Computer Animation and Simulation 2000 (Proceedings of the Eurographics Workshop on Animation and Simulation 2000, Interlaken Switzerland), Springer Wien, pp.59-71, August 21-22, 2000

Motion Tracking with Dynamic Simulation

Masaki OSHITA and Akifumi MAKINOUCHI

Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-Ku, Fukuoka, 812-8581, Japan
E-mail: moshita@db.is.kyushu-u.ac.jp, akifumi@is.kyushu-u.ac.jp

Abstract

This paper presents a physics-based real-time animation system for human-like articulated figures. We introduce a novel method for tracking motion data using dynamic simulation. By tracing a desired motion that is kinematically specified by a user using dynamic simulation, our system produces a motion that dynamically and realistically responds to a changing environment ensuring both controllability and physical realism. A tracking controller uses a human strength model as primary constraints, and controls joint angular acceleration within the available range of torque using inverse dynamics. As secondary constraints, the spatial accelerations of the center of mass and end-effectors are controlled. Unlike existing dynamic controllers that control joint torque for each degree-of-freedom (DOF) separately, our dynamic controller controls joint angular acceleration considering the influence of all DOFs using a pseudo-inverse matrix technique. In addition, this paper proposes two extensions of the Newton-Euler inverse dynamics method. One is a proximate solution for handling the closed loop problem. The other is for computing a minimum-moment point between the supporting segment of a figure and the ground for simulating falling motions. We demonstrate the efficacy of our approach by applying our method to a simple lifting task and generating various motions in response to the weight of the lifted load.

1 Introduction

Generating realistic human animation is a difficult challenge. Currently the most efficient and practical method is motion capture. Because the motion data are obtained from the movements of the real actors through the use of motion capture devices, this technique provides very rich details and a high degree of physical correctness. Recently, a number of techniques to reuse captured motion data have been proposed [10][17][6][5][29]. These techniques make it possible to retarget captured motion sequences on another character that has a different skeleton or apply them to another situation that has additional constraints. However, these methods use only kinematic constraints and do not include any notion of dynamics. Therefore these methods do not guarantee physical realism and cannot handle motions that dynamically and realistically respond to a changing environment (e.g. carrying a heavy load, colliding with other, balancing or falling down). However, generating such a motion is currently the most important issue for real-time applications such as electric game, virtual studio, and collaborative environments in which virtual humans move around and interact with the environments and each other.

This paper presents a physics-based real-time animation system for human-like articulated figures. We introduce a new method for tracking motion data using dynamic simulation. Our system takes a kinematic motion sequence as an input.

Based on the desired motion sequences, a tracking controller controls the joints of a figure considering with the muscle strength of the human body, balance control, and spatial constrains of the motion. By tracking a kinematically specified motion using dynamic simulation, our system produces various motions that dynamically and realistically respond to a changing environment, ensuring both controllability and physical realism. A desired motion consists of angular trajectories for each degree-of-freedom (DOF) and optional constraints. A user of our system creates these motion sequences using existing kinematic methods such as motion capture, keyframe interpolation, inverse kinematics, motion synthesis, and transformation techniques.

There are many systems for generating human animations using dynamic simulation [12][16][24][4]. These methods control joint torques for each DOF and use forward dynamics for computing joint angular accelerations based on the joint torques. On the other hand, our method controls joint angular accelerations for all DOFs directly and uses inverse dynamics for analyzing the torques required to realize the angular accelerations. By controlling angular accelerations in order to track a desired motion and modifying the angular accelerations to satisfy the multiple constraints, our method generates dynamically changing motions ensuring controllability.

In this paper, we present a tracking algorithm for controlling joint angular accelerations considering multiple constraints by using a pseudo-inverse matrix technique. The tracking controller uses a human strength model as the primary constraints and controls the joint angular accelerations within the available torque range that is achieved by the muscle strength. As secondary constraints, the spatial accelerations of the center of mass and end-effectors are controlled. Unlike existing dynamic controllers that determine joint torque for each DOF separately, our dynamic controller determines joint angular accelerations considering the influences of each DOF on the torque of other DOFs.

This paper also presents two extensions of the Newton-Euler method, which is one of the inverse dynamics methods. We introduce an approximate solution to handle a closed loop structure in a multiple support phase of human-like articulated figures. We also extend the Newton-Euler method with additional computations to determine a minimum-moment point on the surface between the supporting segment (e.g. foot) and the ground, and compute the rotational acceleration around the point for simulating falling over motions.

The remainder of this paper is organized as follows. The next section describes how this work relates to other research efforts. Section 3 explains our dynamic simulation system and data representations. Section 4 presents the extensions of the Newton-Euler method. Section 5 introduces the tracking control algorithm. In section 6, an experimental result is demonstrated and discussed.

2 Related Work

There are two main approaches for generating motions with dynamics: spacetime constraints and dynamic simulation. In the spacetime constraints approach [28][7], an optimal motion trajectory is automatically determined from specified spacetime constraints to minimize an objective function. Rose et al. [25] adapted this approach to articulated figures and proposed a keyframe interpolation technique between specified postures minimizing the required torque which is calculated using an inverse dynamics during the motion segments. Komura et al.[15] introduced a muscle model and used an objective function to minimize muscle force instead of joint torque. These methods are effective for generating keyframe animations. However, these methods cannot utilize existing motion data. Recently, Popović and Witkin [23] proposed a transformation technique which included the spacetime constraints approach and notions of dynamics. They extract the essence of physical properties from an original motion for the simplified model using the spacetime constraints approach. Then, they modify the extracted dynamic properties and reconstruct the resulting motion for the original articulated figure model. This method can easily modify the dynamic properties of existing motion data. However, this method does not reflect the character's skeleton and strength. Although the spacetime constraints approach ensures both controllability and physical realism, because solving an optimal problem requires an off-line process, it cannot produce motions interacting with environments in real-time.

The dynamic simulation approach is used for animating figures as they interact dynamically with an environment. These methods use a dynamic controller to compute joint torques based on the current state and a desired motion. Forward dynamics simulation then generates the resulting motions based on the joint torques. Researchers have developed dynamic controllers that are specialized for a particular character's skeleton and a behavior such as walking [4][16][24] and athletic movements [12]. These controllers use proportional-derivative (PD) servos to compute joint torque based on the desired and current angle for each DOF. The PD controller determines the output torque in proportion to the difference between the desired state $\theta_d, \dot{\theta}_d$ and the current state $\theta, \dot{\theta}$ (vector of angles and angular velocities, respectively).

$$\tau = k_p(\theta_d - \theta) + k_v(\dot{\theta}_d - \dot{\theta}). \tag{1}$$

The PD controller is easy to implement. However, it assumes nothing about the dynamic characteristics of the system. Therefore, to produce stable and natural looking motions, proportional gains k_p and k_v need to be tuned for both a character and a motion through trial and error. Once the controllers have been fine tuned and synchronized to each other, the method can produce expressive and physically correct motions. However, although an algorithm that transforms a successful controller on another character has been reported [11], it is still difficult to construct a controller that works successfully.

Recently, more advanced controllers have been proposed for tracking a kinematicaly specified motion using an approach similar to ours. Zordan and Hodgins [30] proposed a dynamic controller for general motions of the human upper-body. They combine the PD controller and optimal control. Their system determines optimal parameters k_p and k_v to minimize the error between the desired and produced motion sequences. However, because determining the parameter requires off-line process, this method is not suitable for real-time applications. Kokkevis et al. [14] used Model Reference Adaptive Control (MRAC) instead of the PD control scheme. They reported that the MRAC, based on feedback control, controls DOFs successfully and relieves the user from having to set explicit parameters. These controllers compute the torque for each DOF separately, not considering the influences of joint torque on another joint's angular acceleration. Therefore it seems to be difficult to adopt these controllers to the full-body motion that includes the movement of the center of mass (e.g. walking and running) or needs the assistance of other DOFs for controlling some DOFs (e.g. lifting and swing). Unfortunately, these existing controllers have not yet been applied to such a motion. These methods based on forward dynamics are aimed at generating natural motions even if they take unnatural trajectories as an input. On the other hand, our method assumes that input motion is already realistic, and is aimed at generating natural and realistic motions when the original motion is difficult to realize due to locking muscle strength, external forces, a collision with others, etc.

Some researchers have developed methods that modify an original motion using dynamic simulation. Ko and Badler[13] developed a system modifying human walking motion with balance and comfort control using inverse dynamics. Their system transforms the positions of the pelvis and torso, and the walking speeds in response to the joint torque calculated by inverse dynamics in real-time. However, the computation of the modification does not include dynamics and depends on parameters that are tuned by hand. Therefore the method cannot handle another motion or interactions with the environment.

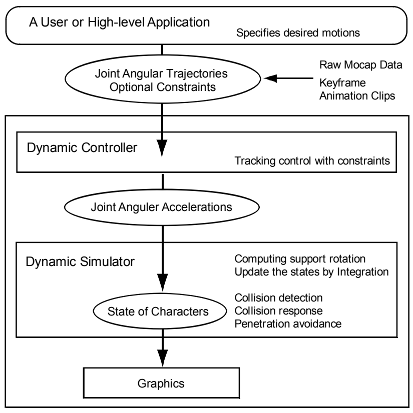
3 Dynamic Simulation System

The structure of the animation system presented in this paper is shown in Fig. 1. The system consists of two main modules: dynamic controller and dynamic simulator. On each simulation step, the dynamic controller computes joint angular acceleration for all DOFs, based on a desired motion that is specified by a user. Then the dynamic simulator updates the state of figures using dynamic simulation.

In standard physics-based animation systems[12][14][30], forward dynamics computes joint angular accelerations based on joint torques generated by a dynamic controller. However, in our system, the dynamic controller controls joint angular accelerations taking into account the required torques using inverse dynamics. Given the joint angular accelerations, the dynamic simulator computes the rotational acceleration of the supporting segment of the figure (e.g. foot) based on the joint angular accelerations (the details of this technique are described in section 4.2). The states of all figures then are updated by an integral computation. In addition, collision detection and response are performed. To handle collision, in the similar way of previous works[14][20], we introduce two stages: impact and contact stage. At the impact stage, when two figures collide with each other for the first time, an impact force works between them and changes their velocities. The variation of the velocities is computed by solving a liner equation [20][14]. At the contact stage, while the two figures contact with each other after the first impact, penetration avoidance works to prevent their penetration. In many physics-based systems, a spring-damper is used to avoid penetration. However, it requires both forward dynamics and a smaller time step for reducing error. Therefore we take another simple approach to control the angles directly using inverse kinematics. Inverse dynamics then compute reacting forces. The reacting forces are considered in the inverse dynamics of the next step of the simulation. If the figure want to remain in contact and the required torque is available, the two figures still contact. If the required torque is not available, they part.

3.1 Human Body Model

Our system uses a human body model consisting of segments and joints. Each rigid


A User or High-level Application
Specifies desired motions
Joint Angular Trajectories
Optional Constraints
Raw Mocap Data
Keyframe
Animation Clips
Dynamic Controller
Tracking control with constraints
Joint Anguler Accelerations
Dynamic Simulator
Computing support rotation
Update the states by Integration
State of Characters
Collision detection
Collision response
Penetration avoidance
Graphics
Fig. 1: The structure of the animation system.

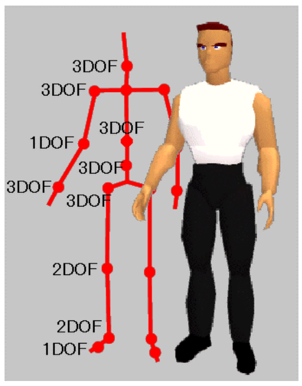


Fig. 2: The human body model.

segment is connected by a rotational joint. Each rotational joint has one, two, or three DOFs. Each DOF has two limits to restrict joint angles within human natural postures. In the experiments presented in this paper, we use a skeleton model composed of 18 segments and 17 joints with a total of 39 DOFs (Fig. 2). For dynamic simulation, the mass and moment of inertia of each segment are calculated based on the polygonal geometry [12]. The polygonal geometry of the human body also is used for collision detection and for determining a supporting point between its surface and the ground.

The dynamic controller uses the available muscle strength of each DOF as constraints. We adopt a simple muscle strength model [13][18]. The two muscle strength functions, the values of maximum and minimum available torque, are used for each DOF. Pandya et al. [21] showed by collecting human strength data that these values are expressed by functions of the joint angle and angular velocity. For the experiments, we assign approximated strength functions to each DOF.

3.2 Motion Data Representation

A desired motion is specified by both joint trajectories and optional constraints. The joint trajectories describe the joint angular displacements over time. These trajectories are used to control the figure’s joint angular acceleration in the dynamic controller. As optional constraints, spatially important segments or the center of mass can be indicated. For some behavior, the position and/or orientation of some segment or the center of mass are more important than individual joint angle. For examples, in picking-up motion, the goal position of the hand is more important than the joint angles along the arm. On the other hand, in walking motion, the horizontal position of the pelvis and the center of mass are important to balance the upper body. These constraints are indicated by a user for individual motion. The spatial or oriental trajectories are given by the user or automatically generated from the joint angular trajectories. The dynamic controller uses these trajectories as the secondary constraints. For a desired motion, motion capture data and any motion sequence created by other animation systems are used as an input to our system. To facilitate the use of existing motion sequences, motion synthesis [3][22][25] and editing

[17][6][10][23] techniques are available. Our system relies on these previous works. Their detailed description is beyond the scope of this paper.

4 Extensions of the Newton-Euler Method

The dynamic controller uses inverse dynamics for tracking control. Given current joint angles and angular velocities, and desired angular accelerations, inverse dynamics computes the joint torques required to realize the angular accelerations. The inverse dynamics problem is well defined, and systematic and efficient methods exist for serial articulated structures. Of the two popular formulations (Newton-Euler and Lagrangian), we adopted the Newton-Euler method. It costs *O(n)* where *n* is the number of DOFs of the figure. The Newton-Euler method computes the torques through two stages: outward iteration and inward iteration. During the outward iteration, accelerations of each segment are propagated from the root to the end-effectors. Then, during the inward iteration, the joint torques are propagated from the end-effectors to the root. For details of this algorithm, we refer the reader to [8]. In the following discussion of this section, we assume that the reader is familiar with the Newton-Euler method.

4.1 Closed-loop Problem in Multiple Supports

Because the inverse dynamics methods are designed for serial structures, it cannot handle a closed loop structure in a multiple support phase of human-like articulated figures (e.g. Fig. 3). The difficulty of this problem comes from the indeterminacy of how much force and torque are distributed to each supporting segment. Ko and Badler [13] introduce an approximate solution for the double support phase during walking motion. They distribute the force and torque from the upper body to each leg in proportion to the relative distances between the projection of the center of mass and the ankles. However, their approximation does not consider the dynamics of motion. Although there are more general methods that treat a closed loop as a non-closed loop with distance constraints [19], the methods need to solve an optimization problem and are not suitable for real-time systems. Therefore, we extend the Ko and Badler method for general postures and introduce dynamics.

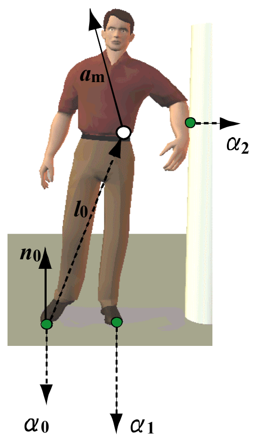


Fig. 3: A multiple supports phase.

When a number of segments make contact with the ground, we calculate the supporting ratio α_i for each segment *i*:

$$\alpha_i = \left(a_m \cdot \frac{l_i}{|l_i|} \right) \cdot \left(n_i \cdot \frac{l_i}{|l_i|} \right) \tag{2}$$

where n_i is a normal vector of the ground or the contact plane, l_i is a vector from the contact point to the center of mass, and a_m is a vector of acceleration of the center of mass (Fig. 4). Equation (2) represents the inner product of a_m and n_i along the direction of l_i. After determining the supporting ratio, we use the Newton-Euler method distributing the force and torque proportional to a_m [13]. To perform the iterations, we chose one segment whose supporting ratio is the largest on a common path, while other segments on the path are neglected. Our solution has the advantage that it is easily computed and reflects the dynamics of the movement of the center of mass. For example, in Fig. 3, when the figure moves toward right side, the acceleration of the center of mass works on the right side, then the supporting ratio of the left hand becomes large.

4.2 Computing Rotational Acceleration around the Minimum-moment Point

In physics-based simulation systems for articulated figures, it is difficult to generate an animation in which a figure falls over. To simulate such a motion, we should determine the supporting point around where the figure rotates. For example, when a figure falls to right side, the supporting point exists on the right side of the right foot. When a figure falls toward the back side, the supporting point exists at the heel of the foot. The supporting point is the minimum-moment point (MMP) at which the moment applied from the supporting segment to the ground is minimum. However, it is difficult to determine where the MPP exists on the supporting surface. Most physics-based systems [12] use a fixed MMP (e.g. middle of sole). In addition, for stable control, they sometimes introduce constraints that do not allow a figure to fall down and perform a forward dynamics. Therefore they cannot simulate falling down motions except on in which the direction of the figures falls over is already known and the supporting points are given. Ko and Badler [13] approximate the position of the MMP during human walking by means of monotonically advancing function from the heel to the tip of the toe. Aydin and Nakajima [1] propose an approximate solution to compute the position of the MMP. However, their algorithm approximates the foot to a rectangle and does not consider rotation around the supporting point.

We extend the Newton-Euler method to compute a MMP and rotational acceleration around the MMP. First, we consider only one main support segment. When a number of segments make contact with the ground, we chose one segment whose supporting ratio is the largest. Generally, when a figure maintains balance, there is a zero-moment point (ZMP) at which the moment applied from the support segment to the ground is zero [13]. Therefore, we are able to determine whether a rotation arises or not, by whether or not the ZMP exists on the supporting segment. To compute the position of the ZMP, we extent the Newton-Euler method and perform additional computations. Given joint angular accelerations, the Newton-Euler method computes joint torques, and the force and moment that is applied to the supporting segment from the previous joint (in Fig. 4, n_1 and f_1 are the moment and the force, respectively). The moment n_0 applied from the supporting segment to the ground is

$$n_0 = n_1 + l_{01} \times f_1 + l_g \times f_g \,. \tag{3}$$

On assumption that the surface of the ground is flat, the vector l_{01} that makes n_0 zero is computed and then the position of the ZMP is determined. At this time, if the ZMP exists inside of the supporting surface between the supporting segment and the ground, no rotation arises (Fig. 4 (a)). On the other hand, if the ZMP exists

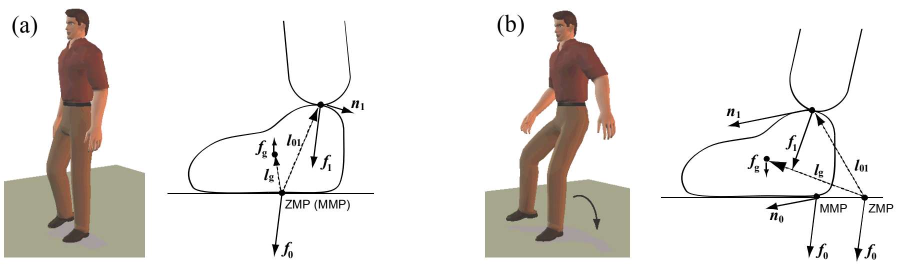


Fig. 4: Posture and forces applied to the supporting segment in (a) a balanced posture and (b) a unbalanced posture.

outside of the surface, a rotation will arise around the MMP that is the closest point to the ZMP (Fig. 4(b)). To determine this condition, our system uses the geometry of the figure and contact information that is reported by the collision handler. The rotational acceleration around the MMP is calculated by n_0 and the moment of inertia of all of the segments. Then the rotation acceleration is applied except when other supporting segments prevent the rotation. For example, in Fig. 3, although the MMP exist on the left edge of the left foot, rotation is prevented by the left hand. Using our algorithm, given joint angular accelerations, the dynamic simulator computes a rotation around a supporting point and generates an animation such that a figure falls over.

5 Motion Tracking Control

The dynamic controller computes joint angular accelerations on each simulation step. This algorithm uses a pseudo-inverse matrix method that is a common technique for inverse kinematics[9]. First, initial angular accelerations are calculated for each DOF to track a desired motion. Then, the angular accelerations are modified in order to satisfy multiple constraints. As primary constraints, the angular accelerations are restricted to be achieved only by available muscle strength. As secondary constraints, the angular accelerations are modified to control the spatial accelerations of the center of mass and the end effectors. Secondary constraints are applied under the condition that they exert no influence on the primary constraints. The remainder of this section describes each step of the algorithm in detail.

5.1 Determining Initial Angular Accelerations

The initial angular accelerations $\ddot{\theta}_{initial}$ are calculated from the differences between the current state $\theta_{curr}, \dot{\theta}_{curr}$ (n dimensional vectors of joint angles and angular velocities for each DOF) and the next state $\theta_{desired}, \dot{\theta}_{desired}$ of the desired motion:

$$\ddot{\theta}_{initial} = \alpha \ddot{\theta}_a + (1-\alpha)\ddot{\theta}_v \tag{4}$$

where $\ddot{\theta}_a$ is an angular acceleration meant to achieve the desired angle $\theta_{desired}$ on next step, $\ddot{\theta}_v$ is one to achieve the desired angular velocity $\dot{\theta}_{desired}$. These values are calculated for each DOF respectively using a differential equation that is used in the dynamic simulation. Since we cannot satisfy both values, we use a blend parameter α (currently, we use $\alpha = 0.2$). On this step, a proportional control is used. However, unlike PD controllers, because all terms are computed in angular acceleration space, stabilization is ensured. If all following constraints are satisfied in $\ddot{\theta}_{initial}$, it is used directly as the output.

5.2 Available Torque Constraints

As the primary constraints, the dynamic controller uses the range of available torque of each DOF. The range of available torque (two vectors of maximum and minimum torques) is obtained from the muscle strength curves described in section 3.1 as:

$$\tau_{\max} = f_1\left(\theta_{curr}, \dot{\theta}_{curr}\right), \quad \tau_{\min} = f_2\left(\theta_{curr}, \dot{\theta}_{curr}\right). \tag{5}$$

The torques to drive the initial accelerations are calculated by the inverse dynamics equation:

$$\tau_{initial} = H(\theta_{curr})\ddot{\theta}_{\text{initial}} + C\left(\theta_{curr}, \dot{\theta}_{curr}\right) + G(\theta_{curr}) + K(\theta_{curr})F \tag{6}$$

where $H(\theta_{curr})$ represents the moment of inertia, and $C\left(\theta_{curr}, \dot{\theta}_{curr}\right)$, $G(\theta_{curr})$, and $K(\theta_{curr})F$ represent the influence on the torques due to coriolis and centrifugal, gravity, and external force, respectively.

If the joint torque of some DOF exceeds its available range, the joint angular accelerations of all DOFs are modified in order to reduce the torque of the DOFs. Let $\Delta\tau'$ be a *k* dimension vector represents the variations of toques for *k* DOFs that exceed the available torque range. Comparing $\tau_{initial}$ with $\tau_{\max}$ and $\tau_{\min}$, the variations of the joint torques $\Delta\tau'$ are calculated:

$$\Delta\tau' = S_1\,\Delta\tau \qquad \Delta\tau_j = \begin{cases} \tau_{\max,i} - \tau_{initial,i} & \text{if } \tau_{initial,i} > \tau_{\max,i} \\ \tau_{\min,i} - \tau_{initial,i} & \text{if } \tau_{initial,i} < \tau_{\min,i} \end{cases} \tag{7}$$

where S_1 is a selection matrix from *n* dimensional vector $\Delta\tau$ to *k* dimensional vector $\Delta\tau'$, $\Delta\ddot{\theta}$ is the *n* dimensional vector that is the variation of the angular accelerations. The relation between $\Delta\tau'$ (*k* dimension) and $\Delta\ddot{\theta}'$ (*n* dimension, $n \geq k$) is

$$\Delta\tau' = H'\Delta\ddot{\theta}' \tag{8}$$

where H' is a $k \times n$ sub-matrix of the moment of inertia $H(\theta_{curr})$. The matrix H' depends on only θ_{curr}, the current joint angles. The H' is computed by using the existing methods [26]. It costs $O(n^2)$ computational time. To solve the redundant linier system (8), a pseudo-inverse matrix is used:

$$\Delta\ddot{\theta}' = H'^{+}\,\Delta\tau' + \left(I - H'^{+}H'\right)x \tag{9}$$

$$H'^{+} = H'^{t}\left(H'H'^{t}\right)^{-1} \tag{10}$$

where H'^{+} is the pseudo-inverse matrix of H' and x is a optimization vector that is used in the secondary constraints. The first term in equation (9) is the least square solution that minimizes the normal of $\Delta\ddot{\theta}$. The second term is the homogeneous portion of the solution, partially performing a desired optimization x under the exact achievement of the primary constraints. This is achieved with the projection to the null space of the linear transformation. A more detailed discussions of the pseudo-inverse matrix method can be found in [9][3][6].

To modify the initial angular accelerations $\ddot{\theta}_{initial}$ in order to satisfy the primary constraints, using the first term of equation (9), the first variation of angular accelerations $\Delta\ddot{\theta}_1$ are calculated.

$$\Delta\ddot{\theta}_1 = H'^{+}\Delta\tau' \tag{11}$$

However, the equations (8), (9), (10) and (11) take into account only *k* DOFs whose torques exceed the available range. Therefore the modified angular accelerations $\Delta\ddot{\theta}_1$ may exceed the available range of the torques besides *k* DOFs. When such excess is present, the exceeded DOFs are added to the group of *k* DOFs, then H'^{+} and $\Delta\ddot{\theta}_1$ are recomputed. This step is repeated until the primary constraints are satisfied. Because the moment of inertia matrix is regular [26][8], the answer $\Delta\ddot{\theta}_1$ is consistently achieved.

5.3 Spatial Constrains of the Center of Mass and End-Effectors

To control the positions and orientations of end-effecters, inverse kinetimatics methods [9][6] are available. In addition, to control both them and the position of the center of the mass, we use an inverse kinetics method [3][1], witch employs a pseudo-inverse matrix technique. The inverse kinetics equation is

$$x = J_g^{\ +}\Delta\ddot{g} + \left(I - J_g^{\ +} \cdot J_g\right)J_e^{\ +}\Delta\ddot{e} \tag{12}$$

where $\Delta\ddot{g}$ is the variation of the position of the center of mass, and $\Delta\ddot{e}$ is composed of the variations of the position and/or orientation of the end-effectors. The dimensions of $\Delta\ddot{g}$ and $\Delta\ddot{e}$ are depend on the constraints specified by a user for individual motion. For example, when the horizontal position of the center of mass and the spatial position of both hands are indicated, the dimension of Δg will be two and the dimension of Δe will be six. J_g is the Jacobian matrix that projects the variation of angles to the variation of the position of the center of mass. In addition, J_e projects the variation of angles to the variation of the position and/or orientation of the end-effectors.

$$\Delta\ddot{g} = J_g\,\Delta\ddot{\theta} \qquad \Delta\ddot{e} = J_e\,\Delta\ddot{\theta}\,. \tag{13}$$

The desired variations of the position of the center of mass $\Delta\ddot{g}$ and the variations of the position and/or orientation of the end-effectors $\Delta\ddot{e}$ are determined calculated by the equation

$$\ddot{g}_{initial} = J_g\left(\ddot{\theta}_{initial} + \Delta\ddot{\theta}_1\right) \qquad \Delta\ddot{g} \;= \ddot{g}_{desired} - \ddot{g}_{initial} \tag{14}$$

$$\ddot{e}_{initial} = J_g\left(\ddot{\theta}_{initial} + \Delta\ddot{\theta}_1\right) \qquad \Delta\ddot{e} \;= \ddot{e}_{desired} - \ddot{e}_{initial} \tag{15}$$

where $\ddot{g}_{initial}$ and $\ddot{e}_{initial}$ are the acceleration of the center of mass and the end-effectors, respectively, that are achieved by the joint angular accelerations which are modified by the primary constraints. $\ddot{g}_{desired}$ and $\ddot{e}_{desired}$ are the desired spatial accelerations that are determined by the desired motion. The second variation of the angular accelerations $\Delta\ddot{\theta}_2$ is calculated as

$$\Delta\ddot{\theta}_2 = \left(I - H''^{+} S_2 H\right)x \tag{16}$$

where S_2 is a mapping matrix that selects only *l* dimensional vector from $H x$ as in equation (7), and I is the identity matrix. We note that *l* DOFs are selected independently of *k* DOFs used in the first modification. Then, as in equation (11), we repeat this computation until the torque of any DOF that is required to $\Delta\ddot{\theta}_2$ does not exceed its available range.

5.4 Determining Output Angular Accelerations

Finally, the output joint angular accelerations $\ddot{\theta}_{output}$ are calculated by the equation

$$\ddot{\theta}_{output} = \ddot{\theta}_{initial} + \Delta\ddot{\theta}_1 + \Delta\ddot{\theta}_2 \,. \tag{17}$$

The full description of (17) is also expressed as

$$\ddot{\theta}_{output} = \ddot{\theta}_{initial} + H'^{+} S_1 \Delta\tau + \left(I - H''^{+} S_2 H\right)\left\{J_g{}^{+}\Delta\ddot{g} + \left(I - J_g{}^{+} \cdot J_g\right)J_e{}^{+}\Delta\ddot{e}\right\}. \tag{18}$$

This algorithm controls angular accelerations in order to achieve a resulting motion that is close to the desired motion, while considering the influence of all DOFs, within the available torque ranges, in the similar fashion as do human beings. As a result, naturally and realistically changing motions are produced.

6 Experiments and Discussion

In this section, we show an experimental result generated by applying our method to a lifting task (Fig. 5, see Appendix). In the three animations, each figure tracks the same motion, while it has a different weight load (1kg, 3kg and 4kg, respectively) with its right hand. In this example, we used a keyframe motion sequence that is made by a hand as a desired motion. The lifting motion consists of the trajectories of all the DOFs and additional spatial constraints of the center of mass and the right hand. Our system works normally at 10fps on a PC (PentiumIII 600MHz Dual). The duration between steps of this simulation is 1/30 second.

In the first animation (1kg), because the constraints are satisfied throughout the entire motion, the original motion is almost exactly tracked. In the second animation, because of the load's relatively heavy weight (3kg), the torques at the DOFs along the right arm exceed the available ranges. In the tracking control, to reduce the stress of DOFs along the right arm, other DOFs (knee, back, etc) that have strongly influence the right arm are forced to move slightly maintaining its balance. As a result, resulting motion close to the original motion is produced, but with an attendant jerking motion. In the last example, because of the heavy load, the back of the figure is forced to bend, and it then lost its balance and falls down. By producing various motions in response to the stress on right arm, the efficacy of our methods is demonstrated.

For this experiment, we used hand-tuned strength function. In our future research, we will plan to clarify the differences of motions produced under different skeletal and strength conditions (e.g. man, woman and child) using actual human strength data. In addition, because measuring human strength is a difficult work, we intend to construct the strength modes from a set of captured motion data.

The dynamic controller presented in this paper is intended only to trace a desired motion. However, in human movements, when it is very difficult to realize a desired motion, more sophisticated control schemes are sometimes used. For example, when a figure loses its balance, it does not only control the center of mass, but also attempts to maintain its balance by moving its foots. If a figure cannot lift a load using only one hand, it automatically uses its other hand. In the future, we are going to introduce such advanced behavior to the current low-level tracking controller.

7 Conclusion

In this paper, we present a physics-based animation system for articulated figures. A new control algorithm to track an original motion using dynamic simulation is introduced. The physics-based approach is not currently adopted in many real-time applications since physics-based systems are difficult to construct and control. However, on-line applications such as electric games or virtual environments have serious limitations in that they cannot generate figures whose motions dynamically change in response to an environment in real-time. We believe that our approach will break through these limitations. Even though a more sophisticated control scheme might be required, the dynamic simulation and tracking control techniques that we present in this paper will form a fundamental basis for further development in the area of real-time computer animation.

Acknowledgements

A part of this research is supported by the Grant-in-Aid for Scientific Research (10308012) from the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] Yahya Aydin, and Masayuki Nakajima, "Realistic Articulated Character Positioning and Balance Control in Interactive Environments", Proceedings of Computer Animation '99, pp.160-168, 1999.

[2] Ronan Boulic, Pascal Be'cheiraz, Luc Emering, and Daniel Thalmann, "Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time Animation", Proceedings of the ACM International Symposium VRST'97, pp 111-118, 1997.

[3] Ronan Boulic, Ramon Mas-Sanso, and Daniel Thalmann, "Complex Character Positioning Based on a Compatible Flow Model of Multiple Supports", IEEE Transactions on Visualization and Computer Graphics, Vol.3, No.3, pp.245-261, July-September 1997.

[4] Armin Bruderlin, and Thomas W. Calvert, "Goal-Directed, Dynamic Animation of Human Walking", Computer Graphics (SIGGRAPH '89 Proceedings), Vol.23, No.3, pp.233-242, 1989.

[5] Armin Bruderlin, and Lance Williams, "Motion Signal Processing", SIGGRAPH '95 Proceedings, pp.97-104, 1995.

[6] Kwang-Jin Choi, and Hyeong-Seok Ko, "On-line Motion Retargetting", Proceedings of International Pacific Graphics, 1999.

[7] Michael F. Cohen, "Interactive Spacetime Control for Animation", Computer Graphics (SIGGRAPH '92 Proceedings), Vol.26, No.2, pp.293-302, 1992.

[8] Roy Featherstone, "Robot Dynamics Algorithms", Kluwer, 1987.

[9] Michael Girard, and A. A. Maciejewski, “Computational Modeling for the Computer Animation of Legged Figures”, Computer Graphics (SIGGRAPH ‘85 Proceedings), Vol.19, No.3, pp.263-270, 1985.

[10] Michael Gleicher, “Retargetting Motion to New Characters”, SIGGRAPH ‘98 Proceedings, pp.33-42, 1998.

[11] Jessica K. Hodgins, and Nancy S. Pollard, “Adapting Simulated Behaviors For New Characters”, SIGGRAPH ’97 Proceedings, pp.153-162, 1997.

[12] Jessica K. Hodgins, Wayne L. Wooten. David. C. Brogan, and James F. O’Brien, “Animating Human Athletes”, SIGGRAPH ’95 Proceedings, pp.71-78, 1995.

[13] Hyeongseok Ko, and Norman I. Badler, “Animating Human Locomotion with Inverse Dynamics”, IEEE Computer Graphics and Applications, Vol.16, No.2, pp.50-59, 1996.

[14] Evangelos Kokkevis, Dimitris Metaxas, and Norman I. Badler, “User-Controlled Physics-Based Animation for Articulated Figures”, Proceedings of Computer Animation '96, 1996.

[15] Taku Komura, Yoshihisa Shinagawa, and Tosiyasu L. Kunii, “A Muscle-based Feed-forward Controller of the Human Body”, Computer Graphics Forum (Proceedings of Eurographics ’97), Vol.16, No.3, pp.165-176, 1997.

[16] Joseph Laszlo, Michiel van de Pann, and Eugene Fiume, “Limit Cycle Control and Its Application to the Animation of Balancing and Walking”, SIGGRAPH ’96 Proceedings, pp.155-162, 1996.

[17] Jehee Lee, and Sung Youg Shin, “A Hierarchical Approach to Interactive Motion Editing for Human-like Figures”, SIGGRAPH ’99 Proceedings, pp.39-48, 1999.

[18] Philip Lee, Susanna Wei, Jianmin Zhao, and Norman I. Badler, “Strength Guided Motion”, Computer Graphics (SIGGRAPH ’90 Proceedings), Vol.24, No.3, pp.253-262, 1990.

[19] J. Y. X. Luh, and Yuan-Fang Zheng, “Computation of Input Generalized Forces for Robots with Closed Kinematic Chain Mechanisms”, IEEE Journal of Robotics and Automation, Vol. RA-1, No. 2, pp.95-103, 1985.

[20] Matthew Moore, and James Wilhelms, “Collision Detection and Response for Computer Animation”, Computer Graphics (SIGGRAPH ’88 Proceedings), Vol.22, No.3, pp.289-298, 1988.

[21] Abhilash K. Pandya, James C. Maida, Ann M. Aldridge, Scott M. Hasson, and Barbara J. Woodford, “The Validation of a Human Force Model To Predict Dynamic Forces Resulting From Multi-Joint Motions”, Technical Report 3206, NASA, Houston, Texas, 1992.

[22] Ken Perlin, and Athomas Goldberg, “Improv: A System for Scripting Interactive Actors in Virtual Worlds”, SIGGRAPH ’96 Proceedings, pp.205-216, 1996.

[23] Zoran Popović, and Andrew Witkin, “Physically Based Motion Transformation”, SIGGRAPH ’99 Proceedings, pp.11-20, 1999.

[24] Marc H. Raibert, and Jessica K. Hodgins, “Animation of Dynamic Legged Locomotion”, Computer Graphics (SIGGRAPH ’91 Proceedings), Vol.25, No.4, pp.349-358, 1991.

[25] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen, “Efficient Generation of Motion Transitions using Spacetime Constraints”, SIGGRAPH ’95 Proceedings, pp.147-154, 1995.

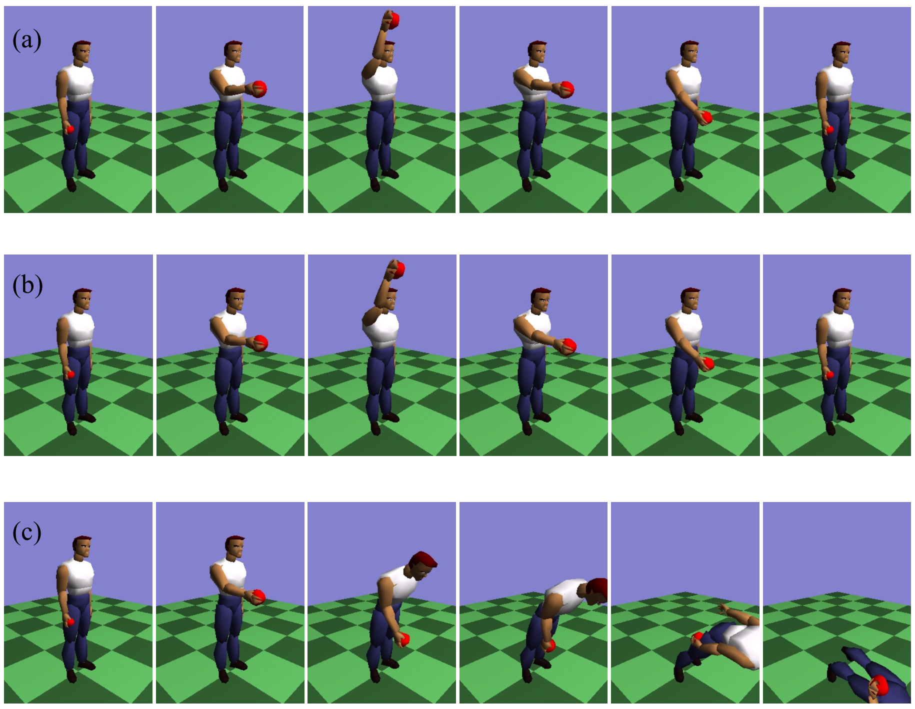
[26] M. W. Walker, and D. E. Orin, “Efficient Dynamic Computer Simulation of Robotic Mechanisms”, Journal of Dynamic Systems, Measurement, and Control, Vol.104, pp.205-211, September 1982.

[27] Douglas J. Wiley, and James K. Hahn, “Interpolation Synthesis for Articulated Figure Motion”, IEEE Computer Graphics and Applications, Vol.17, No.6, pp.39-45, 1997.

[28] Andrew Witkin, and Michael Kass, “Spacetime Constraints”, Computer Graphics (SIGGRAPH ’88 Proceedings), Vol.22, No.4, pp.159-168, 1988.

[29] Andrew Witkin, and Zoran Popović, “Motion Warping”, SIGGRAPH ’95 Proceedings, pp.105-108, 1995.

[30] Victor B. Zordan, and Jessica K. Hodgins, “Tracking and Modifying Upper-body Human Motion Data with Dynamic Simulation”, Computer Animation and Simulation ’99 (Proceedings of Eurographics Workshop on Animation and Simulation `99), 1999.



Simulated lifting up task. In each animation, the figure tracks the same lifting motion while having a different weight load ((a)1kg, (b)3kg, and (c)5kg) (Oshita and Makinouchi, Fig. 5).