CHAPTER IV

Humanoid Crawl Gait

Using the Nao platform, crawling gaits applicable to humanoid robots was explored. Not only is crawling a more stable gaiting strategy than walking, but it gives the robot access to areas of the environment that are inaccessible via walking alone. A low-profile laterally symmetric crawl gait is described, modeling the robot as a closed-chain manipulator with pseudo-static dynamics. This gait was parameterized on three joint variables and optimized using a cubic splines approach via a genetic algorithm.

As was described in Chapter III, the gaiting modality directly affects the navigation strategy. By selecting the appropriate gaiting strategy, the robot can modify its movement to achieve a commanded goal. This modified movement also modifies which environmental objects present as obstacles. Expanding the library of gaits legged robots have access to allows robots to be increasingly more capable and applicable to a wider range of scenarios.

4.1 Humanoid Crawling

Unlike walking and running which enjoy precise definitions, crawling seems to only have a subjective notion. [**?**] asserts that crawling is a statically stable walk. This definition is problematic as bipedal gaits have been demonstrated in [**?**] that are statically stable but would not be classified as crawling. In addition to this, soldiers performing the high army crawl [**?**] can be seen to perform this motion very quickly which introduces a dynamic component to the crawl. The standard or baby crawl is described as using one's hands and knees to produce forward motion. In contrast to this, crawls such as the leopard, tiger, bear, and crab use hands and feet to produce forward motion and the low army crawl uses the hands to drag and one leg to push the body across the ground. Such diversity in crawling motion makes it difficult to differentiate a crawl from a statically stable quadrupedal gait that uses something other than the end effectors to interact

Figure 11: The pane on the left shows the leg configuration of the Nao when the Hip Yaw-Pitch DoF is fully turned in. The right pane show the leg configuration when the Hip Yaw-Pitch is fully turned out. The legs are mechanically linked, making the amount of Hip Yaw-Pitch equal for each leg.

with the environment. Despite this, the presented gait produces a motion that many would associate with humanoid crawling.

4.1.1 Nao Crawling Limitations

A primary limitation to the gaits that can be produced by the Nao is the limited number of degrees of freedom (DoF) of the platform in contrast to the large number of DoF present in humans. While the Nao has 25 DoF, the human body has 244 [**?**]. The human arm and leg each have 7 DoF while the Nao's arms and legs each have 5. Nao's hips have one more degree of freedom, called Hip Yaw-Pitch, which turn the legs together at an angle. Figure 11 illustrates this degree of freedom more clearly. The rest of the DoF of the Nao are in the hands and neck. Notably, Nao has no back joint. This prevents the gait designer from prescribing a twisting motion for use in the crawl gait. These limits in motion preclude the execution of any gaits that require lateral twisting or sagittal arching. Examples of this are shown in Figure 12 .

4.2 Projected Profile Humanoid Crawl Gait

The crawling gait presented in this thesis is based on viewing the humanoid form as a set of manipulators on the sagittal plane. Figure 13 illustrates this concept. When the robot is laying in the prone position, it necessarily makes contact with the ground. If we then view the robot from the side (looking at the sagittal plane) we can model it as a planar kinematic chain. If the chest and knees are making contact with the ground,

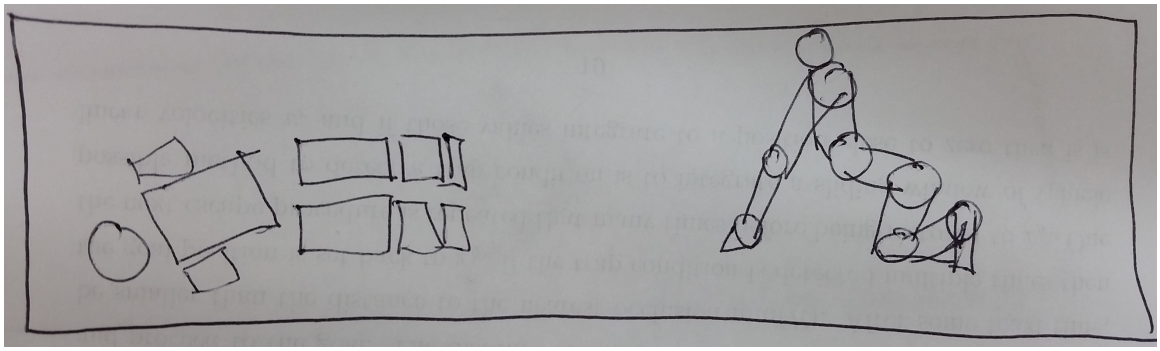

Figure 12: Illustration of crawl motions that involve actuated back joints. The left pane shows lateral twisting during a crawl. The right panes shows sagittal arching while on hands and knees.

then the arms from the shoulder joint to the hand, and the legs from the knees to the toes are free to move without affecting the rest of the body. This produces two open chain manipulators. In the case of the Nao, each has two degrees of freedom in the sagittal pane, allowing the hands and feet to move independently.

If the elbows and toes are placed on the ground, then the body from the toes to the elbow can be viewed as a closed chain manipulator. This allows the joints to work together to move the center of mass. Kinematically, these two phases share two common configurations. The first configuration is when the elbow is at full extension and the toes touching the knees. We will call this the "extension" configuration. This can be viewed as the robot reaching forward. The second is when the elbow is at full flexion and the ankles at extension. We will call this "compression". This can be viewed as the robot having pulled itself forward. These motions can then be combined to produce a full gait.

Using the Nao robot as an example, Figure 14 shows the full gaiting sequence. The robot initializes itself in the open chain configuration. From this, it can position its end effectors (the toes and elbows) into the first common configuration "extension". Next, the robot is in the closed chain configuration in which it can transport its center of mass forward until it reaches the "compression" configuration. Finally the robot is again in the open chain configuration and the cycle can start again.

The gait is laterally symmetric. If we actuate the joints at an appropriate rate, dynamic effects from the robot's motion do not become a significant factor. As detailed in Chapter **??**, this gait can be performed on the Nao at a speed of 1 ft every 6 to 8 seconds. The wide surface area of the forearm provides a high coefficient of friction

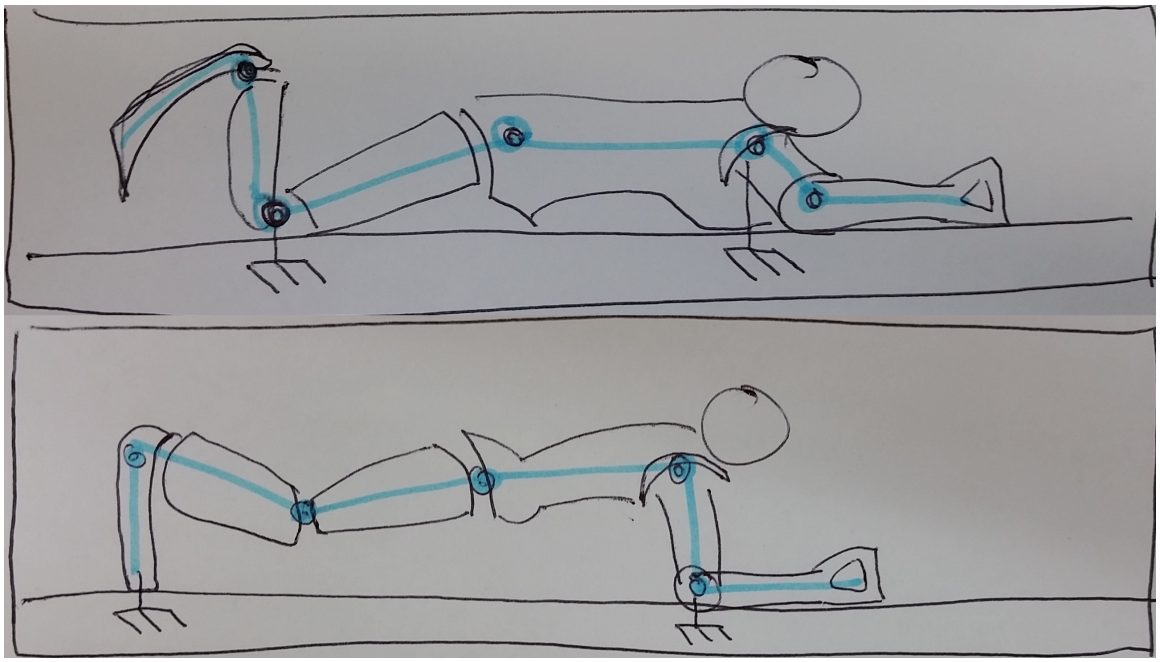

Figure 13: Illustration of Projected Profile concept. Both panes show the saggital view of the Nao with a schematic representation of the kinematics. In the top view, the robot appears as two open chain manipulators. In the bottom view, the robot appears as one closed chain manipulator.

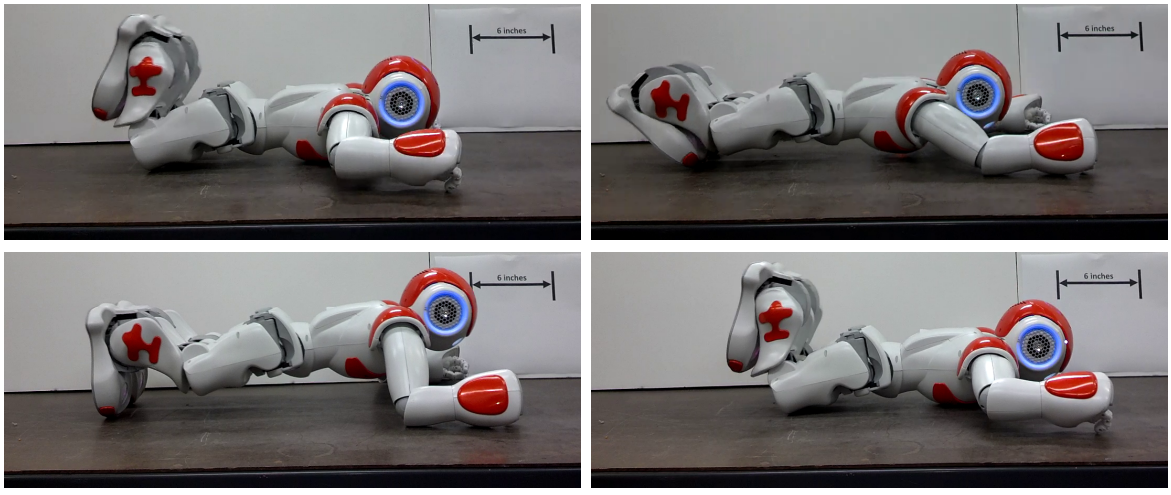

Figure 14: The above sequence shows the motion segments and robot postures in the crawl gait. The upper left pane shows the initial open chain configuration. The upper right pane moves the robot to the "extension" configuration. The lower left shows the "compression" configuration. Finally, the lower right shows the robot, having translated forward, once again in the open chain configuration. A 6-inch marker is shown in the background as a length scale reference.

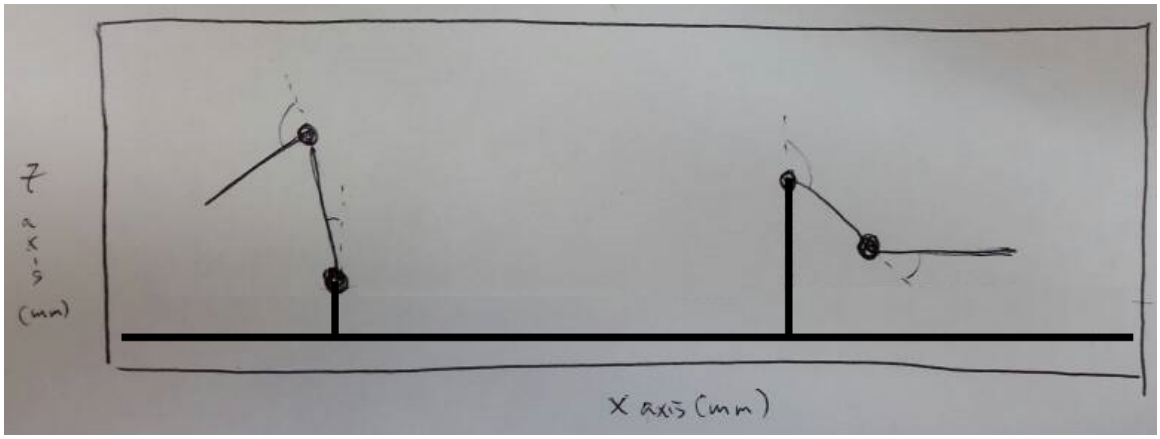

Figure 15: Simplified kinematic model of the sagittal projection of the open chain configuration. The manipulator on the left represents the tibia-foot chain. The manipulator on the right represents the humerus-forearm chain. The origin of the knee and shoulder are represented as having a z-axis offset because the knee and chest of the robot have heights that raise their origins.

against slipping and the small surface area of the toes can act as a point of high pressure which can dig into soft surfaces such as carpets. The gait is statically stable in the sense that the robot's motion can be paused at any point and the robot will maintain that pose. The gait does not depend on the robot sliding along the surface nor does it highly depend on surface friction to pull the robot forward. The gait has a very low profile. The highest point on the robot during the gait (which is the top of the head) is about 8 inches off of the ground. In contrast, during walking, the Nao robot stands 23 inches tall.

4.2.1 Open Chain Kinematics

The Projected Profile crawl gait has two kinematic configurations: open chain and closed chain. In the open chain configuration, the robot acts as two independent planar manipulators. Each manipulator has two degrees of freedom as can be seen in Figure 15 .

With the Nao facing downwards, the feet towards the origin and the head in the positive x direction, the forward kinematics for each manipulator are described by:

$$x = x_0 + l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) \tag{4.1}$$

$$z = z_0 + l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) \tag{4.2}$$

where x_0 and z_0 are the superior and posterior offsets with respect to the global frame, respectively. l_1 is the length of the humerus/tibia, l_2 is the length of the forearm/foot, θ_1 is the angle subtended by the shoulder/knee and the x-axis, and θ_2 is the angle subtended by the ankle/elbow and the x-axis of the humerus/tibia.

The solutions to the inverse kinematics problem can be seen as:

$$\theta_2 = \cos^{-1}\left(\frac{(x-x_0)^2 + (z-z_0)^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{4.3}$$

$$\theta_1 = 2\tan^{-1}\left(\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}\right) \tag{4.4}$$

$$A = (x-x_0) + (z-z_0) + l_1 + l_2(\cos(\theta_2) + \sin(\theta_2)) \tag{4.5}$$

$$B = -2(l_1 + l_2(\cos(\theta_2) - \sin(\theta_2)) \tag{4.6}$$

$$C = (x-x_0) + (z-z_0) - l_1 - l_2(\cos(\theta_2) + \sin(\theta_2)) \tag{4.7}$$

This solution derives from the standard inverse kinematics procedure of inverting the forward kinematics. θ_1 must be chosen such that no part of the robot tries to intersect with the floor. In practice, the two argument arc-tangent function $atan2$ is used instead of $\tan^{-1}$.

4.2.2 Closed Chain Kinematics

The closed chain configuration models the toes and elbows of the robot as being fixed to the ground. As with all closed chain kinematics, describing the forward kinematics requires solving an inverse kinematics equation. Modeling the robot in the same orientation as the open chain, with the toe at the origin and neglecting the thickness of the elbow, the forward kinematics of the closed chain are:

$$d_e = \sum_{i=1}^{5} l_i \cos(\sum_{j=1}^{i} \theta_j) \tag{4.8}$$

$$0 = \sum_{i=1}^{5} l_i \sin(\sum_{j=1}^{i} \theta_j) \tag{4.9}$$

$$\alpha = \sum_{i=1}^{5} \theta_i \tag{4.10}$$

where d_e is the prescribed distance of the elbow from the foot and α is the desired angle created by the x-axis of the humerus and the ground. θ_1 through θ_5 are the angles of

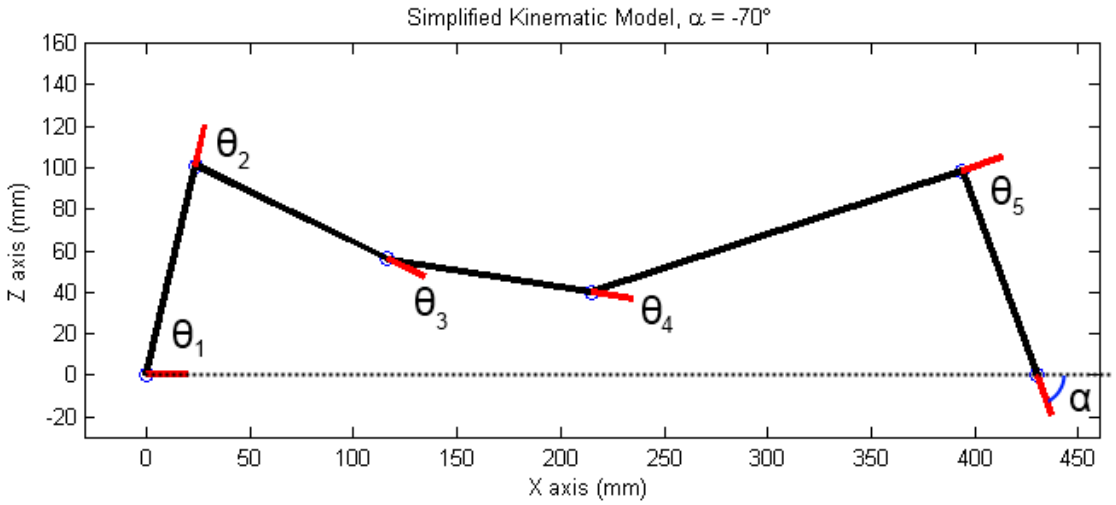

Figure 16: Simplified kinematic model of the sagittal projection of the closed chain configuration.

the following joints with respect to their previous links when projected onto the sagittal plane: toe-to-ground, ankle, knee, hip, shoulder. In Figure 16, the red lines represent the links projected onto the sagittal plane. These equations are the standard planar manipulation equations, treating the foot as the base link with the elbow as the end effector. Equation (4.8) constrains the end effector to be a set distance from the foot and equation (4.9) constrains the end effector to be on the ground. Using equation (4.10), equations (4.8) and (4.9) can be rewritten as:

$$\sum_{i=1}^{4} l_i \cos(\sum_{j=1}^{i} \theta_j) = d_e - l_5 \cos(\alpha) \tag{4.11}$$

$$\sum_{i=1}^{4} l_i \sin(\sum_{j=1}^{i} \theta_j) = -l_5 \sin(\alpha). \tag{4.12}$$

If θ_3, θ_4, and α are prescribed angles, then equations (4.11) and (4.12) are two equations in the two unknowns θ_1 and θ_2. θ_3 and θ_4 can be constant or time-varying as the resultant configuration is a function of these two "free" variables. Taking the square of each of the equations (4.11) and (4.12), an equation in the single variable θ_2

is obtained as:

$$2l_1K_1\cos(\theta_2) + 2l_1K_2\sin(\theta_2) = [d_e - l_5\cos(\alpha)]^2 + [l_5\sin(\alpha)]^2 - l_1^2 - K_1^2 - K_2^2 \quad (4.13)$$

$$K_1 = l_2 + l_3\cos(\theta_3) + l_4\cos(\theta_3 + \theta_4) \quad (4.14)$$

$$K_2 = -l_3\sin(\theta_3) - l_4\sin(\theta_3 + \theta_4) \quad (4.15)$$

The solution to equation (4.13) is of similar form to that seen in (4.4). In general, there will be two solutions for θ_2, but one of them will cause the robot to collide with the ground and must be guarded against. With θ_2 solved, equations (4.11) and (4.12) can be used to solve for θ_1:

$$\theta_1 = tan^{-1}\left(\frac{sin(\theta_1)}{cos(\theta_1)}\right) \quad (4.16)$$

$$cos(\theta_1) = \frac{K_3(d_e - l_5\cos(\alpha)) + l_5K_4\sin(\alpha)}{K_3^2 + K_4^2} \quad (4.17)$$

$$sin(\theta_1) = \frac{K_4(d_e - l_5\cos(\alpha)) - l_5K_3\sin(\alpha)}{K_3^2 + K_4^2} \quad (4.18)$$

$$K_3 = l_1 + K_1cos(\theta_2) + K_2sin(\theta_2) \quad (4.19)$$

$$K_4 = K_2cos(\theta_2) - K_1sin(\theta_2) \quad (4.20)$$

Lastly, θ_5 is solved using equation (4.10).

With these angles solved, the entire robot is parameterized on three angles $\theta_3, \theta_4, \alpha$. If θ_3 and θ_4 are fixed, then starting with the elbow at full extension, bringing the elbow to flexion moves the robot forward. This corresponds to α starting with a small negative angle and ending with a large negative angle. For the Nao, α is initialized at approximately $-30°$ and terminates at about $-90°$. The primary intuition about this procedure is that the closed chain is like a parallelogram that is used to shift the mass of the robot. Any robot (humanoid or not) that can be set into this configuration can use this framework in order to gait the robot.

4.2.3 Application onto Nao

Once the projected profile time sequence of angles has been computed, it needs to be applied to the Nao. Figure 17 illustrates the sagittal view of the robot in the closed chain configuration. When Nao is set to this configuration, the ankle pitch, knee pitch, hip pitch, and shoulder pitch joints of the robot directly correspond to θ_2 through θ_5. θ_1 corresponds to the angle subtended by the robot's foot and the ground. Unlike the first

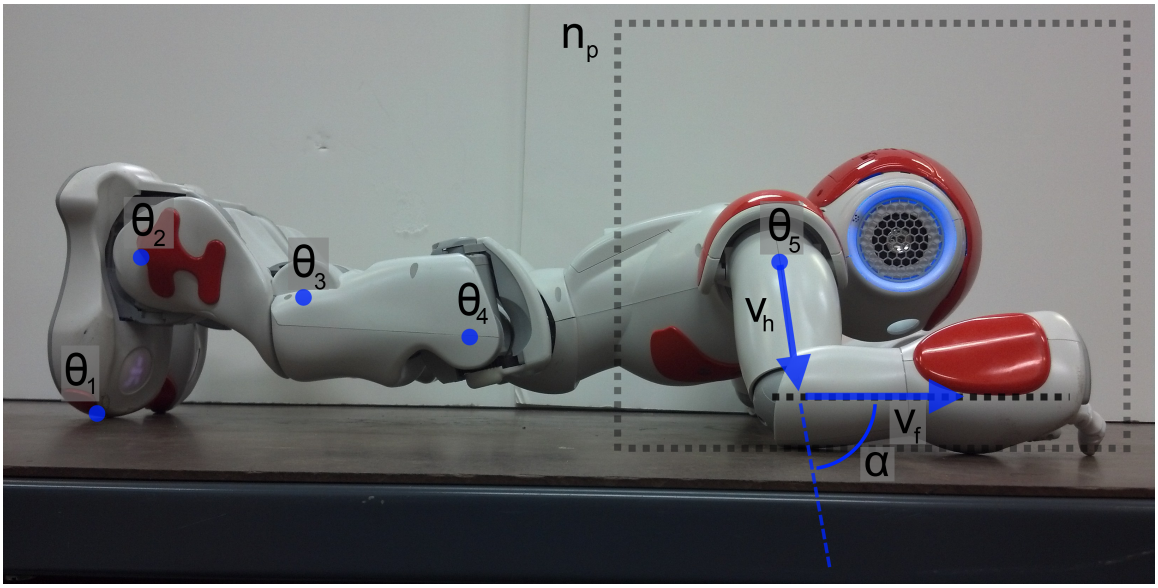

Figure 17: Sagittal view the Nao in the closed chain crawling configuration. The locations of the joints used in the projected profile calculation are shown as blue dots. n_p represents the sagittal plane of the robot.

five joint angles, angle α does not have a direct correspondence and must be derived from the projection of the arm onto the sagittal plane n_p .

[More to come on this part. Continue reading.] This is the part where we talk about the arm kinematics. Unlike the feet, etc, the arms are not inline with the rest of the body. We want to be able to position the arms a certain width apart to give a wider base and allow the elbows to rotate the projection of the forearm more before the actual forearm hits the humerus.

The shoulder roll is computable as a function of the desired width, shoulder width, and the length of the humerus.

The elbow roll is a function of the humerus vector and the desired forearm direction vector. The elbow yaw is a function of the humerus vector, the desired forearm direction vector, and the axis of rotation the the elbow roll revolves around.

With these three angles, the arm can be positioned.

To explain how to compute these, we have to define a whole bunch of vectors and scalars. First the scalars. Desired forearm-to-forearm spacing is denoted d_f. The shoulder-to-shoulder spacing is denoted d_s Then, as can be seen in one of the diagrams, we can define a quantity $d = \frac{d_f - d_s}{2}$ as the y-component of the elbow, where the origin is placed at the shoulder joint, the z-axis pointing up, x-axis facing forward. l_h is the

length of the humerus. $\tilde{l}_h = \sqrt{l_h^2 - d^2}$ is the length of the humerus, projected onto the z-x plane.

Now the vectors. The first vector is the desired forearm orientation $u_f = [1, 0, 0]^T$. We want the forearm to lie on the ground and pointing forward. On the ground to maximize surface contact so the robot is well supported and forward so the ground contact is parallel to the direction of travel. The next vector is the unit vector representing the direction that the humerus is pointing. This vector represents the rotation axis for the elbow yaw joint on the Nao. $v_{ey} = [\tilde{l}_h \cos(\alpha), d, \tilde{l}_h \sin(\alpha)]^T / l_h$. As can be seen by Figure **??** $[\tilde{l}_h \cos(\alpha), d, \tilde{l}_h \sin(\alpha)]^T$ represents the humerus with the shoulder at the origin, which is then normalized by the length of the humerus l_h. The next vector is the unit vector representing the rotation axis for the elbow roll joint on the Nao. When the elbow yaw is zero, the rotation axis for the elbow roll is $v_{er} = [-\sin(\alpha), 0, \cos(\alpha)]^T$. This vector v_{er} is perpendicular to v_{ey}.

The goal here is to put the forearm to be u_f. When elbow yaw and elbow roll are zero, the orientation of the forearm is coincident with the vector v_{ey}. We will denote the forearm vector as v_f which is initialized to $v_f = v_{ey}$. In order to move v_f to u_f we have to rotate v_f around v_{ey} by an amount θ_{ey} and then about v_{er} by an amount θ_{er}.

$$u_f = R_{v_{er},\theta_{er}} R_{v_{ey},\theta_{ey}} v_f \tag{4.21}$$

To find θ_{ey}, we must first compute $v_f \times u_f$ to find the desired vector u_{er} that v_{er} must be rotated to. Then, $\theta_{ey} = \arccos(v_{er} \cdot u_{er})$. Then, $\theta_{er} = \arccos(v_{ey} \cdot u_f)$.

These four angles complete the arm kinematics and compute the joints angles given the projected profile angle α and the desired forearm-to-forearm spacing.

4.3 Optimization

In the previous section, the Projected Profile crawl gait was parameterized on angle triplet $[\theta_3, \theta_4, \alpha]$. To achieve a crawling gait, θ_3 and θ_4 can be set to be constant and α linearly incremented from an initial angle to a final angle as a function of time. While this configuration successfully gaits the robot, it is heuristic. To improve the approach, the selection of a triplet of cubic splines $[\theta_3(t), \theta_4(t), \alpha(t)]$ is considered as an optimization problem. While many different quantities such as gait speed or the levelness of the back (for transportation of payloads) could be considered as optimization metrics, in this thesis the energy usage was minimized via joint torque measurements. The aim was to

increase the amount of time the crawling gait could be performed and reduce the stress on the robot's joints.

4.3.1 Pseudo-static Model

In order to optimize the gait with respect to the joint torques of the robot, a dynamic model of the robot is required. As the Projected Profile crawl gait is conceived as a statically stable gait and performed at slow speeds, the dynamics due to the movement of the robot are not considered to be forces that the motor controllers must counteract. During any part of the gait the robot will not slide if the gaiting direction is orthogonal to gravity and if the robot were to relax its joints it would collapse. Considering this, the resultant joint torques can be seen to be a function of gravity. This pseudo-static model of the robot can then be used to produce a cost metric for the optimization procedure. While conceptually simple, analyzing the projected profile closed chain manipulator to produce the system dynamics is challenging. In lieu of this, the robot was simulated for different values of the angle triplet in the closed chain configuration to generate a table of torques. This table of torques is then interpolated to produce a model of joint torques as a function of the parameterized angles:

$$[\tau_2, \tau_2, \tau_4, \tau_5] = jointTorques(\theta_3, \theta_4, \alpha) \tag{4.22}$$

where $\tau_2, \tau_3, \tau_4, \tau_5$ are the torques of one of the ankles, knees, hips, and shoulders, respectively. In this case, only one of each of the joints is considered because the gait is laterally symmetric. This means the torque of the left joint should be the same of the right joint. τ_1 is not a product of the function as θ_1 is not an actuated joint.

The V-REP simulator by Coppelia Robotics was used to gather joint torque data. It uses the Open Dynamics Engine (ODE) as its dynamics solver and is distributed with a model of the Nao Humanoid Platform. The model of the Nao kinematically corresponds with the Nao V4 and has the same mass values for the links. The Nao V4 is the robot used in this thesis. It has an easy to use API that can interface with C++, Python, or MATLAB.

Figure 18 shows a screen capture of the Nao model in the V-REP simulator. The Nao was configured to be in the closed chain and set to different joints angles. The initial and final values of α were constrained due to the gaiting requirement of the elbow to start at extension and end at flexion. The ranges of θ_3 and θ_4 were defined according to what seemed like plausible knee and hip angles for the gait. Using these limits a discrete

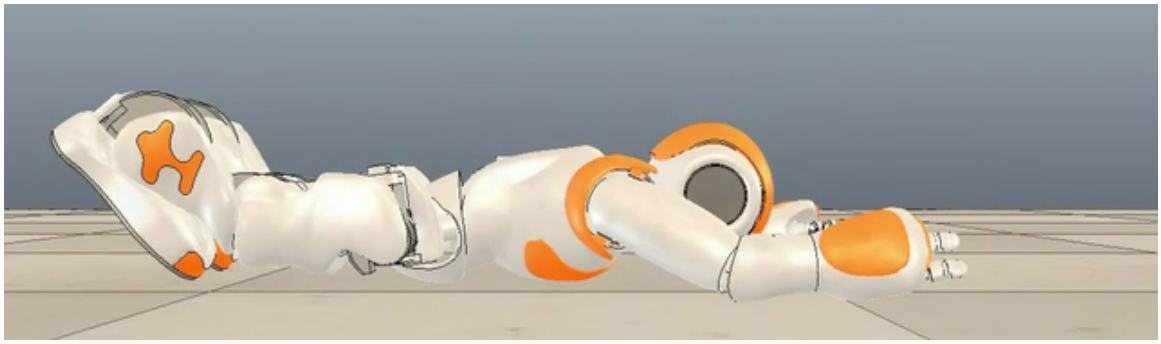

Figure 18: A sample screen capture of the V-REP simulation of the Nao robot in the closed chain configuration. The robot is set to different poses and then the joint torques are read after a short settle time.

Configuration Parameter	**Minimum Angle (degrees)**	**Maximum Angle (degrees)**
θ_3	-5	45
θ_4	15	-30
α	-30	-90

Table 1: Table of initial and final joint angles for each angle in the configuration triplet used to generate the set of angles used to configure the simulated Nao robot. The resultant set had 9,975 configurations with an angular resolution of 2.5°.

set of triplets were defined at a 2.5° resolution for each triplet parameter. These triplets were sent through the kinematics equations to produce the joint commands for the simulated Nao robot. Table 1 lists the parameters that describe the triplets tested. In total, 9,975 different joint configurations were simulated. To allow for the effect of any dynamics generated by the change in configuration to settle, the torque values of each of the joints was recorded after a period of one second.

4.3.2 Cost Functions

To find the optimal parameters for the angle triplet splines $[\theta_3(t), \theta_4(t), \alpha(t)]$, a cost c_s has to be attributable to a given spline triplet as a function of the spline parameters. The cost function contains terms regarding the joint torques given by the pseudo-static model, as the primary goal of the optimization procedure is to reduce the overall usage of

torque (and therefore energy) of the gait. Additionally, a number of indicator functions were introduced to the cost function in order to ensure kinematic constraints were not violated and to deter problems with the algorithm's implementation, which will be discussed.

The cost function based on joint torques was computed as:

$$c_\tau(t) = \sum_{i=1}^{4} w_i \tau_i^2(t) \tag{4.23}$$

In this notation, $\tau_1, \tau_2, \tau_3, \tau_4$ are the torques of the ankle, knee, hip, and shoulder, respectively. These torques are taken from the pseudo-static model and, as stated, are a function of $\alpha(t), \theta_3(t)$, and $\theta_4(t)$. The weight vector is constant and equal to $w = [1, 1, 1, 5]^T$. w_4 has a higher value in order to reduce the use of Nao's shoulder, as its actuator can produce about 3 to 4 times less torque than each of the leg actuators.

The first set of indicator functions introduced deter violation of kinematic constrains, such as joint limits in θ_3 and θ_4 (4.24), starting and stopping conditions for α (4.25), and to prevent the Nao from being in a kinematic configuration where the hips are making contact with the crawling surface (4.26).

$$c_{\theta_i}(t) = \begin{cases} 1 & \text{if } \theta_i(t) > \theta_{i_{max}} \text{ or } \theta_i(t) < \theta_{i_{min}} \\ 0 & \text{otherwise} \end{cases} \tag{4.24}$$

$$c_\alpha(t) = \begin{cases} 1 & \text{if } \alpha(t) > \alpha_{max} \text{ or } \alpha(t) < \alpha_{min} \\ 0 & \text{otherwise} \end{cases} \tag{4.25}$$

$$c_{z_{hip}}(t) = \begin{cases} 1 & \text{if } z_{hip}(t) < z_{hip_{thresh}} \\ 0 & \text{otherwise} \end{cases} \tag{4.26}$$

The function $z_{hip}(t)$ returns the height of the hips at time t using the the forward kinematics of the closed chain model of the gait. $z_{hip_{thresh}} \neq 0$, and is chosen to be the height of the hip joint when the robot is touching the crawling surface.

The next indicator function exists to deter a problem observed with the gaits being produced. Experimentally, it was seen that the optimizer would produce splines in which the parameter α would, for a large part of the gait, either stay nearly stationary or increase for some time, before decreasing by a large amount. To review, α is initialized at approximately $-30°$ and terminates at about $-90°$. This appeared as the robot lunging

forward at the end of the gait. This can most readily intepreted as the robot incurring a large cost for a small time, which is perhaps a local optima. An indicator function (4.27) was added to penalize splines in which $\alpha(t)$ was not monotonically decreasing, which allowed the optimizer to find a more efficient gait.

$$c_{\dot{\alpha}}(t) = \begin{cases} 1 & \text{if } \dot{\alpha}(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.27}$$

The indicator functions are then summed and multiplied by a large constant C_v so that the cost to violate any of these constraints is prohibitively large.

$$c_I(t) = C_v\Big(c_{\dot{\alpha}}(t) + c_{\alpha}(t) + c_{z_{hip}}(t) + \sum_{3}^{4} c_{\theta_i}(t)\Big) \tag{4.28}$$

The cost of the spline triplet is then given by integrating the sum of cost functions (4.23) and (4.28), and can be seen in equation (4.29).

$$c_s = \int_0^T c_\tau(t) + c_I(t)\, dt \tag{4.29}$$

where T is equal to one second, as it was seen that this was a reasonable time frame for this phase of the gait to be performed.

4.3.3 Optimization Procedure

The optimization procedure used to optimize the spline triplet $[\theta_3(t), \theta_4(t), \alpha(t)]$ was a basic genetic algorithm implemented using the genetic algorithm functions available in the MATLAB Global Optimization Toolbox.

Briefly, genetic algorithm optimization works by first creating a "population" of possible solutions, in this case the parameters for the spline triplet. Then the "fitness" of each solution is determined, here by evaluating the cost function for the solution. Finally, the solutions that are more fit are "selected" to for the "parent" set. These parents are then "bred" to form a new set of "child" solutions which will then be evaluated again. After some number of "generations", the most fit solution is selected as the optimal solution.

Genetic algorithms are well suited for problems like this as more classical optimization techniques would require the cost function to be analytical, which the joint torque table is not. As a result, because there is so much freedom in the cost function, different functions can be tried and evaluated to find a suitable result. Lastly, because the gait

design problem presented in this thesis, was not required to run in real time on the Nao, using such a computationally heavy optimization technique did not impact gait performance. One pitfall to this technique is there is no guarantee that the global optima will be found for a given cost function and therefore local optima are still possible to be returned.

Using the cost function presented in Section **??**, the genetic algorithm was used to find the 12 triplet parameters (4 parameters for each cubic spline). The algorithm ran for 67 generations and produced a triplet spline whose cost was approximately six times smaller than the nominal strategy of holding θ_3 and θ_4 constant while linearly increasing α.