TSR specified in (4.3) to place the right hand on the handrail.
3. Plan a step motion to place the left foot on the next step land. The torso is constrained to the bounds specified in (4.4).
4. Given the resulting foot locations, shift the body weight over the left foot, while maintaining foot and hand placement.
5. Plan a second arm motion to relocate the right hand further along the handrail.

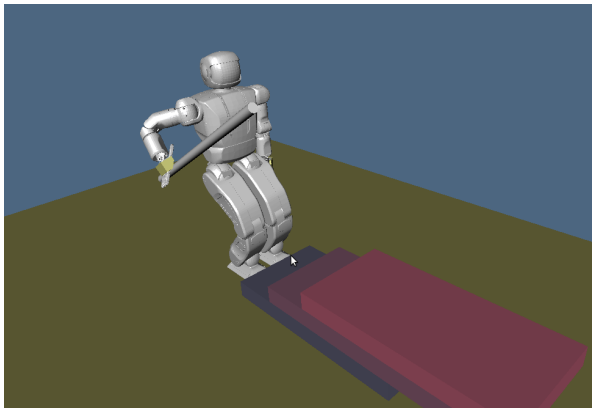

Figure 4.3: The Hubo2 humanoid posed in OpenHubo on scaled stair model.

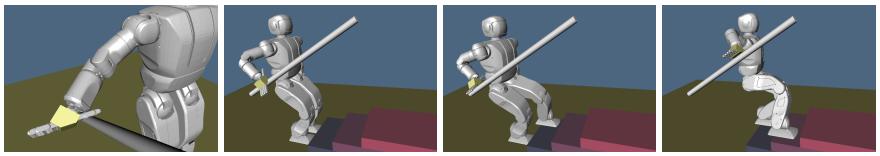

Figure 4.4: End poses of a particular test run of the statically-stable stair-climbing phases

6. Plan a step motion with the right foot to the same step, completing the cycle.

The motivation for dividing the climbing phases in this way is due to the design of the CBiRRT algorithm. For a given planning problem, the support condition must be consistent. This means that each contact transition, such as lifting a foot or grasping the handrail, requires a separate planning phase. This segmentation allows the CBiRRT algorithm to plan a motion towards a transition phase, and another planning cycle to plan the following motion.

For a given pair of motions, the challenge is therefore to find a goal pose that is valid both in the current and next phase. For example, when both feet are on a step, there is a wide range of possible stable poses. However, only a small fraction of them are also stable if one foot is raised. As shown in Figure 4.5, taking a step forward or moving the hand along the handrail changes the support polygon. To pass from the first phase (red) to the second phase (blue), the center of mass must be posed in the in the overlapping region (magenta) during the transition.

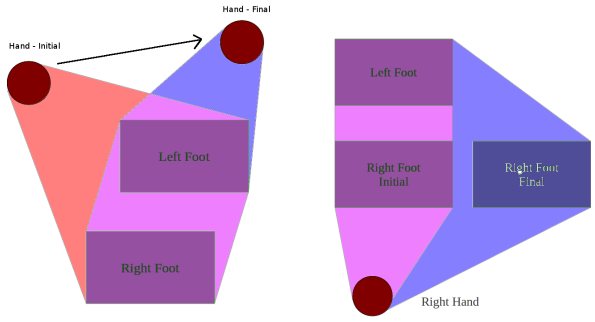

Figure 4.5: The support polygon, or the region over which the center of mass can lie, changes as a hand is lifted and re-placed in another location (left), and as the right foot is lifted and placed forward (right).

4.2.3 Hand and Foot Placements

Foot placements were specified as goal regions at each stair land 20*cm* wide for each foot, the depth of each land, and 2*mm* above the stair land. The slight gap between the stair land and goal region prevented planned goals from colliding with the stair land.

To approximate grasping hand-placement, the grasp is assumed to align with the cylindrical hand-rail. To prevent collisions with the handrail, the ideal hand goal position is specified as a neighborhood around the radius of the handrail. The workspace coordinate system associated with the handrail places the Z axis along the length of the rail, and the x axis aligned with the global XZ plane. To allow the palm to slide along the handrail and rotate around it, the bounding volume is specified in (4.3) with large deviations in the z displacement and rotation.

In general, the initial, goal, and motion constraints are specified using a workspace transformation (4.1) and the Bounding Volume (4.2). The workspace transformation is an approximate center of the goal region, specified in a convenient set of workspace coordinates. The bounding volume specified the range over which the translation (3D) and Euler angles can vary when sampling from that constraint. If the bounding volume is specified as all zeros, then the sampled region is a single point. If the maximum and minimum for a given parameter are very large, then the volume is effectively unconstrained in that dimension of the task space.

$$\mathbf{T}_w = \begin{bmatrix} \mathbf{R}_w & \mathbf{t}_w \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{4.1}$$

$$\mathbf{B}_w = \begin{bmatrix} x_{min} & y_{min} & z_{min} & \psi_{min} & \theta_{min} & \phi_{min} \\ x_{max} & y_{max} & z_{max} & \psi_{max} & \theta_{max} & \phi_{max} \end{bmatrix} \tag{4.2}$$

$$\mathbf{B}_w = \begin{bmatrix} -r & -r & -0.5 & 0 & 0 & -\pi \\ r & r & 0.5 & 0 & 0 & \pi \end{bmatrix} \tag{4.3}$$

$$\mathbf{B}_w = \begin{bmatrix} -0.2 & -0.2 & -0.1 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.1 & 0 & 0 & 0 \end{bmatrix} \tag{4.4}$$

4.3 Results and Lessons Learned

The results of 50 simulation trials are given in Table 4.3.

Table 4.3: Stair Climbing Planner Results

Progress	Successful Trials	Percentage
IK solution for first grasp	41	82%
First hand rail grasp	38	76%
Left foot step	30	60%
Torso shift	14	28%
Right Foot step	8	16%
Right hand re-grasp	5	10%

The climbing sequence shown in 4.4 that whole-body planning is possible using the CBiRRT algorithm. However, the planner as designed results in a significant percentage of planning errors. A common error encountered in approximately 18% of simulations showed that the transition motion failed to bias the body enough to the right to complete the move. While the stability constraint was obeyed in all of the transition moves, there remained a wide range of possible poses that represented a successful goal. Unfortunately, the motions as planned lead to a few critical limitations. One issue is with the reachability constraints of the Hubo2 robot. The wrist pitch and roll joint can tilt approximately 22° in each direction. Because of this narrow range, the IK solution was sensitive to the initial position of the torso relative to the hand-rail. Also, because the center-of-mass support check did not account for grip force, supporting poses are biased towards the grasping hand.

5. Ladder Climbing

This chapter introduces a ladder-climbing planner for the Hubo+ humanoid. The planner uses a numerical inverse kinematics solver to interpolate between motion primitives, and a bisection-projection algorithm to correct for environmental collisions during a planned motion. This work, published in [94] and [95], was collaboration with Prof. Kris Hauser, Yajia Zhang, and Jingru Luo of Indiana University, and with H. Andy Park, Manas Paldhe, and Prof. C. S. George Lee of Purdue University.

5.1 Background

5.1.1 Climbing Motion Planning

Robots that are capable of scaling walls and rocky faces have been in development for many years. For example, Kim et al. developed a robot that can climb on a flat and smooth surface using a hand and foot similar to a gecko [45]. Bretl et al. studied multi-step motion planning for a vertical rock climbing robot [10, 12]. For ladder climbing specifically, special-purpose robots have been in development since 1989 [36], in which Iida et al. developed a ladder-climbing robot called LCR-1 for vertical ladder climbing [36]. Yoneda et al. developed a vertical ladder-climbing humanoid robot that is specially designed for climbing; it uses hook-like hands to maintain balance during the climb, and force sensors to detect whether a rung was successfully grasped [93]. Unlike general-purpose humanoids, these special-purpose climbing robots have limited or nonexistent capabilities to perform manipulation, stair-climbing, and transitioning from flat ground to ladders and vice versa. However, the ability to verify that a hand-hold has been achieved is important with hands as well, since a grip can fail under high loads.

For a successful ladder climbing, motion planning for a sequence of hands-feet placement must be performed to detect collisions between the robot's body and the rungs of the ladder as well as gripping forces/-torques for the hands and the stepping forces/torques of the feet on the rungs. Recently, Hauser et al. [28–30] have extended motion planning to humanoid robots with multi-limb contacts to achieve balance.

5.2 Open Loop Experiments in Simulation

An open-loop ladder-climbing planner was recently published in [94], designed to plan ladder climbing motions for an arbitrary humanoid robot and a parameterized ladder. The planner takes several things into consideration:

- A sequence of limbs to be placed and removed against the terrain.
- Contact points and orientations for those limbs.
- Joint-level trajectories showing the robot's poses that achieve those contacts while avoiding collision and respecting kinematic limits.
- Forces and torques that realize stable balancing at all points along the trajectory.

The concept behind the ladder-climbing planner in [94] was based on the motion primitive idea introduced in [28], that the building blocks of climbing are single motions that make or break a single limb contact. During climbing, up to four limbs will be in contact with the ground or the ladder to provide support. The concept of a *hold* as a region in which the geometry of a single robot link and environment touch. A list of holds yields a *stance*. While on the ground, a stance σ consists of two holds, and during climbing, a stance may consist of three or four holds (i.e., either two feet and a hand, two hands and a foot, or all four limbs in contact). To model a contact region, a finite number of point-contacts where the points $r_1,\ldots,r_k$ on the robot touch the points $x_1,\ldots,x_k$ in the environment, with contact normals $n_1,\ldots,n_k$. For simplicity, hand holds were treated as two point-contacts — one vertically-oriented to allow the hand to push down, and one horizontally-oriented to allow the fingers to pull back — and foot holds on a rung with one vertically-oriented point-contact. Contact friction was modeled as Coulomb friction with a coefficient of 0.4.

5.2.1 Planner Details

A feasible robot configuration q at a stance σ must satisfy the following constraints:

1. Contact constraints: the points $r_1(q),\ldots,r_k(q)$ meet the points $x_1,\ldots,x_k$, for all holds in σ.
2. Within joint limits: $q \in [q_{min}, q_{max}]$.
3. Free of collision with the environment.
4. Free of self-collision.
5. Stable under gravity: the center of mass of the robot should lie within the support polygon formed by the supporting limbs. Note that with uneven contacts, a support polygon does not necessarily correspond to the convex hull of the projection of the contacts. Bretl's method was used to compute this polygon [11].

A solution to the ladder-climbing planning problem is a sequence of stances $\sigma_1, \ldots, \sigma_n$ and a continuous sequence of single-step paths of feasible configurations $p_1, \ldots, p_n$ that are feasible at their respective stances. Note that at the transition between stances σ_i and σ_{i+1}, the robot must pass through a configuration that meets the constraints at *both* stances.

A ladder climbing motion sequence was decomposed into motion primitives which

1. placeHands: place two hands on a (chosen) rung.
2. placeLFoot: place left foot on the first rung.
3. placeRFoot: place right foot on the first rung.
4. moveLHand: lift left hand to the next higher rung.
5. moveRHand: lift right hand to the next higher rung.
6. moveLFoot: lift left foot to the next higher rung.
7. moveRFoot: lift right foot to the next higher rung.

A motion primitive typically causes a limb to be removed and then placed it at a new location, forming two stance changes. We utilize primitives 1–3 to mount a ladder, and primitives 4–7 are then repeated to climb the ladder for a desired number of rungs.

Each motion primitive is designed to contain prior knowledge for solving that portion of the climbing task. It contains an "ideal" set of point-contacts, robot poses, and intermediate waypoints tailored to the action (see Fig. 5.2). In the current implementation, these are designed by a human expert. These values are used as *seeds* to help the planner avoid joint singularities when planning contacts and poses for novel ladders. Since the planner depends on optimization, a good starting point is critical to avoid local minima, and reduce the cost of optimization.

The initial climbing sequence uses primitives 1–7 to climb up two rungs. Climbing multiple rungs extends this process by repeating primitives 4–7. Contact poses are solved by a randomized sequential descent, in which each primitive is slightly perturbed from the seed values at random in order to help find successful solutions. To ensure that paths stay close to the seed primitives, the radius of perturbation starts at zero and increases upon subsequent iterations. This procedure is described by the following pseudocode:

1. Repeat until a solution is found, or a time limit is reached:
2. Let $q_0 \leftarrow q_{cur}$ and $\sigma_0 \leftarrow \sigma_{cur}$.

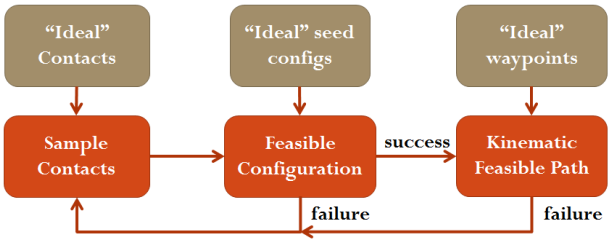

Figure 5.1: Three steps in motion planning for ladder climbing based on motion primitives. If one step fails, the planning process will trace back to the previous step. Prior information is being used to assist in each step.

3. For motion primitives, $1, 2, \ldots, 7$, do:

4. Sample a desired hold h_d near the seed hold.

5. Let σ_i replace the current hold in σ_{i-1} with h_d.

6. Sample a feasible destination configuration q_i at σ_i.

7. Find a feasible path connecting q_{i-1} to q_i at σ_{i-1}.

The innermost loop samples holds, configurations, and paths in that order (see Fig. 5.1). If any step in the innermost loop fails, the planner restarts from step 2. Each innermost sampling step is run for n samples, where n controls the balance of putting more effort on one action or backtracking to get a better start. In our implementation, n is set to 50 after tuning.

Starting from a seed configuration q_{seed}, we use a numerical inverse kinematics (IK) solver to obtain a configuration that satisfies IK and joint limit constraints. Moreover our planning system can retract slightly colliding configurations out of collision by solving a nonlinear constrained optimization process, similar to the Iterative Constraint Enforcement algorithm [29]. If this fails, a perturbation function is used to adjust q_{init} with perturbation drawn uniformly from 0 to some radius c, which is chosen empirically. The perturbation radius increases as the number of failures increases. The process stops until we find a feasible configuration or it reaches the iteration limit. Algorithm 1 describes a configuration-finding procedure.

To generate feasible trajectories that connect the starting and ending configurations of motion primitives, we add intermediate waypoints to avoid collision with the ladder rungs. These waypoints are solved by interpolating the endpoints of the moved limb along an arc in world space. Again, perturbations are used to push waypoints into the feasible space.

We also employ a contact-space interpolation strategy to smoothly connect these waypoints. Simple linear interpolations in joint space does not work because the robot fails to maintain contact at intermediate

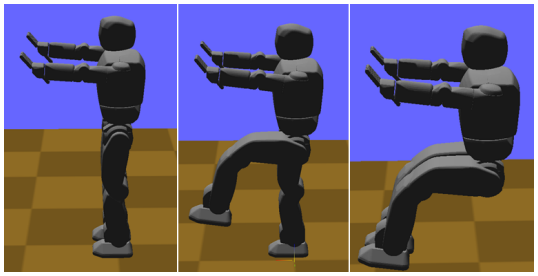

Figure 5.2: Seed examples. From left to right: placeHands, liftLFoot, liftRFoot.

Algorithm 5.1 Finding feasible configuration

```
for i = 0, 1,..., n: do
    q_init = q_seed + perturb(i)
    if find q from IK solver starting from q_init: then
        if no self- and no envr-collision and stable: then
            return q
        end if
        if no self-collision and has envr-collision: then
            if retract(q) succeeds and q is stable: then
                return q
            end if
        end if
    end if
end for
```

configurations (a problem known as foot-skate in the animation literature). Instead we use a recursive interpolation to ensure that that the supporting limbs remain in contact up to some user-defined resolution ε. Given two endpoint configurations, q_1 and q_2, this strategy first finds the middle point $q = \frac{1}{2}(q_1 + q_2)$ in the joint space, then calculates its projection q' in the contact space. The projection function uses numerical IK to find q', which satisfies the IK constraints that $r_1(q'), \ldots, r_k(q')$ meet $x_1, \ldots, x_k$. If successful, we then recursively solve two sub-problems: interpolation between q_1 and q', and interpolation between q' and q_2. The algorithm terminates the recursion once q_1 and q_2 are closer than ε.

5.2.2 RobotSim Results

Extensive computer simulations were performed to test the ladder-climbing motion planner on the Hubo+ robot model and a list of ladders of various specifications as input. Two parameters for ladders are considered: the rung spacing ranging from 20 cm to 35 cm with 1 cm increment, and the incline angle ranging from 70° to 90° with 1° increment. For each ladder, the motion planner was tested by utilizing all 7 motion primitives. All the experiments were carried out on an Intel Core i7 2.8 GHz machine with 4GB RAM. The planner was

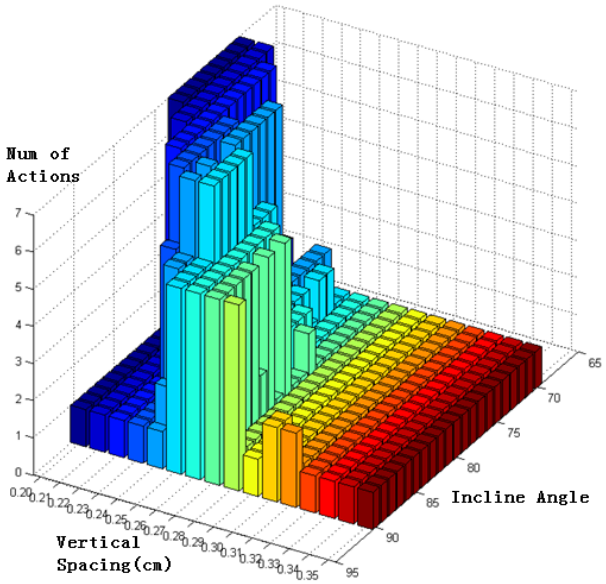

Figure 5.3: Computer simulation results on ladders with inclined angle between 70° and 90°, and the rung spacing between 20 cm and 35cm. The height of the bar indicates how many sequential primitive actions were successfully planned.

run on each ladder with a 60-second cutoff time.

Figure 5.3 shows the motion planning results for each ladder, the height of the bar indicates how many sequential motion primitives were successfully planned. Among all 756 ladders, 15.48% could be fully solved (i.e., succeeded in utilizing all 7 motion primitives) and 25.30% could be solved for all the mounting actions (i.e., the first 3 motion primitives).

By observing the cases that the motion planner failed in finding a solution to a specific ladder, it became clear that two parameters of Hubo+ robot play an important role: the joint limits of the leg pitch and the geometric size of the knee joints. Limited leg pitch prevents the leg lifting higher, which reduces the spacing requirement in a cluttered environment. The large knee joint frequently causes collisions on ladders with high inclination. To verify that these factors were significant, two additional experiments were designed:

1. Increase the leg pitch joint limits by 10°.

2. Shrink the knee joints by 1.5 cm (In practice, this probably can be done by removing the shells of the knee joints since they are not tightly designed).

3. Apply the above both changes.

Every experiment was carried out with other robot and ladder settings remained the same. The experiment

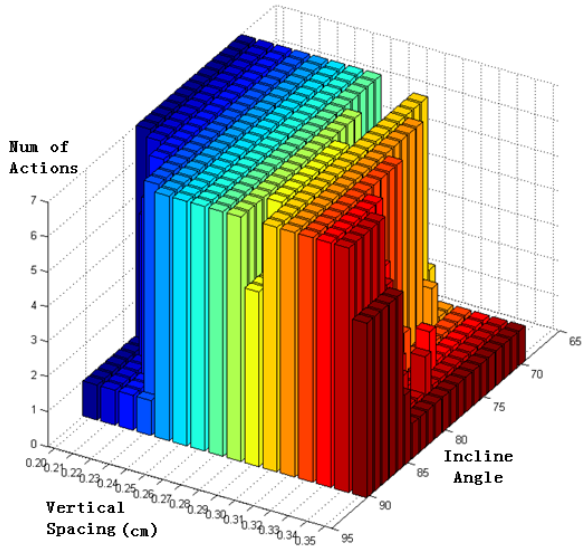

Figure 5.4: Batch test result by using modified Hubo+ robot model. The fully solved rate is boosted to 70.24% while mounting success rate is boosted to 73.81%.

results showed that by increasing the leg-pitch limit alone, the fully solved ladder-climbing ratio changed to 43.45%; by shrinking the knee joints by 1.5 cm alone, the ratio changed to 34.82%; by applying both modifications, the ratio changed to 70.24% (see Fig. 5.4). This demonstrated that our motion planner can be utilized to effect hardware re-design.

5.2.3 OpenHubo Simulation Results

The motion planner produces trajectories that satisfy kinematic and posture constraints of Hubo+ robot. To validate the planner, the OpenHubo platform was used to simulate a subset of these planned motions. The OpenHubo simulation package considers the dynamic behavior of Hubo+ robot as well as the effects of contacts and friction. If a planned motion is executed correctly with full physics considered, then the assumptions and simplifications of the kinematic model and constraints are more trustworthy, and future controller design can be tested and verified in OpenHubo before the controller is ported to the Hubo+ robot.

Enabling physics in the simulation reveals important difficulties in the ladder climbing motion that are not necessarily apparent from kinematic planning. For example, when multiple contacts are established on the ladder, the hands and feet are not always accurately placed, due to the weight of the robot. Position-controlled joints respond aggressively to small joint errors due to their high gains. Thus, in multi-contact climbing poses, incorrect placement of limbs creates extra force that must be cancelled by the other supporting limbs (see Fig. 5.5).

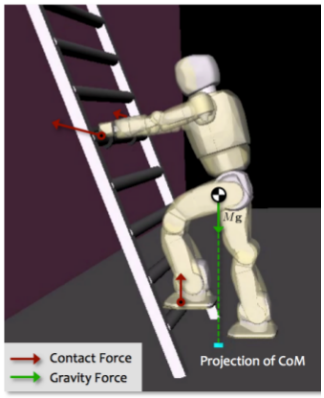

Figure 5.5: Contact forces produced by the Hubo+ robot's hands and feet.

Table 5.1: Predicted successes of select cases with a stock Hubo+ and the modifications recommended in our motion planner.

		Prediction		Result	
Angle	Spacing	Stock	Modified	Stock	Modified
70°	20 cm	7	7	1	7
70°	25 cm	1	7	1	4
75°	22 cm	7	7	1	4
75°	25 cm	1	7	1	4
80°	25 cm	7	7	1	4
80°	30 cm	1	7	1	4
85°	25 cm	7	7	1	2
85°	30 cm	1	7	1	2
90°	25 cm	7	7	0	0
90°	32 cm	1	7	0	0

Table 5.1 shows the set of ladders chosen for full simulation with OpenHubo. For each ladder, there are two independent improvements considered. The first is a 10° increase in each of the three leg pitch joint limits, and the second is the reduction in leg thickness equivalent to the removal of the plastic shells. Grip strength is assumed to be artificially large (10.0 Nm maximum finger joint torque) for this initial validation, while the ladder rungs were assumed to be cylindrical with a diameter of 4 cm. A best-case static friction coefficient of 2.0 was assumed for all contacts due to the use of soft rubber pads on the hands and feet of the Hubo+ robot. During climbing motions, Hubo+ robot was capable of reaching for the next rung and completing the motion for the 70° and 75° cases, but not for steeper angles.

The simulated Hubo+ robot was able to mount eight of the ten ladders only when using the slim legs and increased range of motion of the shell-less version of the robot. Without both of these improvements,

the robot's lower legs collided with the ladder, pushing the robot away and causing it to fall. Contact error was another factor – the large moment created by the robot hanging far from the ladder caused the robot to tilt horizontally away from the ladder, in turn causing the reaching motion moveLHand to fall short (see Fig. 5.6). The left foot also slipped noticeably backwards in these cases.

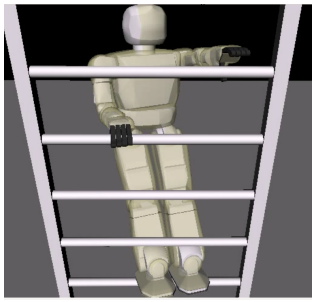

Figure 5.6: Joint limits cause the left foot to fall short (left), and climbing onto a 80°, 25 cm-spaced ladder causes a slip, leading to misalignment (right).

For the 90°-inclined ladder case, the fast swing-up motion of the arms added extra torque to the robot, tipping it backwards. Based on these results, the accuracy of hands and feet placements represents an important challenge for the physical Hubo+ robot. The robot must be able to detect an improperly placed hand/foot, or be able to detect when a hand or foot has slipped from the desired location.

For the most successful case, the 70° and 20-cm-spaced ladder, the previous experiment was repeated using a maximum finger strength corresponding to the actual Hubo+ robot. The current hands have been shown to support a weight of approximately 4.5 kg. Simulation of a hand grasping a weighted bar (see Fig. 5.7) showed that the fingers could support the weight of the bar with 1.6 Nm maximum torque on the thumb joints, and a maximum of 0.8 Nm at each finger joint. This measurement is not meant to exactly model each physical finger, rather, it is a simulation reference point that shows how much stronger the hands must be to successfully climb our chosen range of ladders.

For a 70°, 20-cm-spaced ladder, a complete climb was possible with a maximum finger torque of 4.0 Nm (see Fig. 5.8), or approximately twice the current finger strength. The 4 mounting placements and 3 climbing placements were completed in 31 seconds.

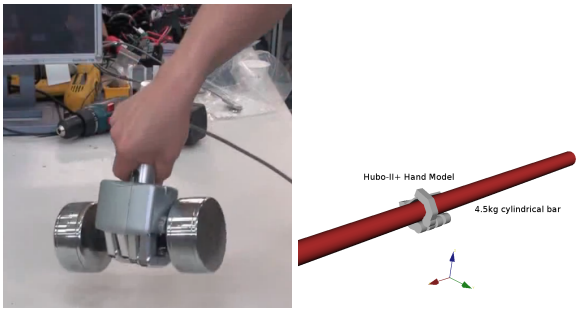

Figure 5.7: Physical Hubo+ robot hand supporting 4.5 kg (left), and a simulated grasp with finger torque of 1.6 Nm.

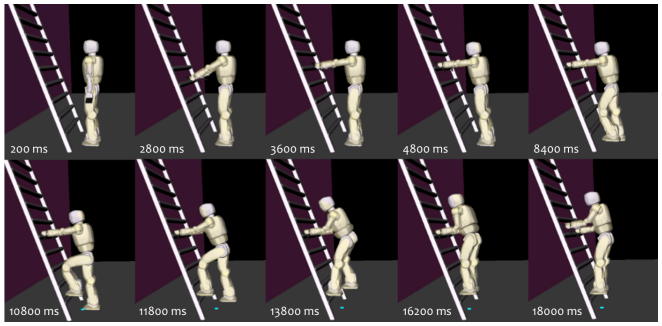

Figure 5.8: The Hubo+ robot climbed onto a 70°-inclined ladder and took a step, using maximum finger torque of 4.0 Nm.

5.3 DARPA Robotics Challenge Mock Trial Results

5.4 Results from DARPA Robotics Challenge

This work was further refined and eventually deployed in the DARPA Humanoid Robotics challenge [95]. In the ladder-climbing event, the DRC Hubo team scored two points out of a possible four for climbing six out of nine rungs on a 60° ship ladder. Despite the great improvements in planning accuracy and robot compliance, the robot was not able to complete the ladder climb due to a grip failure. The control team did not have proper feedback to diagnose the failure, so the subsequent motion sequences were not stable, and led to a fall.

The differences in solution performance between the planned motions and simulation demonstrated that even small placement errors can lead to partial grip failure, disrupting the planned motion. Human intervention from cameras observing the robot could detect and fix gross errors. However, for more subtle errors such as a damaged finger, there was no feedback to the operators besides that of the force-torque sensors in the

wrist.

Based on the competition results, the following needs for a stability estimator were identified:

- The estimator needs to measure grip / placement quality.
- The stability contribution of grasp force must be carefully accounted for. Ignoring the effect of hands leads to overly conservative motions, while assuming too much grip force / torque leads to grip failure and mechanical damage.
- Online stability estimation must account for the tilt of the robot, since the torso needs to be able to tilt to climb the ladder.
- The majority of failures were precipitated by failing to detect a bad placement during a static phase. Therefore, slower and more accurate calculations are permissible.

6. State Estimation

6.1 Introduction

Search and rescue is an application for humanoid robotics that has received significant attention in both popular media and literature. Industrial disasters such as the Fukushima Daichi reactor explosion can trap people, and require expert help to resolve. Rather than put humans in danger, a humanoid rescue robot can withstand radiation, chemical, and biological agents. A humanoid robot's build is a natural fit for human environments as well.

In particular, ladders are a common feature in factories and navy ships. A small mobile robot would have great difficulty navigating a steep ladder. A humanoid has a natural advantage due to the size and form factor. As humanoid robots such as ATLAS, S-one, and DRC-Hubo are developed towards search as rescue work, ladders will become an important challenge to overcome.

Ladder climbing in many ways represents the next big problem to solve in humanoid robotics. Unlike walking and stair-climbing, the majority of movement in ladder climbing is lifting or lowering the robot's weight. The robot is supported with both feet and hands, and in the case of ladders with safety cages (Figure 6.1), cannot depend on climbing with feet alone.

To plan and execute climbing motions effectively, this problem requires a means of estimating stability that accounts for support from the robot's hands and feet simultaneously. In [94], a statically stable planner was demonstrated that used statically stable trajectories. Since it is possible to alternate between three and four-point contact, static stability can be maintained if contacts are stable. To satisfy the assumptions of the planner, however, the robot must be able to maintain contact with the ladder when needed. Failure to measure and react to loss of contact can mean losing support and falling from the ladder.

This chapter introduces an implementation for a humanoid robot stability estimator using Contact Wrench Space methods. Section 6.2 surveys similar methods of stability estimation and the limitations of these methods when applied to ladder climbing. Section 6.3 explains the contact wrench sum formulation as applied to a humanoid robot with articulated fingers. Section 6.4 shows simulation results showing correspondence with the ZMP metric during flat-ground contact. Section 6.5 shows simulation results of measurement of four-point contact stability on a vertical ladder.

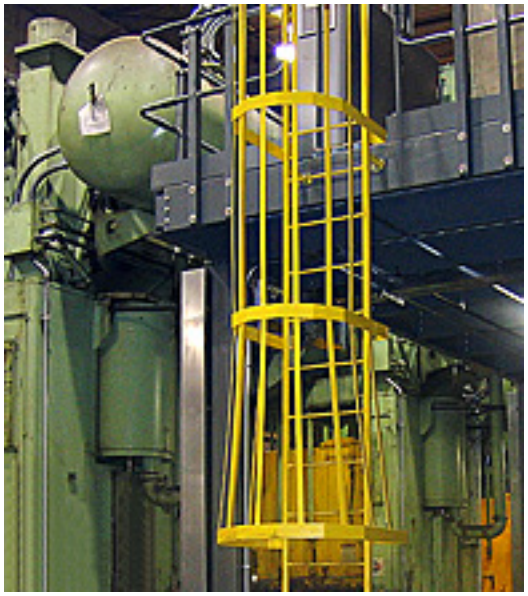

Figure 6.1: An example of a ladder with safety cage for equipment access.

6.2 Humanoid State Estimation

State estimation has been a part of robotics for many years. Biped robot state estimation started with the use of Linear Inverted Pendulum Models (LIPM) and (Zero Moment Point) ZMP control, in works such as [27], in which IMU and force-torque sensors are used to estimate the ZMP of a biped robot. They used a Kalman filter to refine the pose estimate, which is a common feature in these works. For example, [89] introduces a 2D biped model that uses a lumped mass model to predict model states for the sagittal plane.

The goal for a state estimator usable for climbing trajectories is to measure the space of possible contact forces that the robot can exert at any time. While instantaneous force/torque information from a wrist or foot may reveal the current state of reaction forces, they do not directly measure what reaction forces are possible under changing conditions.

The Linear Inverted Pendulum Model (LIPM) and Zero Moment Point (ZMP) concepts introduced in [86] solved this problem for the case of uniform terrain. ZMP-based methods have since been demonstrated with humanoids such as ASIMO, Hubo, HRP, and Wabian. A ZMP measurement from a robot's force-torque sensors predicts how close to instability a robot is as a function of both its current pose, but also the robot's dynamics. The limitation of ZMP-based methods, however, is that the simplified model assume a constant center of mass height. While this is adequate for piecewise-flat terrain like stairs, ladder-climbing operates largely in the vertical direction.

A more advanced method of stability checking draws inspiration from support grasps, or grasps which use passive forces to achieve force closure [88]. The forces and torques applied to an object being grasped are known as contact wrenches, and the combined effect of all of these is known as the Contact Wrench Space. Originally introduced in [33] for rough-terrain walking, this method consists of building a six-dimensional space of wrenches applied to the robot due to each contact. For a given robot pose, if a linear combination of these wrenches can be applied to oppose the robot's weight, then the current pose is stable.

In [98], this concept was applied to a humanoid robot to calculate a CWS for a humanoid robot based on foot contacts. This method applied a modified version of the GJK algorithm for the interior check. Critically, this method introduced the use of polyhedral cones to describe the contact wrench space. Treating contact normal force as infinite allowed a significant increase in calculation speed. When applied to "strong" limbs like a humanoid robot's legs, then this simplification is reasonable, since the strength limits of a leg are not often exceeded during statically stable motions.

For ladder-climbing, however, this assumption can mean overestimating the strength of dexterous hands, leading to grip failure. On the DRC Hubo humanoid for example, the rated grip strength is approximately $15kgF$ perpendicular, approximately %30 of the robot's weight. On a vertical ladder, the moment created due to the robot's distance from the ladder must be matched by the grip strength of the hands. This grip force limit means that some grip locations lead to overly high grip forces, and some motions may put too much load on one gripper. Therefore, for ladder climbing, the stability estimate must account for force limits at each end effector.

6.3 Contact Wrench Sum Formulation

6.3.1 Grasping

A ladder presents a somewhat simplified grasping problem that may not require complex computations of grasps. In fact, an exploratory work shows that ladder climbing is at least possible with pre-selected positions and a shape-adaptive hand [94]. However, the algorithms used to compute support grasps are mathematically very similar to the Contact Wrench Sum methods that form the stability check.

Basic grasping tasks for general grippers and hands have been well-researched; it has been shown repeatedly that a grasp is stable if it achieves force closure [63]. A force-closing grasp is one that can oppose an arbitrary disturbance force applied to the object with a combination of contact forces from each finger. An optimal grasp maximizes the disturbance force that can be canceled for a given force limit. For example, [53] solves optimal force closure via linear programming. In [56], it was demonstrated that shape primitives can

speed the search process for an optimal grasp. [66] produces sets of candidate grasps for odd-shaped objects by analysis of the medial axis transform.

The contact wrench sum concept is based on grasp-planning work such as [9], in which optimal contact locations are calculated to provide force closure around an object. A contact wrench is a combination of the contact force f and the moment τ with respect to a fixed point P_0. Together, the force and moment create a wrench that can be applied to the object. A contact wrench w_i is compactly expressed in the form of (6.1). The Contact Wrench Space is therefore a volume in the six-dimensional force / moment space that contains every possible combination of these wrenches. Dimensions such as volume and minimum diameter of this wrench space have been used to obtain quality metrics of grasps in work such as [90] and [96].

$$\begin{pmatrix} \vec{f_i} \\ \vec{r_{i/0}}_{COM} \times \vec{f} \end{pmatrix} \tag{6.1}$$

These methods can also be applied to the robot itself, which was the major contribution of [33] and [98]. Much as a grasped object is stabilized if an arbitrary disturbance can be opposed, a multi-limbed robot can be considered to be stable if a disturbance wrench $\vec{w_d}$ lies within the contact wrench space. The disturbance wrench consists of several components:

- Inertial wrench $\vec{w_i}$: the equivalent wrench due to the robot's inertia and overall linear and angular acceleration.
- Gravity wrench $\vec{w_g}$ due to the weights of each robot link.
- External applied load $\vec{w_f}$ due to applied forces or carried load.

The disturbance wrench in (6.2) can be simplified to (6.3) if motion is quasi-static, and no external loads are applied to the robot. The moment of the gravity wrench $\vec{w_g}$ is taken with respect to a fixed origin, which can without loss of generality be fixed to a point on the robot.

$$\vec{w_d} = \vec{w_i} + \vec{w_g} + \vec{w_f} \in \Sigma \vec{w_i} \tag{6.2}$$

$$\vec{w_g} = \begin{pmatrix} -m\vec{g} \\ \vec{r}_{G/0} \times -m\vec{g} \end{pmatrix} \tag{6.3}$$

To account for friction forces, the friction cone of each contact can be approximated as a 4-sided friction pyramid, as introduced in [33]. Assume a limiting normal force f_n^i for each point contact, a friction coefficient

of μ^i per contact i, the friction pyramid is represented by the four contact forces in (6.4). The basis vectors $\hat{b}_j$ lie in the null plane of the contact normal, spaced $\frac{\pi}{2} rad$ apart.

$$\vec{f}_i = \vec{f}_n + \hat{b}_j \mu_i f_n, j = 1..4 \tag{6.4}$$

Evaluating stability from the contact wrench space and a known disturbance wrench reduces to an interior point check. If $-\vec{w}_d$ lies within the CWS volume, then the robot's reaction forces can oppose the disturbance. The general procedure is:

1. Build a list of contact points, normals, and friction coefficients.
2. For each contact, create a set of contact forces including normal force and friction force perpendicular to the normal.
3. Compute the contact wrench for each of these possible forces by (6.1).
4. Build a 6D convex hull of these contact wrenches.
5. Perform an interior point check on the CWS and $\vec{w}_d$.

Note that the contact wrench space in this method is a linear combination of contact wrenches. When sufficient strength can be assumed, then the magnitude of contact forces is normalized. While this approach is typical for grasping, it presents certain challenges when applied to a humanoid robot. If the Contact Wrench Space is formulated without force limits, then a pose such as (Figure 6.2) will be considered stable, despite the high forces applied to the fingers to keep from falling backwards.

Hubo in simulation hanging from ladder with fingers

Figure 6.2: Example of a pose that depends on limited grip strength for balance.

In (6.5), a simple planar model approximates grip force F_{grip}. For a grasp height $h = 1m$, a body overhang $d = .3m$, and a lumped mass $m = 48kg$, the grip force required is approximately $14.3kgF$, which is very close to maximum straight-pull rating of $15kgF$. This pose is typical of ladder climbing while facing forward due to knee clearance.

Therefore, the assumption of either a uniform contact force, or infinite maximum normal force as in [98] have the potential to produce false positive estimates of stability.

$$F_{grip} = \frac{mgd}{h} \tag{6.5}$$

A simple interior check algorithm used to prototype the algorithm was used in python. Given that each facet of the convex hull represents a hyperplane in 6D, any interior point in the convex hull will satisfy (6.6), where V is the point in question, x is the normal vector of the facet's hyperplane, and b is the offset from the origin for the hyperplane. This information is available as part of the qhull convex hull solution. Therefore, an easy way to check that a point is interior is to iterate over the list of facets. This procedure requires O(n) time for a success, and worst-case O(n) for a failure.

$$Vx + b \leq 0 \tag{6.6}$$

6.3.2 Simulation in OpenHubo

The simulation used for these experiments was the open source OpenHubo[1] platform. Based on the OpenRAVE environment, openHubo adds important simulation elements such as servo control, along with a complete dynamic model of Hubo2 and DRC Hubo robots. These robot models and the OpenHubo simulator have been employed in [75] and [39] for evaluation of locomotion planning, and in [54] to plan throwing trajectories.

The DRC Hubo humanoid (figure 6.3) was modeled in OpenHubo by deriving rigid body masses and inertias from a detailed CAD model in SolidWorks. The model was exported to Universal Robot Description Format (URDF) via the SolidWorks URDF exporter plugin. The URDF model was then converted to OpenRAVE XML format for use with OpenHubo via a python module included in OpenHubo. Model geometry was exported as shrinkwrapped STL models for each rigid link, which were then manually decomposed into convex sub-bodies using the STL-VRML toolbox for MATLAB[2]. Model specifications for the DRC Hubo are given in Table 6.1.

6.3.3 Contact Measurement

When estimating contact and contact forces in simulation, one challenge that arises with Open Dynamics Engine (ODE) is collision handling. Contact restoring force is proportional to penetration depth between two

[1] https://github.com/daslrobotics/openHubo
[2] http://github.com/robEllenberg/stl-vrml-toolbox

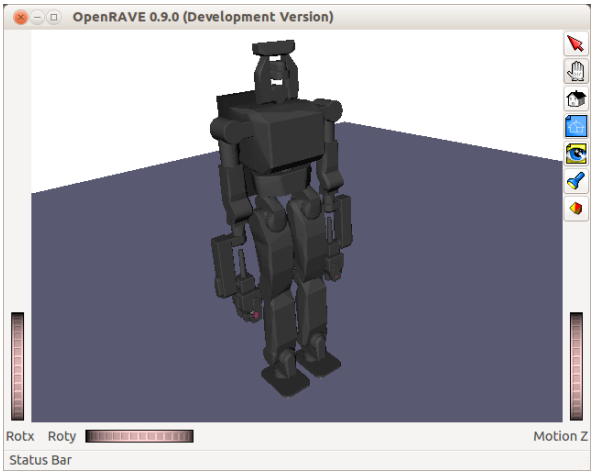

Figure 6.3: DRC Hubo and OpenHubo model

Table 6.1: Link lengths and masses for DRC Hubo robot

Link	Length	Mass
Torso	290mm	12kg
Upper Leg	350mm	5.1kg
Lower Leg	350mm	4.0kg
Foot (length)	220mm	2.2kg
Foot (width)	110mm	-
Upper Arm	250mm	3.5kg
Lower Arm	250mm	2kg

bodies. However, a large articulated system such as a humanoid creates closed kinematic chains, leading to constraint loops that cause solution instability. The result of this is that bodies in contact will not settle to an equilibrium position, but instead "jitter" in place slightly. In a given simulation step, only part of a body may be penetrating, even though on average full contact is made.

The physical DRC Hubo has rubber pads on the feet and fingers to add a small amount of compliance when grasping objects or placing a foot. Shifting the robot's weight between front and rear extremes can cause compression of approximately $1mm$. While soft contact is not supported currently in OpenRAVE with ODE, the effect was approximated by the addition of rectangular primitive contact bodies on the hands and feet. These bodies are fixed to the robot's hands and feed. During solution however, fixed constraints are relaxed slightly to allow quicker convergence. Typically these primitive bodies yield 4-8 collision points, instead of tens of points as would be measured with a detailed mesh model (Figure 6.4).

Directly querying contact locations abstracts away the role of F/T sensors in these experiments. However,

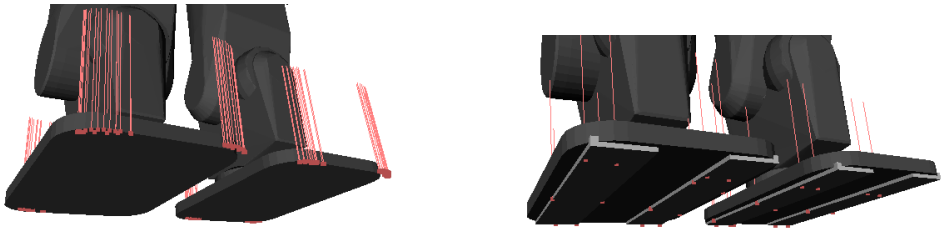

Figure 6.4: Comparison of contacts detected with trimesh-primitive (upper) collision vs. primitive-primitive (lower) collision. Fewer discrete contacts reduce computational burden during convex hull generation.

a set of contact points is similar to the information that would be gathered from an array of simple two-state touch sensors on the hands and feet. Since a precise reaction force is not required, this simplification was considered to be a minor abstraction. A motion planner typically uses contact checking as an easy failure metric, yet at the same time we need those contacts to determine probably reaction forces. A simple solution to this issue for planning is to treat the contact pads as solid objects only during execution. The assumption here is that the thickness of the contact pads represents the maximum reasonable deflection those pads are expected to undergo, and can therefore be ignored during motion planning.

For a single contact pad, this definition is somewhat loose, in that contact could be achieved on one edge of the pad, while the majority remains out of contact. This could be resolved by partitioning the contact body into discrete segments. Unfortunately, simply partitioning the contact body itself leads to increased noise and jitter due to the smaller mass and inertia of each individual piece. The stiffness of the contact relative to the mass/inertia of the body is a major factor in the uncertainty of the contacts. This does add an additional 6DOF to the total solution for ODE, however. This type of optimization of computational efficiency is outside the scope of this work.

6.4 Comparison to ZMP

An initial experiment demonstrates the close correspondence between the CWS prediction and the simulation results for the static case. For this experiment, the robot was biased to lean forward at the hips and ankles, with the torso remaining perpendicular to the ground. The arms were tilted at the shoulder at an angle θ, which shifts the center of mass towards the front of the robot. Figure 6.5 shows the predicted vs actual failure point for a nearly static case for varying torso offsets. The initial pose for the robot is given in Table 6.2. The angle at which the robot fell forward was determined via a bisection algorithm. The prediction from the CWS was recorded before running the dynamic simulation, and the resulting success for failure was

Figure 6.5: Balance failure point from simulation compared to prediction from CWS.

Table 6.2: Initial pose for forward tilt experiment

Joint	Angle (deg)
Hip Pitch	-9.1°
Ankle Pitch	9.1°
Shoulder Pitch Min	-11.5°
Shoulder Pitch Max	-32.1°

logged for each joint angle. For this planar case, no false positive predictions of stability were observed. While the robot physically fell forward at an arm angle greater than $.560 rad$, the CWS predicted failure for tilt angles greater than $0.546 rad$, a margin of approximately 2.5%.

6.5 Stability during Ladder Climbing

Earlier work on ladder climbing in [94] assumed that the contact directions were fixed for a given grasp. For the case of hand grips on ladder rungs, this meant the contact forces were pointed outward towards the robot, while foot contacts were assumed to be vertical. While this adequately characterized the balance that should be achievable with the ideal pose, if the grasp or foot placement varied, then the contact conditions would not match the planned path. Due to the limiting grip strength, even small errors in grip placement

Table 6.3: Table of limits for ladder pose trials

Parameter	Min	Max
Rung Diameter	$0.04m$	-
Rung Spacing	$0.3m$	-
Stringer Spacing	$0.65m$	-
Torso distance from ladder	$0.25m$	$0.65m$
Grasp point above rung	$0.09m$	$0.11m$
Foot placement spacing	$0.24m$	-
Palm yaw angle from +Y	60°	-

could lead to loss of grip.

Maximum torque for each joint of DRC Hubo's fingers was calculated in (6.7). Maximum finger torque τ_f is determined from the total pull force F_pull, the number of fingers per hand n_f, and the experimentally-determined effective finger length L_f. For a $12kgF$ pull force, the minimum finger torque for a stable grip in simulation was determined to be $1.5Nm$.

$$\tau_f = L_f F_{pull} / n_f \tag{6.7}$$

A ladder grasping pose with four points of contact (Figure 6.6) was planned using the GeneralIK whole-body numerical inverse kinematics solver in the CoMPS library [8]. Grasp and foot placement positions were first approximated with manual goal transforms that place the end effectors within $5mm$ of the goal point. A bisection algorithm was then used to close each grasp and foot contact to within a joint angle tolerance of $0.001rad$.

Figure 6.6: DRCHubo in four-point contact on a vertical, cylindrical rung ladder.

Table 6.3 lists the bounds on body posture relative to the ladder that were used for this experiment.

Table 6.4 shows the required scale factors on finger force and foot force to correctly predict a supporting condition. The contact wrenches were each iteratively scaled up until a solution was found, or a scale factor of 10 was reached. Finger torque was adjusted to less than 10% excess for the pose, ensuring that the robot could barely maintain stability in the four-point contact pose.

These results are consistent with what would be expected from a linear combination of contacts. Support must come from a combination of each end-effectors, yet the contact wrench space only encloses a linear

Table 6.4: Scaling of force limits for 4-point contact

Foot Force Scale	Finger Force Scale	Prediction
-	1.0	False
-	2.0	False
-	3.0	False
-	4.0	False
-	5.0	False
8.0	6.0	False
6.0	7.0	True
4.0	8.0	True
3.0	9.0	True

combination of contact forces. To be physically consistent, the coefficients must all be less than unity. However, this restriction means that the net force predicted from both hands will never exceed a single finger's maximum force, and typically be much less. In fact, stability is only correctly predicted when each finger can exert a total force equivalent to both hands combined (scale factor of 6). Maximum force from each foot must be similarly scaled up. This result is a fundamental limitation on the predictive power of a linear combination of contact forces.

6.5.1 Approximating the Minkowski Sum

The theoretical solution demonstrated in [97] is to apply a form of Minkowski sum to the sets of wrenches. Each rigid body is treated as a set of contact wrenches. In the same way that contact forces on a biped foot can be represented by a single center of pressure, the net contact wrench is a linear combination of wrenches applied to the rigid body. The full CWS is therefore the Minkowski sum of each of these independent sets.

Unfortunately, the number of points in the Minkowski sum grows very rapidly as the number of contacts n increases. The total number of contact wrenches V for a pair of bodies with r facets for each friction pyramid is given in (6.9). The nr term represents the contribution of each edge of each contact friction pyramid, plus the origin. The number of vertices is further inflated when additional bodies must be included in the sum. Formulating the Minkowski sum recursively for k bodies, it's clear that the number of vertices is polynomial in n and r, and exponential in b.

$$V = (nr+1)^2 = (nr)^2 + 2nr + 1 \tag{6.8}$$

$$V_k = (nr+1)^k \tag{6.9}$$

When applied to the DRC Hubo, the complexity of the full sum quickly grows unmanageable for online

computation. Table 6.5 itemizes the total points required to form the full Minkowski sum for the DRC Hubo model. To reduce the complexity of this method, a first-order approximation was implemented (Figure

Table 6.5: Dimensions of full Minkowski Sum for DRC-Hubo, assuming point contact for fingers, line contact for each palm, and plane contact for each foot.

Body	Minimum Contact Points	Friction Cone Edges	Contact Wrenches	Recursive Minkowski Sum
leftFoot	4	4	16	16
rightFoot	4	4	16	289
leftPalm	2	4	8	2610
rightPalm	2	4	8	23499
Body_LF12	1	1	1	47000
Body_LF13	1	1	1	94002
Body_LF22	1	1	1	188006
Body_LF23	1	1	1	376014
Body_LF32	1	1	1	752030
Body_LF33	1	1	1	1504062
Body_RF12	1	1	1	3008126
Body_RF13	1	1	1	6016254
Body_RF22	1	1	1	12032510
Body_RF23	1	1	1	24065022
Body_RF32	1	1	1	48130046
Body_RF33	1	1	1	96260094

6.7). Instead of taking the Minkowski sum of the entire set, the sum is only performed over the net contact wrenches. For the same n contacts, r edges, and b bodies, the reduced contact set's size V_r is given in (6.10). Compared with $96M$ points, the full first-order approximation at worst requires only 65595 points. Furthermore, because the sum can be computed sequentially, the process can be aborted at any time, giving a subset of the full CWS that will give a more conservative stability estimate.

$$V_r = bnr + 2^b - 1 \tag{6.10}$$

6.6 Simulation Results with First-Order Approximation

For the four-point contact pose in Figure 6.6, the DRC Hubo model was posed at 20 positions, placing the torso center between $0.34m$ and $0.56m$ distance in the X direction from the ladder plane. At each distance, a whole-body IK solution was found to place the hands and feet at fixed goal poses on the ladder. Foot goals were specified at a $0.24m$ spacing, allowing a random deviation of 5° rotation about the rung's axis. To

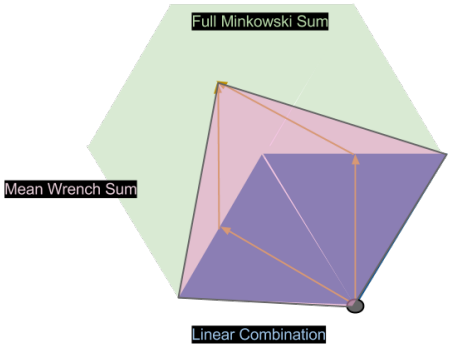

Figure 6.7: Conceptual comparison of full Minkowski sum (green), first order Minkowski sum approximation (red), and linear wrench combination (blue)

control for the effects of grip and contact friction, the hand grasp pose was chosen to align the hands with the plane of the ladder (Figure 6.8). This hand orientation is not optimal for maximum grasp force. However, the orientation simplifies validation against finger torque limits since the net force applied to each finger joint is closely aligned with the contact normal.

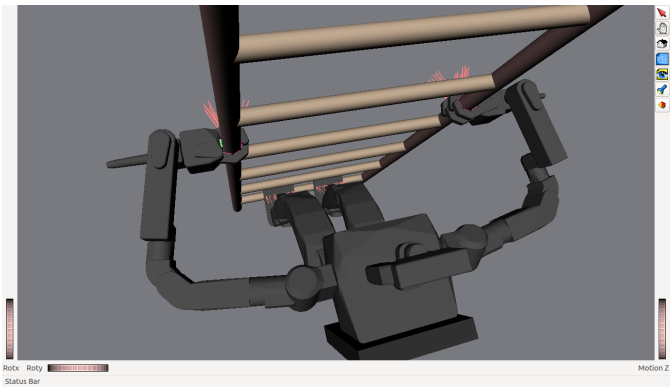

Figure 6.8: Orientation of hands parallel to ladder plane. This sub-optimal orientation minimizes dependence on friction for grip strength.

Since the size of the CWS will vary as a function of the chosen force limits, it was necessary to isolate the effects of small changes on the quality of the stability estimate. After choosing a set of finger force scale factors and torso poses, for every combination thereof, the bisection method in (Algorithm 6.1) was applied

to determine the minimum foot force needed for the gravity wrench to lie in the interior of the CWS.

Algorithm 6.1 Bisection search to identify minimum foot force scale factors.

```
CWS.Init()
lower = 1.0
upper = 50.0
scale = upper
while upper-lower ¿ tol do
    scale = (lower + upper) / 2.0
    CWS.scaleFootForce(scale)
    if CWS.check() then
        lower = scale
    else
        upper = scale
    end if
end while
return scale
```

Figure 6.9 shows the minimum value of the foot force scale factor (relative to the robot's total weight) required for the gravity wrench to lie in the interior of the partial CWS. The horizontal axes represent the distance between the robot torso and ladder plane, and the scale factor of maximum finger contact force. The "sum depth" is the number of partial sums computed for each lumped contact wrench. For example, a sum depth of 3 implies that for n contact wrenches, all possible combinations of 3 and fewer wrenches are computed and added to the set. By comparison, the full Minkowski sum implies computing all combinations of n or fewer wrenches.

For each trial, a grip failure and fall was detected by checking if the torso position has shifted vertically by at least $0.01m$ after $20sec$ of simulation time. Despite small variation in initial pose, the failure point remained fairly consistent at a distance from $0.46m$, as shown in Figure 6.10.

These trials show that for $n_{sum} = 4$, the gravity wrench lies in the interior of the CWS if the finger force scale is approximately 1.25 time greater than the nominal limit. This represents a significant improvement over the linear combination case, which required scale factors of 3 or greater to return the same result. As expected, smaller partial sums lead to a more incomplete CWS, which requires a large scale factor to correctly predict stability. These results suggest that constructing an incomplete CWS can still have predictive power, but the prediction is sensitive to the finger force scale factor.

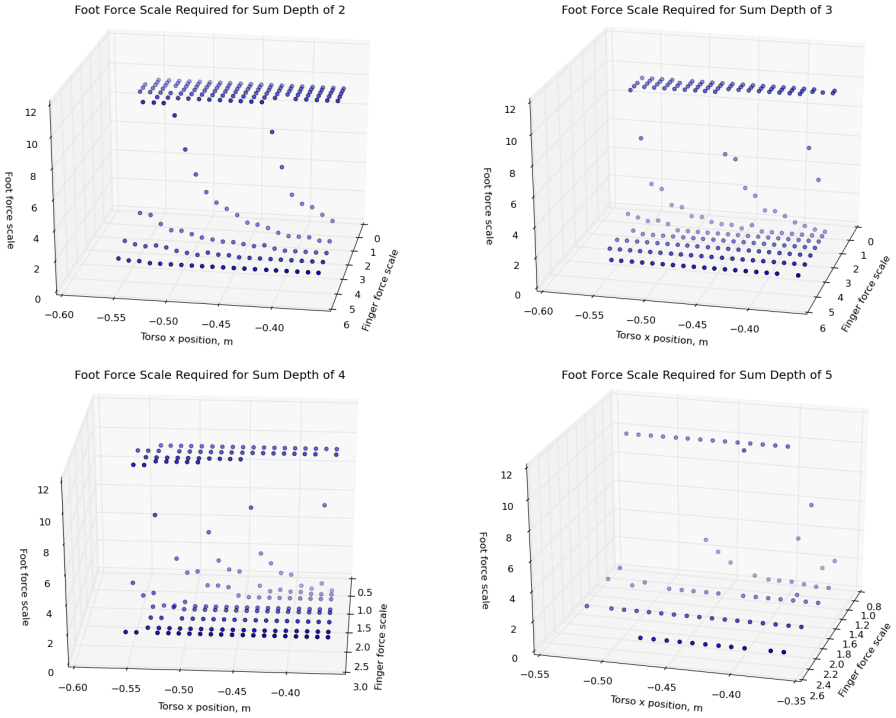

Figure 6.9: Plot of minimum foot force scale required for gravity wrench to be in CWS interior.

6.7 Applications with Robot Sensors

To implement this method on a physical robot such as DRC Hubo, the algorithm first needs to be reformulated in terms of measurable values from robot sensors. Subsection 6.7.1 identifies viable sensors to provide raw data for the CWS algorithm. Subsection 6.7.2 explains how the gravity wrench can be formulated with available sensor data. Finally, subsection 6.7.3 derives the contact normals and positions from robot state information. With these pieces, the CWS can be constructed in the robot's local coordinate system without global knowledge.

6.7.1 Available Sensor Data

Force / Torque Sensors

Force torque sensors do not provide specific insight into contact conditions, since many different contact combinations can produce the same instantaneous force. As such, a force/torque sensor by itself is not suited to estimate the contact state without a higher-level process.

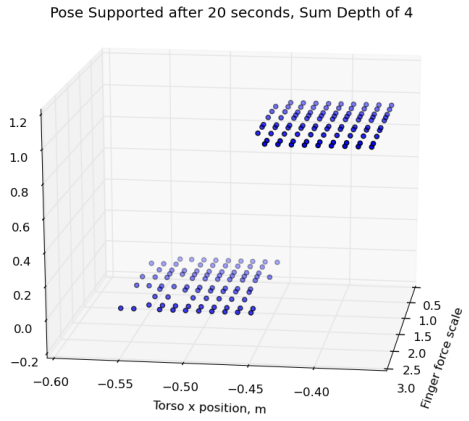

Figure 6.10: OpenHubo simulation results of grip failure with respect to torso position.

Contact-Sensing Pads

Point-contact sensors such as the Takktile, XXX and YYY return an analog force value in proportion to the load applied to the working area. While these force sensing elements are less accurate than a typical force sensor, they are much less expensive (on the order of $10's rather than $1000's), and much more compact. As such, these simple sensors can arranged in an array along the bottom of a robot's foot. Critically, the compliance in the sensor allows a more nuanced sensing of contacts, giving a clearer picture of where the foot is supported.

Inertial Measurement Unit

The Hubo series of robots each have a standard 5-axis IMU. Standard in this and other IMU's is a tri-axial accelerometer. Raw acceleration data from the accelerometer is available to measure the proper acceleration of the robot's Hip body. Industrial IMU's such as the MicroStrain 3DM-GX[3] additionally provide angular rate compensation to reduce the effects of Coriolis acceleration on the accelerometer data when angular velocity is significant.

6.7.2 Finding the Gravity Wrench

Since the IMU is attached to the Hip body within, the proper acceleration in hip coordinates $\vec{a}_{imu}$ is measured from the tri-axis accelerometer in the robot's hip body. Without loss of generality, the reference

[3]http://files.microstrain.com/3DM-GX3-25-Attitude-Heading-Reference-System-Data-Sheet.pdf

point is set to the origin of the accelerometer. Thus, the gravity wrench $\vec{w}_g$ in (6.3) becomes (6.11).

$$\vec{w}'_g = \begin{pmatrix} -m\vec{a}_{imu} \\ \mathbf{0} \end{pmatrix} \tag{6.11}$$

A critical property of this gravity-aligned representation is that we don't need to know what the actual ground orientation is. The net acceleration as measured by the accelerometer is effectively the instantaneous direction of gravity, even when accounting for dynamics.

6.7.3 Contact Normals and Positions

Using an array of point-contact sensors such as FSR's or contact switches, each measured contact can be assigned a position on the surface of a contacting body. For the DRC Hubo, each of the feet, palms, and finger surfaces could be covered with a flat array of these sensors.

To parameterize these locations, a coordinate system C_k is assigned to each of the k contacting bodies. C_k is aligned with the surface such that the local $\hat{\imath}$ and $\hat{\jmath}$ are aligned with the contacting plane, and $\hat{k}$ is along the contact normal, facing outwards from the face. We can then assign a local displacement p_i to each sensor from (6.12).

$$\vec{p}_i = \begin{pmatrix} p_x \\ p_y \\ 0 \end{pmatrix} \tag{6.12}$$

Table 6.6 shows the relationship of C_k to the corresponding end-effector body, along with the angle and position offsets. Angles α, β, and γ are Tait-Bryan angles about the $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ axes of the end-effector body. Following the coordinate system convention of Hubo2+ and DRC-Hubo, each body in home position is oriented such that the x-axis points forward, the y axis points left, and the z axis point up. The contact coordinate system is a simple permutation of the body coordinate system.

Table 6.6: Coordinate system offsets from end-effector bodies to contact coordinates

Body index	Joint Name	Child Contact Body	α	β	γ	x (mm)	y (mm)	z (mm)
1	LAR	leftFoot	π	0	0	0	0	-140
2	RAR	rightFoot	π	0	0	0	0	-140
3	LWP	leftPalm	$\frac{-\pi}{2}$	0	0	100	0	0
4	RWP	rightPalm	$\frac{-\pi}{2}$	0	0	100	0	0

Now that C_k is specified in terms of the appropriate end-effector link, forward kinematics can be used to calculate the instantaneous transform from the hip coordinates to the contact coordinates in (6.13). The transformation matrix $T_a^b(\vec{q})$ transforms from body a to body b as a function of the joint angles $\vec{q}$.

$$T_0^k = T_0^e T_e^k T_e^k = \begin{pmatrix} \mathbf{R}_k & \vec{p}_k \\ 0 & 1 \end{pmatrix} \tag{6.13}$$

To compute a given contact wrench requires the following properties:

1. position $p_{i,k}$ for i contacts each on k bodies.
2. normal $\hat{n}$ from C_k
3. force limit f_k

The position and normal are given from the local sensor position and forward kinematics, and the force limit is assigned a constant value from experiment. By following algorithm 6.2, the contact wrench space can be built in the Hip coordinates based on the robot's joint state and known kinematics.

Algorithm 6.2 CWS Location Formulation

```
CWS.init()
for k in bodies do
    for i in contactIndices do
        if inContact(i) then
            T_k = makeTransform(k,q)
            w_{i,k} = makeWrench(i,k,T_k)
            CWS.addWrench(i,k,p_{i,k})
        end if
    end for
end for
return CWS
```

7. Conclusions

As verified in 6, the contact wrench sum is equivalent to ZMP in the case of flat level ground. Furthermore, this method allows estimation of stability in multi-limb contact without a-priori knowledge of contact directions. Finally, experiments with four-point contact have shown that a linear combination of contact wrenches can predict stability. However, the maximum contact forces require unreasonable scaling to do so. Thus, a linear combination of contact forces is ill-suited to contact stability when the maximum contact force is not uniform. By including the largest terms of the Minkowski Sum, the CWS can be significantly increased in volume given identical contact conditions. The value of the simplifications presented are that the trade-off between calculation complexity and CWS accuracy could be selected to fit computation limits. While outside the scope of this work, an optimized implementation of this algorithm on a physical platform would be a useful next step in establishing the potential of this method in quasi-static applications.

The specific novel contributions demonstrated in this dissertation are:

- The OpenHubo humanoid simulator demonstrated planning and dynamic simulation of the Hubo2, Hubo+, and DRC-Hubo robots. OpenHubo has further been validated and deployed in multiple publications.
- The CBiRRT planner was adapter to solve motion planning for stair climbing, and adapted in OpenHubo to handle contact mode switching.
- An iterative method for constructing the Contact Wrench Sum was demonstrated to allow tradeoff between computational complexity and estimate accuracy.
- A Contact Wrench Sum implementation was developed in python for OpenHubo for application to motion planners.

Bibliography

[1] D. Alves Barbosa de Oliveira Vaz, R. S. Inoue, and V. Grassi. Kinodynamic motion planning of a skid-steering mobile robot using rrts. In *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*, pages 73–78, 2010.

[2] L. Baudouin, N. Perrin, Moulard T., Lamiraux F., Stasse O., and Yoshida E. Real-time replanning using 3d environment for humanoid robot. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 584–589, 2011.

[3] K.E. Bekris, B.Y. Chen, Ladd A.M., Plaku E., and Kavraki L.E. Multiple query probabilistic roadmap planning using single query planning primitives. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 656–661 vol.1, 2003.

[4] Dmitr. Berenson, Joe. Chestnutt, Srinivasa Siddharth., Kuffner James, and Kagami Satoshi. Pose-constrained whole-body planning using task space region chains. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids09)*, 2009.

[5] Dmitr. Berenson, Siddhartha S. Srinivasa, Ferguson Dav., Collet Alvaro, and Kuffner James J. Manipulation planning with workspace goal regions. *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 618–624, 2009.

[6] Dmitr. Berenson, Siddhartha S. Srinivasa, Ferguson Dave., and Kuffner James J. Manipulation planning on constraint manifolds. *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 625–632, 2009.

[7] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A robot path planning framework that learns from experience. In *ICRA*, pages 3671–3678. IEEE, 2012.

[8] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *International Journal of Robotics Research (IJRR)*, 30(12):1435–1460, October 2011.

[9] Ch. Borst, M. Fischer, and G. Hirzinger. Grasp planning: how to choose a suitable task wrench space. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 319 – 325 Vol.1, april-1 may 2004.

[10] T. Bretl. Multi-step motion planning: Application to free-climbing robots. *PhD Thesis, Stanford University*, Stanford, CA, June 2005.

[11] T. Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *International Journal of Robotics Research*, 25(4):317–342, April 2006.

[12] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock. Multi-step motion planning for free-climbing robots. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2004.

[13] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. October 2009.

[14] J. Chestnutt, J.J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *Proceedings of the IEEE International Conference on Humanoid Robotics*, 2003.

[15] Baek-Kyu Cho, Sang-Sin Park, and Jun-Ho Oh. Controllers for running in the humanoid robot, hubo. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 385–390, 2009.

[16] Rafael Limon Cisneros, Eiichi Yoshida, and Kazuhito Yokoi. Ball dynamics simulation on openhrp3. In *Proc. 2012 IEEE Int. Conf. on Robotics and Biomimetics*, pages 871–877, 2012.

[17] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, 2010.

[18] Alexander Dietrich, Thomas Wimbock, Alin Albu-Schaffer, and Gerd Hirzinger. Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. *IEEE Robotics & Automation Magazine*, 19(2):20–33, June 2012.

[19] R. Ellenberg, D. Grunberg, Paul Y. Oh, and Youngmoo Kim. Using miniature humanoids as surrogate research platforms. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 175 –180, dec. 2009.

[20] Robert Ellenberg, David Grunberg, Youngmoo Kim, and Paul Oh. Exploring creativity through humanoids and dance. In *5th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Seoul, South Korea, November 2008.

[21] R.W. Ellenberg, R. Vallett, R.S. Gross, B. Nutt, and P.Y. Oh. Development of the skewed rotation plane (srp) waist joint for humanoid robots. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6, April 2013.

[22] E. Foo and Thanh Tung Le. Starcly robot; a novel compact stair climbing robot. In *Mechanical and Electronics Engineering (ICMEE), 2010 2nd International Conference on*, volume 2, pages V2–75 –V2–78, 2010.

[23] Chenglon. Fu, Me. Shuai, Xu Ka., Zhao J. andon., Wang Jianme., Huang Yuanlin, and Chen Ken. Planning and control for thbip-i humanoid robot. In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, pages 1066–1071, 2006.

[24] M. Fuchs, C. Borst, P.R. Giordano, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimboeck, and G. Hirzinger. Rollin' justin - design considerations and realization of a mobile platform for a humanoid upper body. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4131 –4137, may 2009.

[25] J. S. Gutmann, M. Fukuchi, and Fujita M. A floor and obstacle height map for 3d navigation of a humanoid robot. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1066–1071, 2005.

[26] K. Harada, M. Morisawa, Miura K., Nakaoka S., Fujiwara K., Kaneko K., and Kajita S. Kinodynamic gait planning for full-body humanoid robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1544–1550, 2008.

[27] I. Hashlamon and K. Erbatur. Center of mass states and disturbance estimation for a walking biped. In *Mechatronics (ICM), 2013 IEEE International Conference on*, pages 248–253, 2013.

[28] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2006.

[29] K. Hauser, T. Bretl, and J.-C. Latombe. Non-gaited humanoid locomotion planning. In *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*, pages 7–12, Dec. 2005.

[30] Kris Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada, and Brian Wilcox. Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research*, 27(11-12):1325–1349, 2008.

[31] S. D. Herbert, A. Drenner, and Papanikolopoulos N. Loper: A quadruped-hybrid stair climbing robot. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 799–804, 2008.

[32] W.M. Hinojosa, N.G. Tsagarakis, G. Metta, F. Becchi, G. Sandini, and D.G. Caldwell. Performance assessment of a 3 dof differential based waist joint for the "icub" baby humanoid robot. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 195 –201, sept. 2006.

[33] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa. A universal stability criterion of the foot contact of legged robots - adios zmp. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1976 –1983, may 2006.

[34] R.D. Howe, N. Popp, P. Akella, Imin Kao, and M.R. Cutkosky. Grasping, manipulation, and control with tactile sensing. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1258–1263 vol.2, 1990.

[35] Weiwe. Huang, Chee-Men. Chew, Zheng Yu., and Hong Geok-Soon. Pattern generation for bipedal walking on slopes and stairs. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 205–210, 2008.

[36] H. Iida, H. Hozumi, and R. Nakayama. Development of ladder climbing robot lcr-1. *Journal of Robotics and Mechatronics*, 1(4):311–316, 1989.

[37] Youngbum Jun, R. Ellenberg, and P. Oh. Realization of miniature humanoid for obstacle avoidance with real-time zmp preview control used for full-sized humanoid. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 46 –51, dec. 2010.

[38] Youngbum Jun, Robert Ellenberg, and Paul Y. Oh. From concept to realization: Designing miniature humanoids for running. In *IIIS Proceedings of The 2nd International Multi-Conference on Engineering and Technological Innovation: IMETI 2009*, June 2009.

[39] Youngbum Jun and Paul Oh. A 3-tier infrastructure: Virtual-, mini-, online-hubo stair climbing as a case study. In *International Association of Science and Technology for Development-Robo2011*, Pittsburgh, PA, November 2011.

[40] S. Kajita, F. Kanehiro, Kaneko K., Yokoi K., and Hirukawa H. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239–246 vol.1, 2001.

[41] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1620 – 1626 vol.2, sept. 2003.

[42] Fumio Kanehiro, Hirohisa Hirukawa, and Shuuji Kajita. Openhrp: Open architecture humanoid robotics platform. *International Journal of Robotic Research*, 23:155–165, 2004.

[43] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1083 – 1090 Vol.2, 26-may 1, 2004.

[44] J.T. Kider, M. Henderson, Likhachev M., and Safonova A. High-dimensional planning on the gpu. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2515–2522, 2010.

[45] Sangbae Kim, M. Spenko, S. Trujillo, B. Heyneman, V. Mattoli, and M.R. Cutkosky. Whole body adhesion: hierarchical, directional and distributed control of adhesive forces for a climbing robot. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1268–1273, April 2007.

[46] Dennis Krupke, Guoyuan Li, Jianwei Zhang, Houxiang Zhang, and Hans Petter Hildre. Flexible modular robotic simulation environment for research and education. In *26th EUROPEAN CONFERENCE ON MODELING AND SIMULATION*. Universitat Koblenz, May 2012.

[47] J.J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. In *Autonomous Robots (special issue on Humanoid Robotics)*, volume 12, pages 105–118, January 2002.

[48] S. M. LaValle and James Kuffner. Randomized kinodynamic planning. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 1, 1999.

[49] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

[50] Steven M. Lavalle, James J. Kuffner, and Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[51] DavidN. Lee and Eric Aronson. Visual proprioceptive control of standing in human infants. *Perception & Psychophysics*, 15(3):529–532, 1974.

[52] Guangri Li, Qiang Huang, Yanping Tang, Guodong Li, and Min Li. Kinematic analysis and motion planning of a biped robot with 7-dof and double spherical hip joint. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 2982 –2987, june 2008.

[53] Yun-Hui Liu. Qualitative test and force optimization of 3-d frictional form-closure grasps using linear programming. *Robotics and Automation, IEEE Transactions on*, 15(1):163 –173, feb 1999.

[54] D. Lofaro and P. Oh. Humanoid throws inaugural pitch at major league baseball game: Challenges, approach, implementation and lessons learned. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, nov. 2012.

[55] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[56] A.T. Miller, S. Knoop, H.I. Christensen, and P.K. Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1824–1829 vol.2, 2003.

[57] A.M.M. Omer, Y. Ogura, H. Kondo, A. Morishima, G. Carbone, M. Ceccarelli, Hun ok Lim, and A. Takanishi. Development of a humanoid robot having 2-dof waist and 2-dof trunk. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 333 –338, dec. 2005.

[58] S. Osswald, J.-S. Gutmann, Hornung A., and Bennewitz M. From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 93–98, 2011.

[59] F.C. Park and Jin Wook Kim. Manipulability and singularity analysis of multiple robot systems: a geometric approach. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1032 –1037 vol.2, may 1998.

[60] Ill-Wo. Park, Jung-Yup. Kim, and Oh Jun-Ho. Online biped walking pattern generation for humanoid robot khr-3(kaist humanoid robot - 3: Hubo). In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 398–403, 2006.

[61] In-Won Park, Yong-Duk Kim, Bum-Joo Lee, Jeong-Ki Yoo, and Jong-Hwan Kim. Generating performance motions of humanoid robot for entertainment. *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pages 950–955, Aug. 2007.

[62] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

[63] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *Robotics and Automation, IEEE Transactions on*, 11(6):868 –881, dec 1995.

[64] D. Pongas, M. Mistry, and Schaal S. A robust quadruped walking gait for traversing rough terrain. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1474–1479, 2007.

[65] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200–207, 2006.

[66] M. Przybylski, T. Asfour, and Dillmann R. Unions of balls for shape approximation in robot grasping. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1592–1599, 2010.

[67] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, and the BigDog Team. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress The International Federation of Automatic Control*, July 2008.

[68] Alfred Sherwood Romer and Thomas S. Parsons. *The Vertebrate Body*. Holt-Saunders International, Philadelphia, PA, 1977.

[69] United States Occupational Safety and Health Administration. *Fixed Industrial Stairs*, chapter Walking-Working Surfaces, page 1910.24. Government Printing Office, 1984.

[70] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(5):630 – 637, sept. 2004.

[71] Ching-Long Shih. Ascending and descending stairs for a biped robot. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 29(3):255 –268, may 1999.

[72] Ching-Long. Shih and C.A. Klein. An adaptive gait for legged walking machines over rough terrain. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(4):1150–1155, 1993.

[73] B. Siciliano and J.-J.E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1211 –1216 vol.2, jun 1991.

[74] Greg Slabaugh. Computing euler angles from a rotation matrix. Technical report, Georgia Institute of Technology, August 1999.

[75] Kiwon Sohn and Paul Oh. Applying human motion capture to design energy-efficient trajectories for miniature humanoids. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3425–3431, 2012.

[76] O. Stasse, B. Verrelst, B. Vanderborght, and K. Yokoi. Strategies for humanoid robots to dynamically walk over large obstacles. *Robotics, IEEE Transactions on*, 25(4):960 –967, aug. 2009.

[77] T. Takubo, Y. Imada, Ohara K., Mae Y., and Arai T. Rough terrain walking for bipedal robot by using zmp criteria map. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 788–793, 2009.

[78] Yaroslav Tenzer, Leif P. Jentoft, and Robert D. Howe. Inexpensive and easily customized tactile array sensors using mems barometers chips. *Microelectromechanical Systems, Journal of*, 2012. Under review.

[79] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. accepted for publication.

[80] Zhao Tiejun, Tan Dalong, and Zhao Mingyang. The development of a mobile humanoid robot with varying joint stiffness waist. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3, pages 1402 –1407 Vol. 3, 2005.

[81] N. Vahrenkamp, T. Asfour, and Dillmann R. Efficient motion planning for humanoid robots using lazy collision checking and enlarged robot models. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3062–3067, 2007.

[82] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2464 –2470, oct. 2009.

[83] N. Vahrenkamp, C. Scheurer, Asfour T., Kuffner J., and Dillmann R. Adaptive motion planning for humanoid robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2127–2132, 2008.

[84] M. Vande Weghe, D. Ferguson, and S.S. Srinivasa. Randomized path planning for redundant manipulators without inverse kinematics. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 477 –482, 29 2007-dec. 1 2007.

[85] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev, and D.D. Lee. Search-based planning for a legged robot over rough terrain. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2380 –2387, may 2009.

[86] M. Vukobratovic, A.A. Frank, and D. Juricic. On the stability of biped locomotion. *Biomedical Engineering, IEEE Transactions on*, BME-17(1):25–36, 1970.

[87] Miomir Vukobratovic and Davor Juricic. Contribution to the synthesis of biped gait. *Biomedical Engineering, IEEE Transactions on*, BME-16(1):1–6, jan. 1969.

[88] M.Y. Wang and Yun-Hui Liu. Force passivity in fixturing and grasping. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 2236–2241 vol.2, 2003.

[89] Wei. Wang and Yan Li. Path planning for redundant manipulator without explicit inverse kinematics solution. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 1918–1923, 2009.

[90] Jonathan Weisz and P.K. Allen. Pose error robust grasping from contact wrench space metrics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 557–562, 2012.

[91] Zeyan. Xia, Guodon. Chen, Xiong Jin., Zhao Qunfei., and Chen Ken. A random sampling-based approach to goal-directed footstep planning for humanoid robots. In *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*, pages 168–173, 2009.

[92] J. Yamaguchi, S. Inoue, D. Nishino, and A. Takanishi. Development of a bipedal humanoid robot having antagonistic driven joints and three dof trunk. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 96 –101 vol.1, oct 1998.

[93] H. Yoneda, K. Sekiyama, Y. Hasegawa, and T. Fukuda. Vertical ladder climbing motion with posture control for multi-locomotion robot. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3579 –3584, sept. 2008.

[94] Yajia Zhang, Jingru Luo, Kris Hauser, Robert Ellenberg, Paul Oh, Andy Park, Manas Paldhe, and C.S. George Lee. Geometric constraints for ladder climbing for humanoid robots. In *Proceedings of TePRA*, 2013.

[95] Yajia Zhang, Jingru Luo, Kris Hauser, H. Andy Park, Manas Paldhe, C. S. George Lee2, Robert Ellenberg, Brittany Killen, Paul Oh, Jun Ho Oh, Jungho Lee, and Inhyeok Kim. Motion planning and control of ladder climbing on drc-hubo for darpa robotics challenge. In *ICRA*, 2014.

[96] Y. Zheng, M. C. Lin, and D. Manocha. On computing reliable optimal grasping forces. *Robotics, IEEE Transactions on*, 28(3):619 –633, june 2012.

[97] Yu Zheng and Chee-Meng Chew. Distance between a point and a convex cone in -dimensional space: Computation and applications. *Robotics, IEEE Transactions on*, 25(6):1397 –1412, dec. 2009.

[98] Yu Zheng, M.C. Lin, D. Manocha, A.H. Adiwahono, and Chee-Meng Chew. A walking pattern generator for biped robots on uneven terrains. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4483 –4488, oct. 2010.

[99] Changji. Zhou, Yue., Pik Kon., Ni Jun., and Chan Shan-Ben. Dynamically stable gait planning for a humanoid robot to climb sloping surface. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 1, pages 341–346 vol.1, 2004.

[100] M. Zucker, J.A. Bagnell, C.G. Atkeson, and J. Kuffner. An optimization approach to rough terrain locomotion. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3589 –3595, may 2010.

Appendix A. Whole-Body Inverse Kinematics

The general Inverse Kinematics (IK) problem for a single chain of joints is formulated as follows. A robot manipulator is composed of a kinematic sequence of joints. These joints can be prismatic (motion is rectilinear) or rotational. The position and orientation of each successive body can be expressed as a sequence of homogeneous transformations, which are functions of the joint positions. For a typical 7DOF robotic arm, the position of the end effector is represented as a sequence of 7 transformations. The 9 terms of the rotation matrix and 3 terms of the translation vector are in general functions of the 7 joint positions.The goal pose for the end effector of the robot manipulator has up to 6 configuration constraints imposed as a subset of Cartesian position R^3 and spatial orientation SO^3. By taking advantage of symmetries in the rotation matrix, there are ultimately 6 independent constraint equations, expressed as a combination of the 12 terms of the transformation matrix. The typical numerical methods of inverse kinematics solution for a single manipulator are well established in literature. Jacobian -inverse, transpose, and damped least squares methods [13] all share a common basis. The Jacobian of the m constraint equations with respect to n joint variables produces an m x n symbolic matrix of partial derivatives. When evaluated at the current joint space position, the Jacobian matrix represents the vector space tangent plane. Thus, a small displacement in joint space, when multiplied by the Jacobian, produces the corresponding change in end effector position.

A.1 MiniHubo IK Solution

Once a trajectory of hip and foot positions is computed, the inverse kinematics solver computes the appropriate joint angles at each step. While closed form solutions of Hubo2 IK were introduced in [60], the solution was restricted to keep the feet parallel to the forward direction of the robot. To relieve this limitation, a numerical IK solver based on the pseudoinverse and Selectively-Damped Least-Squares (SDLS) methods in [13] was was created in MATLAB, and exported as MEX functions to improve computational performance.The kinematic structure of the mini-Hubo and Hubo2 has 6 DOF in each leg, with the order of rotation joints common to both (Fig. A.1). In the hips and shoulders, three perpendicular joint axes intersect at a common point to simulate a spherical joint. The ankle pitch and roll axes also intersect, simulating a universal joint.

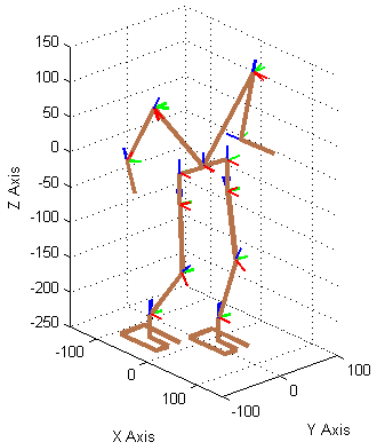

Figure A.1: The kinematic skeleton of the Mini-Hubo robot, used to implement the inverse kinematics solver. Shown are 6 joints each for the legs, the hip, and 4 joints for each arm.

Table A.1: MiniHubo Link coordinate displacements, mm

Body	dx	dy	dz
Torso	0	0	0
Hip	0	37.9	0
Left Hip Yaw	0	0	-44.7
Left Hip Roll	0	0	0
Left Hip Pitch	-21	0	-88
Left Knee Pitch	21	0	-87.69
Left Ankle Pitch	0	0	0
Left Ankle Roll	0	0	-34

A.1.1 Solution Constraints

To define the robot's pose, each end-effector is given constraints on its position and orientation. For the MiniHubo, these constraints are given in Table A.2, totalling 18 constraints on the 21 DOF of the robot. (A.1). Treating both feed as a coupled system has the advantage of increasing the maximum step length. As the foot goals are spread farther apart, the Torso yaw joint rotates to twist the hips in the direction of the stepping foot.

$$J_{robot} = \begin{pmatrix} J_{legs,12x13} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & J_{Larm,3x4} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & J_{Rarm,3x4} \end{pmatrix} \tag{A.1}$$

The IK results showed that, within the specified step length, the IK solver could move from any point to any other point within the solution bounds specified in Table A.3. However, outside of this volume, the solution

Table A.2: IK Constraints for each limb of the MiniHubo robot

End-Effector	Position Constraints	Orientation Constraints
Left Foot	XYZ	RPY
Right Foot	XYZ	RPY
Left Arm	XYZ	-
Right Arm	XYZ	-

Table A.3: Bounds of Foot position (relative to Hip center) that are numerically stable.

Foot X displacement	$-120mm$	$120mm$
Foot Y displacement	$-40mm$	$40mm$
Foot Z displacement	$-20mm$	$80mm$

would not always converge, leaving some constraints unsatisfied. For example, if a foot position goal was specified farther than the foot could reach, then the foot orientation constraints would no longer be preserved. While this kind of failure might be tolerable in an arm, it is dangerous for a leg and food. Since the robot depends on proper foot placement for balance, if that constraint is violated, then the robot will no longer be stable.

A.2 Hierarchical Constraints

Since the behavior of the solution is poorly defined if the constraints are not all solvable, a hierarchical approach was implemented following [73]. This method ensured that a set of constraints can be solved in priority order, so a valid solution is produced even if some constraints are violated.The formulation begins by expressing a set of constraints at a given hierarchy level i as a Jacobian approximation (A.2).

$$\dot{\mathbf{x}}^i = \mathbf{J}^i(\mathbf{q})\,\dot{\mathbf{q}} \tag{A.2}$$

An augmented constraint Jacobian for constraints 1 to i is shown in (A.3).

$$\mathbf{J}_A^i = \begin{bmatrix} \mathbf{J}^1 \\ \mathbf{J}^2 \\ \vdots \\ \mathbf{J}^i \end{bmatrix} \tag{A.3}$$

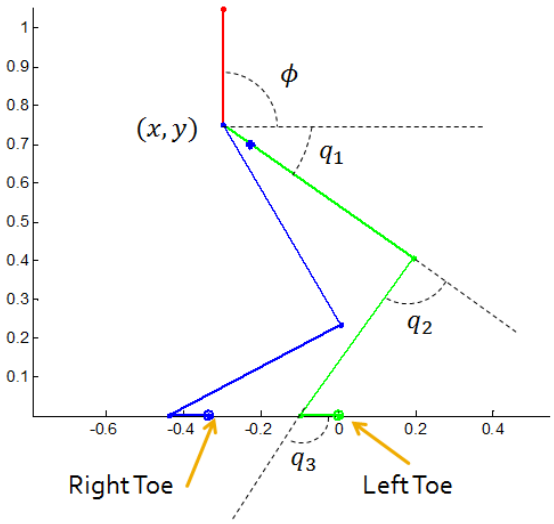

Figure A.2: The simplified 2D model of a humanoid robot lower body in the Sagittal plane

A projection matrix **P** given in (A.4) projects any vector into the nullspace of the augmented Jacobian.

$$\mathbf{P}_A^i = \mathbf{I} - \mathbf{J}_A^{i\dagger}\mathbf{J}_A^i \tag{A.4}$$

A recursive formulation for the change in joint angles is given in (A.5). The innermost term represents the constraint function minus the net motion in the constraint direction due to the accumulated change in joint angles q_{i-1}. $\bar{J}^i$ represents the projection of the i^{th} constraint's Jacobian onto the nullspace of higher-priority constraints (A.6).

$$\dot{\mathbf{q}}^i = \dot{\mathbf{q}}^{i-1} + \bar{\mathbf{J}}^{i\dagger}\left(\dot{\mathbf{x}}^i - \mathbf{J}^i\dot{\mathbf{q}}^{i-1}\right) \tag{A.5}$$

$$\bar{\mathbf{J}}^i = \mathbf{J}^i\mathbf{P}_A^{i-1} \tag{A.6}$$

To demonstrate hierarchical IK solution for a humanoid robot, a simplified 2D example problem was developed around a simplified 2D lower body model (Fig. A.2). The 2D model consists of 6 actuated joints connecting simple beam links as shown in Table A.4.The constraints for IK solution are defined in Table A.5. The foot placement constraints are treated as most important, followed by the foot orientation constraint to preserve contact, then the center of mass location to preserve balance. To demonstrate the orderly failure of the constraints, a sequence of poses was commanded, moving the desired center of mass increasingly far forward. As shown in Figure A.3, all constraints are satisfied for the first five poses. As the desired center of

q_1	Left hip
q_2	Left knee
q_3	Left ankle
q_4	Right hip
q_5	Right knee
q_6	Right ankle
L_i	Length of i'th joint

Table A.4: Joints and parameters for the simplified 2D humanoid

Constraint	Description	Priority
x_L	x position of left toe	1
y_L	y position of left toe	1
x_R	x position of right toe	2
y_R	y position of right toe	2
ϕ_L	global orientation of left foot	3
ϕ_R	global orientation of right foot	3
x_{CoM}	overall center of mass, x	4
y_{CoM}	overall center of mass, y	5
ϕ	torso global orientation	6

Table A.5: Breakdown of constraints to specify body pose

mass continues to move forward, the torso must tip forward, and body must drop slightly to satisfy the center of mass constraint.Despite the error in the torso orientation and the center of mass height, the high-priority constraints are preserved. Returning a valid solution has potential to improve the performance of IK-based RRT methods such as the CBiRRT, which forms the basis of the proposed humanoid planner. By being able to directly specify a goal that is balanced, the planner does not have to search for a valid supporting pose by trial and error.

A.2.1 Hierarchy Specifications

1. Supporting Link positions
2. Supporting Link Planar orientations (local roll / pitch)
3. Supporting Link Axis orientations (i.e. rotation about goal contact normal)
4. Balance Constraints (center of mass X/Y position)
5. Reaching End Effector Pose

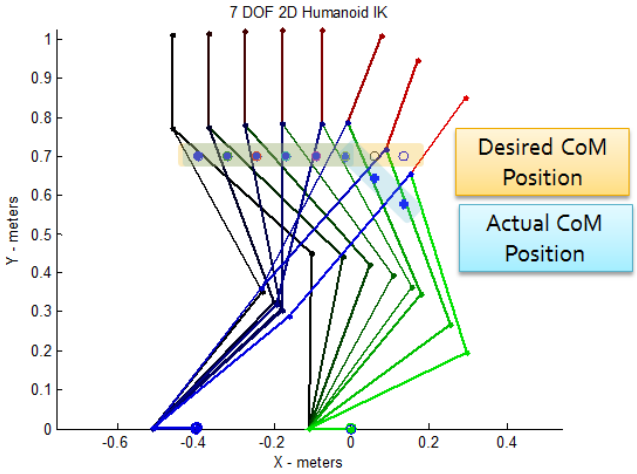

Figure A.3: Pose sequence of 2D Humanoid as desired COM position increases in forward direction. The Torso orientation constrain relaxes to satisfy higher-priority constraints.

One issue that arises with this type of formulation is that the solutions for the balance constraints are not unique. Any combination of joint angles that places the center of mass over the support polygon will technically be stable. This method is only effective at optimally solving a set of equality constraints. While it is possible to choose a particular solution, such as the center of the support polygon, higher-level constraints such as the reaching limb may be unduly constrained. If the support polygon is large, or a strong grip is used, then the geometric center may not necessarily lie within the reachable space. Therefore, a pure hierarchical solution that prioritizes balance above reach will be "wasteful", while prioritizing reach over balance will not generally be stable. The best application for this hierarchical approach is at a low-level. At the motion controller level, the IK solver would be tasked with always providing a stable / safe pose, even if other constraints fail. The solver would be primarily responsible for small updates per timestep, and not gross movements. A simple way to apply this IK solver in a planning process is given in Figure A.4. In particular, this process resolves the inequality constraint of the support pose by randomly sampling a goal constraint within the support polygon. The randomization routine is the key to this step. If the chosen stability constraint is not satisfied, then it is updated with an corrector as follows:

1. Find Jacobian J_c of n higher-level constraints wrt 6DOF body pose.

2. Use pseudo-inverse of J_c to estimate the best direction for the torso to move to satisfy the failing constraints.

3. Shift COM constraint in hierarchical IK in the same 6D direction while keeping it inside the support

polygon.

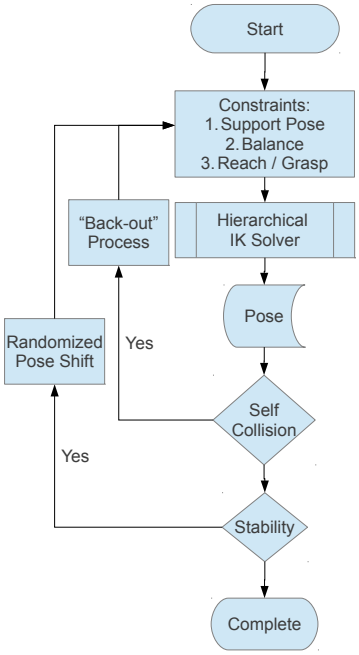

Figure A.4: Flowchart of simple pose generation process. Start of the process contains initial goals from planner (updated at each timestep).

Appendix B. Skewed Rotation Plane Joint Development

The development of the Skewed Rotation Plane joint was a collaboration with Richard Vallett, Roy Gross, and Brittany Nutt of Drexel University. The Mechanical prototype was originally developed for a capstone design project, and the joint design has since been published in [21].

B.1 Motivation

Traditionally, humanoid robots have a slightly simplified body plan compared to that of humans. Robots such as the Hubo KHR4 [60], HRP-2 [43], and ASIMO[1], for example, all have 6 degrees of freedom in each leg, 7 in each arm, and a single rotary joint at the waist, connecting the upper and lower bodies. While the kinematics of these arms and legs give similar ranges of motion to human arms, the waist joints of many humanoids do not match the range of motion of the human torso [68] (Table B.1). For locomotion research such as dynamic walking [40], [60], the torso's primary role was to compensate for angular momentum of the legs during walking.

More challenging locomotion problems, such as uneven terrain or stair-climbing, a fully-articulated upper body becomes useful to help stabilize the robot. In [26], an HRP-2 humanoid was shown to walk over sloped surfaces, while grasping a hand-rail. The stable region, and the range of possible walking trajectories, was shown to be significantly larger when holding a handrail. With a single-DOF waist, however, the humanoid was limited in how far it can reach to grasp the handrail.

Robots that lack a multi-DOF torso must move many more joints to tilt the torso. For example, a robot such as the Hubo2 must bend forward at the hip joints to tilt forward, and actually bend one or both legs to tilt sideways (Figure B.2). These restrictions can reduce the range of motion when the robot must reach over a tall obstacle (Figure B.1). A multi-DOF torso can bend around the obstacle, allowing an arm to reach farther past the edge. Using leg joints to tilt the robot can also make planning whole-body motions more difficult, since lower body tasks (balance, foot placement) and upper body tasks (manipulation) are more coupled. More recent humanoid research has focused on a whole-body locomotion and manipulation [82], [18]. In general these tasks require higher-level planning to satisfy the many constraints imposed by the task. Methods based on Rapidly-Exploring Random Trees (RRT) [49], for example, can plan motions to satisfy multiple, simultaneous task-based constraints [4]. In this case, additional DOFs give the planner freedom to explore for solutions that would over-constrain a simpler robot. Particularly for service robotics, many humanoid

[1] http://asimo.honda.com/

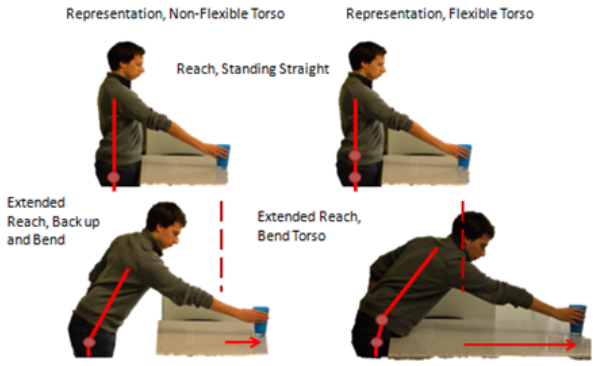

Figure B.1: A human bending from the back (left) can reach forward over an obstacle, where bending from the hips (right) would cause a collision.

Left Lateral Bending	50°
Right Lateral Bending	50°
Flexion (Front)	45°
Extension (Back)	30°
Right Rotation	40°
Left Rotation	40°

Table B.1: Range of motion for a typical human spine.

designs have appeared that have more than one DOF in the waist joint. For example, the iCub [32] uses a 3-DOF differential drive joint, allowing the waist to achieve similar bend angles to a human waist, while in [80], a 2 DOF differential drive gives a similar range of motion to a YiREN mobile robot. In [52], a humanoid robot was developed with a "double spherical" joint, which allowed the legs to effectively rotate about a single center point. The Rollin' Justin [24] uses an active/passive combination of joints to provide yaw (twist) and pitch motion for the torso. [57] develops a humanoid robot with a 3-DOF yaw-pitch-roll joint, showing a range of motion similar to the human back. The HRP4[2] has a 2DOF orthogonal joint that allows side-to-side tile motion. Almost all of these joint designs use orthogonal axes that directly tilt the torso in each direction. While simple to model, orthogonal axes require each actuator to directly support the weight of the upper body when tilted.

This chapter presents a design for a humanoid robot waist: the Skewed Rotation Plane (SRP) joint (Figure B.3), a compact, joint to approximate the range of motion of the human waist. The joint design was inspired by joints found in atmospheric diving suits[3], where joints must sustain high pressures without buckling.

[2] http://global.kawada.jp/mechatronics/hrp4.html
[3] http://www.oceanworks.com/hsCommercial.php

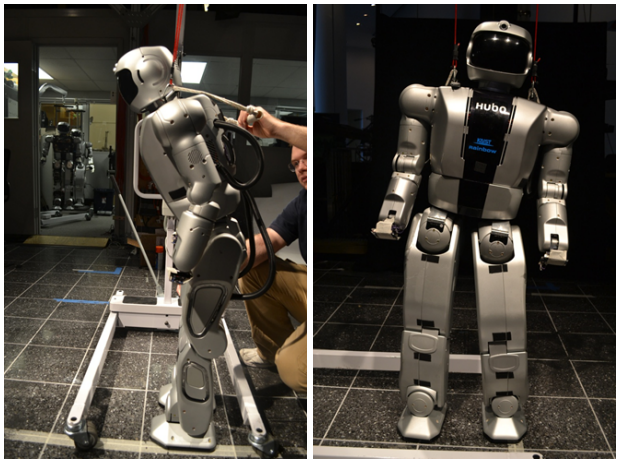

Figure B.2: Pitching the torso of the KAIST Hubo2 Humanoid (left) requires bending at the hips. (Right) Roll side-to-side requires significant bending at the knees to tilt the hips.

Similar kinematics are also found in robots such as the Kinova Jaco[4], though to the authors knowledge, the design has not previously been implemented as a waist joint. The main contributions of this chapter are:

1. The SRP joint is shown theoretically and experimentally in Section B.2 to match or exceed the range of motion of the typical human torso.

2. A Jacobian-Pseudoinverse Inverse Kinematics solver is developed in Section B.3, and is experimentally shown to solve arbitrary torso rotation trajectories.

3. Improvements in energy consumption are explored in Section B.4, showing reduced motor torque at large tilt angles.

B.2 Kinematics and Reachability

The SRP joint in general is an arrangement of three or more joint axes that are tilted a small angle θ_b from each other (Figure B.3). Unlike the orthogonal axes of traditional Roll-Pitch-Yaw (RPY) gimbals, the SRP joint axes have $\theta_b << 90^\circ$ between each body. The sequence of rotations from the fixed base to the end of the joint can be derived from this angle and each of the joint angles $\theta_j, j \in \{1,2,3\}$, the rotation of the j^{th} joint axis. The final orientation of the joint is given (B.1) where $R_i(\theta), i \in \{x,y,z\}$ represents a rotation by θ

[4]http://kinovarobotics.com/products/jaco-research-edition/

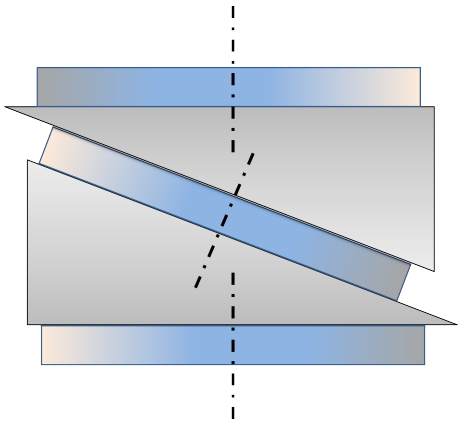

Figure B.3: The simplest SRP joint consists of three rotation axes. Each body in the joint has a rotation axis, tilted at at an angle of θ_b to the previous body.

about the i^{th} coordinate axis. This is commonly known as the extrinsic formulation.

$$R = R_z(\theta_1)R_y(-\theta_b)R_z(\theta_2)R_y(\theta_b)R_z(\theta_3) \tag{B.1}$$

Rotating each joint segment has the effect of precessing the next joint axis. By setting θ_b to be identical for the first $n-1$ joints, maximum precession ψ_{max} is achieved when each segment is rotated 180° with respect to the previous segment. This can be shown analytically by considering the relationship between this maximum overall bend angle of the torso, ψ_{max}, and the Torso's $\hat{k}$ vector. The 3^{rd} component of the Torso's $\hat{z}$ vector is found by expanding (B.1), resulting in (B.2). The maximum precession / bend is therefore the maximum angle between the $\hat{z}$ and $\hat{k}$ vectors, which is related to $R_{3,3}$ by (B.2). The minimal value of $R_{3,3}$ occurs when $\psi_2 = \pi$ in (B.4). Using the half-angle identity, the maximum value is shown to be $2\theta_b$.

$$R_{3,3} = 1 - 2\sin^2\left(\frac{\psi_2}{2}\right)\sin(\theta_b)^2 \tag{B.2}$$

$$\cos(\psi_{max}) = \arg\min_{\psi_2}(R_{3,3}) \tag{B.3}$$

$$\cos(\psi_{max}) = 1 - 2\sin^2(\theta_b) = \cos(2\theta_b) \tag{B.4}$$

Finally, the rotation introduced by the last joint, $R_z(\psi_3)$, is identical to the yaw motion of the last joint of the RPY joint γ. Therefore, within the mechanical limits of the joint, any reasonable desired yaw is possible.

Note that this yaw rotation is about the local $\hat{z}$ axis. Together, these limits (B.4) bound the SRP joint's kinematic reachability.

The reachable space was also established by a sampling method over all possible joint angles. Sampling randomly from $[0, 2\pi]$ for each joint angle produces a random torso orientation R_s for the SRP joint. This rotation matrix can be expressed in the equivalent pose of an RPY joint by by extracting an equivalent combination of α,β, and γ that represents R. The Tait-Bryan angles α about the x-axis, β about the y-axis, and γ about the z-axis can be determined from the components of R. If condition (B.5) is imposed, then the conversion in [74] can be simplified to (B.6).

$$\alpha, \beta, \gamma \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \tag{B.5}$$

$$\begin{aligned} \beta &= asin(R_{1,3}) \\ \alpha &= atan2(\frac{R_{2,3}}{cos(\beta)}, \frac{R_{3,3}}{cos(\beta)}) \\ \gamma &= atan2(\frac{R_{1,2}}{cos(\alpha)}, \frac{R_{2,2}}{cos(\beta)}) \end{aligned} \tag{B.6}$$

The choice of the number of DOF and θ_b, should ideally achieve the same range of motion as the human spine. θ_b must be large enough to accommodate extension, flexion, and lateral rotation in Table B.1. This constraint is given in (B.7), resulting in a minimum θ_b of 23°. Sampling 10,000 random orientations resulted in the reachability space of Figure B.4. Each point represents a tuple of the Tait-Bryan angles. The roll-pitch view includes the subset of points such that $\gamma \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$.

$$\theta_b > \frac{max(\theta_f, \theta_r)}{2} \tag{B.7}$$

An important concern in the design of non-orthogonal joints like the RPY joint is the manipulability of the joint over its range of motion. Kinematic manipulability, explored in [59], relates the ability of a manipulator to move an end effector in a task space. For a typical 7-DOF robotic arm, the task space consists of three Cartesian dimensions and three orientation dimensions, or a 6-DOF space. A kinematic singularity occurs when a manipulator cannot create a differential movement in one or more directions in the task space. Manipulability is essentially a measure of the condition of the constraint Jacobian. For an $m \times n$ Jacobian

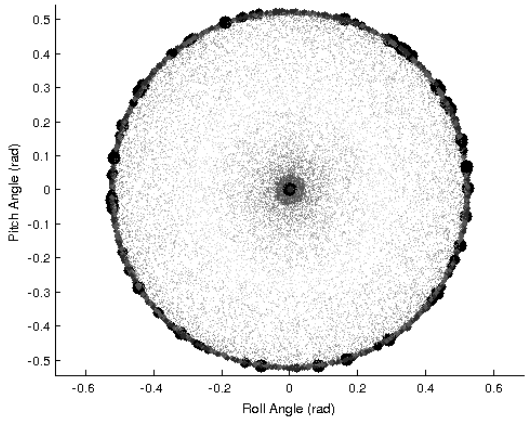

Figure B.4: The reachability space of the 3-DOF SRP joint. The size of the markers is proportional to the condition estimate $log_{10}(C)$

Matrix, the condition C is given in (B.8).

$$C = \frac{\sigma_1}{\sigma_m} \tag{B.8}$$

As the ratio C becomes larger, the manipulability grows poorer: essentially, some task-space motions require large changes in joint angles. Figure B.4 shows a region of poor manipulability near the origin, as well as at the edge of the reachable space. While increasing θ_b and soft-limiting the range of motion can avoid the 2^{nd} region, it is impractical to simply avoid the origin, as this is often the home position for a humanoid robot. By introducing an extra initial bend by θ_b, however, this singularity can be shifted away from the origin (B.9). The resulting reachability of this modified configuration is shown in figure B.5. While the singular region is not eliminated in this case, it can be placed outside the nominal range of motion. This result makes the SRP-style of joint uniquely suited to a waist joint's small range of motion. In a robotic arm that requires rotations of on the order of 180°, it becomes impractical to avoid this singularity.

$$R_{new} = R_y(-\theta_b)R \tag{B.9}$$

Another way to mitigate the kinematic singularity is to add an additional degree of freedom to the joint. Figure B.6 shows two alternative configurations for the additional DOF. The resulting ranges of motion for each case are shown in figure B.7. The additional DOF does not remove the central singularity, and in the case of the second modification, introduces an additional singular region. Increasing the number of joints does increase the overall reachability. If the joint in constrained to only positive pitch angles, the operating

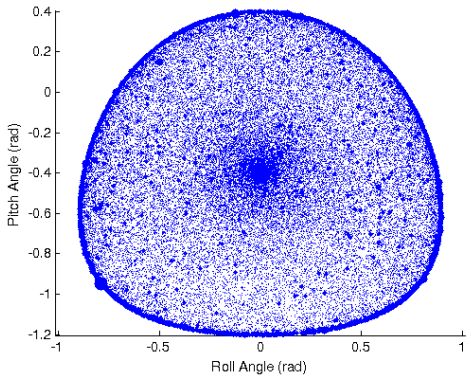

Figure B.5: An initial tilt of the joint shifts the singular region away from the equilibrium pose

region becomes singularity-free. To achieve full range of motion, the Torso can simply be mounted at a fixed angle. When the torso is rotated forward to be level, the joint will be pitched forward.

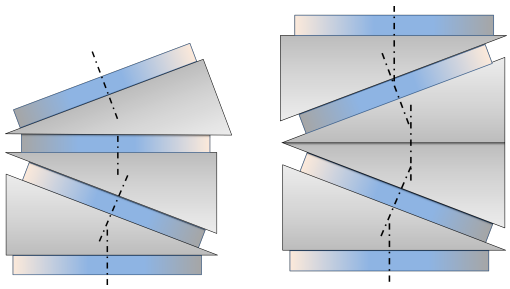

Figure B.6: (Left) An additional DOF is added in between the 2nd and 3rd joints of the basic design. (Right) The additional DOF is added at the end of the kinematic chain

These reachability spaces for the 3 DOF and 4 DOF designs show mixed but promising results. All tested designs show at least one near-singular region within the reachable space, requiring that the IK solver be able to cope with these regions. However, by reducing the operating region in software, the desired range of motion can still be achieved while also avoiding these problematic regions.

B.3 Inverse Kinematics Solver

Recalling that the limits of the human torso are significantly less than 90° of rotation in roll, pitch, and yaw, we claim the following:

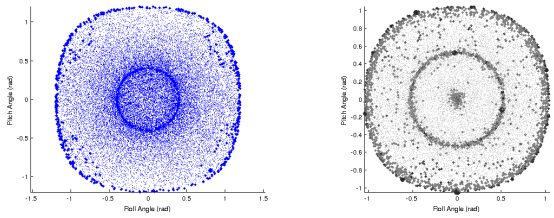

Figure B.7: Reachability polt for 4 DOF designs. Both plots show regions of near-singularity within the reachable space.

1. For "human-like" torso rotations, it is sufficient to constrain the possible rotations of the SRP joint to +/- 90 deg in any direction.

2. This reduced space of rotations is uniquely parameterized by 3 components of the rotation matrix.

Based on human anatomy, the typical humanoid coordinate system has its x axis pointing forward from the torso, the y axis to its left, and the z axis upwards through the head. These directions are perpendicular to the standard anatomical planes: the coronal, sagittal, and transverse planes, respectively. In particular, the transverse plane, or X-Y plane, divides $\mathbb{R}^3$ into half-spaces. Recalling the maximum pitch and roll of the torso (table B.1, the $\hat{z}$ vector of the torso is never rotated farther than 45° from the Hip's $\hat{K}$ axis. Thus, the vector is constrained to lie in the positive half-space defined by $R_{3,3} > 0$. Given the unit magnitude of $\hat{z}$, the constraint (B.10) implies that its x and y components will uniquely determine the vector.

$$\sqrt{1-\hat{k}_x^2-\hat{k}_y^2} = \hat{k}_z = 1, z \in (0,1] \tag{B.10}$$

A half-space constraint is also imposed on the $\hat{x}$ basis vector, constraining the yaw angle γ. Thus, one additional parameter is sufficient to specify the y vector: its z component. The third unit vector is trivially found from the cross product of the first two. Therefore, R_{21}, R_{31}, and R_{32} form a possible constraint vector.

$$\begin{aligned} \hat{k}\hat{j} &= 0 \\ \hat{j}_y &= \sqrt{1-\hat{j}_x^2-\hat{j}_z^2} \end{aligned} \tag{B.11}$$

These constraints are not intuitive, however, so it is desirable to specify the desired orientation constraints with conventional Tait-Bryan angles. A trivial answer is to simply compare the joint rotation matrix R directly with the desired rotation matrix given in (B.13). Unfortunately, the gradient of each term in the rotation matrix

is not conducive to numerical solution if a given component is close in magnitude to 1. A small rotation in any direction will tend to reduce the magnitude of the rotation at this point.

A better way is to use the desired orientation as an inverse rotation. R_tb is transposed and left-multiplied to the joint rotation matrix R. When a set of joint angles is a solution, this overall rotation matrix R_c given in (B.12) is approximately the identity matrix.

$$R_c = R_{tb}^T R \tag{B.12}$$

$$R_{tb} = R_\alpha R_\beta R_\gamma \tag{B.13}$$

Because only off-diagonal terms are specified in the constraints, the three constraints are zero at the solution. These off-diagonal terms are also nicely formulated in terms of their gradient: a small displacement of rotation in any direction is expressed well in the constraints (B.14).

$$C = \begin{pmatrix} \cos(\theta)\sin(\phi) \\ \sin(\theta) \\ \cos(\theta)\sin(\psi) \end{pmatrix} \tag{B.14}$$

Taking the Jacobian (B.15) shows that the constraints are decoupled in the local neighborhood of the solution. This formulation can then used with numerical Jacobian Pseudo-Inverse methods to find inverse kinematics solutions.

$$J\Big|_{\vec{0}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{B.15}$$

To solve inverse kinematics of the joint, a Jacobian Pseudo-Inverse solver was developed in MATLAB. Because of the relatively low number of DOFs, the constraints were directly formulated as a sequence of rotation matrices, such as in (B.1). The three constraint terms developed in B.2 are functions of the joint angles, the wedge bend angle θ_b, and the desired orientation angles α, β, and γ. In this formulation, the bend angle and desired angles are treated as constant parameters. The Jacobian of these constraints with respect to the joint variables is calculated symbolically, then stored as a function. Thus, the Jacobian at an arbitrary position can be exactly evaluated simply by calling this function.

Once a goal orientation is chosen, (B.16) is iteratively evaluated until the orientation error is below a specified tolerance.

$$\Delta\vec{\psi} = J_r\vec{\psi}^{\dagger}\Delta\vec{p}$$
$$\vec{\psi} = \vec{\psi}_0 + \Delta\vec{\psi} \tag{B.16}$$

B.4 Motor Torque

Under static loading, the SRP joint is advantageous over the RPY joint in that it requires less motor torque to support a given pose. Because the joint axes are oriented along the body, much of the torque applied due to pitching or rolling the torso, $\vec{\tau}$, is supported directly by reaction forces in the mechanism. Simplifying the torso to a lumped-mass approximation, the load applied to the waist during a static pose is given in (B.17), where m is the mass of the upper body and $\vec{c}$ is the center of mass position with respect to the Torso frame of reference.

$$\vec{\tau} = R\vec{c} \otimes m\vec{g} \tag{B.17}$$

The load torque can then be projected onto each motor axis to estimate the required motor torque to hold that pose in(B.18), where τ_j is the motor torque for joint j, and $\hat{z_j}$ is the z unit vector for the j^{th} joint segment. If the torso is assumed to be a centered lumped mass, then $\vec{c}$ reduces to $[0,0,c_z]^T$. Similarly, the theoretical static load for the equivalent RPY joint can be calculated by projecting the load torque onto the roll, pitch, and yaw axes. Since current is proportional to torque at stall in a DC motor, and assuming identical motors, the sum of motor torques is an equivalent measure to the sum of motor currents. Therefore, a given joint's operating cost f can be characterized by (B.19).

$$\vec{\tau_j} = \vec{\tau} \cdot \hat{z_j} \tag{B.18}$$

$$f = \sum_j |\tau_j| \tag{B.19}$$

To show the torque savings of the SRP joint, the minimum holding torques were calculated for every pose in the reachability space constructed previously. The ratio of the cost f to that of the RPY joint is shown in figure B.8. At most, the SRP joint requires 39% of the torque that an RPY joint requires, and the mode is only 27.5%.

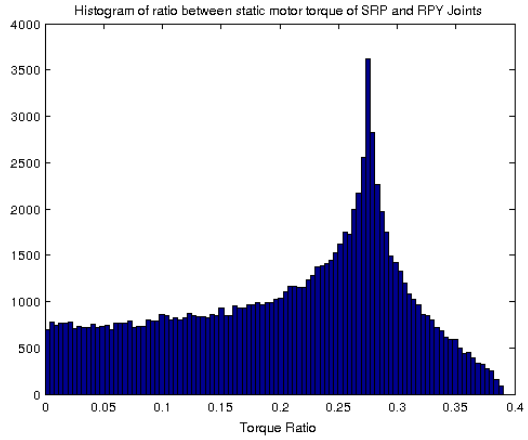

Figure B.8: Total torque cost as a fraction of the equivalent RPY joint, for positions in the reachability space.

B.5 Prototype Design

The design for the mechanical prototype is based on the 3DOF concept introduced in section B.2. Shown in Figure B.9, this design has a notable change: The top half of the middle body is rotated 90° about the vertical axis with respect to the bottom half. This rotation has the effect of moving the joint singularity from the origin (Figure B.10). The singularity present at the origin has become an unreachable region of diameter 11.5° centered at $\alpha = 0.26rad, \beta = 0.0rad$. The shaded region shown represents the convex region of pitch and roll angles in which there is no joint singularity. Table B.2 summarizes the physical properties of the

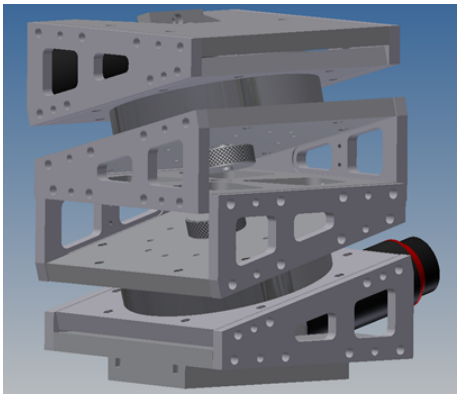

Figure B.9: Modified SRP joint with a $90^\circ\hat{k}$ rotation offset between the 2nd and 3rd joints.

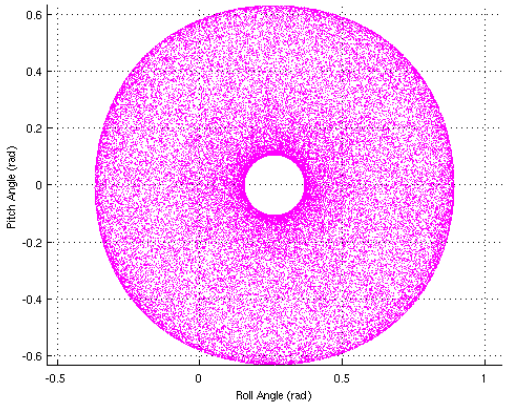

Figure B.10: The modified design shows an unreachable region of approximately 11.5° centered at $\alpha = 15°, \beta = 0°$.

Table B.2: Properties of SRP joint mechanical prototype

Length	10cm
Width	10cm
Height	13.2cm
Mass	3kg
Roll Range	$\pm 36.1°$
Pitch Range	$-10.0°, +22.1°$
Maximum Joint Velocity	$270 \frac{deg}{sec}$
Drive Motor (3)	Maxon 200W, 314176
Gear Reduction (3)	Harmonic Drive SHD-20-160-2SH
	32 tooth, 0.5mm module 45° miter gear

prototype (Figure B.11). All parts of the frame were machined from 8*mm*, 6061-T6 plate stock on a Tormach PCNC-1100 CNC milling machine. The three harmonic drives (labeled) are connected to the 200W Maxon Brushless Motors via a set of 32-tooth, 0.5*mm* module miter gears. The miter gears allowed the motors to be mounted perpendicular to the drive axes. This reduces the space required between the joints, which is essential in reducing the height of the mechanism.

The completed prototype (Figure B.11) has been mounted to a Hubo2 torso. Using two Hubo motor controllers, the prototype was able to actuate through the range of motion. Future iterations of this design will focus on weight and size reduction. Good candidates for size reduction include using smaller harmonic drives and thinner structural plates to reduce weight.

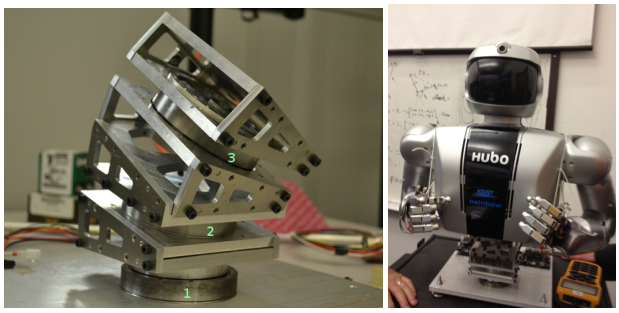

Figure B.11: Prototype design assembled on test fixture with joint angles of 15°, 90°, and 90°.

B.6 Conclusions

This work demonstrated a waist joint design for the Hubo2 humanoid robot that provides a range of motion similar to a human. Kinematic simulation showed that collision-free, arbitrary trajectories were possible through out the specified range of the joint. A simple modification was also demonstrated that moved the kinematic singularity outside the range of motion. The full-scale prototype was then developed and implemented to verify and validate the design. Plate manufacturing techniques allowed the design to be made with 2.5D machining methods, reducing machining time associated with complex 3D joints.

Vita

Robert W. Ellenberg was born on November 7, 1984, to William C. Ellenberg and Pasqualina M. Ellenberg in Philadelphia, PA. After graduating from Cheltenham High School in 2002, Robert attended Drexel University under a Presidential Scholarship to complete a B.S. degree in Electrical Engineering in 2007, and a B.S. in Mechanical Engineering in 2010. During his time at Drexel, Robert has participated in Robotics competitions including the Indoor Aerial Robotics competition (from 2006-2008). With a team of students from the Autonomous Systems Lab, the custom-made ATLAS humanoid was entered in the autonomous dash competition at Robogames 2009, winning first place. Robert has also taught classes at Drexel University as both a teaching assistant and adjunct professor for over 14 terms, including Freshman design lab, Dynamic Systems Lab, and Fundamentals of Vibrations.

Robert also contributed to Drexel's entry in the DARPA Humanoid Robotics Challenge (DRC), working on the simulation model of the DRC Hubo robot as well as the ladder-climbing portion of the competition. His simulation and modeling work has been released in several open-source packages, including the OpenHubo Simulator and stl-vrml toolbox for MATLAB. Robert has also working with CNC milling machines, and contributed an improved trajectory planner to the LinuxCNC project.

Permanent Address:

10 Asbury Ave.

Melrose Park, PA 19027