# Methods for Motion Generation and Interaction with a Humanoid Robot: Case Studies of Dancing and Catching

**Marcia Riley[1,3], Ales Ude[2], Christopher G. Atkeson[1,3],**

[1]ATR Human Information Processing Research Laboratories
2-2 Hikaridai Seika-cho, Soraku-gun, Kyoto, Japan
[2]ERATO Kawato Dynamic Brain Project, Japan Science and Technology Corporation
2-2 Hikaridai Seika-cho, Soraku-gun, Kyoto, Japan
[3]College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

**Abstract**

We focus on creating realistic, adaptable movement for humanoid robots and virtual characters. Here we present motion synthesis of dance movements for a humanoid robot, and interactive behavior for catching. Our approach to motion generation includes collection of example human movements, handling of marker occlusion, extraction of motion parameters, and trajectory generation, all of which must be handled in such a way as to be faithful to the style of the original movements. In our interactive behavior, we generate ball-glove impact predictions and intercept motion trajectories for a real time catching task. In this paper we present our results and discuss ideas for future improvements.

## Introduction

Generating human-like motion for virtual characters and humanoid robots facilitates our goal of building more engaging machine behaviors and human-machine interactions. Evidence and experience show that agents with human-like traits provide an effective way to interact with a person (Nass, Steuer & Tauber 1994; Sloman & Croucher 1981). Generation of motion for humanoid robots is different from that of standard robots because of the complexity of the humanoid robots' kinematics and dynamics. Additionally, we encounter challenges not found when generating motion for a virtual human because we must obey the physics of the real world. Here we present work on generating dance and catching behaviors on a humanoid robot. We first discuss motion synthesis of the dance movements, and then discuss the implementation of the interactive catching task.

## Dancing

Our first task was to enable the 30-degree-of-freedom humanoid robot[1] in our lab to perform an Okinawan folk dance learned from human examples. Our approach consisted of selecting the type of motion capture used to collect the dance movements, processing this input for noise and occluded data, building a kinematic representation of the dancer, extracting motion parameters from the human motion, adjusting the parameters for the robot's capabilities, and synthesizing combinations of motion sequences on the robot. We assessed the effect of our motion data processing by comparing it with the original data in 3D graphics visualizations and by comparing the human and robot motion.

In the following sections we present the details of the task. Following this, we identify the key issues which arose during this work, including handling kinematic and dynamic mismatch between the human performer and the robot, and discuss different approaches such as alternate formulations of the occluded data and motion parameter extraction problems which may improve the results.

### Motion Capture

We considered three types of motion capture techniques for the Okinawan dance sequences: a marker-based system, the Optotrak[2], a system of goniometers strapped to the performer, the Sensuit[3], and computer vision.

In the Optotrak system the data is collected from infrared-emitting identifiable active markers attached to the human and connected by wires to a central controller. Special cameras track the 3D positions of each marker over the duration of the movement. This system reliably captures complex full body motions, although the markers are subject to noise and occlusion. Some marker systems rely on magnetic rather than visual information, and therefore eliminate the occlusion problem. Goniometer-based systems also do not suffer from occlusion. However, these systems are subject to larger noise than optical tracking devices. In addition, magnetic markers are sensitive to metal objects in the environment, which further restricts their usefulness.

Measuring complex movements such as dance requires the use of many markers. As the number of markers increase, they become more cumbersome to wear. Care must be taken to keep their trailing wires out of the way during movements. In part because of these encumbrances, computer vision may seem preferable. Vision, being non-invasive, provides a freedom of movement not present with other methods. However, even with the advances in computer vision with work such as (Ude 1999) and others, the field has not yet overcome the difficult problems posed by complex human motions. Many cameras may be needed to minimize occlusions, and analysis of vision data is compu-



[1]http://www.sarcos.com/

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

[3]http://www.sarcos.com/

tationally expensive. For these reasons we elected to use the Optotrak system.

## Treatment of Motion Capture Data

As noted above, even with good marker data, problems with missing data sometimes still occur. Self-occlusion by a limb, or wrist rotation into the torso, for example, may make total marker visibility very difficult.

We used an Optotrak system with three stereo cameras to collect 40 10-second dance trials from 22 markers attached to the dancer and sampled at 60 Hz. We found two types of missing data: systematic, where the cameras could not see the marker during most of the trials, and intermittent, where a marker was occluded for a short time during a movement, and then returned into view.

We discovered the systematic missing data during the motion capture session, where both the right and left heel markers were missing. However, this did not pose a significant problem, as we used redundant markers in problematic areas such as the feet. This strategy provided us with alternative means to get foot information. We also experimented with approximating the heel marker positions by reconstruction using rigid body constraints. We had measured and recorded all marker positions on the body after marker placement, so relative approximate positions were easy to calculate since we knew the position and orientation of the body parts from the visible markers.

The intermittent marker data was especially interesting for us. We needed to recover good approximations in order to extract continuous motions closely resembling the original movements. We used a cubic spline interpolation between visible data points to account for the missing data. This straightforward technique works well for short durations of missing data and provided us with enough sample movements for our task, but is unsatisfactory over large regions of missing data. Also, the method does not exploit other available information about the movement, such as kinematics or dynamics, to aid in the data recovery. In the dance results section we suggest different approaches to this problem.

## The Kinematic Model and Trajectory Extraction

Once enough missing marker data was accounted for, we extracted the motion parameters using a kinematic body representation. Our humanoid robot consists of 15 rigid parts divided into 6 interconnected kinematic chains: the head, the upper arms, lower arms, and hands; the lower and upper torso; the upper legs, lower legs and feet (Fig. 1).

The trajectories of joints connecting these body parts define the complete motion of the robot, as the robot pelvis is fixed in space. The robot has 26 degrees of freedom (DOFs), plus four degrees of freedom for eye movements, which are not considered here. (See Fig. 2). The dependencies between trajectories of different body parts are defined by the robot's geometric structure. It is thus appropriate to represent full-body motion trajectories in terms of independent variables, e.g. joint angles, because joint space trajectories automatically conform to the robot geometric constraints.

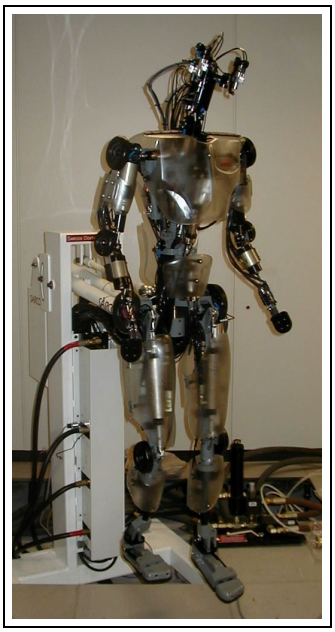

Figure 1: The humanoid robot in our laboratory

We model the performer's kinematics using a kinematic model similar to the robot's, but scaled to the performer. All corresponding degrees of freedom are available, but only those relevant for the desired movement need to be used.

The placement of a human body in a Cartesian space is fully described by the position and orientation of a global body coordinate system rigidly attached to one of the body parts and by the values of the joint angles. We use twist coordinates to model the kinematics (Murray, Li & Sastry 1994). For a revolute joint, the twist has the form

$$\boldsymbol{\xi}_i = \begin{bmatrix} -\boldsymbol{n}_i \times \boldsymbol{q}_i \\ \boldsymbol{n}_i \end{bmatrix}, \tag{1}$$

where $\boldsymbol{n}_i$ is the unit vector in the direction of the joint axis and $\boldsymbol{q}_i$ is any point on the axis, both given in a global body coordinate system. The 4 by 4 homogeneous transformation specifying the rotation about such a joint for an angle $\theta_i$ is given by the exponential map

$$\exp(\theta_i \boldsymbol{\xi}_i). \tag{2}$$

See (Murray, Li & Sastry 1994) for mathematical details.

If the coordinates of a marker in a local coordinate system of a rigid body part to which it is attached are given by $\boldsymbol{y}_j$, then its 3-D position at body configuration $(\boldsymbol{R}, \boldsymbol{d}, \theta_1, \ldots, \theta_n)$ can be calculated as follows

$$\begin{aligned} \tilde{\boldsymbol{y}}_j &= \boldsymbol{g}(\boldsymbol{R}, \boldsymbol{d}) \cdot \exp(\theta_{i_1} \boldsymbol{\xi}_{i_1}) \cdot \ldots \cdot \exp(\theta_{i_{n_j}} \boldsymbol{\xi}_{i_{n_j}}) \cdot \boldsymbol{G}_j \cdot \boldsymbol{y}_j \\ &= \boldsymbol{h}_j(\boldsymbol{R}, \boldsymbol{d}, \theta_1, \ldots, \theta_n). \end{aligned} \tag{3}$$

Here $\exp$ is the function mapping twists $\boldsymbol{\xi}_i$ representing the body kinematics into rigid body transformations, $\boldsymbol{G}_j$ is the homogeneous matrix combining the position and orientation of the local body part coordinate system to which the marker is attached with respect to the global body coordinate system

at zero configuration, $\boldsymbol{R}$ and $\boldsymbol{d}$ are the orientation and position of a global body coordinate system with respect to the world coordinate system, and $\boldsymbol{g}(\boldsymbol{R}, \boldsymbol{d})$ denotes the homogeneous matrix corresponding to $\boldsymbol{R}$ and $\boldsymbol{d}$. Note that the set of twists affecting the motion of a marker varies according to the identity of the body part to which the marker is attached.

Since our goal was to match the motion of the original performer as closely as possible, we required a close match between the performer's actual joint angles and the calculated joint angles. To attain this, we should minimize the difference between the measured marker positions and the marker positions generated by the recovered joint angles for each frame of motion over the set of body configurations $(\boldsymbol{R}(t_k), \boldsymbol{d}(t_k), \theta_i(t_k))$:

$$\begin{aligned} &\tfrac{1}{2}\sum_{j=1}^{N} \|\boldsymbol{h}_j(\boldsymbol{R}(t_k), \boldsymbol{d}(t_k), \theta_1(t_k), \ldots, \theta_n(t_k)) - \boldsymbol{y}_j(t_k)\|^2 \\ &= \tfrac{1}{2}\|\boldsymbol{h}(\boldsymbol{R}(t_k), \boldsymbol{d}(t_k), \theta_1(t_k), \ldots, \theta_n(t_k)) - \boldsymbol{y}(t_k)\|^2, \end{aligned} \tag{4}$$

where $\boldsymbol{y}_j(t_k)$ denotes the $j$-th measured marker at time $t_k$, $\boldsymbol{h} = [\boldsymbol{h}_1^T, \ldots, \boldsymbol{h}_N^T]^T$ and $\boldsymbol{y}(t_k) = [\boldsymbol{y}_1(t_k)^T, \ldots, \boldsymbol{y}_N(t_k)^T]^T$. To be able to apply a standard method like Gauss-Newton to this nonlinear optimization problem, we had to rewrite criterion (4) to account for the fact that the coefficients of the rotation matrix are not independent. The modified iteration exploits the properties of the exponential map $\exp$[4] which maps a 3-D real vector space onto the space of all rotation matrices and can be used to parameterize the neighborhood of each rotation matrix. Instead of directly minimizing (4), we looked for a minimum of

$$\begin{aligned} &\tfrac{1}{2}\|\boldsymbol{h}[\exp(\Delta\boldsymbol{r})\boldsymbol{R}(t_k), \boldsymbol{d}(t_k) + \Delta\boldsymbol{d}, \theta_1(t_k) + \Delta\theta_1, \ldots, \\ &\qquad \Delta\theta_n + \theta_n(t_k)] - \boldsymbol{y}(t_k)\|^2 = \\ &\tfrac{1}{2}\|\tilde{\boldsymbol{h}}(\Delta\boldsymbol{r}, \Delta\boldsymbol{d}, \Delta\theta_1, \ldots, \Delta\theta_n) - \boldsymbol{y}(t_k)\|^2. \end{aligned} \tag{5}$$

Here $(\boldsymbol{R}(t_k), \boldsymbol{d}(t_k), \theta_i(t_k))$ denotes the current estimate for the body configuration. This resulted in the following overconstrained system of linear equations that had to be solved at each iteration step

$$\begin{aligned} &\boldsymbol{J} \cdot [\Delta\boldsymbol{r}^T, \Delta\boldsymbol{d}^T, \Delta\theta_1, \ldots, \Delta\theta_n]^T = \\ &\boldsymbol{y}(t_k) - \boldsymbol{h}(\boldsymbol{R}(t_k), \boldsymbol{d}(t_k), \theta_1(t_k), \ldots, \theta_n(t_k)), \end{aligned} \tag{6}$$

where $\boldsymbol{J}$ is the Jacobian of $\tilde{\boldsymbol{h}}$ at $0$. It can be calculated using standard techniques from robotics (Murray, Li & Sastry 1994). The next estimate for the body configuration was calculated by

$$\begin{aligned} \tilde{\boldsymbol{R}}(t_k) &= \exp(\Delta\boldsymbol{r})\boldsymbol{R}(t_k), \\ \tilde{\boldsymbol{d}}(t_k) &= \boldsymbol{d}(t_k) + \Delta\boldsymbol{d}, \\ \tilde{\theta}_i(t_k) &= \theta_i(t_k) + \Delta\theta_i,\ i = 1, \ldots n. \end{aligned}$$

This modification enabled us to represent the global orientation by rotation matrices and thus avoid the pitfalls of minimal representations such as Euler angles.

[4]Note that this exponential map is different than the one in Eq. (2).

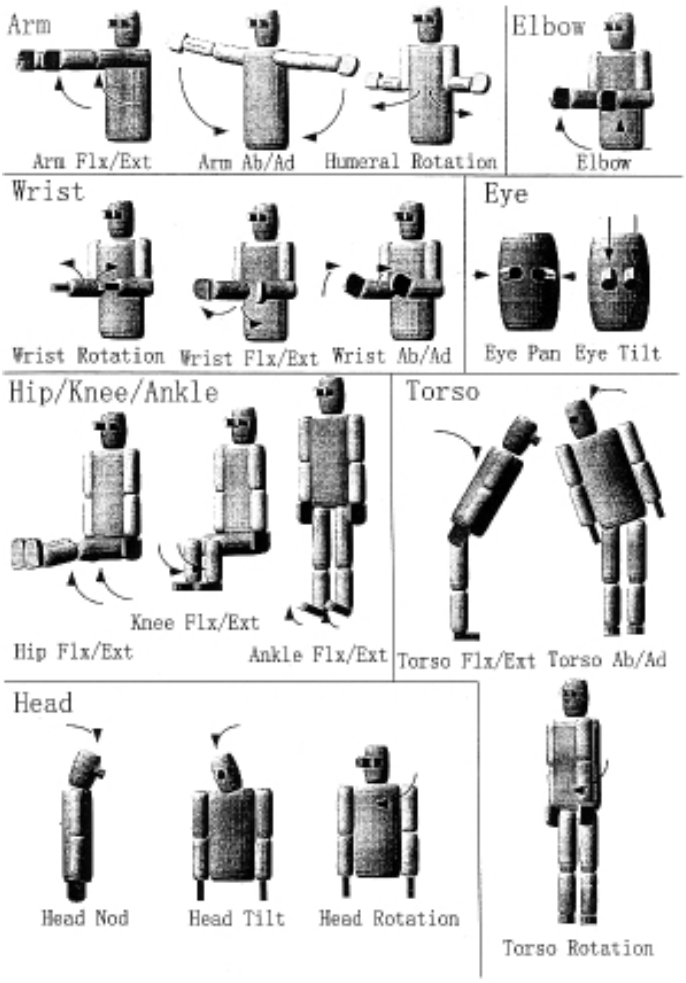


Figure 2: The robot's degrees of freedom.

We implemented our approach on a Dec Alpha workstation as a combination of Matlab[5] and C routines. The processed motion sequences were typically 5 seconds long, since we rejected regions of sparse data, and so included about 300 frames. Using Gauss-Newton to iteratively solve the nonlinear optimization problem required a good initial guess for the first frame, which was found either manually or by search. The method converged in an average of 25 iterations per frame at a precision of $10^{-8}$. However, we found that we were unable to estimate all 26 degrees of freedom, and had to lock the torso DOFs and some of the wrist DOFs in order to successfully estimate the other DOFs. Thus, we estimated 19 of the desired 26 DOFs by this method.

## Model Change

Synthesizing motion for models of different kinematic and dynamic structures has been an interesting topic for many researchers (Hodgins & Pollard 1997; Gleicher 1998). Motion generalization realized automatically or semi-automatically

[5]The MathWorks, Inc., http://www.mathworks.com

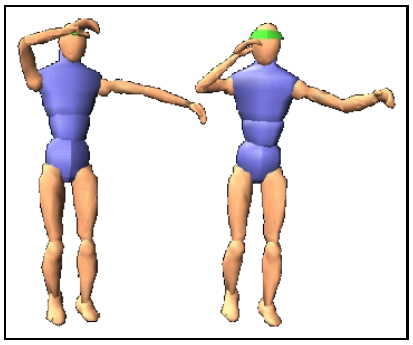

Figure 3: A frame from a graphics visualization of the reconstructed motion. The robot is visualized on the reader's right. Note the constraints on the shoulder and elbow degrees of freedom as compared to the human visualization on the left.

is valuable for computer animation and the gaming industry, where reuse of motion on multiple characters is advantageous. In our case, the target and source models were more similar in joint space (considering kinematic configurations and discounting joint limit differences) than in Cartesian space, so we exploited this joint space similarity to sidestep motion scaling. Because limb-length dependent movements did not play an important role in the dancing motion, this worked well. However, in other tasks such as clapping this kinematic mismatch would become an important issue in joint space, requiring a scaling approach to fit the motion to a new model.

Another consideration in changing models was the difference between the physical constraints of the person and the robot. Since the robot has joint angle limits which constrain some of its movements to a portion of what a human can do, the recovered trajectories were adjusted to fit its limits. Where possible, the entire movement was translated into the robot's range of motion. Where necessary, the trajectory was scaled to fit this range (Fig. 3).

Our kinematic solution gave us joint angle trajectories, but we still needed dynamic information to tell the robot how to move along the trajectory and to ensure target joint angles were met for each frame. Desired velocities and accelerations were calculated between sequential joint angle targets and smoothed to avoid jerky movements. An inverse dynamics algorithm[6] was used to compute the forces necessary to attain these desired trajectories. Whether dynamic mismatch between the source and the target causes a problem for motion synthesis is somewhat task-specific. We discuss this in the following section.

## Dance Results and Discussion

Comparing the measured marker motion with the reconstructed motion in 3D graphics visualizations revealed that the largest visual difference resulted from trajectory changes made in adjusting to the robot's joint limits. However, the style and similarity of the motion was preserved, albeit at a new position.

[6]Provided by Stefan Schaal, Kawato Dynamic Brain Project, http://www-slab.usc.edu/.

Figure 4: A frame of motion from the robot's dance performance.

For the dance motion, the desired velocities and accelerations were attainable, and slight differences between the desired and realized trajectories did not play a big role in the task. In other movements, however, this is not the case. Juggling, for example, requires precisely generating velocities and accelerations to attain the desired ball trajectories. If the forces computed to attain these trajectories do not result in a very close match between the desired and realized targets, the juggling will not be successful. Since human and robot dynamics differ, the new dynamics must be learned or somehow given to the robot to ensure successful motion. Catching is another example of a dynamics-sensitive task where path planning with good velocity and acceleration profiles becomes important for the robot's accuracy, smoothness and speed.

Besides kinematic and dynamic mismatch, key issues include the relationship of the optimization task and the reconstruction of occluded markers. The straightforward approach of sequentially minimizing the criterion function at consecutive time steps has the advantage of simplicity of formulation and fast convergence. However, the optimization process will fail when markers are occluded, which necessitates a preprocessing step addressing the problem of occluded marker data. Although we were able to recover only part of the trajectories by interpolating the marker positions from the visible frames through the intervals of occlusion, the information obtained was sufficient to describe the dance, since there is no clear discrete movement sequence or ending to the dance. If necessary, we could use different methods to recover the complete trajectories, such as reconstruction of marker position using rigid body constraints.

We must also consider the degrees of freedom which we could not estimate (2 wrist DOFs for each hand and the 3

torso DOFs). Our difficulties in estimating the wrist DOFs did not reflect the method, but rather our choice of how many markers were used to collect the data. Thus, recovery of the wrist DOFs can be corrected by adding additional markers to the wrist during the capture session. However, the torso degrees of freedom proved problematic. To estimate them, we would need to add additional criteria to our method to regularize the objective function. Other approaches to this problem include estimating the complete trajectory in one large optimization process instead of separately estimating single configurations. In (Ude, Atkeson & Riley 1999) we test this approach. Although this approach is promising, it requires much more processing time to recover the motion than the current sequential approach. We are continuing to develop both approaches, and compare them to determine which method results in motion more similar to the original motion.

## Catching

We now discuss the implementation of interactive tasks for a humanoid robot, using catching as an example. Interactive tasks are especially interesting because they engage a person in a behavior with a robot, and require real time solutions from the robot to participate in that behavior. A humanoid robot with its increased complexity (30 degrees of freedom compared with 6 degrees of freedom of a typical robot arm), challenges us to find satisfactory real time solutions for interaction. Additionally, because it is humanoid, the robot lends itself to solutions inspired by theories of human movement. There is a wealth of literature from the computational neuroscience community about human motor control and trajectory planning which can be applied to the robot's behaviors. We exploit this knowledge of human movement in our catching task solution.

To successfully complete the task the robot must detect the ball, determine when it is in flight, track and predict the ball's trajectory, and plan and execute a movement to intercept the ball. We prefer an iterative prediction algorithm which adjusts the prediction according to new sensory input. Additionally, we want an online motion trajectory generator, which changes the intercept trajectory according to the revised impact prediction information. Below we discuss our strategy to attain these goals.

### Implementation

The QuickMag[7] color vision system is used to detect the ball and track its flight. It supplies position data 60 times a second. To detect when the ball is in flight, the thrower's hand is also tracked, so that the ball position relative to the human's hand position can be determined. To the robot, the ball is in flight when the distance between the ball and hand surpasses a pre-defined threshold distance.

Once the ball is in flight, the robot starts collecting ball positions $(x_i, y_i, z_i)$ to use in predicting the time and place of ball-glove impact. (The robot is wearing a child's baseball glove on its left hand to make the catch.) Enough data should be available at the first prediction to provide a good solution. Here, a good solution is one in which subsequent predictions will result in relatively small changes to the original solution in order to avoid large changes in direction for the robot movement.

The current prediction algorithm assumes that the z-component of the ball trajectory is a parabola. Thus we can solve the following system of equations to calculate the parameters of the parabola

$$z_i = at_i^2 + bt_i + c, i = 1, \ldots, n, \tag{7}$$

where $n \geq 5$ is the number of measurements. The resulting normal system of equations is given by

$$\begin{bmatrix} \sum_{i=1}^{n} 1 & \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 \\ \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i^3 \\ \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^4 + \lambda \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} z_i \\ \sum_{i=1}^{n} z_i t_i \\ \sum_{i=1}^{n} z_i t_i^2 - \lambda \frac{1}{2} g \end{bmatrix} \tag{8}$$

where the additional parameters in the system ($\lambda$ and - $\lambda \frac{1}{2} g$) account for gravity.

For a desired pre-determined impact height, the impact time is estimated using the calculated parabola. Then the corresponding $x$ and $y$ impact coordinates for this impact time are estimated assuming that the $x$ and $y$ components of the trajectory are linear. With each new frame of vision data, this prediction is recalculated incorporating all current available frames of vision data. Predictions are currently begun after 5 ball position frames are collected. The process continues until the catch is made, or another end condition is triggered, such as the ball being lower than the impact height (a miss), or the task exceeding a maximum time limit.

We prefer a trajectory which is efficient and natural. Thus, we have drawn from studies of human movement and chosen a planner which results in bell-shaped velocity curves for point-to-point movements, where the desired velocity and acceleration at the movement beginning and end are zero. In catching, this point-to-point movement is from the initial hand position to the predicted ball-hand impact position. The motion primitives used to generate this movement are programmable pattern generators or PPGs, and are based on non-linear attractor dynamics. Details are described in (Schaal & Sternad 1998) and (Schaal, Kotosaka & Sternad 1999).

Since the ball position and impact estimate, and thus the desired end effector position, are given in Cartesian space, we must solve the inverse kinematics problem to arrive at a desired body configuration described by joint angles which will place the end effector in the target Cartesian location. To accomplish this we use the solution described in (Tevatia & Schaal 2000). This allows us to convert quickly and easily from Cartesian to joint space.

Using the inverse kinematics solver and Cartesian trajectory planner, we can attain the desired final position incrementally as follows. For our current impact prediction, the PPG specifies a partial movement toward the goal, specify-

[7] OKK Inc., Japan

Figure 5: A frame of motion showing the end of a catching sequence.

ing both desired position and velocity. The inverse kinematics solver is then used to follow in joint space the Cartesian path specified by the PPG. The inverse kinematics solution is calculated 480 times a second (the robot's task servo rate). As new ball position data becomes available, it can be used to modify the Cartesian target used by the PPG to plan the movement. In this way, we are always moving toward the current goal in small steps, and alter the motion trajectory as the impact prediction coordinates change.

## Catching Results and Discussion

Typical hand velocities needed by the robot to catch the throws may vary between 0.160 meters/second to 1.2 meters/second. Distances from the initial hand position to the final position may range between approximately 0.07 meters (almost thrown into the glove), and over 1 meter. The catching motion is completed in about 350 to 850 msec. The baseball glove used for catching is accounted for by changing the local coordinate system of the end effector to which it is attached, accounting for the extension of the hand.

Missed catches are caused by different conditions. The most common misses are due to the ball being thrown outside the robot's workspace, and the discrepancies which exist between the robot and vision coordinate systems, despite reasonable calibration. Additionally, as discussed below, the position data provided by the vision system can be noisy. We discuss one possible improvement to noise below. Despite these problems, the robot catches balls reliably. It fits a parabolic trajectory accurately at heights varying from .32 meters to .4 meters. (The usual starting height for the robot's left hand was .27 meters, with the origin of the coordinate system located near the pelvis of the robot.)

Figure 6 shows an estimation of the ball position provided by the vision system. Noise is evident, especially in the $y$ coordinate, which is the depth coordinate with respect to the vision system, and thus the most difficult to accurately estimate. To improve the estimation and minimize the noise, we are currently testing adding Kalman filtering (Maybeck 1979) to the prediction algorithm. The disadvantage is that a Kalman filter has trouble handling the start up transient. If the gain for observations is low, it takes a long time for the filter to match what the ball is actually doing. If the

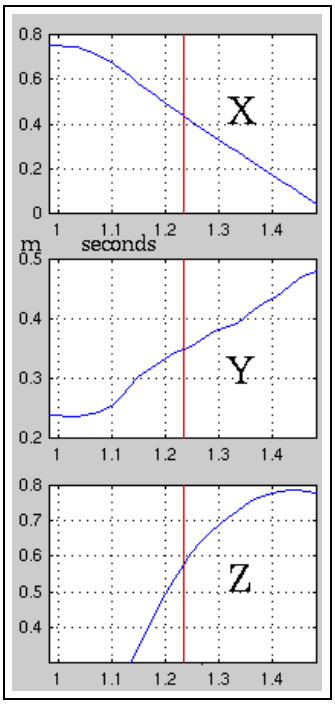


Figure 6: A measured ball trajectory.

gain is high, the start up transient is short, but the tracking is noisy. Therefore, a blend of the current prediction method with Kalman filtering may be a better solution if predictions with Kalman filtering prove to be more accurate. Since the current algorithm allows us to get accurate predictions after only a small number of vision frames, it could be used for early predictions, and a Kalman filter added for later predictions.

Our formulation for Kalman filtering is derived from the following physical equations of motion:

$$\begin{aligned} x_t &= x_0 + \dot{x}_0 t, \\ y_t &= y_0 + \dot{y}_0 t, \\ z_t &= z_0 + \dot{z}_0 t + 1/2 g t^2, \end{aligned}$$

where $x$, $y$, and $z$ describe the current ball position, $x_0$, $y_0$, and $z_0$ describe the initial ball position, $\dot{x}_0$, $\dot{y}_0$, and $\dot{z}_0$ are respectively the $x$, $y$, and $z$ initial velocities, $t$ is the time change from the initial to current state, and $g$ is the gravitational acceleration. Thus our state vector is given by $\boldsymbol{X} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$ and we measure the 3-D position $\boldsymbol{M} = [x, y, z]$. Our state and measurement equations are

$$\boldsymbol{X}_{k+1} = \begin{bmatrix} \boldsymbol{I}_3 & \Delta t_k \boldsymbol{I}_3 \\ 0_3 & \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{X}_k + \boldsymbol{u}_k + \boldsymbol{w}_k, \quad (9)$$

$$\boldsymbol{M}_k = [\ \boldsymbol{I}_3 \quad 0\ ]\, \boldsymbol{X}_k + \boldsymbol{e}_k, \quad (10)$$

where the control input $\boldsymbol{u}_k$ is equal to $\left[0, 0, 1/2 g \Delta t_k^2, 0, 0, g \Delta t_k\right]^T$, $\boldsymbol{w}_k$ is the process noise and $\boldsymbol{e}_k$ is the measurement noise.

Besides exploring Kalman filtering for possible improvements in ball trajectory predictions, we wish to improve the

naturalness of the catching movement. We will implement simple head movements corresponding to the ball movement to give the feeling that the robot is watching the ball's trajectory with the head-mounted cameras that serve as its eyes. (Currently, the robot is using the external QuickMag color vision system, with the cameras located in front of it.) In the future, we will also implement the catching task using vision data from these head-mounted cameras. We are also comparing human catching data from a motion capture session with the robot's catching. From this, we will look for further ways to improve the perceived naturalness of the motion.

Currently, the task allows the user to set the following parameters: the desired hand distance threshold to signal flight of the ball, the ball impact height for the prediction algorithm, and the minimum number of vision frames necessary to begin predicting the ball-glove impact time and place for the given height. These parameters can instead be specified by decision rules or learning. For example, if the robot desires more time to catch the ball, he may decide to lower the target impact height.

We have observed two broad types of catches: inside catches, where the robot reaches toward its body, and outside, where it extends its arm away from its body to catch the ball. The inside catch often requires pulling the arm back while rotating around the elbow, while the outside catch requires extension of the arm from the shoulder and elbow. At present, one default position is used to resolve redundancies in the inverse kinematics solver. To ensure good inverse kinematic solutions for a given type of catch, a preferred posture corresponding to the desired posture for the catch can be given to the inverse kinematics solver, thus biasing the solution toward the advantageous posture.

Finally, sometimes a catch may fail because the desired trajectories were not reached. This could result from an incorrect desired or attained force. To improve this, more precise trajectory following based on machine learning could be used.

## Conclusion

Being able to faithfully reproduce human-like movements for virtual characters and humanoid robots from human data helps create engaging human-machine interactions. Humanoid robots especially provide a rich and challenging platform for implementing such behaviors. In this work, we have discussed our methods for generating dance movements for a humanoid robot, and for catching. In the case of dancing, we started from human examples, and compared the generated dance motion to the human dance motion at several steps in the process. Our goal was to extract the parameters of the dance motion and give them to the robot while staying true to the dance's original style. In our interactive catching behavior, we generate point-to-point movements derived from studies of human motion with an iterative impact prediction algorithm and an adjustable motion trajectory generator. We suggest various strategies to improve the robustness of these methods, some of which are now in progress.

Videos showing our humanoid robot performing dance and catching movements can be viewed at http://www.hip.co.jp/∼mriley.

**Acknowledgments:** M. Riley is on leave from the College of Computing at the Georgia Institute of Technology, Atlanta, Georgia. A. Ude is on leave from the Department of Automatics, Biocybernetics and Robotics, Jozef Stefan Institute, Ljubljana, Slovenia. This work is part of the CyberHuman Project at ATR-HIP 3.

## References

B. Bodenheimer and C. Rose, The process of motion capture: Dealing with the data. In *Proc. Computer Animation and Simulation '97, Eurographics Workshop*, pages 3-18, Budapest, Hungary, September 1997.

M. Gleicher, Retargeting motion to new characters. In *Computer Graphics, Proc. SIGGRAPH '98*, pages 33-42, July 1998.

J. K. Hodgins and N.S. Pollard, Adapting Simulated Behaviors For New Characters. In *Computer Graphics, Proc. SIGGRAPH 97*, Los Angeles, CA, 1997.

P. Maybeck, *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, 1979.

T. Lozano-Perez, Robot programming, *Proc. IEEE*, 71(7):821-841, July 1983.

R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Boca Raton, New York, 1994.

C. Nass, J. Steuer, and E. Tauber, Computers are Social Actors. In *Proc. CHI '94*, pages 72-78, April 1994.

Schaal, S., Sternad, D., Programmable pattern generators. International Conference on Computational Intelligence in Neuroscience (ICCIN'98), pages 48-51. Research Triangle Park, NC, October 1998.

Schaal, S., Kotosaka, S., and Sternad, D., Nonlinear dynamical systems as movement primitives. International Conference on Computational Intelligence in Robotics and Automation. Monterey, CA, October, 1999.

A. Sloman and M. Croucher, Why robots will have emotions. In *Proc. 7th International Joint Conference on Artificial Intelligence*, Vancouver, 1981.

Tevatia, G., and Schaal, S., Inverse kinematics for humanoid robots. Submitted to IEEE International Conference on Robotics and Automation. San Francisco, CA, April, 2000.

A. Ude, Robust Estimation of Human Body Kinematics from Video. In *Proc. IEEE/RSJ Conf. Intelligent Robots and Systems*, pages 1489-1494, Kyongju, Korea, 1999.

A. Ude, C. G. Atkeson, M. Riley, Planning of joint trajectories for humanoid robots using B-spline wavelets. Submitted to IEEE Conference on Robotics and Automation, San Francisco, CA, April 2000.