Controlling a Marionette with Human Motion Capture Data

Katsu Yamane, Jessica K. Hodgins, and H. Benjamin Brown
The Robotics Institute, Carnegie Mellon University
E-mail: {kyamane|jkh|hbb}@cs.cmu.edu

Abstract

In this paper, we present a method for controlling a motorized, string-driven marionette using motion capture data from human actors. The motion data must be adapted for the marionette because its kinematic and dynamic properties differ from those of the human actor in degrees of freedom, limb length, workspace, mass distribution, sensors, and actuators. This adaptation is accomplished via an inverse kinematics algorithm that takes into account marker positions, joint motion ranges, string constraints, and potential energy. We also apply a feedforward controller to prevent extraneous swings of the hands. Experimental results show that our approach enables the marionette to perform motions that are qualitatively similar to the original human motion capture data.

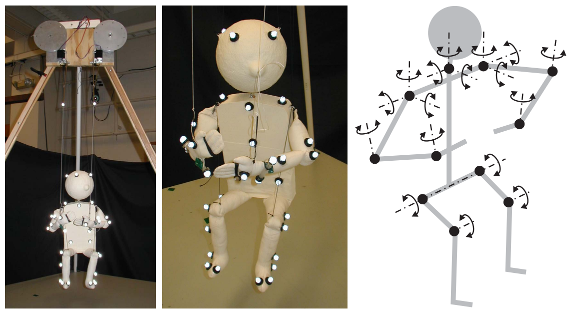
Figure 1: The motor-driven marionette and its model. The marionette is about 60cm tall. The shoulder and elbow joints have cloth stops to prevent unrealistic joint angles.

1 Introduction

Entertainment is one of the more immediately practical applications of humanoid robots and several robots have recently been developed for this purpose [1, 2, 3]. In this paper, we explore the use of an inexpensive entertainment robot controlled by motion capture data with the goal of making such robots readily available and easily programmable. The robot is a marionette where the length and pivot points of the strings are controlled by eight servo motors that bring the hands and the feet of the marionette to the desired positions (Figure 1).

Standard marionettes are puppets with strings operated by a human performer's hands and fingers. Creating appealing motion with such a puppet is difficult and requires extensive practice. Although for our marionette the servo motors move the strings, programming a robotic version of such a device by hand to produce expressive gestures would also be difficult. We solve this problem by using full-body human motion data to drive the motion. The human motion data is recorded as the positions of markers in three dimensions while an actor tells a story with his or her hands. After adaptation, the data are used to drive the motion of the marionette by taking into account the marionette's swing dynamics. These adaptations and dynamics compensation are necessary because the marionette has many fewer degrees of freedom and much smaller size than the human actor, strings rather than actuators at the joints, and no sensors except for the rotation of the motors.

The method described in this paper consists of four steps: (1) identify the swing dynamics of the hands and design a feedforward controller to prevent swinging and obtain a desired response, (2) obtain the translation, orientation, and scaling parameters that map the measured marker positions for the human motion into the marionette's workspace, (3) apply the controller to modify the mapped marker positions to prevent swing, and (4) compute the motor commands to bring the (virtual) markers attached to the marionette to the revised positions computed in step (3).

The relationship between the four steps is illustrated in Figure 2. In steps (2) and (4), we solve the inverse kinematics problem with many different constraints including marker positions, joint motion ranges, strings, and gravity. This algorithm is an extension of the first author's previous work [4]. In step (1), we model the dynamics of swing by capturing the response to a step input of each desired marker position and then use that response to design a feedforward controller to compensate for the swing motion.

As a demonstration of the algorithm, we include exper-

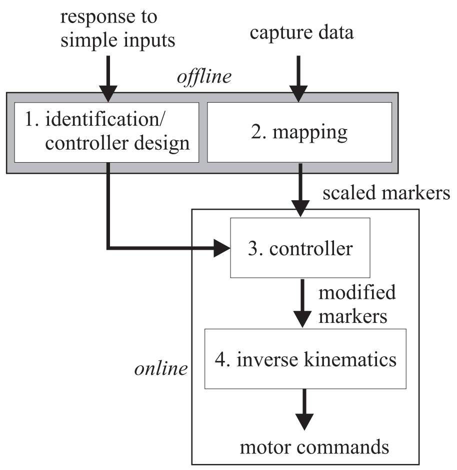

Figure 2: Overview of the marionette control system. The blocks in the top gray box are processed offline: identification/controller design for each marionette and mapping for each motion sequence. Those in the bottom white box is processed in realtime during a performance.

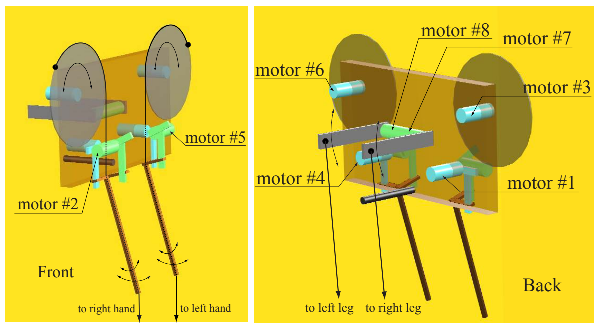

Figure 3: Closeup of the motors and pulleys; front of marionette (left), back (right).

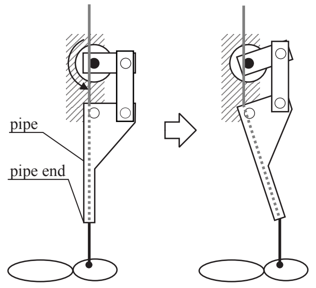

Figure 4: The mechanism for moving the hand in the horizontal direction.

imental results to compare the marionette motion to that of human actors. The motions of the marionette and the human actor are similar enough to distinguish different styles for the same story.

2 Related Work

Hoffmann [5] developed a human-scale marionette and controlled it to perform dancing motions using human data. The size and controllable degrees of freedom of the marionette are much closer to those of human than ours. The research is therefore focused more on image processing for measuring human motion than on mapping between human and marionette motions.

Mapping motion data is a common problem in applying motion capture data to a real robot or to a virtual character. The factors considered in previous work include joint angle and velocity limits [6], kinematic constraints [7], and physical consistency [8, 9]. However, the original and target human figures are usually more similar in degrees of freedom, dimensions, and actuators than the marionette is to a human actor.

The mechanism and dynamics of a string-driven marionette are quite similar to those of wired structures such as a crane. A number of researchers have worked on controlling a crane to bring an object to a desired position without significant oscillations [10]. This work assumes that the position of the object is known through the direction of the wire. Although we measure the position of the hand and feet for recording the swing dynamics, we do not have this information during a performance. Our accuracy requirements are much less because the marionette is gesturing in free space rather than precisely positioning an object.

3 Experimental Setup

The marionette is modeled as a 17DOF kinematic chain (Figure 1 right). Closeups of the motors and pulleys are shown in Figure 3. The marionette has eight servo motors (Airtronics Servo 94102); six control the arms and two control the legs. The motors are controlled by position commands sent from a serial port of a PC via an 8-channel controller board (Pontech SV203).

Motors 3 and 6 change the length of the strings connecting the hands. Motors 7 and 8 move the knee up and down. Motors 1, 2, 4 and 5 move the hands in horizontal directions by rotating the “pipes” and moving the pipe ends via four independent planar linkages (Figure 4).

We used a commercially available motion capture system from Vicon for capturing the actor’s performance and the marionette’s motion for identification of the swing dynamics. The system has nine cameras, each capable of recording images of 1000×1000 pixels resolution at 120Hz. We used different camera placements for the human subject and the marionette to accommodate the smaller workspace of the marionette and to ensure accurate measurements.

4 Inverse Kinematics

The inverse kinematics algorithm is used to enforce constraints to bring the markers representing the desired motion into the workspace of the marionette and to determine the motor angles that satisfy the desired marker positions and the physical constraints, including the desired marker positions, joint motion ranges, length and orientation of the strings, and potential energy. The potential energy constraint is introduced to model the effect of gravity. The inverse kinematics algorithm computes the joint angles and the motor commands that locally optimize the constraints. Because the algorithm was described in a previous paper [4], we present a short outline here.

Often, all the constraints cannot be satisfied due to the singularity of the configuration or to inconsistencies between the constraints. Therefore, the user is asked to divide the constraints into two groups: those that must be satisfied and those where some error is acceptable. The algorithm applies singularity-robust (SR) inverse [11] (also known as damped pseudo inverse [12]) to the lower-priority constraints. As described below, the SR-inverse distributes the error among the lower-priority constraints according to the given weights so that the resulting joint velocity does not become too large even if there are singularities or inconsistencies in the constraints.

We design a feedback controller for each constraint to ensure that the lower-priority constraints are satisfied as much as possible and to eliminate integration errors in both higher- and lower-priority constraints. The controller computes the required velocity when constraints are violated. For example, the feedback controller to bring a link to its reference position $\boldsymbol{p}^{ref}$ is $\dot{\boldsymbol{p}}^{des} = k_p(\boldsymbol{p}^{ref} - \boldsymbol{p})$ where k_p is a positive gain, $\boldsymbol{p}$ is the current position, and $\dot{\boldsymbol{p}}^{des}$ is the desired velocity. Note that this velocity is not always realized for lower-priority constraints due to the nature of the SR-inverse algorithm.

With n_1 higher-priority constraints and n_2 lower-priority constraints, we have the following equations in generalized velocity $\dot{\boldsymbol{\theta}}$:

$$\begin{aligned} \boldsymbol{J}_1\dot{\boldsymbol{\theta}} &= \boldsymbol{v}_1^{des} \quad (1)\\ \boldsymbol{J}_2\dot{\boldsymbol{\theta}} &= \boldsymbol{v}_2^{des} \quad (2)\end{aligned}$$

where $\boldsymbol{v}_1^{des}$ and $\boldsymbol{v}_2^{des}$ are the desired velocities corresponding to higher- and lower-priority constraints respectively, and $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ are the Jacobian matrices of the constraints with respect to $\boldsymbol{\theta}$.

We solve this equation for the generalized velocity as follows. First, we compute the set of exact solutions of Eq.(1) by

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_1^{\sharp}\boldsymbol{v}_1^{des} + (\boldsymbol{I} - \boldsymbol{J}_1^{\sharp}\boldsymbol{J}_1)\boldsymbol{y} \quad (3)$$

where $\boldsymbol{J}_1^{\sharp}$ is the pseudo inverse of $\boldsymbol{J}_1$, $\boldsymbol{I}$ is the identity matrix of the appropriate size, and $\boldsymbol{y}$ is an arbitrary vector. We can rewrite this equation as $\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_1 + \boldsymbol{W}\boldsymbol{y}$ where $\dot{\boldsymbol{\theta}}_1 \overset{\triangle}{=} \boldsymbol{J}_1^{\sharp}\boldsymbol{v}_1^{des}$, and $\boldsymbol{W} \overset{\triangle}{=} \boldsymbol{I} - \boldsymbol{J}_1^{\sharp}\boldsymbol{J}_1$. Next, we compute the $\boldsymbol{y}$ with which $\dot{\boldsymbol{\theta}}$ would satisfy Eq.(2) as closely as possible by

$$\boldsymbol{y} = (\boldsymbol{J}_2\boldsymbol{W})^*(\boldsymbol{v}_2^{des} - \boldsymbol{J}_2\boldsymbol{\theta}_1) \quad (4)$$

where $(\boldsymbol{J}_2\boldsymbol{W})^*$ is the SR-inverse of $\boldsymbol{J}_2\boldsymbol{W}$. Finally, the generalized velocity $\dot{\boldsymbol{\theta}}$ is computed by substituting $\boldsymbol{y}$ into Eq.(3), which is then integrated to compute the generalized coordinates in the next step.

In order to add a new constraint, we must design a feedback controller to compute the desired velocity and derive the corresponding Jacobian matrix. We describe the string and potential energy constraints in detail because the other constraints were described in the earlier paper [4].

4.1 String Constraints

Each string has a start point, an end point, and some number of intermediate points (Figure 5 left). The string can slide back and forth at the intermediate points. The current length of a string, l, must always be equal to or smaller than its nominal length l_0. l is computed by summing the length of all segments:

$$\begin{aligned} l &= \sum_{i=0}^{N-1} l_i = \sum_{i=0}^{N-1} |\boldsymbol{p}_{i+1} - \boldsymbol{p}_i| \\ &= \sum_{i=0}^{N-1} \sqrt{(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)^T(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)} \quad (5)\end{aligned}$$

where N is the number of segments, l_i $(0 \le i \le N-1)$ is the length of segment i, $\boldsymbol{p}_i$ $(0 \le i \le N)$ is the position of the i-th point. The Jacobian matrix of l with respect to the generalized coordinates $\boldsymbol{\theta}$ is computed by

$$\begin{aligned} \boldsymbol{J}_{str} &= \frac{\partial l}{\partial \boldsymbol{\theta}} = \sum \frac{\partial l_i}{\partial \boldsymbol{\theta}} \\ &= \sum \frac{1}{l_i}(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)^T \left(\frac{\partial \boldsymbol{p}_{i+1}}{\partial \boldsymbol{\theta}} - \frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{\theta}}\right) \\ &= \sum \frac{1}{l_i}(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)^T(\boldsymbol{J}_{i+1} - \boldsymbol{J}_i). \quad (6)\end{aligned}$$

Note that the Jacobian matrix is not defined for segments with $l_i = 0$, although we never encounter such situations in

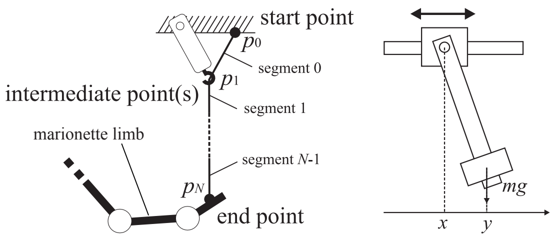

Figure 5: String models for inverse kinematics (left) and swing controller (right).

physical mechanisms. The feedback law for a string constraint is $v_{str}^{des} = k_{str}(l_0 - l)$.

In addition to the length, we also constrain the $(N-1)$th segment of the string to be vertical due to gravity. The two points $\boldsymbol{p}_{N-1}$ and $\boldsymbol{p}_N$ should then be vertical:

$$\boldsymbol{h}_1 \cdot \boldsymbol{d}_{N-1,N} = 0,\ \boldsymbol{h}_2 \cdot \boldsymbol{d}_{N-1,N} = 0 \tag{7}$$

where $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are independent unit vectors in the horizontal plane (e.g. $\boldsymbol{h}_1 = (1\ 0\ 0)^T, \boldsymbol{h}_2 = (0\ 1\ 0\ )^T$ if the gravity is in z direction) and $\boldsymbol{d}_{N-1,N}$ is the unit vector from $\boldsymbol{p}_{N-1}$ to $\boldsymbol{p}_N$, namely $\boldsymbol{d}_{N-1,N} = (\boldsymbol{p}_N - \boldsymbol{p}_{N-1})/l_{N-1}$. The Jacobian matrix for this constraint is

$$\boldsymbol{J}_v = \frac{1}{l_{N-1}} \begin{pmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \end{pmatrix} (\boldsymbol{J}_N - \boldsymbol{J}_{N-1}). \tag{8}$$

The desired velocity for this constraint is

$$\boldsymbol{v}_v^{des} = -k_v \begin{pmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \end{pmatrix} \boldsymbol{d}_{N-1,N} \tag{9}$$

where k_v is a positive gain.

4.2 Potential Energy

Because the joints that do not have strings directly attached to them will bend downward due to gravity, we also constrain the potential energy to be as small as possible by constraining the center of mass of the whole body to be as low as possible. The Jacobian matrix for this constraint is computed as $\boldsymbol{J}_{pe} = \boldsymbol{d}_g^T \boldsymbol{J}_{COM}$ where $\boldsymbol{d}_g = (0\ 0\ 1)^T$ is the unit vector in the direction of the gravity and $\boldsymbol{J}_{COM}$ is the Jacobian matrix of the center of mass with respect to the generalized coordinates. A method for computing $\boldsymbol{J}_{COM}$ can be found in [13]. The desired velocity for this constraint is a negative constant $-k_{pe}$.

5 Mapping

Before applying the measured marker positions to a marionette, we need to map them into new positions not only to adapt to the size of the marionette but also to comply with such physical constraints as the strings. Our marionette, for example, does not have a mechanism to move the pelvis. Therefore, if the captured motion contains translation or rotation of the pelvis, the motion should be translated or rotated so that the pelvis motion is eliminated.

In this section, we describe an algorithm to compute seven parameters for translation, rotation, and scaling that map the measured marker positions into new positions that satisfy the constraints of the inverse kinematics model described in Section 4. We compute the mapping parameters independently for each frame rather than using the same parameters for all frames. Although it might seem natural to fix a parameter such as scaling for a particular human actor, we have found that, because of the marionette's limited range of motion, using the best possible mapping for each frame is preferable to using a fixed mapping bounded by the most difficult posture in the motion clip.

Suppose we use N markers in frame k as reference and denote the positions of the markers attached to the marionette by $\boldsymbol{p}_{k,i}^M$, those of the captured markers by $\boldsymbol{p}_{k,i}^C$ and those of the mapped markers by $\boldsymbol{p}_{k,i}^S (i = 1 \ldots N)$. We represent the translational, rotational, and scaling parameters of frame k by a position vector $\boldsymbol{t}_k$, a 3-by-3 rotation matrix $\boldsymbol{R}_k$, and a scalar s_k, respectively. Using these parameters, we compute the mapped position $\boldsymbol{p}_{k,i}^S$ of marker i from its original captured position $\boldsymbol{p}_{k,i}^C$ as $\boldsymbol{p}_{k,i}^S = s_k \boldsymbol{R}_k \boldsymbol{p}_{k,i}^C + \boldsymbol{t}_k$.

The system first computes the scaling, translation, and orientation parameters that minimize the total square distance between the measured markers and the virtual markers on the marionette in a fixed configuration. We can then use the inverse kinematics algorithm to compute the joint angles and string lengths that provide the best match between the two sets of markers. These two steps are repeated a number of times to refine the result.

The system finds the translation, rotation, and scaling parameters $\boldsymbol{t}_k, \boldsymbol{R}_k$, and s_k that minimize the evaluation function

$$J_k = \frac{1}{2} \sum_{i=1}^{N} |\boldsymbol{p}_{k,i}^S - \boldsymbol{p}_{k,i}^M|^2. \tag{10}$$

$\boldsymbol{p}_{k,i}^M$ are constant because the configuration of the marionette is fixed during this frame by frame computation.

We combine the unknowns into one variable $\boldsymbol{q}_k \in \boldsymbol{R}^7$, where the rotation matrix is represented by three independent variables whose time derivatives correspond to the angular velocity. We then use the common gradient method [14] to compute the optimum $\boldsymbol{q}_k$ incrementally as

$$\boldsymbol{q}_k = \boldsymbol{q}_k + \Delta \boldsymbol{q}_k,\ \Delta \boldsymbol{q}_k = -k \frac{\partial J_k}{\partial \boldsymbol{q}_k}. \tag{11}$$

The partial derivative of the mapped position $\boldsymbol{p}_{k,i}^S$ with respect to $\boldsymbol{q}_k$ is computed as

$$\boldsymbol{H}_{i,k} \triangleq \frac{\partial \boldsymbol{p}^S_{k,i}}{\partial \boldsymbol{q}_k} = \left(\; \boldsymbol{I}_3 \; \middle| \; [\boldsymbol{r}\times] \; \middle| \; \boldsymbol{R}_k \boldsymbol{p}^C_{k,i} \; \right) \tag{12}$$

where $\boldsymbol{I}_3$ is the 3-by-3 identity matrix, $\boldsymbol{r} \triangleq s_k \boldsymbol{R}_k \boldsymbol{p}^C_{k,i}$, and $[\boldsymbol{r}\times]$ is the cross product matrix of $\boldsymbol{r}$. Using $\boldsymbol{H}_{k,i}$, the partial derivative of J_k is computed by

$$\frac{\partial J_k}{\partial \boldsymbol{q}_k} = (\boldsymbol{p}^S_{k,i} - \boldsymbol{p}^M_{k,i})^T \boldsymbol{H}_{k,i}. \tag{13}$$

We apply this process to each frame independently by starting from the same initial guess. Using the result of previous frame as the initial guess would reduce the computation time, but the algorithm might not recover from a failure to obtain good mapping parameters in one frame due to, for example, missing markers. Regardless of the initial guess, the resulting mapping parameters may not be continuous because the algorithm is finding only a local minima. Small discontinuities are not a problem, however, because the marker positions are “filtered” by the feedback controller and by the SR-inverse used in the inverse kinematics computation.

6 Controlling Swing

If the mapped motion is applied directly to the marionette, the hands of the marionette will swing and the motion will not be a good match to that of the human actor. We solve this problem by building a simple linear model for the swing dynamics and experimentally identifying its parameters. An alternative approach would be to model the full dynamics of the marionette, but this tactic is not practical because of uncertainty in the model parameters and the limitations of the motors and sensors. Because marionette is made of wood and cloth, it is difficult to precisely determine the mass, inertia, and friction parameters of the joints. The joints are cleverly designed to prevent unrealistic joint angles (Figure 1), but this design also makes modeling of the system more difficult. The motors are inexpensive hobby servos and do not provide precise control. Furthermore, we do not have sensors that measure the current state of the marionette during a performance.

For the simple model of the dynamics, we make three assumptions. (1) Swinging of the hands occurs in the horizontal plane. Pulling the hands or legs up or down does not create a swinging motion. (2) The motion of a hand along the x axis (left/right) and the y axis (forward/back) are independently controlled by one motor each. (3) There is no coupling between the swinging of the left and right hands. These simplifying assumptions allow us to model swing as four independent systems, two for each hand.

Some situations occur in which the second and third assumptions do not hold. The hand marker sometimes moves along a circular trajectory rather than a straight line. The markers with fixed inputs inevitably move slightly when other markers are moved, violating the last assumption. Both problems are most likely to occur when the hand is relatively close to the body because the stiffness of the elbow and shoulder joints forces the hand away from the body.

6.1 Modeling of Swing Dynamics

In this section we describe the swing dynamics model that, when combined with the feedback controller of the inverse kinematics algorithm in Section 4, predicts the swing motion.

The inverse kinematics algorithm included a proportional controller, where the velocity of the pipe end $\dot{x}$ is computed from the current position of the pipe end x and the desired position u as $\dot{x} = k(u - x)$ where k is a constant gain. Therefore the transfer function from the marker trajectory to the motion of the pipe end takes the form

$$x = \frac{1}{a_{ik}s + 1} u \tag{14}$$

where u is the input (marker trajectory), x is the output (motion of the pipe end), s is the Laplace transformation operator, and a_{ik} is the parameter that determines the amount of delay.

The motion of the hand for a given trajectory of the pipe end can be modeled as a pendulum with a moving base (Figure 5 right). Using the length of the pendulum l and the damping term d, the equation of motion of the pendulum under gravity g is linearized around $x = y$ as $\ddot{y} = l/g(x - y) + d(\dot{x} - \dot{y})$. In general, therefore, the transfer function from the motion of the pipe end to the marker motion is written as

$$y = \frac{b_s s + 1}{a_s s^2 + b_s s + 1} x \tag{15}$$

where y is the output (actual marker trajectory) and a_s and b_s are the parameters that determine the frequency and damping respectively.

Combining Eqs.(14) and (15), the system computing the desired marker trajectory from the actual trajectory will be a 3rd-order system. We estimate the three parameters from motion capture data.

The gains of both systems were assumed to be 1, which turned out to be not true, probably because the joint motion ranges of the pipe prevented the pipe end from reaching the desired position or because the stiffness of the arm joints did not allow the string to be perpendicular. We decided not to consider these model errors because the desired marker position is not achievable if it violates the joint range constraint and the stiffness strongly depends on the configuration of the arm making the system too complicated.

6.2 Feedforward Controller

The feedforward controller K is formed by connecting the desired response G_D and the inverse of the estimated model P_m in series, that is, $K = G_D P_m^{-1}$. In order for the controller to be proper (the order of the denominator of the transfer function is larger than that of the numerator), the order of G_D must be larger than 2. We selected a 3rd-order G_D so that the output of the controller is continuous. We can also improve the response of the total system by selecting G_D with a smaller delay. In practice, however, we cannot use an arbitrarily fast G_D because as the gain of the controller increases, it becomes sensitive to modeling errors.

The parameters of the string dynamics model, a_s and b_s, depend on the length of the strings; therefore, we repeat the identification process for several different heights for each hand and design a controller for each model. We then apply the weighted sum of the outputs of the three controllers, where the weights are determined according to the actual height during a performance.

7 Results

The inverse kinematics computation to obtain the motor commands was repeated four times for each frame to ensure convergence. The total computation time was about 36ms per frame on a laptop PC with a Mobile PentiumIII 1GHz processor. Motor commands were sent every 50ms.

Based on the inverse kinematics computation, we developed an online control interface for the marionette. The model consists of nine string length constraints, joint motion range constraints for eight joints, two string direction constraints, and the potential energy constraint. The user can select a marker and drag it to any position. The inverse kinematics algorithm then computes the motor commands and the joint angles to move the marker to the specified position. Figure 6 shows several snapshots of the marionette model and the corresponding postures of the actual marionette.

The swing controller was designed for three different heights (-0.59m, -0.44m, and -0.29m, measured from the center of the panel where the motors and pulleys are attached). We had a total of twelve controllers for the x and y directions of both hands. Table 1 lists the parameter sets for the right hand in the x direction. The parameters were tuned manually, although it should also be possible to apply standard system identification techniques [15].

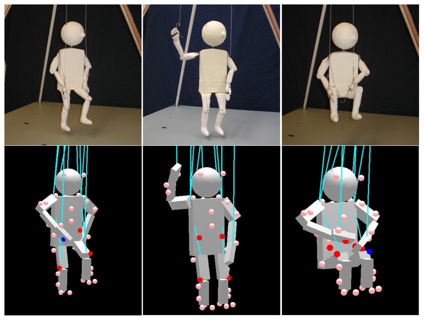
Figure 6: Postures generated by the interactive interface. Above: marionette, below: simulation.

Table 1: Parameters of the string dynamics models for the x direction of the right hand.

height [m]	-0.59	-0.44	-0.29
a_{ik}	0.8		
a_s	0.063	0.061	0.059
b_s	0.07	0.06	0.02

Figures 7 and 8 show the results of identification, controller design, and verification processes at the height of -0.29m. We used the motion capture system to measure the motion of the pipe end and right hand when a step input in x direction (left to right) was given as the desired marker trajectory (Figure 7). Then we designed a swing controller with the desired response $G_D = 1/(0.2s + 1)^3$.

Finally, the designed controller was applied to the same desired marker trajectory used for the identification and the response was measured (Figure 8). The swing controller reduced the width of the first vibration by 40%. The trajectory of the hand without the swing controller is different from that used for parameter identification (Figure 7), although we used the same reference trajectory. This discrepancy probably explains why the controller could not remove the vibration completely, thereby illustrating that a small difference in the configuration results in a relatively large difference in the swing dynamics due to the stiffness of the arms.

To test the motion of the marionette on a longer performance, we recorded the motions of two actors for two stories: “Who Killed Cockrobin?” and “Alaska.” Figure 9 compares the motions based on “Alaska” performed by actor 1. We used 32 reference markers and the two steps for mapping (computing approximate parameters and computing exact parameters) were repeated up to 500 times at each frame. The iteration was suspended if the total error of the marker positions were larger than the previous iteration. The computation time was approximately 5 seconds

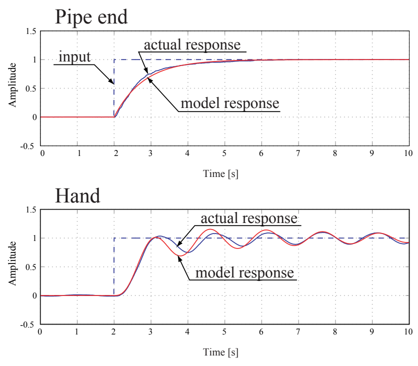

Figure 7: Actual and model responses to a step input. The amplitude of each motion is normalized. The hand of the marionette comes close to its head at this height.

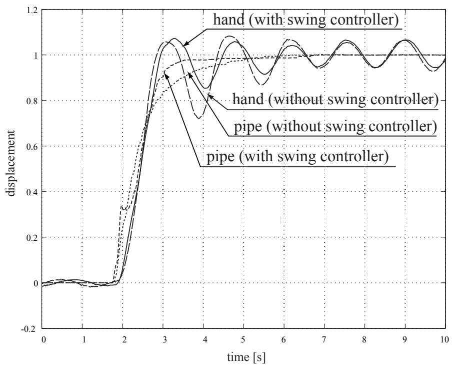

Figure 8: Response of the pipe and the hand to a step input.

per frame.

Figures 9 and 10 illustrate the same story performed by two different actors. The gestures are taken from approximately the same point in the story. The motion in Figure 11 is based on a different story performed by actor 1. The video clips are available online at http://humanoids.cs.cmu.edu/projects/marionette/.html, which also includes comparisons between the motions with and without the swing controller. The marionette's feet touch the floor as in a real performance.

8 Discussion

The motions of the actor and the marionette showed good correspondence, and we were able to distinguish two different styles for the same story (Figures 9 and 10). However, significant differences between the actor's and the marionette's postures were sometimes visible because of the limited range of motion of the pipes (for example the middle column of Figure 9). The marionette also had difficulty with fast motions because of the latency in the feedback controller of the inverse kinematics computation. This problem could be solved with a faster computer that could execute more iterations per step of the inverse kinematics computation, thereby increasing the stability of the computation and allowing larger gains.

Although the swing controller had a significant effect in isolated experiments, its effect during longer performances was quite small. We believe this discrepancy occurred because the stiffness of the arms is highly dependent on the configuration and this effect was not taken into account in the swing model. We could include this effect by testing the response of the system for both pipe position and string length.

The examples in this paper were limited to motions where the actor was told to stand in place during the perfor-

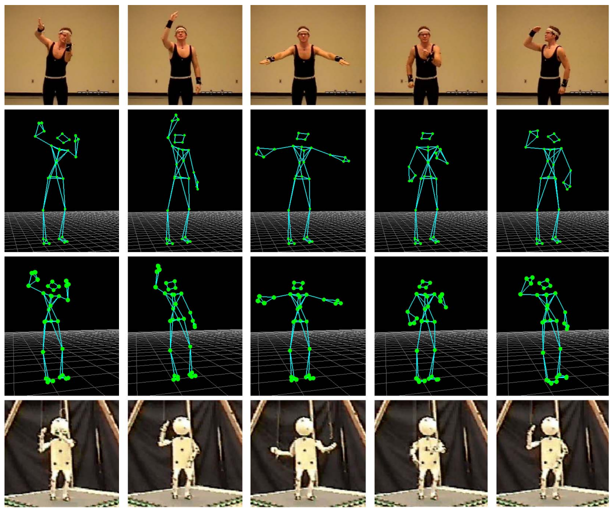

Figure 9: From the top: performance of actor 1 for "Alaska," the motion capture data, mapped marker positions, and the marionette's motion.

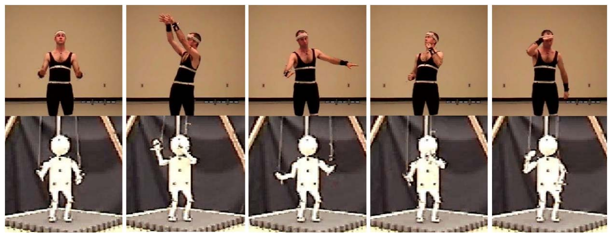

Figure 10: Marionette's motion for "Alaska" performed by actor 2.

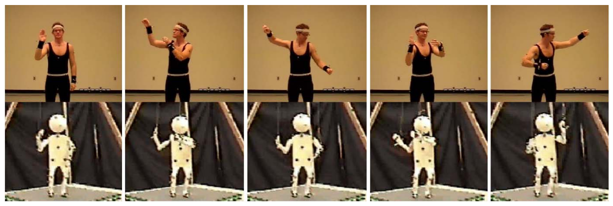

Figure 11: Marionette's motion for "Who Killed Cock-robin?" performed by actor 1.

mance. We could extend the range of feasible motions by adding more controllable strings and degrees of freedom. For example, a motor to control the string connecting the back would allow the marionette to bow. We could also add a pair of strings and motors to control the elbows independently or to move the entire marionette as a human operator would do for walking. In the construction of marionettes for human-operated performances extra strings are often added to enable a particular set of behaviors for that marionette's character.

We did not consider self collisions between the puppet and the strings or interaction with the environment. In the motions shown here we did not encounter situations where self collisions caused significant change of motion, but this issue is a serious concern in the design of performance marionettes with clothing that may catch on the strings. We kept the feet in contact with the floor to reduce the swing of the pelvis but did not explicitly consider contact with the environment in the control system. If the marionette had the additional degrees of freedom for such whole-body motions as walking, modeling of the interaction with the environment would be essential.

We explored two interfaces for driving the marionette: direct input of marker positions for realtime control and offline processing of human motion data. A third alternative would have been to capture a human-operated marionette performance to take advantage of the talent of a professional operator. The control scheme for this interface would presumably be significantly less complex because the motions would already be appropriate to the dynamics of the marionette. Such a system, however, could not easily be operated by an untrained user. In contrast, the control scheme described in this paper enables a naive user to program a motorized marionette to create entertaining performances simply by performing the gestures in a motion capture system.

Acknowledgments The authors would like to thank Rory Macey and Justin Macey for their assistance in capturing the human and marionette motions. This research was supported in part by NSF 0196089 and 0196217.

References

[1] "The Honda Humanoid Robot ASIMO," http://world.honda.com/ASIMO/.

[2] Y. Kuroki, T. Ishida, J. Yamaguchi, M. Fujita, and T. Doi1, "A Small Biped Entertainment Robot," in *Proceedings of Humanoids 2001*, Tokyo, Japan, November 2001.

[3] "Sarcos High Performance Robots," http://www.sarcos.com/entspec_highperfrobot.html.

[4] K. Yamane and Y. Nakamura, "Synergetic CG Choreography through Constraining and Deconstraining at Will," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, May 2002, pp. 855–862.

[5] G. Hoffmann, "Teach-In of a Robot by Showing the Motion," in *IEEE International Conference on Image Processing*, 1996, pp. 529–532.

[6] N. Pollard, J. Hodgins, M. Riley, and C. Atkeson, "Adapting Human Motion for the Control of a Humanoid Robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, May 2002, pp. 1390–1397.

[7] M. Gleicher, "Retargetting Motion to New Characters," in *Proceedings of SIGGRAPH '98*, 1998, pp. 33–42.

[8] S. Tak, O. Song, and H. Ko, "Motion balance filtering," *Eurographics 2000, Computer Graphics Forum*, vol. 19, no. 3, pp. 437–446, 2000.

[9] K. Yamane and Y. Nakamura, "Dynamics Filter—Concept and Implementation of On-Line Motion Generator for Human Figures," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, San Francisco, CA, April 2000, pp. 688–695.

[10] C. Rahn, F. Zhang, S. Joshi, and D. Dawson, "Asymptotically stabilizing angle feedback for a flexible cable gantry crane," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 121, pp. 563–566, September 1999.

[11] Y. Nakamura and H. Hanafusa, "Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 163–171, 1986.

[12] A. Maciejewski, "Dealing with the Ill-conditioned Equations of Motion for Articulated Figures," *IEEE Computer Graphics and Applications*, vol. 10, no. 3, pp. 63–71, May 1990.

[13] T. Sugihara, Y. Nakamura, and H. Inoue, "Realtime Humanoid Motion Generation through ZMP Manipulation based on Inverted Pendulum Control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, May 2002, pp. 1404–1409.

[14] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C Second Edition*. Cambridge, UK: Cambridge University Press, 1999.

[15] L. Ljung, *System Identification – Theory for the User*. Prentice – Hall, 1987.