UNIVERSITY OF CALIFORNIA

Santa Barbara

ZMP and Value Iteration Based Planning to Assist Fast Walking for RoboSimian

A Thesis submitted in partial satisfaction of the

requirements for the degree Master of Science

in Electrical and Computer Engineering

by

Peter Seungsoo Ha

Committee in charge:

Professor Katie Byl, Chair

Professor Joao Hespanha

Professor Brad Paden

March 2014

The thesis of Peter Seungsoo Ha is approved.

_____

Joao Hespanha

_____

Brad Paden

_____

Katie Byl, Committee Chair

December 2013

ZMP and Value Iteration Based Planning to Assist Fast Walking for RoboSimian

by

Peter Seungsoo Ha

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Katie Byl, for providing me with an opportunity to work in the robotics lab at UCSB for I had a great time learning about legged robots and working on a cool project. She has always been kind in answering any questions I had even if it meant explaining it more than once so that I would have a good understanding of the subject matter. Also, I would like to thank Professor Joao Hespanha and Professor Brad Paden for being on my committee and teachings I received in their courses throughout my graduate career.

Additionally, I would like to thank my colleagues, especially Howard Hu, for their advice in both academics and non-academics. Every piece of advice, whether it was from academic knowledge or from their experience, helped me stay focused and have confidence in everything I did.

Finally, I would like to thank my family and friends, especially Alice Lee, for praying for me and providing support morally and financially so that I could make it this far. They have always provided me with encouragement when I doubted myself and pushed me to strive for a higher place.

ABSTRACT

ZMP and Value Iteration Based Planning to Assist Fast Walking for RoboSimian

by

Peter Seungsoo Ha

This thesis presents a ZMP based motion planning using a simplified model of RoboSimian, a quadruped disaster response robot by Jet Propulsion Laboratory (JPL) and Stanford University that is to compete in the DARPA Robotics Challenge (DRC), along with a method of planning for a swing trajectory for a limb using value iteration to optimize the walking speed of RoboSimian. The ZMP based planning is shown to be difficult for large step sizes because of the joint velocity limit. However, the value iteration for planning a swing trajectory was effective in finding an optimal trajectory which improves the overall walking speed of RoboSimian. Lastly, comparing the two methods, we show that when the joint velocity limit is low it is better to use the value iteration based gait but if the joint velocity limit is increased, the ZMP based method will generate a faster gait.

TABLE OF CONTENTS

LIST OF FIGURES

I. Introduction

The concept of walking robots is a highly researched field in this age. In the past, getting a legged robot to walk has been a challenge, but now the challenge is to be able to make them walk faster using smarter ways to develop trajectories for the legs and the body. Improving the speed of legged locomotion is an important area that needs to be tackled. Nowadays, especially after the Fukushima nuclear disaster, many disaster response robots (e.g., RoboSimian) are being developed to be able to perform tasks in environments that are hazardous for humans. For these types of robots, it is crucial to reach a hazardous site in a timely manner to be able to prevent more harm from happening. At this point, you may think that if a robot had wheels, it would be able to move faster. Although that may be true when there are no obstacles, legged robots can provide a significant advantage when there are obstacles, which is particularly likely in a disaster situation.

A. Fast Walking

There are several robots in existence today that are able to move at high speeds such as the Cheetah and the WildCat by Boston Dynamics [14]. These robots are dynamic robots. They rely on constant movement in order to ensure stability while walking / running. Depending on the capabilities and design of a robot, there are many different approaches that have been used for maintaining stability and walking fast.

Figure 1. RoboSimian under construction at JPL

http://www.jpl.nasa.gov/spaceimages/details.php?id=PIA17301 [15]

The most popular approaches for planning humanoid locomotion focus on regulation of center of pressure, referred to as the so-called Zero Moment Point (ZMP) which was introduced by Miomir Vukobratović in 1968. The ZMP represents the point on the ground where the horizontal moments become zero. This is an important concept in legged robots because using ZMP, we are able to have the robots perform dynamic movements opposed to only statically stable movements, allowing the robot to be able to move with higher speeds. In traditional ZMP

methods, motions for a robot are planned such that the resulting ZMP lies strictly within the convex hull of the robot's contacts with the ground, called the support polygon, thereby ensuring there are no toppling moments about any edge of the support polygon for the planned motion. This general approach is used widely for bipedal robots [7, 9, 11]. For a legged robot, the ZMP trajectory can be designed to jump instantaneously from being under one foot to the other foot. This is possible because the ZMP is a function of both the positions and accelerations of the distributed masses in the robot, and because we assume we can instantaneously apply torques to set accelerations at the joints. For quadruped robots, ZMP has also been used to perform maneuvers such as lunging and walking [4]. However, using the ZMP for walking is not for every robot. For example, if the robot has joint velocity limits that are low enough to prevent the robot from performing dynamic motions, there needs to be another approach in getting the robot to walk faster.

In cases where joint velocity limits are particularly low, a more practical method of planning fast walking is to optimize the trajectory of the swinging limb. This can be done by using third-order spline interpolation for the swing leg [2, 5]. However, in this thesis, I will be using value iteration to find a trajectory for the swing limb that will maximize the speed of walking. In this thesis, we consider both ZMP methods and a minimum-time swing leg approach because we consider the choice of gear ratio (and, correspondingly, of the joint velocity limits) to be a potentially open parameter for future revisions of the robot hardware we consider.

B. Executive Summary

In this paper, I will be discussing the use of ZMP and value iteration to increase the walking speed of RoboSimian, a disaster relief robot designed by Jet Propulsion Laboratory (JPL) and Stanford University. In Section II, there will be an overview of the models I have used to test the walking speed of RoboSimian. Section III discusses some assumptions made, two planning methods, ZMP based and Value Iteration based, and how they were implemented to carry out the experiments. Afterwards, results will be presented in Section IV and conclusion and future work will follow in Sections V and VI.

II. Models

A. Cart and Table

The cart and table model **(**Figure 2**)** is a model introduced by Shuuji Kajita et al. to calculate the ZMP for a mass at a set height moving in the x and y directions and to implement a preview control algorithm to find feasible trajectories for the center of mass that minimize a quadratic cost function that penalizes variation from a desired, reference ZMP over time [1]. As I have stated in the introduction, this work by Kajita is one of the popular ways to control robots using ZMP trajectories. Using this model, I also was able to generate COM trajectories to use with RoboSimian toward determining if the kinematics of the robot will be able to use ZMP controlled motion with the joint velocity limitations it has.

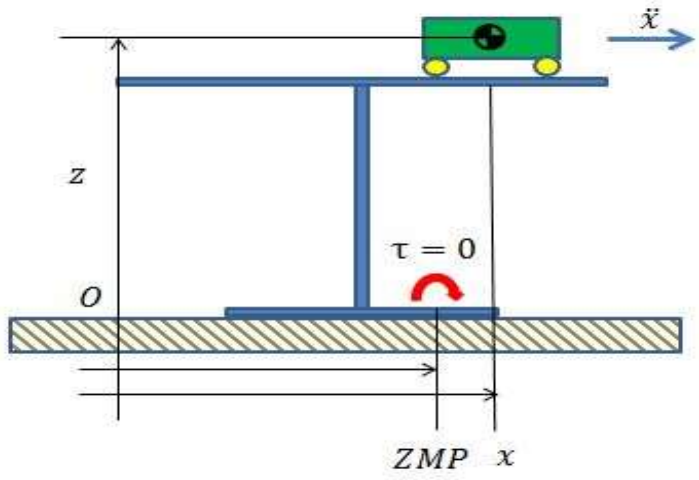

Figure 2. Cart and table model

B. Nine-link Model

The model shown in Figure 3 is a nine-link system. This is a simplified model of RoboSimian that was used to generate a Center of Body (COB) trajectory using the preview control by Kajita [1]. Each leg has two links to simplify the 7 Degrees-of-Freedom (DOF) limbs of RoboSimian. This model was a closer approximation to the actual RoboSimian robot than the single point-mass cart and table model because it also included point masses for each link, which would spread the mass to better approximate forces due to rotational accelerations of limb segments and show that there are differences in the COB, which is a fixed point near the center of the chassis of the robot, and the actual COM, which depends on the robot configuration. This was important because when simulating, we can see that the COM moves significantly less than the COB.

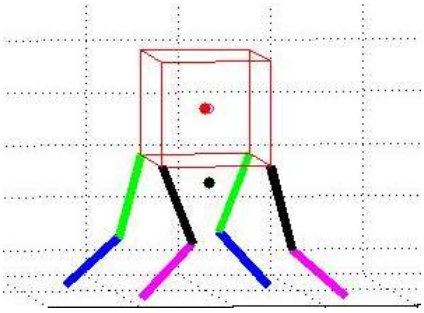

Figure 3. Nine-link model

C. RoboSimian

Robosimian is a quadruped robot developed JPL and Stanford. This robot was designed to be a disaster response robot and is to participate in the Defense Advanced Research Projects Agency (DARPA) Robotics Challenge, abbreviated as DRC. A model of RoboSimian (Figure 4), along with the inverse kinematics (IK), was built in MATLAB by Katie Byl, professor of the Robotics Lab at the University of California, Santa Barbara, which was made available for us to use.

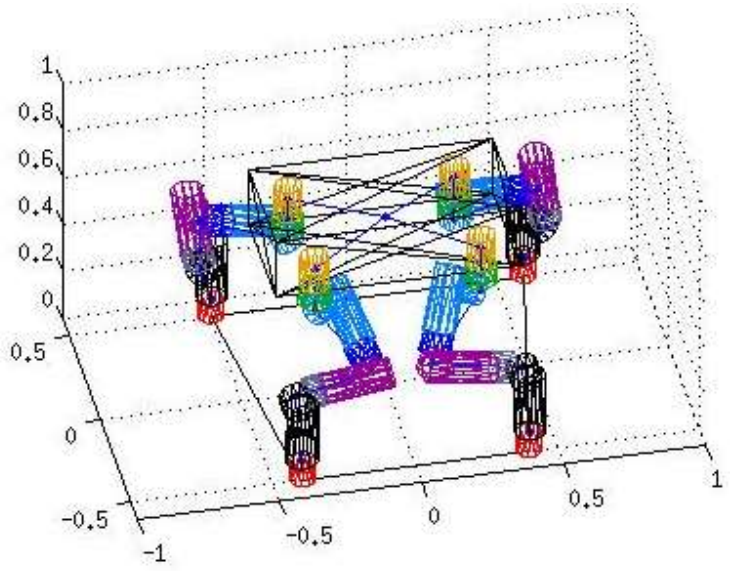

Figure 4. MATLAB model of RoboSimian created by Katie Byl

Since the limbs of RoboSimian are 7-DOF, there are in general an infinite number of IK solutions for each feasible end effector position. The 7th joint, shown in red in Figure 4, is a wrist that can effectively be ignored for walking tasks, leaving 6 joint angles to set the 6-DOF position and orientation of the last rigid limb segment, shown in black. Even for this 6-DOF reduced model, there can still be a finite number of redundant solutions. These redundant IK solutions can be divided into 8 different families depending on the configuration of the joint angles. The families were created by using the binary representations of 0~7 and using the three digits to determine which solution to choose for each of the joints. At a high level, the kinematic solutions for RoboSimian involve three geometric choices, akin to deciding whether to bend an elbow forward or backward, resulting up to 2^3 possible feasible solutions. Each of these solutions has its own feasible workspace.

To achieve smooth limb trajectories, the robot must either remain within a particular kinematic family or intentionally pass through a singularity, and even when remaining in a particular family, it is possible for discrete jumps in joint angles, so that testing trajectories IK for smoothness is always a necessity. For this thesis, I used the 3rd and 8th IK solution families to compare since $2_{10} = 010_2$ and $7_{10} = 111_2$, they would give me two different solutions for all of the joint angles (Figure 5).

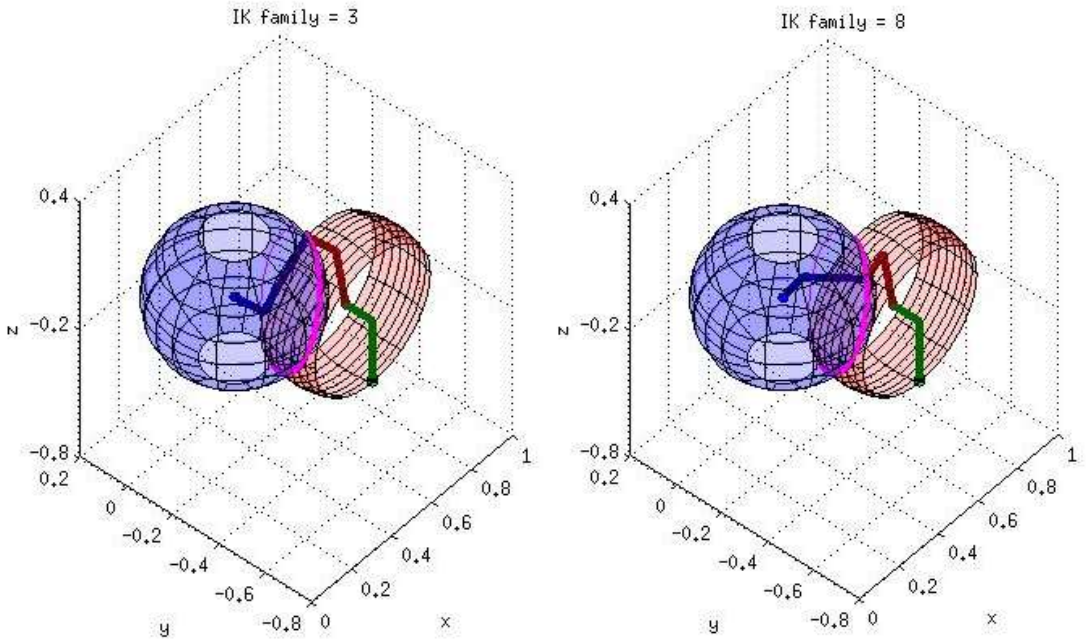

Figure 5. Comparison between the 3rd family (left) of IK solution and 8th family (right) of IK solution with the end effector at x = 0.6, y = -0.54, z = -0.6.

The partial blue sphere shows the reachable workspace of the end of the second link, assuming the base of the kinematic chain for the robot is fixed at the center of the blue sphere. The partial red sphere shows the reachable workspace of this same

point in the kinematic chain when the end effector location and orientation (shown in green) are fixed. The actual location of the end of the second link must lie somewhere on the magenta arc(s) where these two spherical surfaces intersect, and it must also result in an orthogonal intersection between the blue and red links that meet at this point.

D. Limitation on RoboSimian

The most significant motion planning limitation with RoboSimian is that it currently has a velocity limit at each joint of 1 rad/sec, which limits the ability of the robot to do any dynamic motion. The joint velocity limits affect both the speed with which the body can be moved while the stance feet are planted on the ground and the speed with which the swing leg can be moved to reposition itself of upcoming terrain. For each full gait cycle, each of four legs must move one-at-a-time a distance of one step length, plus a vertical pick-up and put-down distance, while the body needs to move a total distance of one step length. Correspondingly, swing leg motions account for more than 80% of the total time during steady-state walking in a crawl gait. Thus, it is the limitation of swing leg trajectory speed which has a far greater impact on the average forward velocity of the robot during a steady, repeating gait. Also, because the swing leg is free to travel in any continuous, non-colliding path through from its start to its end pose, the trajectory for the limbs to produce the fastest motion is not obvious. This motivates the use of a search algorithm to find the optimal path for the limb to move.

III. Planning Methods

A. Assumptions

To simplify the problem a little, we had to make some assumptions. We assumed that the robot would be walking on completely flat ground with no obstacles. Also, it was to walk in a straight line. As for the limbs, we simplify the problem by assuming that both stance and swing legs do not have the ability to pitch or roll the end effector. Both stance and swing leg end effectors were only able to modify yaw, so that they remained perpendicular to the ground. For all calculation of IK's, because the four limbs are identical, the front-right limb solutions was always used, with local coordinates in x and y simply mirrored as appropriate to calculate local end effector locations for the other three limbs. Throughout this thesis, local limb positions are defined using a right-handed coordinate system with the COB at the origin.

B. ZMP Based

Using the preview control planning approach for ZMP by Shuuji Kajita [1], we are able to get a trajectory for the Center Of Mass (COM) for a single mass at a constant height for the desired ZMP. For our point-mass model, the ground reaction force must always point directly toward the COM, which greatly simplifies the mathematical calculation of ZMP location on the ground. The ZMP of a cart and table model is found by using the dynamics:

$$\ddot{x} = \frac{g}{z}(x - ZMP)$$

where, x is the displacement of the COM in the x direction, g is gravity, z is the constant height of the COM, and ZMP is the x location of the ZMP on the ground. Also, using the preview control algorithm from Kajita, we are able to find a COM trajectory that will allow the cart to have the desired ZMP trajectory.

1. Implementation

As RoboSimian has limbs that together account for approximately 60 percent of the entire mass, the COB and COM locations vary significantly as the robot moves, requiring care in planning. Below, we describe the general process of planning desired, stable walking behaviors for RoboSimian.

Our first step was to determine the desired ZMP trajectory. This is done by using diagonal pairs of limb end effector positions. Because as long as the ZMP is inside the support polygon RoboSimian will be stable, using the two different end effector position of the limbs, we determined where the mid-point between the limbs were for the ZMP to be at.

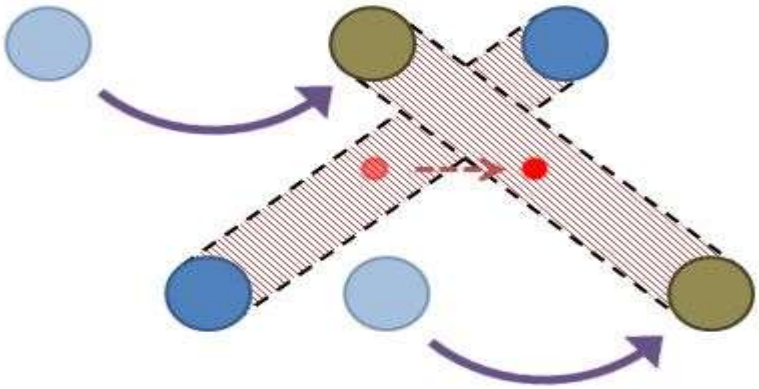

Figure 6. ZMP trajectory

As the light blue footholds move to the brown footholds, the ZMP should move from the light red position to the dark red positions because the support polygon (shaded area) changes.

As Figure 6 shows, the ZMP moves ½ the distance of the step size because the ZMP will have to move twice per cycle.

To obtain the COB trajectory for RoboSimian involved using the cart and table model and the nine-link model. First, we used Kajita's preview control method to obtain a COM trajectory for a single mass at a set height [1]. Afterwards, because RoboSimian cannot be represented with just a point mass, we used the nine-link model which each limb has two point masses, one for each link. For this model, we used the COM trajectory from the cart and table model as our desired motion plan, modifying the COB trajectory such that the resulting COM of the nine-link planar model is as desired. Finally, that new COB trajectory is fed into the RoboSimian model and the resulting ZMP of the full model, shown in Figure 4, is calcuated. This entire process is represented as a flowchart in Figure 7. Within the process of simulating the movements, we included code to get the trajectories of the end

effector of the limbs which allowed us to obtain IK solutions for the RoboSimian to get joint angles and joint velocities.

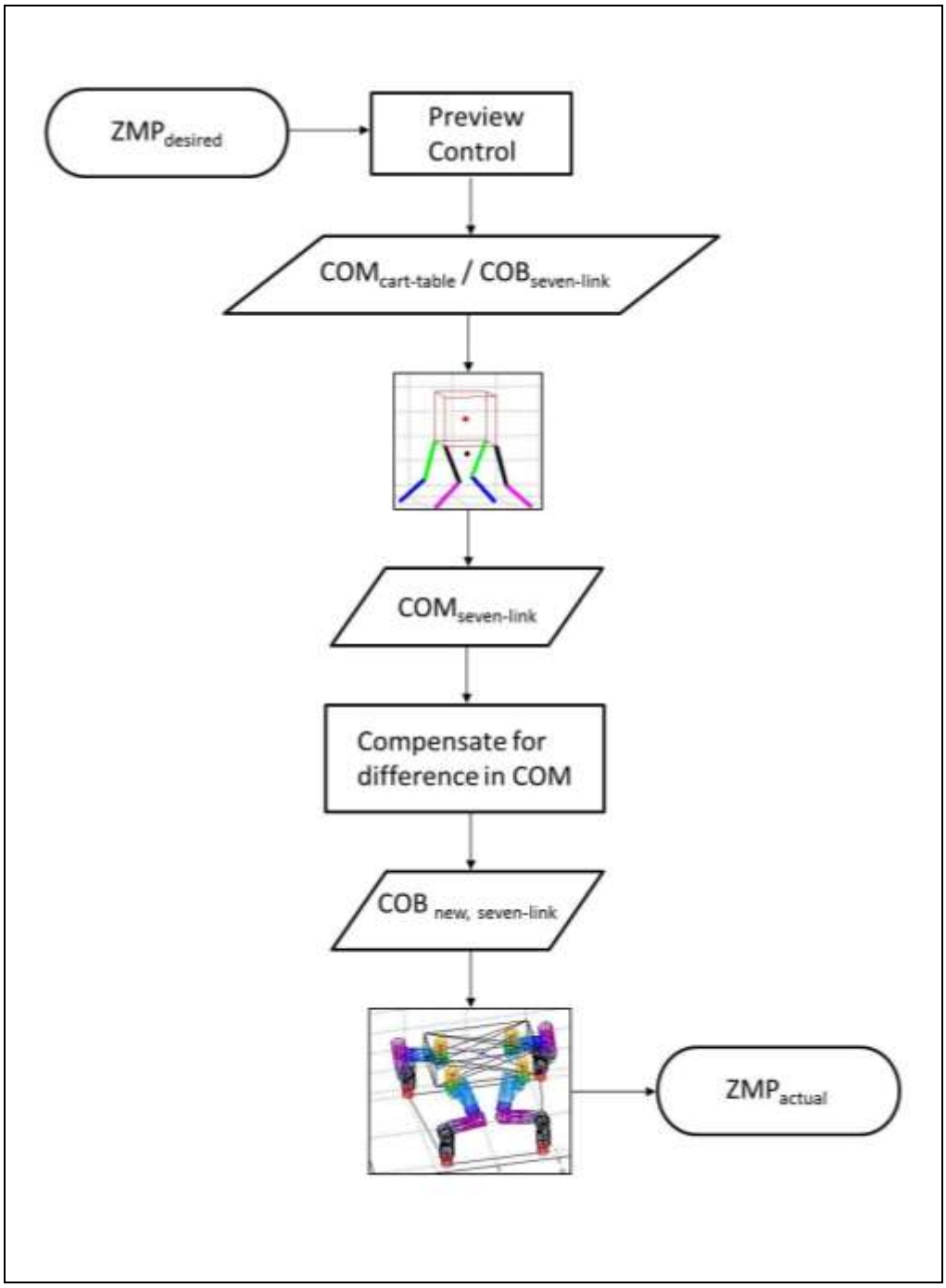

Figure 7. Flowchart for calculating actual ZMP from desired ZMP

C. Value Iteration

To find paths for a 7-DOF limb to be able to start at one point and end at another point in space, one now-common approach in robotics is to use a Rapidly-exploring Random Tree (RRT) search algorithm. However, the RRT finds a path out of many different possible paths and it will probably not be the optimal path to move. This method is useful when the speed of the limb is not crucial to how the robot performs and when avoiding collisions between the limbs and nearby obstacles is an important challenge. A popular case where RRTs are used is for robots that perform manipulation with arms. As for RoboSimian's walking fast, that is not the issue. Because our primary goal was to be able to walk faster on flat terrain, we used a different algorithm called value iteration.

Value iteration [13] is also known as backward induction. It is a process that derives the optimal policy and the corresponding "cost-to-go" from any initial condition to a desired final state. The four elements to this algorithm are the states, S, value of each state, V, actions that can be taken at each node, A, and a one-step cost, C, for transitioning from S to the next state, S'.

Initially, the different states are assigned a value which does not matter since after several iterations, the value of the states will converge to some other value. The value of the states over the iterations will be updated according to equation (1):

$$\mathrm{V(s)} = \min_{A}\{C(S, A, S') + \gamma * V(S')\} \tag{1}$$

where, γ denotes a discount factor, which was not used for my implementation. While iterating for all the different possible states, we have to make sure that the

goal state has a fixed value since once the path reaches the final goal, it should not go anywhere and add to the final cost. For our problem formulation, the swing leg only moves in x and y after it is a safe distance about the ground and so the fixed value at the goal state represents the time required to lower the leg in a straight path down to contact the flat ground surface. Since the final cost is fixed, with n iteration, all the states that can reach the goal state within n steps using their actions will have an updated value. Once the value iteration runs for a while, the values for the states will converge to an optimal cost which represents how long it will take for the limb to swing from any of the states in the mesh to the goal state. For each potential initial state in the mesh, the total time for swing leg motion also includes the time to lift the leg solely in the z direction, to move from the flat ground to initial point in the mesh. Note that both the initial pick-up and final set-down of the swing leg are fixed costs that do not depend on the value iteration solution.

1. Implementation

For the problem of finding the optimal path for a swing leg, the states consisted of x, y, z positions and the yaw of the end effector and created a 4D-mesh to work with. The grid end-points were determined by looking at the reachable workspace and by observing the regions where the joint velocities will be the lowest when moving the limb a small distance. This is a crucial step because we have a 4D problem and each extra point that we introduce in any dimension will add a number of points to the grid equivalent to the number of elements in the other dimensions

multiplied together. As for the different actions, different distance, azimuth, elevation and dyaw were chosen so that for each state, the reachable S' were on spheres around the state with different radiuses and the one-step cost was assigned by how long it will take the limb to perform that action. Because of the limitation of 1 rad/sec on the joint velocities of RoboSimian, the time was measured by saturating the fastest joint in the movement to be 1 rad/sec. However, because the grid is a finite grid and the action may end up causing S' to be outside the grid, it was made sure that if this was the case for an action, S' was looped back to be at the edge of the grid in order to avoid generating non-number values due to interpolating using states outside of the grid. To determine the value of S', the one-step cost was assigned by using interpolation with the values of the states around S'. The initial values for the states were assigned to be zero. However there were certain situations where we needed to modify the values after every iteration and after the values converge.

When assigning values of the states after each iteration, there were two main cases to consider. First, as it was stated in Section C, the value of the goal state had to be fixed as a low cost, in our case, zero to ensure the trajectory given after the value iteration would lead up to the desired goal. Secondly, there was the issue of the limb not being able to reach a given state in the mesh. This was dealt with by assigning a very large initial cost to all infeasible states, so that they were never included in any optimal solutions found by the value iteration algorithm. Once the values of each state converge, since the path is for a swing leg, we needed to

consider the time it takes for RoboSimian to lift up and put down the leg as previously mentioned. Without taking this into account, the optimal path that is given by the value iteration will simply say that starting at the goal state is the fastest way to get to the goal because there is no penalty in moving the limb up before swinging and down after swinging. This was handled by figuring out the time it takes for the limb to move 10 cm along the z-direction from each state and adding it to the value of each state once V converges. Again, there will be states that are not able to reach 10 cm below, and these states will also be assigned a large cost so that they are not chosen as initial states for the path.

Once the initial state is chosen by finding the state with the lowest cost, starting from that point, we would look through the actions and execute the action with the lowest cost to determine the next state in the swing leg trajectory. Then this process was repeated with the next state until the trajectory arrived at the goal state.

IV. Simulation / Results

Note that all gait speeds are in feet/min, unless stated otherwise, and also the gait speed from value iteration is calculated using (2)

$$\text{Gait Speed} = \frac{Step\ Size}{4 * Swing\ Time + (Swing\ Time - Leg\ Up - Leg\ Down)} \quad (2)$$

(Swing Time - Leg Up - Leg Down) represents the time it takes for the body to move forward.

A. ZMP Based Gait Planning for RoboSimian

1. Cart and Table Model and Nine-Link Model Good Enough Estimation?

First, we needed to confirm that using the cart and table model to get a preliminary COM trajectory and then using that and the nine-link model to get a COB trajectory for RoboSimian was a good enough estimate. To do this, for each step size, we generated COB trajectories and ran the trajectories on the full RoboSimian model and compared the outputted ZMP and the desired ZMP we used to generate the COB trajectories. The results (Figure 8) shows that the error in the actual ZMP and desired ZMP increases as the step sizes become larger.

The end effector has a shape of a circle with a radius of approximately 0.055 m, and we used that to decide the worthiness of the models because the size of the end effector dictates the support polygon size. Also, we had to consider the changes that were made to the actual RoboSimian and the vibrations it may have. With all of this taken into account, it was concluded that the simplified model will be useful up to a step size of around 0.4 m.

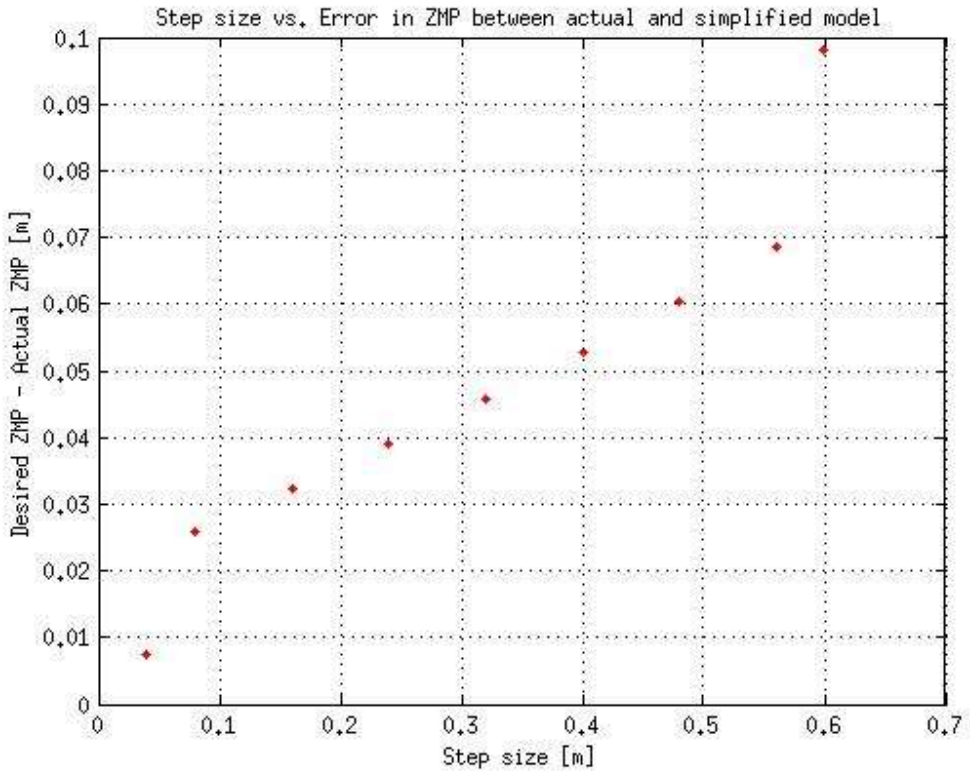

Figure 8. Maximum error in ZMP at each step size

As the feet for RoboSimian are approximately a radius of 0.055 m, trajectory generated by using the simplified models seem to work until around 0.4 m steps assuming that there are vibrations and changes to the robot itself.

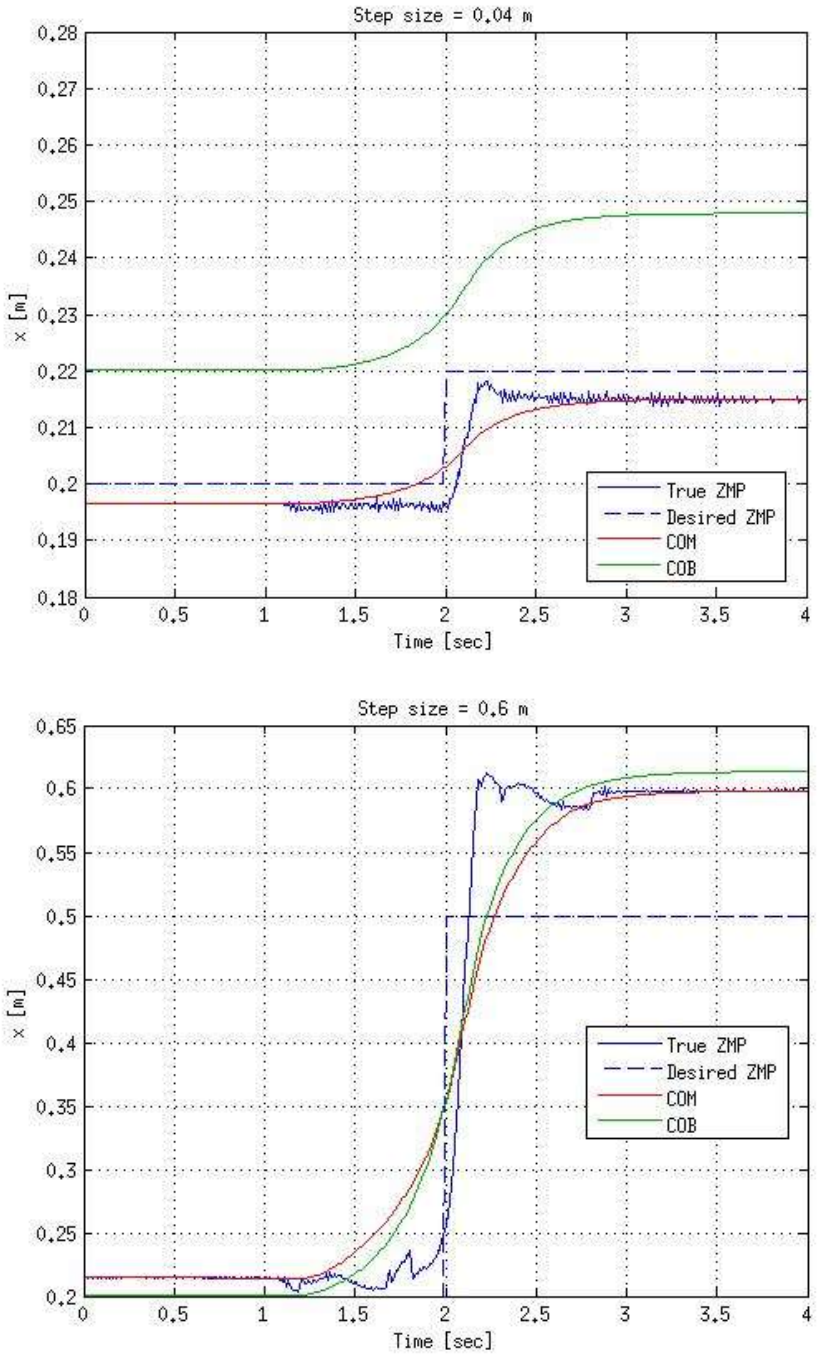

Figure 9. Actual ZMP, COM, COB for step size = 0.04 m (top) and step size = 0.6 (bottom)

We can observe that as the step size gets larger, the simplified model is no longer a good estimate.

2. Test Max Joint Velocity for ZMP Trajectory

Using the process explained in section III.B.1, we simulated what the maximum joint velocity for the stance legs would be for different step sizes.

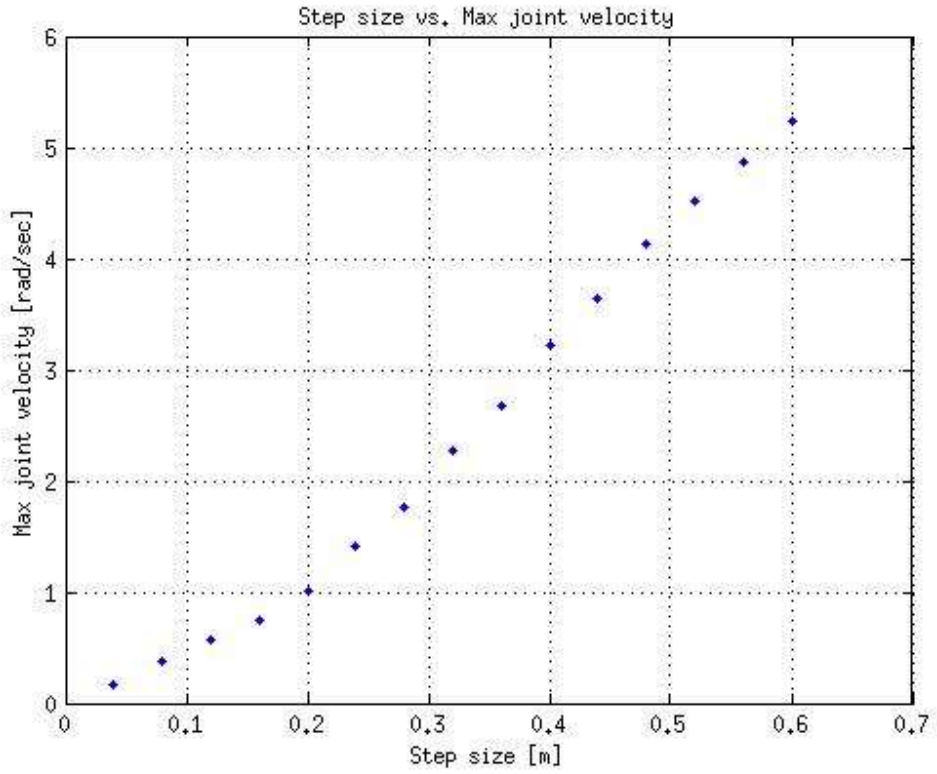

Figure 10. Step size vs. Maximum joint velocity needed

This plot is shows the maximum joint velocity when designing a gait using ZMP. Remember that the ZMP actually only needs to move ½ of the step size since the ZMP needs to move twice during one cycle.

As shown in Figure 10, the maximum joint velocity needed increases almost linearly as the step size increases for the gait. However, since the maximum joint velocity allowed for RoboSimian is 1 rad/sec, with using ZMP, it will only be able to

take steps slightly less than 0.2 m. This data can also be interpreted as the step size vs. time to take that step when the maximum joint velocity limit is 1 rad/sec.

B. Swing Leg Value Iteration for RoboSimian

When determining what the optimal path for RoboSimian should be, we looked at different step sizes, height of the swing leg and the difference between two families of IK solutions.

1. Determining Grid Size for Value Iteration

As mentioned in section III.C.1, the first step was to determine the grid size for the states to perform value iteration. Using the IK solutions, and for a wide range of points, we observed what the maximum joint movement was for a small perturbation in the x and y direction for different z heights for RoboSimian. From this data **(**Figure 11, 12**)**, it was determined that the best heights to move the legs were at 60 cm and 70 cm below the COB. We needed two different heights because we needed to consider the case where it is actually swinging its limb and the case where it will be moving its body, which is equivalent to swinging a limb 10 cm below the point where RoboSimian would be swinging its limb to take an actual step.

Then the next step was to determine what the grid size in the x and y directions should be and shed off points that will have no chance of being part of an optimal trajectory since if the grid is too large, the value iteration process would take too long. By observing the plots in Figure 12, we determined that the grid size should be at least from -0.65 to -0.4 in the y direction and from 0.2 to 0.8 in the x direction.

These values were chosen because it shows that in those ranges, the joint velocities are smaller relative to the other points.

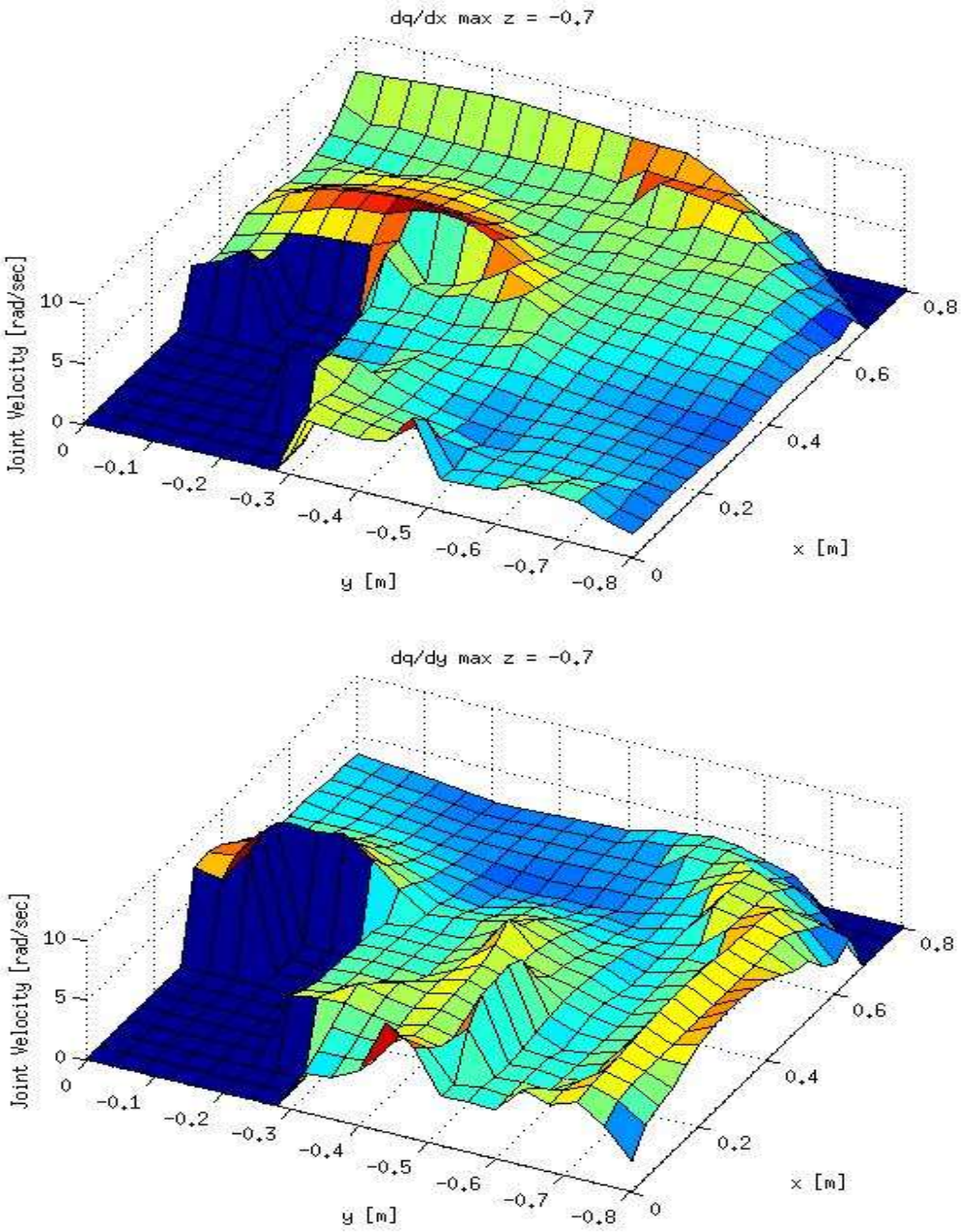

Figure 11. Joint velocity when each node is perturbed by a small amount in the x direction (top) and y direction (bottom) with RoboSimian's COB at the height of 0.7 m.

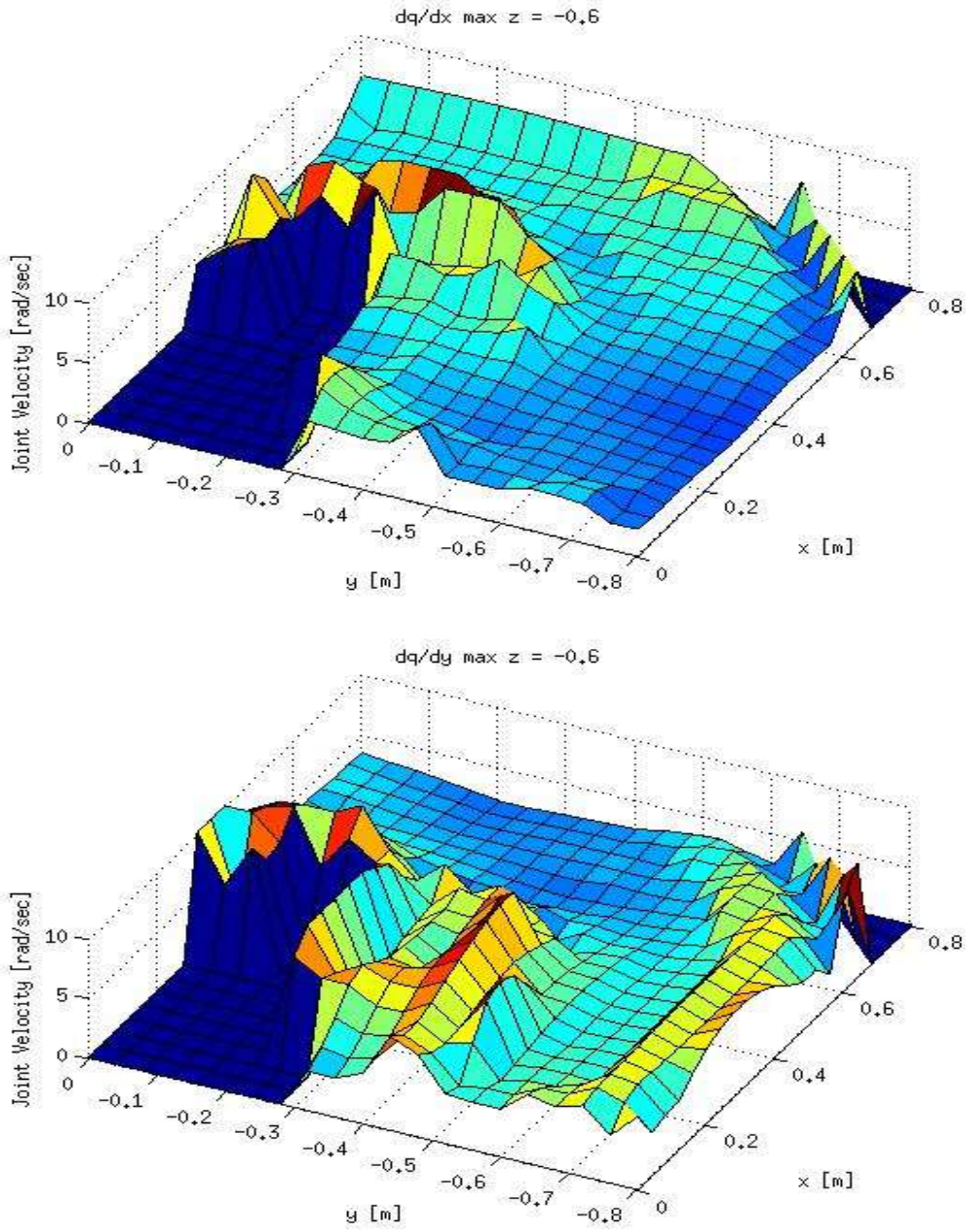

Figure 12. Joint velocity when each node is perturbed by a small amount in the x direction (top) and y direction (bottom) with RoboSimian's COB at the height of 0.6 m.

From this, we can observe that the grid size should at least be from -0.65 to -0.4 in the y direction and from 0.2 to 0.8 in the x direction. (Infeasible nodes received a value of 0 rad/sec)

2. Distance for Actions

When choosing the actions, we originally used a single distance slightly larger than dx in the grid, but then realized that when that was done, as the path gets closer to the goal state, it would diverge away as in Figure 13. This phenomenon was due to the fact that the action for states that are actually close to the goal state could not reach the goal and was trying to get to a spot that would actually be able to reach the goal with the exact step size of the action. To solve this problem, several distances were chosen so that it would allow smaller steps and the result was as follows.

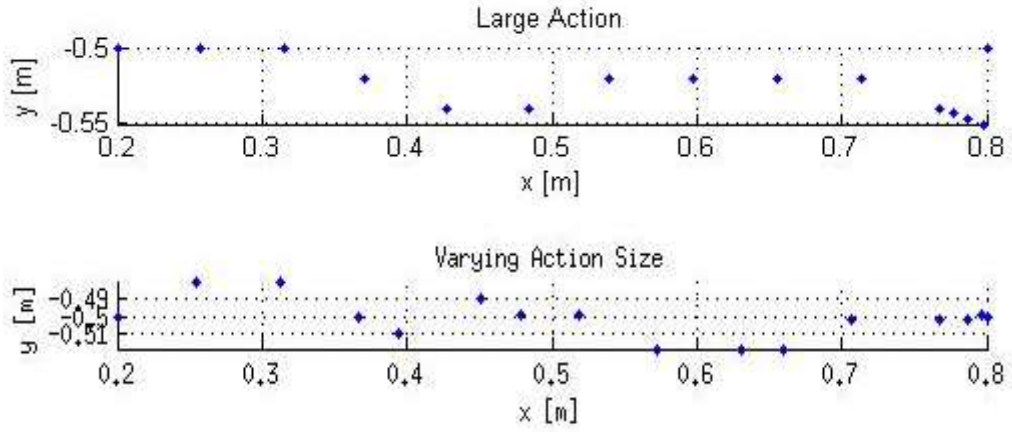

Figure 13. Large action size (top) vs. varying action size (bottom).

If the action is chosen to be a fixed size of larger than the dx in the mesh, when the path is close to the goal, it will diverge away, trying to get to a position where a single step to the goal would require that large action. (Note: These waypoints are changing in the z-direction as well)

By allowing to choose from different distances to move, there was no diverging happening near the goal state.

3. Optimal Step Size

Next, using value iteration, we observed what the optimal step size would be for Robosimian. Also, we determined what the optimal path would be for those step sizes. The simulation was conducted by setting the x-coordinate for the goal state to be different. For all of these simulations, the y and z-coordinates were set to be the same for the goal state and the initial state which were to be: y = -0.5 m and z = -0.6 m so that each gait cycle will have the same pattern.

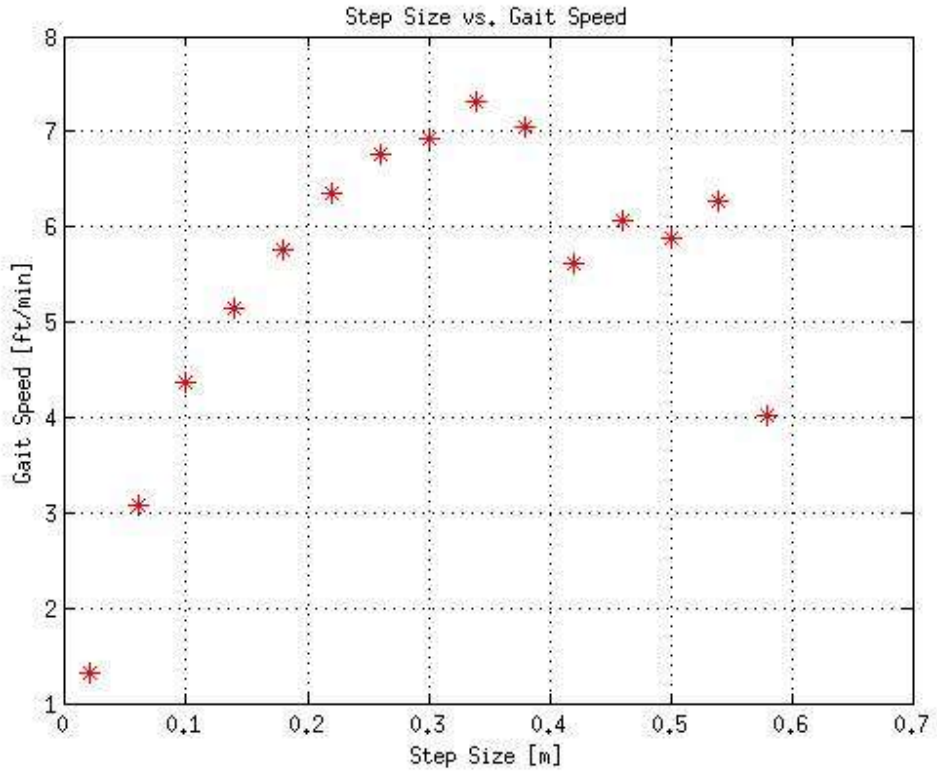

Figure 14. Step size vs. Gait speed.

This particular plot is for the 3rd family of IK solutions with a step height of 0.1 m.

At first thought it seemed like the gait would be the fastest when the steps are larger because picking up and putting down the limb takes up a fair amount of the time for swinging the limb. However, we observed that as the steps got larger, it required the limbs to start closer to the body where the joints require more movement than when it is further away from the body, which consequently decreased the gait speed.

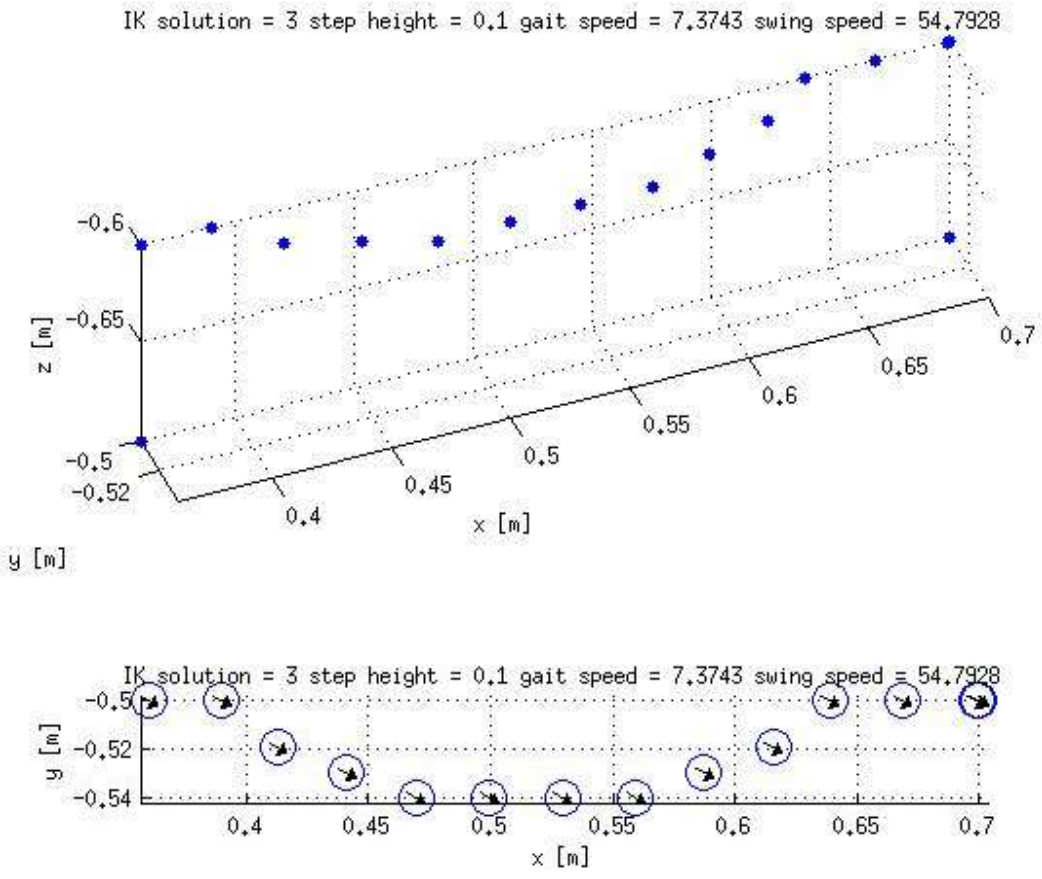

Figure 15. Optimal path using the 3rd family of IK solutions with a step height of 0.1 m (top) and an overhead view with the arrows representing the yaw of the end effector (bottom). Notice that the optimal path swings out instead of moving directly straight.

From Figure 14, we can see that a step size of 0.34 m achieves the fastest gait speed when the step height is 0.1 m (Figure 15). Once the optimal step size is determined, we ran the value iteration again to find the optimal trajectory for the limb, which turned out to be a curved path outwards in the y direction instead of a straight path. This is due to the fact when the limb tries to move straight, it requires more joint movements (Figure 16) hence increasing the time needed to move the same amount.

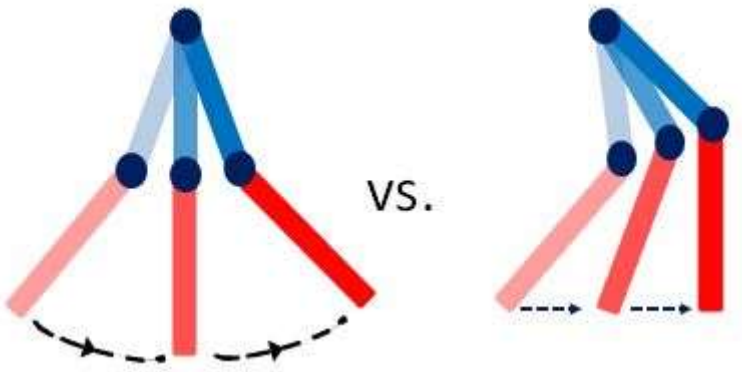

Figure 16. Curved path vs. Straight path

More joint movement is needed when moving straight. Hence increasing the time needed to move.

4. Optimal Speed and Different Family of IK Solutions

Since the limbs are 7-DOF each, there are many IK solutions for a single xyz-coordinate in space. As explained in section I.C, there are 8 different families of the IK solutions. Out of those, we chose to compare the 3rd and 8th families to see how they would differ in terms of different step sizes and gait speeds **(**Figure 17).

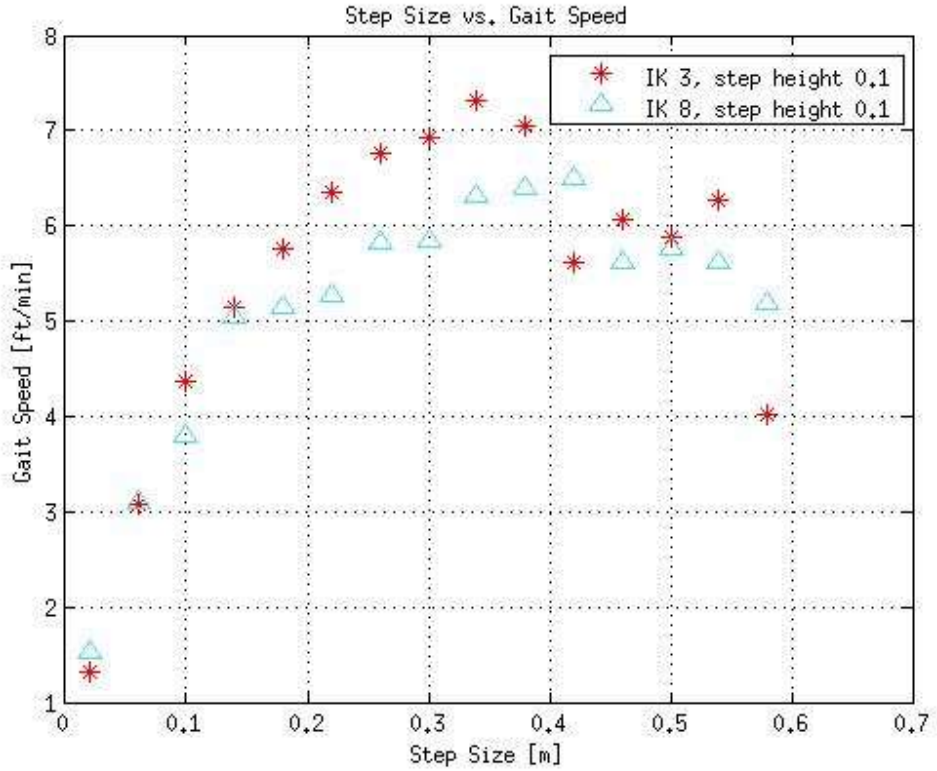

Figure 17. Step Size vs. Gait speed comparison between two families of IK solutions.

From running value iteration for the 3rd and 8th IK solution families with different step sizes, although the 3rd family had a significantly slower speed for 0.42 m and 0.54 m sized steps, it produced a faster speed overall.

5. Optimal Speed and Height of Swing Limb

To figure out what height was better for the swing limb to be at, we compared the difference in optimal speed for 10 cm and 6 cm off the ground. When looking at humans walking, we barely take our foot off the ground. However for a robot, we have to take the vibration and uncertainty into consideration and plan for a higher lift of the limb to make sure it does not collide into the ground when swinging its limb.

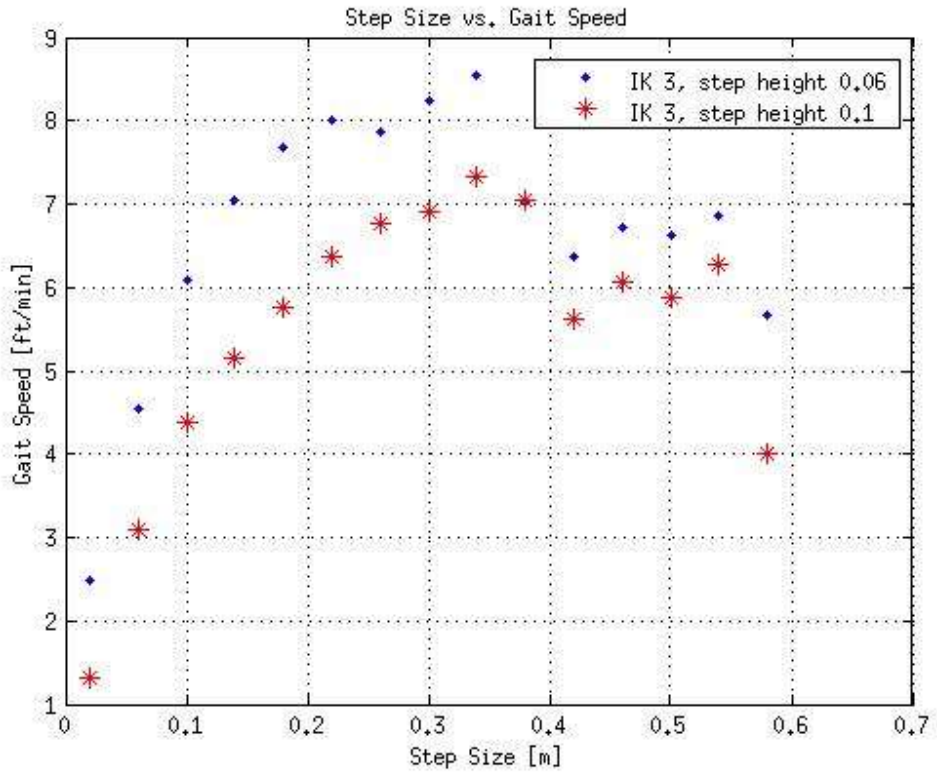

Figure 18. Step Size vs. Gait speed comparison between 0.06 m and 0.1 m step heights.

Because lifting the limb 0.06 m requires less time than lifting 0.1 m, it has a better performance. Without any pitching on the end effector, meaning that the end effector is always perpendicular to the ground, it was interesting to see the percentage of the time it took the limb to be lifted and lowered. For example, for the optimal swing trajectory found in section IV.B.3, the lifting and lowering process of the limb takes 38.76 % of the total time.

6. Optimal Overall

Because there are different factors such as using different family of IK solutions and step height, I combined four different plots to see which combination generates

the optimal speed for the gait. It turned out that lifting the limb off the ground by 0.06 m and using the 3rd family of IK solutions produced the fastest gait speed when taking 0.34 m sized steps which was 8.542 ft/min.

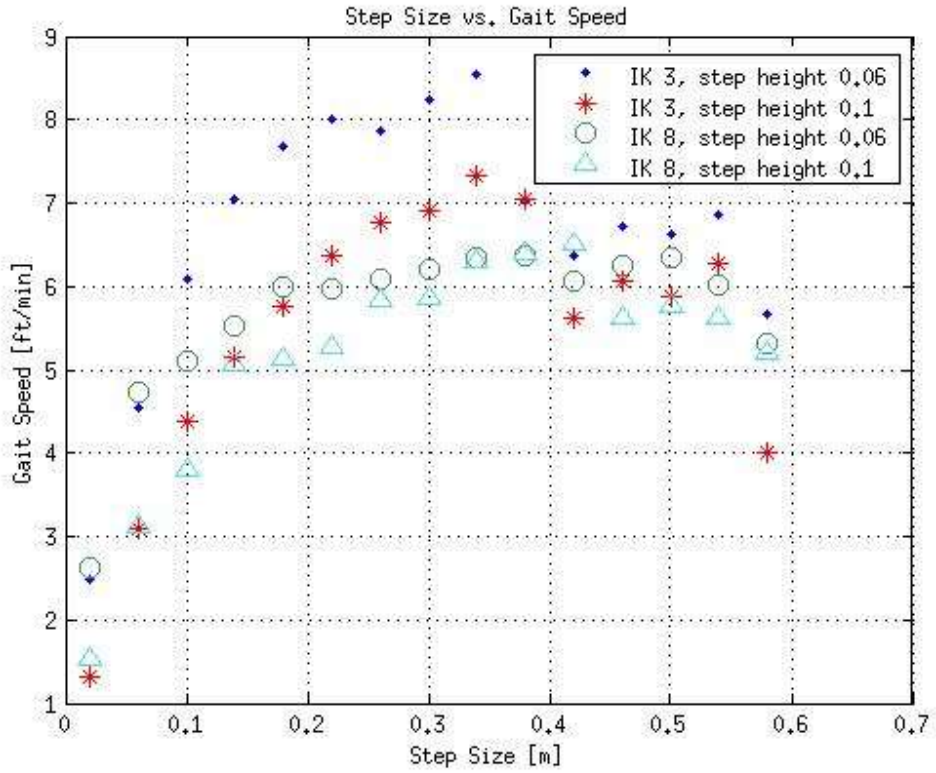

Figure 19. Overall comparison of Step Size vs. Gait speed

7. Optimal?

The last step of finding an optimal step size and path was to confirm that it is actually optimal. As the work optimal suggests, any kind of perturbation I give the optimal path should cause the gait to be slower. Since the value iteration is supposed to find the optimal path, the path should be faster than a path with random perturbations in x, y, and/or z direction for each waypoint in the path. It turned out

that swinging the limb at 0.06 m with 0.5 m sized step using the 3rd IK solution was a locally optimal path.

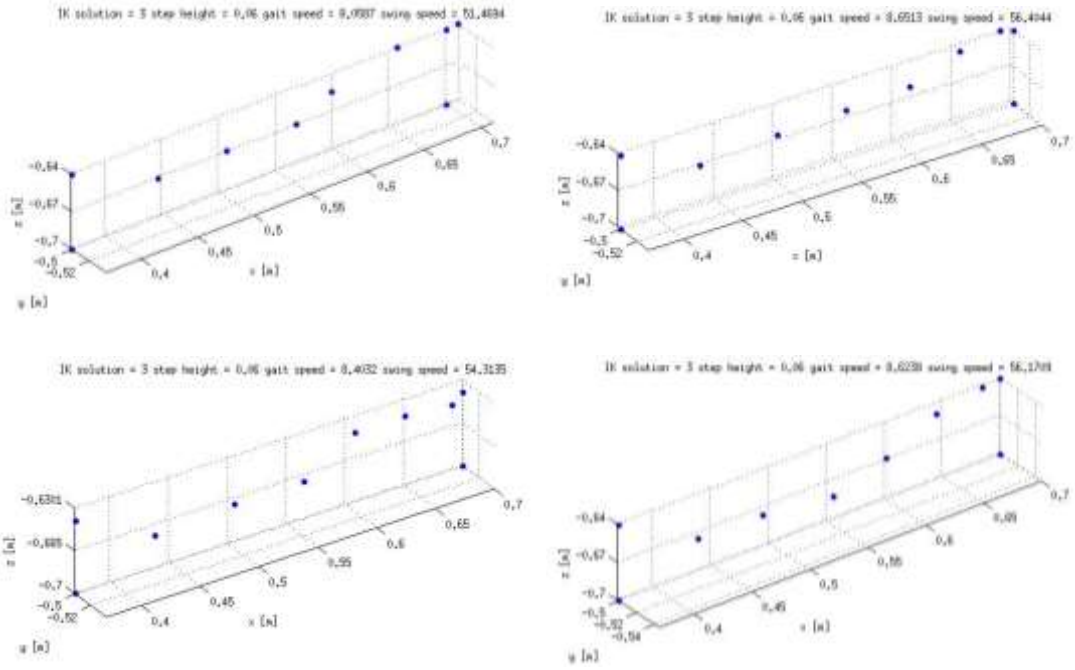

Figure 20. An example of gait speed comparisons with different perturbations in x (top left), y (top right), z (bottom left), xyz (bottom right) directions.

From Figure 15 and Figure 20, we can see that the optimal path that was found had a gait speed of 8.542 ft./min but when perturbations in x, y, and z directions separately and x, y, and z directions all together were applied, after simulating several times, the gait speed decreased with an average of 0.5026 ,0.1513, 0.1803, 0.4353 ft./min respectively.

C. When to Use ZMP Based or Value Iteration based

After exploring the two different methods for planning, we observed when it will be appropriate to use one method over the other. Since the main concern throughout the paper was the limited joint velocity on the robot, we took a look at how much the joint velocity should be increased to be able to perform a dynamic ZMP based gait.

With the help from Katie Byl, we found the required maximum joint velocity depending on the step sizes for the ZMP based gait and the speed of the gait based on the joint velocity limits using the method explained in Section III.B. This data was compared with the gait speed of the gait using value iteration. As for value iteration, since the gait speed increases linearly according to the joint velocity limit, the fastest gait speed observed from Section IV.B for when the velocity limit is 1 rad/sec was used as the guideline.

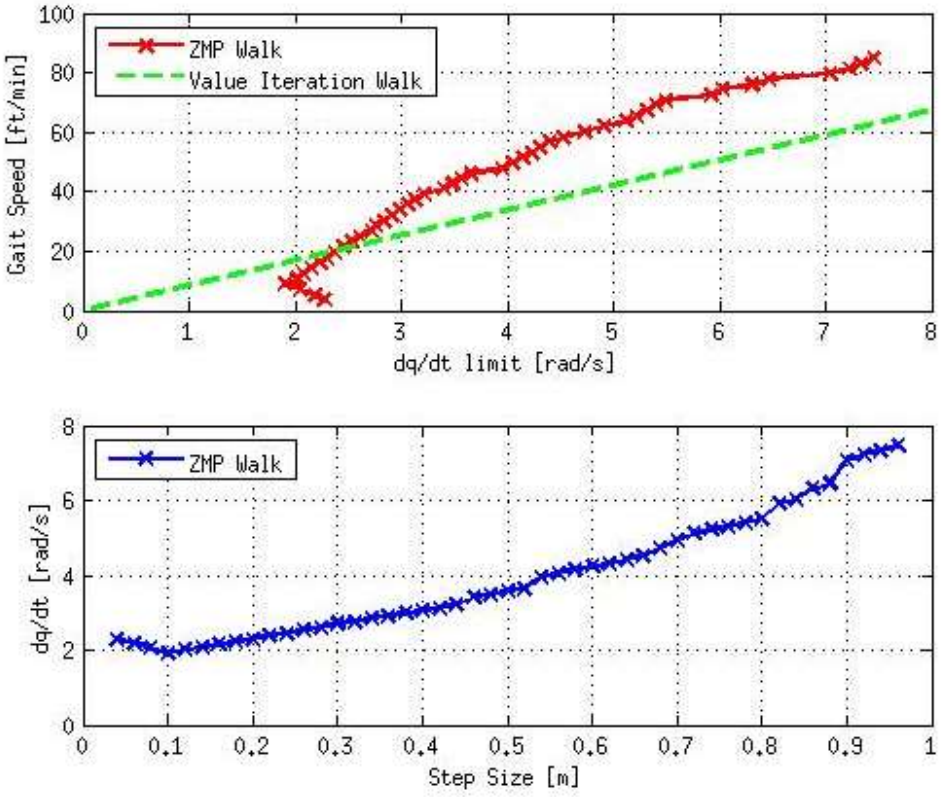

Figure 21 ZMP based vs. Value Iteration based.

From Figure 21, we can see that if the joint velocity limit is under 1.9 rad/sec, it is not possible to use a ZMP based gait planning. However, once the joint velocity limit is greater than approximately 2.4 rad/sec, using the ZMP based gait will be faster than using the value iteration based gait.

V. Conclusion

Throughout this thesis, we investigated ZMP based planning and value iteration based planning in order to develop a way to make RoboSimian walk faster. First, we looked at ZMP based planning to determine how well RoboSimian would be able to perform dynamic movement given the limitation in joint velocity. In order to do this,

a simplified model of the RoboSimian was created to plan a COB trajectory to ensure the ZMP to be at where it needs to be. We determined that the model will be useful up to a step size of approximately 0.4 m since for anything beyond that, there is the danger of the ZMP being outside the support polygon, resulting in RoboSimian to fall over.

Then, we observed the maximum joint velocity needed for certain step sizes. As the step sizes increased, the maximum joint velocity needed increased almost linearly. Unfortunately, with the 1 rad/sec joint velocity limit, it was shown that RoboSimian would only be able to perform about 0.2 m sized steps.

Knowing that RoboSimian could not do much dynamic movements, we optimized the swing trajectory for the limbs using value iteration. Taking the joint velocity limit into account, first, we observed the regions in the reachable workspace where the limb needed the least joint movement for a small perturbation. Using that data, a 4D grid consisting of x, y, z, and yaw was created and value iteration was implemented while making sure that the action sizes vary and that it does not carry the state to be outside the grid. Once the value iteration was implemented, we compared the gait speed with the step sizes for different IK solutions and step heights.

In the end, the trajectory provided when the 3rd family of IK solutions was used for 0.06 m step heights turned out to be the optimal trajectory. The optimality of the solution given by value iteration was tested by imposing perturbations in x, y, and z

directions separately and together on the waypoints produced by value iteration. Through this experiment, it was proved that the trajectory is locally optimal.

Comparing the two methods, we were also able to conclude that the ZMP based gait is useless unless the joint velocity limit is at least 1.9 rad/sec. However, it is better than the value iteration method if the joint velocity limit is greater than about 2.4 rad/sec.

VI. Future Work

So far, as it has been stated before, there has been no pitch involved in the end effector. However, when I simulated a simple straight forward swing trajectory for a limb with and without pitching the end effector, it resulted in the trajectory with pitching produced a speed almost twice as fast as the gait without pitching. This would be an interesting feature to look at. Although it would be adding another dimension for the value iteration, it would produce a faster trajectory.

Also, the results from this thesis should be applied to the actual RoboSimian at JPL to confirm the simulations. I would be interested to see upgrades to the joints of RoboSimian for improved joint velocity limits. If that were to be done, RoboSimian would be able to perform more dynamic movements, allowing for it to walk faster.

On top of that, combining the results of value iteration and ZMP based planning can be combined to fully optimize the speed of RoboSimian's walking.

Another task that could be added is to find optimal trajectories when it is not on even ground. For example, if RoboSimian needs to climb a flight of stairs, we could

search a wider range in the z direction to find optimal trajectories for the limb from the ground to the next step on the flight of stairs.

References

[1] S. Kajita *et al*., "Biped Walking Pattern Generation by using preview control of Zero-Moment Point," in *Proc. ICRA*, Taipei, Taiwan, 2003, pp. 1620-1626.

[2] Z. Tang, C. Zhou, and Z. Sun. "Trajectory Planning for Smooth Transition of a Biped Robot," in *Proc. ICRA*, Taipei, Taiwan, 2003, pp. 2455-2460.

[3] T. Sugihara and Y. Nakamura, "A Fast Online Gait Planning with Boundary Condition Relaxation for Humanoid Robots." in *Proc. ICRA*, Barcelona, Spain, 2005, pp. 305-310.

[4] K. Byl. "Metastable Legged-Robot Locomotion," Ph.D. dissertation, Dept. Mech. Eng., Massachusetts Institute of Technology, Cambridge, 2008.

[5] H. Dong *et al*., "Gait Planning Of Quadruped Robot Based On Third-Order Spline Interpolation," in *Proc. Int. Conf. on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 5756-5761.

[6] N. Kohl and P. Stone, "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion," in *Proc. ICRA*, New Orleans, LA, 2004, pp. 2619-2624.

[7] S. Kagami *et al*., "A Fast Dynamically Equilibrated Walking Trajectory Generation Method of Humanoid Robot," *Autonomous Robots*, vol. 12, pp. 71-82, 2002.

[8] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, 2003.

[9] S. Kajita *et al*., "Biped Walking Pattern Generator allowing Auxiliary ZMP Control," in *Proc. Int. Conf. on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 2993-2999.

[10] J. J. Kuffner, Jr. and S. M. LaValle, " RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proc. ICRA*, San Francisco, CA, 2000, pp. 995-1001.

[11] K. Nishiwaki *et al*., " Online Generation of Humanoid Walking Motion based on a Fast Generation Method of Motion Pattern that Follows Desired ZMP," in *Proc. Int. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002, pp. 2684-2689.

[12] S. Kajita, O. Matsumoto, and M. Saigo, "Real-time 3D walking pattern generation for a biped robot with telescopic legs," in *Proc. ICRA*, Seoul, Korea, 2001, pp. 2299-2306.

[13] "CHEETAH – Fastest Legged Robot," Internet: http://www.bostondynamics.com/robot_cheetah.html, [Nov. 18, 2013].

[14] "RoboSimian Under Construction," Internet: http://www.jpl.nasa.gov/spaceimages/details.php?id=PIA17301, Jul. 11, 2013, [Nov. 18, 2013].

Appendix

A. Psudo Code: Value Iteration Trajectory Search

Variables:

S: states

V: value of the states

A: actions

C: one-step cost

while err > 1e-4

$V_{orig} = V$

for A

V_{temp} = interpolate V for S + A

$V_A = V_{temp} + C_A$

end

$V = \min\{V_A\}$

V (goal) = 0

$err = \max\{V - V_{orig}\}$

end

$V_{total} = V$ + time to lift leg at each S

for x

$xmin = \min\{V_{total}(x, y_{init}, z_{init}, \text{all yaw})\}$

speed = travel distance / speed

end

x_{init} = argmin{speed}

yaw_{init} = argmin{V_{total} (x_{init}, y_{init}, z_{init}, all yaw)}

current = [x_{init}, y_{init}, z_{init}, yaw_{init}]

waypoints = current

while current ~= goal

for A

V_{cost} = interpolate V for current + A

end

A_{min} = min{V_{cost}}

current = current + A_{min}

waypoints = [waypoints; current]

end

B. RoboSimian

Figure 22. RoboSimian at DRC Trials in December 2013