In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup-2004*, pp. 297-309, Springer Verlag, 2005.

A Model-Based Approach to Robot Joint Control

Daniel Stronger and Peter Stone

Department of Computer Sciences, The University of Texas at Austin
stronger,pstone@cs.utexas.edu
http://www.cs.utexas.edu/~{stronger,pstone}

Abstract. Despite efforts to design precise motor controllers, robot joints do not always move exactly as desired. This paper introduces a general model-based method for improving the accuracy of joint control. First, a model that predicts the effects of joint requests is built based on empirical data. Then this model is approximately inverted to determine the control requests that will most closely lead to the desired movements. We implement and validate this approach on a popular, commercially available robot, the Sony Aibo ERS-210A.

Keywords: Sensor-Motor Control; Mobile Robots and Humanoids

1 Introduction

Joint modeling is a useful tool for effective joint control. An accurate model of a robotic system can be used to predict the outcome of a particular combination of requests to the joints. However, there are many ways in which a joint may not behave exactly as desired. For example, the movement of a joint could lag behind the commands being sent to it, or it could have physical limitations that are not mirrored in the control software.

In this paper, we consider the case in which a joint is controlled by repeatedly specifying an angle that the joint then tries to move to (e.g. as is the case for PID control[1]). These are the *requested angles*, and over time they comprise a *requested angle trajectory*. Immediately after each request, one can record the *actual angle* of the joint. These angles make up the *actual angle trajectory*.

This paper presents a solution to a common problem in robot joint control, namely that the actual angle trajectory often differs significantly from the requested trajectory. We develop a model of a joint that predicts how the actual angle trajectory behaves as a function of the requested trajectory. Then, we use the model to alter the requested trajectory so that the resulting actual trajectory more effectively matches the *desired trajectory*, that is, the trajectory that we would ultimately like the joint to follow.

At a high level, our proposed approach is to:

1. Determine the various features of the joint that need to be taken into account by the model.

[1] http://www.expertune.com/tutor.html

2. By experimenting with the joint and the various ways to combine these features, establish a mathematical model for the joint whose behavior mimics that of the joint when given the same input sequence.
3. Use the model to compute a series of requests that yields a close approximation to the desired trajectory. If the model is accurate, then these requests will cause the joint to behave as desired.

For expository purposes, we demonstrate the use of this technique on the Sony Aibo ERS-210A robot,[2] whose motors use PID control. However, this general methodology is potentially applicable to any situation in which robotic joints do not behave exactly as requested. We present empirical results comparing the direct approach of setting the requested angles equal to the desired angles against an approach in which the requested angles are set to a trajectory motivated by knowledge of the joint model.

The remainder of this paper is organized as follows. Section 2 relates our work to previous approaches. Section 3 introduces the Aibo robot and our model of its joint motion. Section 4 describes the process of inverting this model. Section 5 presents the empirical results validating our approach, and Section 6 concludes.

2 Related Work

One common approach to model-based joint control is to determine the complete physical dynamics of the system, which can in turn be used to construct a model-predictive control scheme. For example, Bilodeau and Papadopoulos empirically determine the dynamics of a hydraulic manipulator [1]. While this is a valuable technique, it is only applicable in situations where the physical parameters of the relevant joints can be ascertained. However, in many robotic systems, determining accurate values for these parameters can be very difficult or impossible. Another potential difficulty is that the low-level joint control policy is unattainable. This is the case for the Aibo; although we know that the joints are controlled by a PID mechanism, there is no information available about how the angle requests are converted to motor currents, nor about the motor specifications for the Aibo's joints. Our approach circumvents these problems by experimentally modeling the behavior of each joint as a function of the high-level angular requests that it receives. Furthermore, our approach extends beyond the construction and testing of a model to using the model to motivate more effective joint requests.

Although others have previously looked at the problem of correcting for joint errors, to our knowledge the approach proposed here has not been used. English and Maciejewski track the effects of joint failures in Euclidean space, but focus on joints locking up or breaking rather than correcting for routine inaccuracies [2]. An alternative approach to robot joint control is presented by Kapoor, Cetin, and Tesar [3]. They propose a system for choosing between multiple solutions to the inverse kinematics based on multiple criteria. Their approach differs from ours in that it is designed for robotic systems with more manipulator redundancy.

[2] http://www.aibo.com

Our approach is better suited to situations in which the inverse kinematics has a unique solution, but in which the joints do not behave exactly as requested.

3 Developing a Model

The Aibo robot has four legs that each have three joints known as the rotator, abductor, and knee. The robot architecture allows each joint to receive a request for an angle once every eight milliseconds. For all experiments reported in this paper, we request angles at that maximum frequency. The Aibo also reports the actual angles of the joints at this frequency.

Although we only have direct control over the angles of the three joints, it is often desirable to reason about the location of the robot's foot. The process of converting the foot's location in space into the corresponding joint angles for the leg is known as *inverse kinematics*. Since inverse kinematics converts a point in space to a combination of angles, it also converts a trajectory of points in space to an angle trajectory for each joint. We have previously solved the inverse kinematics problem for the legs of the ERS-210A robot [4].

As our primary test case for this project, we use a spatial trajectory for the foot that is derived from the walking routine of an entry in the 2003 RoboCup robot soccer competitions [5]. This trajectory is based on a half ellipse, and it is shown in Figure 1. The details of the trajectory and walking routine are given in [4]. The high-level motivation for this research is to enable more direct tuning of the legs' trajectories while walking.

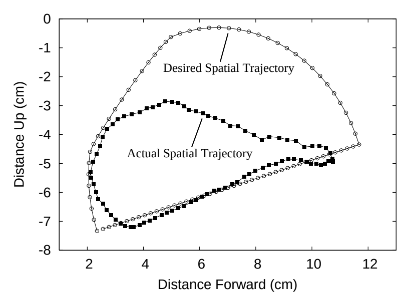

Fig. 1. Desired and actual spatial trajectory of the robot's foot. Only two of the three spatial dimensions are depicted here.

In an actual walking situation, the ground exerts significant and unpredictable forces on the Aibo's leg. In order to isolate the behavior of the joints unaffected by external forces, we perform all the experiments for this project with the robot held in the air, so that there is no interference with the leg's motion.

The most natural approach to trying to produce a desired trajectory for the foot is to convert the desired foot locations into joint angles by inverse kinematics, and then to set the requested joint angles equal to the resulting desired angles. To evaluate this method one can record the actual angle values of the leg's joints, convert them into an actual spatial trajectory with forward kinematics, and compare the desired spatial trajectory to the actual one. This comparison is shown in Figure 1.

The difference between these two trajectories is best understood in terms of the behaviors of the specific joints. We temporarily restrict our attention to only the rotator joint and determine the facets of the difference between its trajectory and the desired trajectory.

The goal of the joint model is to take as its input a sequence of requested angles and return a sequence of angles that mimics as closely as possible the angles that would actually be attained by the robot joint. We denote the sequence of requests by $R(t)$, where values of R are given in degrees, and the units for t is the amount of time between consecutive requests (eight milliseconds in our case). Furthermore, we restrict our attention to a period of time lasting the length of an Aibo step. We call this length of time t_{step}, which is 88 in our units (i.e. each step takes $88 \cdot 8 = 704$ milliseconds). During all of our experiments, we let the Aibo run through many steps continuously, so that an equilibrium is reached before measurements are taken.

To construct a model of how the joint responds to different requests, we observe the behavior of the joint as a result of various requested trajectories. Figure 2 shows how the joint responds to the sequence of requests that equal our desired angles for it. From this, we can start to infer the properties that our model needs to capture.

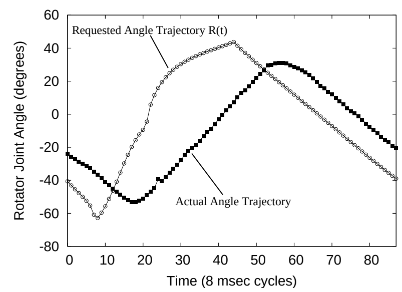

Fig. 2. This graph shows what happens when the angles requested of the rotator joint are set equal to our desired angle trajectory for that joint. These requests are compared to the actual angles that are reported by the robot at each time. Note that the graphs are cyclical, e.g. R(0) = R(88).

First, the actual angle lags behind the requests in time. Second, there appears to be a maximum slope by which the actual angle can change. This would amount to a physical limit on the angular velocity of the joint. Finally, the joint's velocity appears to be unable to change very quickly. In order to isolate these features so that they can be quantified and more precisely characterized, we perform a series of experiments on the joints.

For these experiments, we set $R(t)$ to a test trajectory given by

$$R(t) = \begin{cases} \theta_{test} \text{ if } t < \frac{t_{step}}{2} \\ 0 \quad\quad \text{if } t \geq \frac{t_{step}}{2} \end{cases} \tag{1}$$

Since the sequence of requests is continuously repeated, this has the effect of alternating between requesting angles of 0 and θ_{test}. A graph of $R(t)$ with θ_{test} equal to 40 and the resulting actual angle is shown in Figure 3.

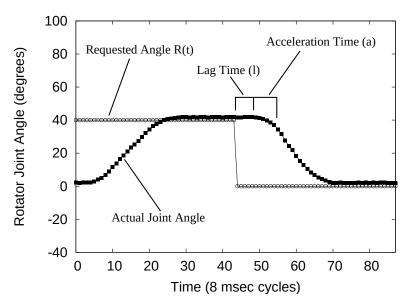

Fig. 3. Requested and actual angular trajectory of the rotator joint in test case with $\theta_{test} = 40$.

This figure identifies a number of important facets that a model of the joint must have. First, when the requested angle suddenly changes by 40°, there is a period of time before the joint responds at all. We denote this *lag time* by l. Then, after the lag time has elapsed, there is a period of time during which the joint accelerates toward its new set point. This *acceleration time* is denoted by a. After this, the joint's angular speed appears to plateau. We postulate that this maximum angular speed is the physical limit of the joint's speed, and we denote it by v_{max}.

At this point, two questions need to be answered. First, will a larger angle difference induce an angular speed greater than v_{max}, or is v_{max} really the joint's top speed? Second, is the acceleration time best modeled as a limit on the joint's angular acceleration, or as a constant acceleration time? These questions can both be answered by performing the same test but with θ_{test} equal to 110 degrees. The results of this test are shown in Figure 4.

In this situation, the joint has the same maximum angular speed as in Figure 3. This confirms that the joint cannot rotate faster than v_{max}, regardless of the difference between the requested and actual angles. Meanwhile, this test disproves the hypothesis that the acceleration time is due to a constant limit on the angular acceleration of the joint. This is because the joint takes the same amount of time to accelerate from angular velocity 0 to $-v_{max}$ in Figure 3 as it does to go from v_{max} to $-v_{max}$ in Figure 4. A constant acceleration limit hypothesis would predict that the second acceleration would take twice as long as the first one.

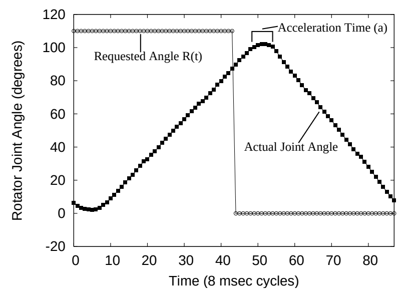

Fig. 4. Requested and actual angular trajectory of the rotator joint in test case with $\theta_{test} = 110$.

Although the joint's angular velocity is bounded by v_{max}, it is still the case that, within a certain range of differences between the requested and actual angle, the higher that difference is, the faster the joint will tend to rotate. This suggests the use of a function f defined as:

$$f(x) = \begin{cases} v_{max} & \text{if } x \geq \theta_0 \\ x \cdot \frac{v_{max}}{\theta_0} & \text{if } -\theta_0 < x < \theta_0 \\ -v_{max} & \text{if } x \leq -\theta_0 \end{cases} \tag{2}$$

where θ_0 is a constant that denotes the size of the difference between the requested and actual angle that is needed for the joint to move at its maximum angular speed. Figure 5 depicts the function f.

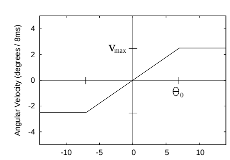

Fig. 5. The function f.

In order to capture the effect of an acceleration time, we set our model's velocity to an average of a values of f. The model's values for the joint angle, which we denote by $M_R(t)$, are defined by:

$$M_R(t) = M_R(t-1) + \frac{1}{a} \sum_{i=l+1}^{l+a} f(R(t-i) - M_R(t-1)) \tag{3}$$

where $R(t)$ is the sequence of requested angles. Since the most recent value of R that is included in this definition is $R(t-l-1)$, the model captures the notion that any request takes a lag of l time steps before it begins to affect the joint. Finally, since the model's velocity is the average of a values of f, and f's absolute value is bounded by v_{max}, the model captures the fact that the joint never moves at a speed greater than v_{max}.

Although the model and its parameters were determined using only the request trajectories in Equation (1), it is highly accurate under a wide range of circumstances. Measurements of the fidelity of the model in our experiments are given in Section 5.

Table 1 summarizes the model's parameters and their values, as determined experimentally, for the Aibo. We use t_u to denote our eight millisecond unit of time.

Parameter	**Description**	**Value**
v_{max}	Maximum Angular Speed	2.5 degrees/t_u
θ_0	Angle Difference Threshold	7.0 degrees
l	Lag Time	4 t_u
a	Acceleration Time	6 t_u

Table 1. Model Parameters.

It is worth noting that although the Aibo's joints are PID controlled, the features captured by the model are not predicted by that fact. PID control does not predict a maximum angular speed, and it does not explain the lag between the requests and their effects. This suggests that for the purposes of joint modeling, it is not particularly helpful to think of the Aibo joints as being PID controlled.

4 Inverting the Model

In order to compel the joint to move in our desired trajectory, which we call $D(t)$, we need to be able to convert it into a set of requests $I(t)$ such that when the values given by I are sent to the joint, the resulting behavior of the joint matches $D(t)$ as closely as possible. In terms of the model, given $D(t)$ we would like to find $I(t)$ such that $M_I(t) = D(t)$. This is the process of inverting the model.

4.1 Inverting the Model Explicitly

The first problem we encounter when trying to invert the model is that by the model, regardless of $I(t)$, the joint's angular speed is bounded by v_{max}. That is, $|M_I(t) - M_I(t-1)| \leq v_{max}$. If $D(t)$ violates this constraint, it is not in the range of the model, and there are no requests $I(t)$ such that $M_I(t) = D(t)$. In fact, many of the angular trajectories we get from inverse kinematics violate this constraint. It is theoretically impossible for the model to return these trajectories exactly.

Even when we know it is impossible to invert the model on a particular trajectory, one possibility is to try to construct an approximation to $D(t)$ that is in the range of the model and invert the model on that instead. However, doing so in general is complicated by the fact that f is not invertible, and in the range that it is invertible, there are points with infinitely many inverses.

We circumvent this problem by restricting our approximation of $D(t)$ to be a piecewise linear trajectory, which we call $P(t)$, with the property that the slope of each line segment is less than or equal to v_{max} in absolute value. Note that since $P(t)$ is a trajectory of joint angles, these segments represent an angle that varies linearly with respect to time (i.e. with a constant angular velocity). An example piecewise linear approximation is shown in Figure 6. One may be able to automate this approximation, even subject to the restriction on minimum and maximum angular velocity. However, for the purposes of this paper the approximations are constructed manually.

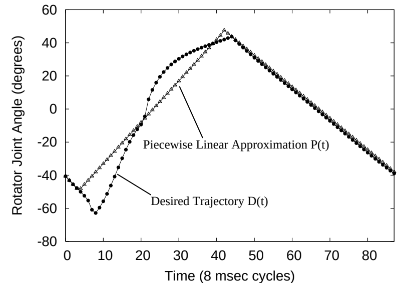

Fig. 6. Piecewise linear approximation. Note that the approximation diverges from the desired trajectory on the left side due to the slope restriction on the linear segments.

Although $P(t)$ is not invertible in the model (due to instantaneous velocity changes), we can invert the mathematical model on the component line segments. Recombining these inverse line segments appropriately yields a series of requests, $R(t)$, such that the result of applying the model to it, $M_R(t)$, is a close approximation to $P(t)$, which is in turn a close approximation to $D(t)$.

4.2 Inverting Lines

Our proposed method relies on being able to determine the inverses of lines according to the model. That is, given a particular linear trajectory, what angles should be requested so that the given trajectory is actually achieved by the joint? The answer is a linear trajectory that has the same slope as the desired line but differs from it by a constant that depends on its slope. This is only possible when our line's slope corresponds to an angular velocity that is actually attainable, so we will restrict our attention to the case where the absolute value of the slope of the line is less than or equal to v_{max}.

Consider a linear series of requests, $L(t)$. We say that $L(t)$ has slope m, where m is $L(t) - L(t-1)$ for all t. We temporarily restrict our attention to the case where $|m| < v_{max}$ and furthermore assume without loss of generality that $m \geq 0$. We will determine the image of $L(t)$ in the model, $M_L(t)$. Since this will turn out to be a line of the same slope, it will enable us to compute the inverse of any line whose slope is in range.

We would like to be able to reason about the angular distance between points on the requested line and those on the resulting line according to the model, and specifically how that distance changes over time. Thus we denote the angular distance $L(t) - M_L(t)$ by $\delta(t)$. First, however, we must understand how $M_L(t)$ changes as a function of $\delta(t)$. This can be seen by plugging L into Equation (3) (for R):

$$M_L(t) = M_L(t-1) + \frac{1}{a} \sum_{i=l+1}^{l+a} f(L(t-i) - M_L(t-1)) \tag{4}$$

Taking advantage of the fact that L is linear, we replace $L(t-i)$ with $L(t-1)-m(i-1)$. Thus $L(t-i)-M_L(t-1)$ is equal to $L(t-1)-M_L(t-1)-m(i-1)$, or $\delta(t-1)-m(i-1)$. This enables us to rewrite Equation (4) as:

$$M_L(t) = M_L(t-1) + \frac{1}{a}\sum_{i=l+1}^{l+a} f(\delta(t-1) - m(i-1)) \tag{5}$$

This equation tells us how the the model's value for the angle varies as a function of the angular distance $\delta(t)$. In order to capture this relationship more concisely, we define the function S to be:

$$S(x) = \frac{1}{a}\sum_{i=l+1}^{l+a} f(x - m(i-1)) \tag{6}$$

Thus we can characterize the relationship between M_L and δ as $M_L(t) = M_L(t-1) + S(\delta(t-1))$. Now that we understand how δ influences M_L, it is useful to analyze how $\delta(t)$ changes over time. We can isolate this effect by using the definition of $\delta(t)$ to replace $M_L(t)$ with $L(t)-\delta(t)$. This gives us:

$$L(t) - \delta(t) = L(t-1) - \delta(t-1) + S(\delta(t-1)) \tag{7}$$

Then, since $L(t) = L(t-1) + m$, we can rearrange as:

$$\delta(t) = \delta(t-1) + m - S(\delta(t-1)) \tag{8}$$

This equation indicates that as time progresses, $\delta(t)$ approaches an equilibrium where $S(\delta(t)) = m$. That is, $\delta(t)$ approaches a constant, which we denote by C_m, such that $S(C_m) = m$. Since $M_L(t) = L(t)-\delta(t)$, this means that $M_L(t)$ approaches $L(t) - C_m$.

Given a desired linear angular trajectory, $D_L(t)$, with slope m, C_m is the amount that must be added to D_L to get a sequence of requests such that the actual joint angles will approach our desired trajectory. Thus inverting each of our component line segments involves computing C_m for m equal to the slope of that segment. Unfortunately, calculating C_m in terms of m explicitly is not straightforward. Since S is defined as the average of a instances of the function f, we would need to determine which case in the definition of f is appropriate in each of those a instances. Nonetheless, we are able to approximate C_m quite accurately by approximating the sum in Equation (6) with an integral. The computation of C_m and an overview of its derivation are in the appendix.

In the case where $m = v_{max}$, C_m is not well defined, since $S(x) = v_{max}$ for any sufficiently large value of x. In fact, if $f(x - v_{max}(l+a)) \geq v_{max}$, then all of the values of f being averaged in the definition of S take on the value of v_{max}, and thus $S(x)$ equals v_{max}. This occurs when $x \geq \theta_0 + v_{max}(l+a)$. In this situation, $L(t)$ and $M_L(t)$ will both keep increasing at a rate of v_{max}, so the distance between them will not change. Thus we use the threshold value of $\theta_0 + (l+a)v_{max}$ for C_m and rely on switching between the inverses of the line segments at the right time to ensure that the actual angle trajectory stays close to the desired line.

4.3 Combining Inverted Line Segments

Our piecewise linear approximation, $P(t)$, is comprised of line segments $D_L(t)$ with slopes m. For any one of these line segments, we know that by requesting values of $D_L(t) + C_m$, the joint angle will closely approximate $D_L(t)$. For the joint to follow $P(t)$ to a close approximation the whole way through, we must transition between these lines at the appropriate times.

After the requests switch from one inverted linear trajectory to another, how long will it take for the joint to switch between the corresponding desired trajectories? According to the model, there is a lag time, l, before any effect of the change will be observed. After that, the joint will accelerate from one linear trajectory to the other over the course of an acceleration time of length a. Ideally then, the desired piecewise linear trajectory, $P(t)$, would transition between components in the middle of this acceleration period. In order to achieve this, we transition between inverted line segments $l + \frac{a}{2}$ time units before the transition between desired line segments. This completes the specification of our approach.

The whole process takes a desired angle trajectory, constructs a piecewise linear approximation to it, and formulates requests to the joint based on that approximation. These three trajectories and the resulting actual trajectory are shown in Figure 7. Notably, the actual trajectory runs very close to the piecewise linear approximation.

5 Experimental Results

The goal of this process is for the robot to move each joint so that it follows a desired trajectory as closely as possible. We can evaluate the success of the method by calculating the angular distance between the desired trajectory and the actual one (as depicted in Figure 7).

Although we have described our approach thus far using the Aibo's rotator joint as an example, we have implemented it on all three of the joints of an Aibo leg. Interestingly, we found the parameters of our model to be exactly the same for all three of the leg's joints. We analyze the method's success on all three joints.

We treat an angle trajectory as a t_{step}-dimensional vector, where t_{step} is the number of requests that comprise one Aibo step (88 in our case), and calculate the L_2 and L_∞ norms between the vectors corresponding to two trajectories. We consider distances between four different trajectories for each joint: Des, the desired trajectory, Dir, the actual angles under the direct method of setting the requests equal to the desired angles,

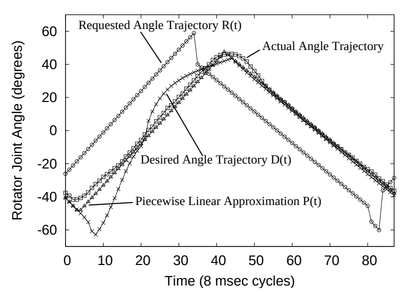

Fig. 7. This graph depicts the original desired angle trajectory, the piecewise linear approximation, the requests that are derived from that approximation, and the actual angle trajectory that results from this process.

Pwl, the piecewise linear approximation, and MB (Model-Based), the actual angle trajectory achieved by the our approach. Since these distances can vary from one Aibo step to the next, the numbers given in Table 2 are averages and standard deviations taken over 20 steps.

Comparison	Rotator	Abductor	Knee
$\boldsymbol{L_2(Des, Dir)}$	**31.0(±0.2)**	**29.0(±0.2)**	**20.1(±0.1)**
$L_\infty(Des, Dir)$	57.2(±0.3)	59.5(±0.5)	42.6(±0.3)
$\boldsymbol{L_2(Des, MB)}$	**9.1(±0.2)**	**10.4(±0.1)**	**5.6(±0.2)**
$L_\infty(Des, MB)$	29.4(±0.8)	24.5(±0.7)	11.1(±0.5)
$L_2(Pwl, MB)$	2.7(±0.4)	2.7(±0.3)	2.6(±0.2)
$L_\infty(Pwl, MB)$	6.4(±0.6)	6.0(±0.4)	6.2(±0.7)

Table 2. Distances between angle trajectories.

The actual angles achieved by our method come much closer to our desired angle trajectories than the ones obtained by setting the requested angles equal to the desired angles, as shown in the two bold rows. The very small distances between Pwl and MB indicate the strength of the fidelity of the model.

We also compare the attained spatial trajectory of the Aibo's foot to the desired spatial trajectory (see Figure 8). Here we measure the improvement in the Euclidean distance between the desired and attained foot trajectories. We calculate the distance between the desired and actual foot location at each time and apply the L_2 and L_∞ norms to these distances. The direct method yields an L_2 distance of 3.23 ± 0.01 cm and L_∞ of 4.61 ± 0.05 cm. Our model-based method gives us an L_2 of 1.21 ± 0.04 cm and an L_∞ of 2.34 ± 0.12 cm.

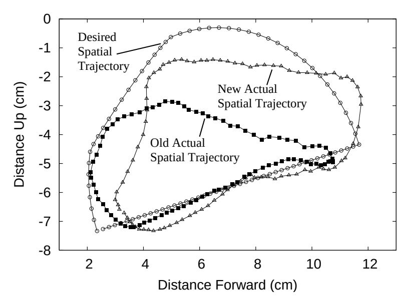

Fig. 8. This graph depicts the desired spatial trajectory compared to the effect of simply requesting the desired angles (old) and to the effect of the approach described in this paper (new). The spatial trajectories are computed by forward kinematics from the recorded actual joint angles.

Finally, we compare our method to the following process. For any value of k, consider setting requested angles $R(t)$ equal to $D(t+k)$. That is, let the requests be exactly the same as the desired angles, but offset by a fixed amount of time. For each of the t_{step} possible values of k, we can compute the distance between the resulting actual angle trajectory and the desired one. The minimum of these distances with respect to k provides a measure of how much of our improvement can be attributed to modeling the lag in the joints. The distances returned by this approach were an L_2 of 1.55 ± 0.04 cm and an L_∞ of 3.57 ± 0.08 cm. These distances are smaller than those achieved by the direct method, but still significantly greater than the distances attained by our model-based method. This result indicates that the success of our approach is due to more than its ability to model lag in the joint.

6 Conclusion and Future Work

This paper demonstrates the development of a detailed joint model of a popular, commercially available robotic research platform. We show all the steps of the derivation of this model using a generally applicable methodology. We then approximately invert the model to determine control requests that cause the robot's joints to move in a desired trajectory. Using this approach, we successfully bring the robot's actual motions significantly closer to the desired motions than they were previously.

The high-level motivation for this research is to enable direct tuning of the legs' trajectories while walking. In addition to applying the proposed approach towards that task, there are three main ways in which the work can be extended in future research. First, since the experiments reported in this paper were performed with the robot held in the air, we have a model of how the joints behave when the external torques being exerted on them are relatively small. An extension of this work would be to model how the joints respond to significant external torques, e.g. the torques that are exerted when a robot walks on the ground. Second, our approach could also be extended by implementing it on other platforms. Doing so will help elucidate the class of robotic problems on which these techniques are effective. Third, a possibility for future work is for the robot to learn the model of its joints automatically from experience based on its knowledge of the joint requests and actual angles over time. This would have the benefit that the robot could adjust its model over time to compensate for changing properties of the joints. While it would be challenging to learn a model of arbitrary functional form, we surmise that tuning the parameters of a model, such as l and a in our case, would be relatively straightforward. In this regard, one contribution of the research reported here is an identification of a class of functions that could be used as the space of models to explore during the learning process.

Acknowledgments

We would like to thank the members of the UT Austin Villa team for their efforts in developing the software used as a basis for the work reported in this paper. Thanks also to Chetan Kapoor and Ben Kuipers for helpful discussions. This research was supported in part by NSF CAREER award IIS-0237699.

Appendix

This appendix describes the computation of C_m from m and gives an overview of the derivation. As discussed in Section 4.2, C_m is the amount that must be added to a line of slope m to get its inverse, and we are considering the case where $m\epsilon[0, v_{max})$. The definition of C_m is that $m = S(C_m)$, where S is defined in Equation (6). The first step is to replace the sum in that definition with the corresponding integral, so that $m = S(C_m)$ becomes:

$$m = \frac{1}{a}\int_{l}^{l+a} f(C_m - mi)\, di \tag{9}$$

Next, since the definition of f is split into cases based on whether its argument is greater than $-\theta_0$, and θ_0, important thresholds in the analysis of Equation (9) are values of i for which $C_m - mi = -\theta_0$, and θ_0. We then divide our analysis into cases based on where these two thresholds fall with respect to our limits of integration (e.g. less than both, between them, or greater than both).

This results in three cases, each of which can be analyzed independently. Due to the particular parameters in our model, this analysis reduces to two cases. Finally, C_m is computed as follows. First, determine whether or not the following inequality holds:

$$m\left(\frac{\theta_0}{v_{max}} + \frac{a}{2}\right) < \theta_0 \tag{10}$$

If it does, C_m is given by the equation:

$$C_m = m\left(\frac{\theta_0}{v_{max}} + l + \frac{a}{2}\right) \tag{11}$$

If not, it is instead given by:

$$C_m = \theta_0 + m\left(l + a - \sqrt{\frac{2\theta_0}{m} \cdot \frac{a(v_{max} - m)}{v_{max}}}\right) \tag{12}$$

This computation of q relies on two numerical facts regarding the parameters of our model. These are:

$$\theta_0 > v_{max} \qquad \text{and} \qquad 4\theta_0 > v_{max}a \tag{13}$$

If these are not true, there may be more cases involved in the computation of C_m.

References

1. G. Bilodeau and E. Papadopoulos, "Modelling, identification and experimental validation of a hydraulic manipulator joint for control." in *Proceedings of the 1997 International Confrerence on Intelligent Robots and Systems (IROS '97)*, Victoria, BC, October 1998.
2. J. English and A. Maciejewski, "Measuring and reducing the euclidean-space measures of robotic joint failures," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 1, pp. 20–28, Feb. 2000.
3. C. Kapoor, M. Cetin, and D. Tesar, "Performance based redundancy resolution with multiple criteria," in *Proceedings of 1998 ASME Design Engineering Technical Conference (DETC98)*, Atlanta, Georgia, September 1998.
4. P. Stone, K. Dresner, S. T. Erdoğan, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, D. Stronger, and G. Hariharan, "UT Austin Villa 2003: A new RoboCup four-legged team," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-03-304, 2003.
5. P. Stone, T. Balch, and G. Kraetzschmar, Eds., *RoboCup-2000: Robot Soccer World Cup IV*, ser. Lecture Notes in Artificial Intelligence. Berlin: Springer Verlag, 2001, vol. 2019.