Dynamics Filter—Concept and Implementation of Online Motion Generator for Human Figures

Katsu Yamane, *Member, IEEE*, and Yoshihiko Nakamura, *Member, IEEE*

***Abstract*—In this paper, we describe the concept and implementation of a *dynamics filter*, an online, full-body motion generator that converts a physically infeasible reference motion into a feasible one for the given human figure. Our implementation of the dynamics filter only uses time-local information, that is, does not require the whole motion sequence in advance. Therefore, the reference motion may be changed online in response to the interaction with a human or the environment. The dynamics filter is implemented based on an efficient rigid-body collision/contact model. This model itself provides an efficient algorithm for dynamics simulation of collisions and contacts. We demonstrate the power of the dynamics filter by several example motions that use motion capture data as a reference.**

***Index Terms*—Collision/contact model, human figures, motion generation, motion synthesis, physical consistency.**

I. Introduction

INTERACTIVITY is a key issue in many applications of humanoid robots working in a changing or unknown environment with a human. A robot may need to avoid an obstacle suddenly appearing in front of it, resist external forces applied to its body, or abruptly change the walking direction as ordered by its master. Most previous research has discussed the interactivity in humanoid robot motion on the behavioral level. Although such approaches are capable of generating intelligent behaviors, there still remains the problem of attaching physically feasible and natural full-body motion to the behavior.

Physical consistency, the condition that the motion is physically possible for some choice of internal forces, is essential for stable and reliable motions of humanoid robots. The total angular momentum, for example, should be constant while the human figure is in the air. The zero moment point (ZMP) should be in the contact area if one or more links are in contact with the environment. There may be other constraints depending on a specific robot such as joint torque and angle limits.

Manuscript received April 16, 2002; revised October 16, 2002. This paper was recommended for publication by Associate Editor H. Arai and Editor A. De Luca upon evaluation of the reviewers' comments. This work was supported by the Humanoid Robotics Project, NEDO, Japan, and the CREST Program of the Japan Science and Technology Corporation. The work of K. Yamane was supported by the Japan Society for the Promotion of Science. This paper was presented at the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 24–28, 2000.

K. Yamane was with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA. He is now with the Department of Mechano-Informatics, University of Tokyo, Tokyo 113-8656, Japan (e-mail: kyamane@ynl.t.u-tokyo.ac.jp).

Y. Nakamura is with the Department of Mechano-Informatics, University of Tokyo, Tokyo 113-8656, Japan (e-mail: nakamura@ynl.t.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TRA.2003.810579

In this paper, we propose the concept of a *dynamics filter* and present an implementation with high interactivity. The basic function of the dynamics filter is to convert a physically inconsistent motion into a consistent one. Its main focus is in generating a motion sequence that is theoretically feasible for a given human figure. Any motion data may be input to the filter as a reference: human motion capture data, motions created by hand using animation software, or kinematically synthesized motions. The dynamics filter provides large flexibility in applying the existing motion data to a different model or environment.

The implementation of the dynamics filter is based on our previously proposed method for computing the dynamics of structure-varying kinematic chains [1]. The original method is extended to handle collisions and contacts between the robot and the environment. This extension is also described in detail because it has not been published yet. The method also serves as an efficient collision/contact model for dynamics simulation.

II. Related Work

Motion generation considering the dynamics of human figures has been discussed in both the humanoid robotics and computer animation fields. One of the major approaches is to describe the motion by a few parameters and optimize them using the ZMP constraint ([2]–[5]), inverted pendulum model [6], or learning [7]. The problems of such approaches are that the parameterization scheme depends on the motion and that the optimization usually runs offline due to the heavy computational load. Moreover, parameterization of a motion tends to yield an artificial, unfriendly motion. Some online controllers are proposed ([8]–[10]), but they focus on unexpected disturbances during the operation and only allow small differences between the ideal and actual environments and models.

Another approach is to use integrated control schemes to create various behaviors ([11]–[15]). A problem with this approach is that we need to prepare different controller for each behavior, and it is difficult to generate intermediate behaviors. [16] is an attempt to solve this problem.

Some researchers have tried to combine dynamics and motion capture data to generate feasible or realistic motions. For computer animation applications, Popović *et al.* [17] and Pollard *et al.* [18] used simplified human models to reduce the computational complexity of the dynamics. Zordan and Hodgins [19] built an online controller to track the upper-body human motion data that allows interactions with the environment. DasGupta and Nakamura [20] proposed a method to generate physically

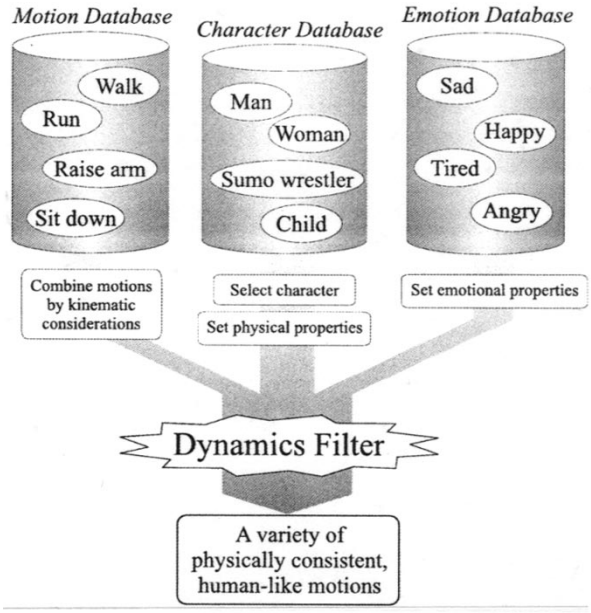

Fig. 1. A motion generator using a dynamics filter.

feasible walking motion by modifying the angles of waist joints with a Fourier series function such that the ZMP stays in the footprint. The computation is offline and their approach is limited to cyclic motions. Tak *et al.* [21] proposed an alternative method for the same purpose and their method can be applied to wider range of motions. However, their method is also an offline process.

III. Concept of a Dynamics Filter

Most of the existing methods for generating motions of human figures have problems from the standpoint of interactivity due to their poor flexibility, offline computation scheme, or long computation time. A dynamics filter is expected to provide a solution for these problems. An example of a motion generation system utilizing a dynamics filter is illustrated in Fig. 1. First, several properties, such as the motion (walk/run/sit, ...), the model (mass/link length, ...), character (male/female, adult/child, ...), and emotion (happy/angry/sad, ...) are selected and combined kinematically. We can employ various techniques for the kinematic synthesis of motions ([22]–[25]) at this stage. Next, the combined reference motion is input to the dynamics filter, which outputs a physically consistent motion, preferably close to the reference. Users may make some trial-and-error experiments with the dynamics filter to meet their taste. In interactive systems, the reference motion may change during the computation according to the user inputs.

Implementation of the dynamics filter may be offline or online. An offline filter could take advantage of knowing the whole sequence of the input motion in advance and would be able to generate the globally optimum motion. This type of dynamics filter would be useful for creating artistic films in computer graphics or a motion library for humanoid robots. An online dynamics filter, which this paper describes, faces a more difficult task because the available information is limited, but would be essential for real-time and interactive applications.

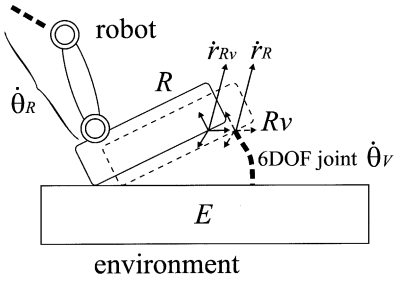

Fig. 2. Virtual link for contact computation. Original and virtual links are actually in the identical position.

It is also worth pointing out that the dynamics filter approach is reasonable from the viewpoint of the learning process of humans in the following sense: we first imitate just the *kinematics* of a motion watching how the others execute it, then adapt the motion to the *dynamics* of our own body and the environment by actually executing or practicing by ourselves. Applying an offline dynamics filter may correspond to practicing a difficult task many times. Simple tasks, on the other hand, may be achieved by a single trial, which in turn may be achieved by an online dynamics filter.

IV. Basic Equations

Sections IV–VI describe the collision/contact model used in our implementation of the dynamics filter. The model is an extension of the forward dynamics algorithm presented in [1] for structure-varying kinematic chains, which only handles *bilateral* constraints where the joints can constrain the links in any direction. Collisions and contacts, on the other hand, are subject to *unilateral* constraints where the constraint conditions change according to the direction of the motion. For example, the relative motion of two links in contact is constrained in the interpenetrating direction but unconstrained in the reverse direction.

This section extends the previous method to compute the constraint forces and the following section describes a fast iterative procedure to find the constraint conditions that satisfy the unilateral constraints. Section VI describes the method for computing the discontinuous change of the velocity due to a collision. The methods presented in these three sections enable fast simulation of collisions and contacts based on a rigid-body contact model. A simple reorganization of the equations derived here serves as the basis for the implementation of the dynamics filter as described in Section VII.

Suppose a link in a robot *Link R* is in contact with an environment link *Link E*. Following the strategy described in [1], we create a virtual link of *Link R*, named *Link Rv*, as a child link of *Link E* as illustrated in Fig. 2.

In rigid joint constraint, *Link Rv* is connected to *Link E* through a joint of a known type. In contact constraint, on the other hand, we do not know the constraint condition at this stage. We therefore place a 6-degrees-of-freedom (DOF) joint between *Link Rv* and *Link E* and impose an additional constraint on the joint velocity of the 6-DOF joint.

Let N_{DOF} denote the total DOF of the robot including the 6 DOF of the root link, $\dot{\boldsymbol{\theta}}_J \in \boldsymbol{R}^{N_{\mathrm{DOF}}}$ the vector composed of the joint velocities of the robot, $\dot{\boldsymbol{\theta}}_V \in \boldsymbol{R}^6$ the joint velocity of the 6-DOF joint between *Link Rv* and *Link E*, and $\dot{\boldsymbol{r}}_R \in \boldsymbol{R}^6$ and $\dot{\boldsymbol{r}}_{Rv} \in \boldsymbol{R}^6$ the spatial velocities of *Link R* and *Link Rv*, respectively. Again following our strategy, we virtually cut the joint between *Link Rv* and *Link E* to obtain a virtual open kinematic chain. The two sets of joint velocities, $\dot{\boldsymbol{\theta}}_A$ including those of the actuated joints, and $\dot{\boldsymbol{\theta}}_O$ including those of the virtual open kinematic chain, are written as follows, respectively:

$$\dot{\boldsymbol{\theta}}_O = \dot{\boldsymbol{\theta}}_J \tag{1}$$

$$\dot{\boldsymbol{\theta}}_A = \begin{pmatrix} \dot{\boldsymbol{\theta}}_V \\ \dot{\boldsymbol{\theta}}_J \end{pmatrix} \tag{2}$$

and the generalized coordinates $\boldsymbol{\theta}_G$ are yet to be selected.

$\dot{\boldsymbol{r}}_R$ and $\dot{\boldsymbol{r}}_{Rv}$ are described as

$$\dot{\boldsymbol{r}}_R = \boldsymbol{J}_R \dot{\boldsymbol{\theta}}_J \tag{3}$$

$$\dot{\boldsymbol{r}}_{Rv} = \boldsymbol{J}_V \dot{\boldsymbol{\theta}}_V \tag{4}$$

where $\boldsymbol{J}_R \in \boldsymbol{R}^{6\times N_{\mathrm{DOF}}}$ and $\boldsymbol{J}_V \in \boldsymbol{R}^{6\times 6}$ are the Jacobian matrix of $\boldsymbol{r}_R$ with respect to $\boldsymbol{\theta}_J$ and $\boldsymbol{r}_{Rv}$ with respect to $\boldsymbol{\theta}_V$, respectively, that is

$$\boldsymbol{J}_R = \frac{\partial \boldsymbol{r}_R}{\partial \boldsymbol{\theta}_J} \tag{5}$$

$$\boldsymbol{J}_V = \frac{\partial \boldsymbol{r}_{Rv}}{\partial \boldsymbol{\theta}_V}. \tag{6}$$

Because $\dot{\boldsymbol{r}}_{Rv}$ should be equal to $\dot{\boldsymbol{r}}_R$, the following equation holds:

$$(\boldsymbol{J}_V \ -\boldsymbol{J}_R)\begin{pmatrix} \dot{\boldsymbol{\theta}}_V \\ \dot{\boldsymbol{\theta}}_J \end{pmatrix} = O \tag{7}$$

where $\boldsymbol{O}$ denotes a zero matrix/vector. This equation corresponds to the closed-loop constraint in [1, eq. (9)]. We assume there are no other contacts between the robot and the environment for simplicity, but the equations can be extended to multiple contacts with only straightforward changes.

The next step is to extract six independent columns of the coefficient matrix of the left-hand side of (7) to select the generalized coordinates $\boldsymbol{\theta}_G$ and compute the two Jacobian matrices $\boldsymbol{W} \triangleq \partial\boldsymbol{\theta}_O/\partial\boldsymbol{\theta}_G$ and $\boldsymbol{S} \triangleq \partial\boldsymbol{\theta}_A/\partial\boldsymbol{\theta}_G$. $\boldsymbol{J}_V$ is usually a coordinate transformation from $\dot{\boldsymbol{\theta}}_V$ to $\dot{\boldsymbol{r}}_{Rv}$ because they represent the same physical values (linear and angular velocity of *Link Rv*) in two different coordinate systems. The columns of $\boldsymbol{J}_V$ are, therefore, independent of each other, which means that $\boldsymbol{\theta}_J$ contains the generalized coordinates of the system, and that the Jacobian matrix of dependent joints (in this case, $\boldsymbol{\theta}_V$) with respect to the generalized coordinates is computed by

$$\boldsymbol{H} = -\boldsymbol{J}_V^{-1}(-\boldsymbol{J}_R) = \boldsymbol{J}_V^{-1}\boldsymbol{J}_R \tag{8}$$

as described in [1, eq. (17)]. Therefore, $\boldsymbol{W}$ and $\boldsymbol{S}$ are formed as

$$\boldsymbol{W} = \boldsymbol{I} \tag{9}$$

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{H} \\ \boldsymbol{I} \end{pmatrix} \tag{10}$$

where $\boldsymbol{I}$ is the identity matrix of the appropriate size.

The fact that the joint values of the robot are the generalized coordinates means that we do not have to reselect the generalized coordinates and recompute the inertia matrix even if the constraint condition or the number of contacts changes. In addition, we can easily compute the inertial matrix in the generalized coordinate space because it is simply the joint space inertial matrix of the robot.

Now we derive the equations to compute the constraint forces. Let $\boldsymbol{A} \in \boldsymbol{R}^{N_{\mathrm{DOF}}\times N_{\mathrm{DOF}}}$ denote the inertial matrix in the generalized coordinate space and $\boldsymbol{b}$ denote the velocity product and gravity terms, both of which can be computed easily by applying well-known methods such as [26]. Let $\boldsymbol{\tau}_j \in \boldsymbol{R}^{N_{\mathrm{DOF}}}$ denote the vector composed of the joint torques of the robot and $\boldsymbol{\tau}_V \in \boldsymbol{R}^6$ denote the joint force/torque of the 6-DOF joint between *Link E* and *Link Rv*. $\boldsymbol{\tau}_V$ is essentially the constraint force and moment. Note that the elements of $\boldsymbol{\tau}_J$ corresponding to the six DOF of the root link are always zero. To make this point clear, we introduce another vector $\boldsymbol{\tau}_A \in \boldsymbol{R}^{N_{\mathrm{DOF}}-6}$ that includes only the torques of actuated joints, and another matrix $\boldsymbol{H}_J \in \boldsymbol{R}^{N_{\mathrm{DOF}}\times(N_{\mathrm{DOF}}-6)}$ that maps $\boldsymbol{\tau}_A$ to $\boldsymbol{\tau}_J$ as

$$\boldsymbol{\tau}_J = \boldsymbol{H}_J^T \boldsymbol{\tau}_A. \tag{11}$$

Using these notations, the generalized force acting on the system $\boldsymbol{\tau}_G \in \boldsymbol{R}^{N_J}$ is computed as

$$\begin{aligned} \boldsymbol{\tau}_G &= \boldsymbol{S}^T \begin{pmatrix} \boldsymbol{\tau}_V \\ \boldsymbol{\tau}_J \end{pmatrix} + \boldsymbol{f}_{\mathrm{ext}} \\ &= \boldsymbol{H}^T \boldsymbol{\tau}_V + \boldsymbol{H}_J^T \boldsymbol{\tau}_A + \boldsymbol{f}_{\mathrm{ext}} \end{aligned} \tag{12}$$

where $\boldsymbol{f}_{\mathrm{ext}}$ is the effect of known external forces.

The equation of the motion of the system is [27]

$$\boldsymbol{A}\ddot{\boldsymbol{\theta}}_J + \boldsymbol{b} = \boldsymbol{\tau}_G \tag{13}$$

because the generalized coordinates of the system are $\boldsymbol{\theta}_J$. On the other hand, the relationship between $\ddot{\boldsymbol{\theta}}_J$ and $\dot{\boldsymbol{\theta}}_V$ is described by

$$\ddot{\boldsymbol{\theta}}_V = \boldsymbol{H}\ddot{\boldsymbol{\theta}}_J + \dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_J. \tag{14}$$

If *Link R* is fixed to *Link E*, namely $\ddot{\boldsymbol{\theta}}_V = \boldsymbol{O}$, we obtain the following equation with the unknowns $\ddot{\boldsymbol{\theta}}_J$ and $\boldsymbol{\tau}_V$ from (12)–(14):

$$\begin{pmatrix} \boldsymbol{A} & -\boldsymbol{H}^T \\ \boldsymbol{H} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{\theta}}_J \\ \boldsymbol{\tau}_V \end{pmatrix} = \begin{pmatrix} \boldsymbol{H}_J^T \boldsymbol{\tau}_A + \boldsymbol{f}_{ext} - \boldsymbol{b} \\ -\dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_J \end{pmatrix}. \tag{15}$$

The coefficient matrix of the left-hand side of this equation is always square, and invertible if $\boldsymbol{J}_R$ has full row rank. We can, therefore, compute the joint accelerations $\ddot{\boldsymbol{\theta}}_J$ as well as the constraint force $\boldsymbol{\tau}_V$.

This equation is identical to the equation of motion of constrained systems derived in various methods ([28], [29]), except that the generalized coordinates and the inertial matrix are based on joint values. Also, we encounter the matrix $\boldsymbol{H}\boldsymbol{A}^{-1}\boldsymbol{H}^T$ while solving (15), which is identical to the operational space inertia matrix [30] in single contact case or the extended operational space inertia matrix [31] in multiple contact case.

If *Link R* and *Link E* are in contact, the constraint condition varies depending on the contact state. Suppose the constraints for the acceleration $\ddot{\boldsymbol{\theta}}_V$ are described as

$$\boldsymbol{K}_C\ddot{\boldsymbol{\theta}}_V = \boldsymbol{O}. \tag{16}$$

Using this notation, (15) becomes

$$\begin{pmatrix} \boldsymbol{A} & -\boldsymbol{H}_C^T \\ \boldsymbol{H}_C & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{\theta}}_J \\ \boldsymbol{\tau}_C \end{pmatrix} = \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{a} \end{pmatrix}$$

where

$$\begin{aligned} \boldsymbol{H}_C &\triangleq \boldsymbol{K}_C \boldsymbol{H} \\ \boldsymbol{\tau}_C &\triangleq \boldsymbol{K}_C \boldsymbol{\tau}_V \\ \boldsymbol{t} &\triangleq \bar{\boldsymbol{H}}_J^T \boldsymbol{\tau}_A + \boldsymbol{f}_{ext} - \boldsymbol{b} \\ \boldsymbol{a} &\triangleq -\dot{\boldsymbol{H}}_C \dot{\boldsymbol{\theta}}_J. \end{aligned} \tag{17}$$

Equation (17) is the general equation of motion of a human figure, including the constraints with the environment, and serves as the basic equation for collision and contact simulation.

V. Contact

A. Related Work

Simulation of collisions and contacts has been discussed for many years and a number of methods have been proposed. They are basically divided into two categories: penalty-based methods and analytical methods.

In penalty-based methods ([32], [33]), contact forces are generated by virtual springs and dampers at the contact points. These approaches are the most common ones in commercial software packages for dynamics simulation because of the easy implementation. However, the problem is that this approach requires extremely precise and time-consuming simulation due to the stiff system. It is also difficult to find parameters that yield realistic results.

Analytical methods compute the contact forces that satisfy the unilateral conditions using optimization techniques such as quadratic programming (QP) [34], linear complementarity problem (LCP) solvers [35], [36], or other simplified techniques [37]. These methods can produce relatively stable results with large sampling time, but solving optimization problems tends to be time consuming and requires a simplification of the problem. There are some other approaches such as impulse-based method [38], which is powerful in systems where bouncing occurs more frequently than stick contacts.

As a whole, efficient and precise simulation of collisions and contacts still remains an open research issue. One approach to overcome this problem is to make use of the nature of collisions and contacts in question to simplify the problem [37]. We develop an efficient method for collision/contact simulation taking advantage of the observation that most contacts in human motion do not produce bouncing.

Our method is basically an analytical approach but, instead of the mathematical optimization techniques, we apply a trial-and-error process to find the constraint condition and contact forces that satisfy the unilateral conditions. Collisions are modeled as discontinuous change of the joint velocities to enforce zero relative velocity between the contact pairs and to avoid impulsive accelerations on impact.

B. Overview

We can compute the contact force for a specific constraint condition by solving (17). In this section, we present a method to find the constraint condition and the constraint force that satisfy the unilateral conditions. Here we assume that the normal relative velocity is zero. If the links come into contact with nonzero normal relative velocity, it is set to zero by the collision computation described in the next section.

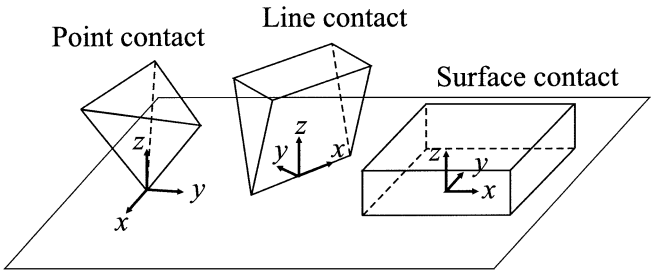

Fig. 3. Contact coordinates for each contact state.

The idea of the trial-and-error procedure is to assume a constraint condition and compute the constraint forces and moments to satisfy the constraints, and then check whether they satisfy the unilateral conditions. If they do not, we change the constraint condition and compute the constraint force again. The process is repeated until all the constraint forces and moments satisfy the unilateral conditions.

The whole procedure is summarized as follows.

1) Set all possible constraints at each contact pair.
2) Compute the constraint forces and moments by (17).
3) Check whether the constraint forces and moments satisfy the unilateral conditions as described in Section V-E.
4) If invalid constraint forces were found, modify the constraint as described in Section V-F and return to (2), otherwise proceed to (5).
5) Compute the joint accelerations using (17).

C. Contact Coordinate

We place the virtual link frame, or the contact coordinate frame, as illustrated in Fig. 3 for the simple representation of various constraints. The z axis is always identical to the normal vector of the contact surface. The other axes and the position are set as follows, depending on the contact state.

- Point contact: position is set to the contact point, and the direction of x and y axes are arbitrary.
- Line contact: the x axis is taken in the direction of the contact line, and the position is set to any point on the line
- Face contact: position is set to any point in the contact surface, and the directions of x and y axes are arbitrary.

Under this assumption, the constraint matrix $\boldsymbol{K}_C$ has a very simple structure composed of only 1's and 0's for all practical constraint conditions. For example, if a pair of links in line contact is fully constrained, $\boldsymbol{K}_C$ becomes

$$\boldsymbol{K}_C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

TABLE I
POSSIBLE CONSTRAINT CONDITIONS FOR EACH CONTACT STATE

ID	constraints						state		
	v_x	v_y	v_z	ω_x	ω_y	ω_z	face	line	point
C_0	◯	◯	◯	◯	◯	◯	◯	×	×
C_1	◯	◯	◯	×	◯	◯	×	◯	×
C_2	◯	◯	◯	×	×	◯	◯	◯	×
C_3	◯	◯	◯	×	×	×	×	×	◯
C_4	×	×	◯	◯	◯	×	◯	×	×
C_5	×	×	◯	×	◯	×	×	◯	×
C_6	×	×	◯	×	×	×	◯	◯	◯
C_7	×	×	×	×	×	×	◯	◯	◯

because only the rotation around x axis is unconstrained. If the links are slipping, $\boldsymbol{K}_C$ would be

$$\boldsymbol{K}_C = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{19}$$

because the motion in x and y axes is not constrained either.

In the following discussion, the spatial velocity of the contact coordinate and the constraint force/moment are expressed in the contact coordinate and denoted by vectors $(v_x v_y v_z \omega_x \omega_y \omega_z)^T$ and $(f_x f_y f_z n_x n_y n_z)^T$, respectively.

D. List of Constraint Conditions

Table I shows all the possible constraint conditions (sets of constrained directions) $C_0 - C_7$ and the feasibility of the constraint conditions in each contact state. The column *constraints* shows whether each direction is constrained (◯) or not (×) in each constraint condition, while the column *state* shows whether the constraint condition is feasible (◯) or not (×) for each contact type. In point contact, for example, only constraint conditions C_3 (position constrained), C_6 (only normal direction constrained) and C_7 (no constraint) are feasible, because point contacts cannot produce any moment.

Conditions C_1 and C_5, where ω_y is constrained but ω_x is not, are possible only at line contacts. Similarly, condition C_3, where v_x and v_y are constrained but ω_z is not, is possible at point contacts. The reason is that in face and line contacts, if ω_z is not constrained, namely, if the links are rotating around the normal vector, the total friction force is determined uniquely because all points in the contact area, except for the center of rotation, are slipping. Although it is possible to consider a situation where the slipping of the center of rotation is constrained, it is omitted for simplicity.

There are also limitation from the current relative velocity of the links in contact.

- If $v_x^2 + v_y^2 > 0$ or $\omega_z \neq 0$, directions in v_x, v_y and ω_z are not constrained. That is, if the links are slipping, the relative motion in the tangential plane is not constrained.
- If $\omega_x^2 + \omega_y^2 > 0$, directions in ω_x and ω_y are not constrained.

E. Constraint Force Validity Check

For all constraint forces and moments corresponding to the constrained directions, we execute the validity checks listed below. If any one of the checks fails, we immediately change the constraint condition as mentioned later.

1) Normal force
   The normal force f_z should be in the repulsive direction, namely, $f_z \geq 0$.
2) Center of pressure (COP)
   n_x and n_y should satisfy the condition that the COP, computed by $(n_y/f_z, -n_x/f_z, 0)$, is in the contact area. If not, the actual COP is set to the point in the contact area closest to the computed COP.
3) Friction
   The friction force should be smaller than the maximum static friction, namely
   $$\sqrt{f_x^2 + f_y^2} \leq \mu_S f_z \tag{20}$$
   where μ_S is the static friction coefficient. If the friction force exceeds this limit, the links in contact start slipping in the direction computed by
   $$\left(-\frac{f_x}{\sqrt{f_x^2 + f_y^2}} \quad -\frac{f_y}{\sqrt{f_x^2 + f_y^2}} \quad 0 \right)^T. \tag{21}$$
4) Twist moment
   When rotation around z axis is constrained, n_z is the sum of the moments due to the static friction forces distributed over the contact area. Upper and lower limits of n_z depend on the distribution of the normal forces, and cannot be determined only by the total normal force f_z. Before dealing with this problem, we first focus on the twist moment around the COP $\hat{n}_z$ when $v_x^2 + v_y^2 \neq 0$ or $w_z \neq 0$. We consider two extreme cases.
   1) When the links are slipping and not spinning, namely, $\sqrt{v_x^2 + v_y^2} \neq 0$ and $\omega_z = 0$, the friction forces are in the same direction and the net friction acts at the COP because each friction force is proportional to the normal force acting at each point. Therefore, the friction forces do not produce moment around z axis at COP, so $\hat{n}_z = 0$.
   2) When the links are spinning around the COP, namely, $\sqrt{\hat{v}_x^2 + \hat{v}_y^2}$ and $\hat{\omega}_z \neq 0$, where $\hat{v}_x$, $\hat{v}_y$, and $\hat{\omega}_z$ denote the velocities at COP, the friction forces act as concentric circles, so the torque around z axis is maximized. Although it is impossible to compute the torque exactly, we can approximate it by
      $$\hat{n}_z^{\max} = -\mathrm{sgn}(\omega_z) l \mu_D f_z \tag{22}$$
      where l is the dimension of the contact area and μ_D is the slip friction coefficient, and $\mathrm{sgn}(x)$ denotes the sign of x.

   Taking the above observations into account, we approximate $\hat{n}_z$ by the following function:
   $$\hat{n}_z = -\mathrm{sgn}(\omega_z) p(r) l \mu_D f_z \tag{23}$$
   where r is the distance between the COP and the center of rotation, and $p(x)(x \geq 0)$ is a function which returns

TABLE II
NEW CONSTRAINT CONDITION; THE NUMBER OF THE CHECK ITEMS COINCIDES WITH THE ITEM NUMBER OF THE LIST IN SECTION V-E

from	check failed			
	(1)	(2)	(3)	(4)
$C_{0(1)}$	$C_{2,4(5),6,7}$	$C_{2,4(5),6,7}$	$C_{4(5),2,6,7}$	$C_{4(5),2,6,7}$
C_2	$C_{4(5),6,7}$	—	$C_{4(5),6,7}$	$C_{4(5),6,7}$
C_3	$C_{6,7}$	—	$C_{6,7}$	—
$C_{4(5)}$	$C_{2,6,7}$	$C_{2,6,7}$	—	—
C_6	C_7	—	—	—

0 for $x = 0$ and converges to 1 as x increases. Through a similar method, we can estimate the maximum and minimum values of n_z when v_x, v_y, and ω_z are constrained. In this case, we have computed f_x, f_y, and n_z required to constrain the motion in the three axes. If the friction force exceeds its limit described in (20) and the tangential directions have turned out to be unconstrained, we do not have to check the validity of n_z because the rotation around z axis is not constrained either, as discussed in Section V-D. Otherwise, we first compute the ratio of the actual friction force with respect to the maximum friction force k_f by

$$k_f = \frac{\sqrt{f_x^2 + f_y^2}}{\mu_S f_z} \tag{24}$$

which is always between 0 and 1. Large k_f implies that the distributed friction forces directs almost the same directions. In the above discussion, we observed that in slipping without spinning, where the directions of slipping frictions are exactly the same, we have $n_z = 0$. Therefore, it can be estimated that if k_f is large, the absolute values of the limits for n_z will be small. Small k_f, on the other hand, implies the directions can vary, in which case n_z is maximized by applying static frictions tangential to concentric circles around the COP. For small k_f, therefore, the absolute values of the limits for n_z will be large. Finally, the maximum absolute value of n_z can be estimated by

$$n_z^{\max} = \mathrm{sgn}(\omega_z)(1 - k_f) l \mu_D f_z. \tag{25}$$

Note that $(1 - k_f)$ is used in place of $p(r)$ in (22). To check the validity of n_z, we first convert it to the moment around the COP n_z^{COP} by

$$n_z^{\mathrm{COP}} = n_z - c_y f_x + c_x f_y \tag{26}$$

where $(c_x\ c_y\ 0)^T$ is the position of COP, and then check if $|n_z^{\mathrm{COP}}| \leq n_z^{\max}$.

F. Transition Among Constraint Conditions

Table II shows the transition among the constraint conditions when invalid contact forces were found. The rows show the current constraint conditions, whereas the columns show the new constraint condition for each check. The constraint condition numbers in the brackets are used in place of those out of the brackets for line contact because those constraint condition are not feasible in line contact. "—" indicates that the check is not executed because the corresponding directions are not constrained. For example, in condition C_4, which means that the links are slipping, check on friction forces is not applicable because the friction forces are explicitly computed from f_z.

For a pair of the current constraint condition and the check failed, the new constraint condition candidates in the corresponding cell are evaluated from left. The conditions previously visited or inadequate for the contact state are ignored, and the first available condition becomes the next constraint condition.

Note that, for example, negative f_z does not immediately lead to constraint condition C_7, namely, no constraint. In some situations, even if we had negative f_z for a certain constraint condition, we might get positive f_z after removing constraints in (v_x, v_y, n_z) or (ω_x, ω_y) directions.

The computational load for recomputation of constraint forces after switching constraint condition is quite small.

1) Compute $\boldsymbol{H}_C$. Practically, the work needed here is just to select appropriate rows of $\boldsymbol{H}$.
2) Compute $\boldsymbol{H}_C \boldsymbol{A}^{-1} \boldsymbol{H}_C^T$ and its inverse. This is not time consuming either, because the matrix to be inverted is usually small, and we do not need to recompute $\boldsymbol{A}$ and $\boldsymbol{A}^{-1}$.
3) Compute $\boldsymbol{\tau}_C$.

G. Forces Depending on f_z

When the links are slipping, the v_x and v_y directions are not constrained, and the friction forces are computed explicitly by

$$f_x = -s_x \mu_D f_z$$
$$f_y = -s_y \mu_D f_z$$

where $\boldsymbol{s} = (s_x\ s_y\ 0)^T$ is the unit vector in the slipping direction. We have to put the effect of friction forces into the equation of motion (17). However, f_z, on which the friction forces depend, is computed by solving this equation itself and f_x and f_y are not included in $\boldsymbol{\tau}_C$.

Let us denote the rows of $\boldsymbol{H}$ corresponding to x, y, and z directions, namely, the first, second, and third rows, by $\boldsymbol{h}_x$, $\boldsymbol{h}_y$, and $\boldsymbol{h}_z$, respectively. Also denote the rest of $\boldsymbol{H}$ by $\boldsymbol{H}_n$ and the vector $(n_x\ n_y\ n_z)^T$ by $\boldsymbol{n}$. The contribution of contact forces to the generalized force is computed by

$$\begin{aligned}
\boldsymbol{H}^T f &= \begin{pmatrix} \boldsymbol{h}_x^T & \boldsymbol{h}_y^T & \boldsymbol{h}_z^T & \boldsymbol{H}_n^T \end{pmatrix} \begin{pmatrix} f_x \\ f_y \\ f_z \\ n \end{pmatrix} \\
&= \left(-s_x \mu_D \boldsymbol{h}_x^T - s_y \mu_D \boldsymbol{h}_y^T + \boldsymbol{h}_z^T \right) f_z + \boldsymbol{H}_n^T \boldsymbol{n} \\
&= \begin{pmatrix} -s_x \mu_D \boldsymbol{h}_x^T - s_y \mu_D \boldsymbol{h}_y^T + \boldsymbol{h}_z^T & \boldsymbol{H}_n^T \end{pmatrix} \begin{pmatrix} f_z \\ \boldsymbol{n} \end{pmatrix} \\
&= \hat{\boldsymbol{H}}_C^T \boldsymbol{\tau}_C
\end{aligned} \tag{27}$$

where

$$\hat{\boldsymbol{H}}_C \triangleq \begin{pmatrix} -s_x \mu_D \boldsymbol{h}_z - s_y \mu_D \boldsymbol{h}_y + \boldsymbol{h}_z \\ \boldsymbol{H}_n \end{pmatrix}. \tag{28}$$

Equation (27) means that adding the effect of slip friction to generalized force is equivalent to replacing $\boldsymbol{H}_C^T$ by $\hat{\boldsymbol{H}}_C^T$ in (17).

n_x, n_y, and n_z are also handled in similar manners. When ω_x and ω_y directions are not constrained and the COP are known

to be at $(p_x\ p_y\ 0)$ in the contact coordinate, n_x and n_y are computed by

$$n_x = \quad p_y f_z \tag{29}$$
$$n_y = -\,p_x f_z. \tag{30}$$

When ω_z is not constrained, n_z is computed by (23).

Let $h_{nu}(u = x, y, z)$ denote the rows of $\boldsymbol{H}$ corresponding to ω_u. If each direction is not constrained, the row of $\hat{\boldsymbol{H}}_C$ corresponding to v_z is increased by the following vectors:

$$\begin{aligned} \omega_x &: \ p_y \boldsymbol{h}_{nx} \\ \omega_y &: \ -p_x \boldsymbol{h}_{ny} \\ \omega_z &: \ -sgn(\omega_z) p(\tau) l \mu_D \boldsymbol{h}_{nz}. \end{aligned}$$

H. Limitations

The method has two major limitations as the drawbacks of the simplifications.

- Multiple Contacts

  The problem of handling multiple contacts is the coupling between the contacts. We basically reduce the number of constraints if invalid constraint forces are found, and each contact pair never revisits the constraint conditions that have been already checked. This scheme enables us to obtain the feasible constraint forces with $O(mn)$ computations, where m denotes the number of pairs in contact and n the number of possible contact conditions for each pair, while checking all possible sets of constraint conditions requires $O(n^m)$ computations. The problem is that, however, it might happen that the most desirable solution is never tried in the course of the trial-and-error process. This failure could lead to negative normal acceleration (or velocity in collisions) of the contact coordinate, in which case, the links would interpenetrate. It is difficult to prove that this situation never occurs in general cases. However, we have never observed unrealistic behaviors so far in a number of simulations. Even if interpenetrations occur, they can be recovered by giving a nonzero value to $\ddot{\boldsymbol{r}}_{Rv}$ in (14).
- Varying Normal Vectors

  We cannot completely handle situations where the normal vectors of the contact surface are not uniform in the contact area, because the validity check is conducted against the sum of the contact forces at all contact points described in the contact coordinate. Summing up the contact forces into a single set of forces and moments eliminates the direction information of the original contact forces, which is essential for the validity check because the conditions are completely different for each direction. A solution for this problem would be to set multiple contact points for one contact pair, in which case, we then encounter the problem of indeterminate contact forces.

VI. Collision

The method for simulating collisions is derived by applying the same extension to the method for computing the discontinuous changes of joint velocities using Newton's Impact Law and conservation of momentum [1].

The conservation of momentum is described as

$$\boldsymbol{H}_C^T \boldsymbol{F}_C = -\boldsymbol{A} \Delta \dot{\boldsymbol{\theta}}_J \tag{31}$$

where $\boldsymbol{F}_C$ is the impulse and $\Delta \dot{\boldsymbol{\theta}}_J$ is the change of joint velocities. The kinematic constraint after the collision is described as

$$\boldsymbol{H}_C(\dot{\boldsymbol{\theta}}_J + \Delta \dot{\boldsymbol{\theta}}_J) = \boldsymbol{v}_C \tag{32}$$

where $\boldsymbol{v}_C$ is the velocity of the contact coordinate after the collision.

$\boldsymbol{v}_C$ is computed using Newton's Impact Law. Newton's Impact Law describes the relationship of the normal relative velocities before and after the collision. In order to handle the impulses in tangential and rotational directions, we extend the discussion to all constrained directions. The modified impact law is written as

$$\boldsymbol{v}_C = -\boldsymbol{e}\boldsymbol{v}_C 0 \tag{33}$$

where $\boldsymbol{v}_C 0$ is the vector of relative velocities of constrained directions before the collision, and $\boldsymbol{e}$ is a diagonal matrix containing the coefficients of restitution in the constrained directions.

Based on the observation that most contacts in human motions are inelastic, the element of $\boldsymbol{e}$ corresponding to the normal direction is set to 0, namely, the links stay in contact after the collision, although it may be set to nonzero values in other situations. If the tangential directions v_x and v_y and the rotational directions ω_x, ω_y and ω_z are constrained, we set the corresponding elements of $\boldsymbol{e}$ to 0, which means that the links stop slipping or rotating if the tangential or rotational impulses were large enough. Therefore, in contrast to contact, even if the links had tangential or angular velocities before a collision, we start by setting all constraints possible for the contact state shown in Table I.

The validity check is conducted against $\boldsymbol{F}_C$ using the same method as described in Section V-E, although this is not an exact model. For example, a positive normal impulse only guarantees that the integrated normal force during the collision phase is positive. We might have negative contact force during the collision, in which case, the links would separate and the actual normal impact force become smaller than the computed one. However, we ignore these special cases and execute the same validity check as during contact.

VII. Dynamics Filter Implementation

A. Basic Equation

Equation (17) gives the unique solution for the joint accelerations and the constraint forces when the constraint condition and the actuator torques $\boldsymbol{\tau}_A$ are known, which is the case for dynamics simulation. Our purpose in the dynamics filter, on the other hand, is to generate the motion itself based on some optimization scheme while the actuator torques are initially unknown. Moreover, we need an equation with some redundancy

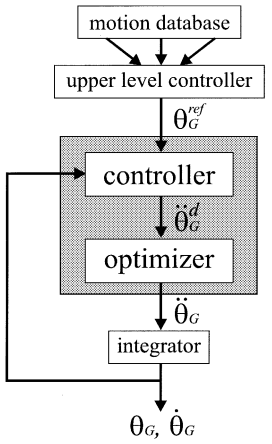

Fig. 4. Structure of the dynamics filter.

for optimization. Taking these facts into account, we modify (17) for the dynamics filter as

$$\begin{pmatrix} \boldsymbol{A} & -\boldsymbol{H}_C^T & -\boldsymbol{H}_J^T \\ \boldsymbol{H}_C & \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{\theta}} \\ \boldsymbol{\tau}_C \\ \boldsymbol{\tau}_J \end{pmatrix} = \begin{pmatrix} -\boldsymbol{b} \\ -\dot{\boldsymbol{H}}_C \dot{\boldsymbol{\theta}}_G \end{pmatrix} \tag{34}$$

or in a simpler form

$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{u} \tag{35}$$

where

$$\boldsymbol{W} \triangleq \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{H}_C^T & -\boldsymbol{H}_C^T \\ \boldsymbol{H}_C & \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \tag{36}$$

$$\boldsymbol{x} \triangleq \begin{pmatrix} \dot{\boldsymbol{\theta}}_C \\ \boldsymbol{\tau}_C \\ \boldsymbol{\tau}_J \end{pmatrix} \tag{37}$$

$$\boldsymbol{u} \triangleq \begin{pmatrix} -\boldsymbol{b} \\ -\dot{\boldsymbol{H}}_C \dot{\boldsymbol{\theta}}_G \end{pmatrix}. \tag{38}$$

Equation (34) is a redundant linear equation in $(\dot{\boldsymbol{\theta}}_J\, \boldsymbol{\tau}_C\, \boldsymbol{\tau}_J)^T$, whose solutions represents all the physically feasible motions under the given set of constraints. The role of the dynamics filter is to select the most appropriate solution of (34), taking various factors into account. As described in Section V, if $\boldsymbol{\tau}_C$ in the solution does not satisfy the unilateral conditions for the contact forces, we change the constraint condition and recompute a new solution.

B. Outline

Fig. 4 shows the structure of the dynamics filter and its typical application, where the shadowed box represents the main part of the dynamics filter. The filter we developed consists of two parts, controller and optimizer. The controller part computes the desired (but not always feasible) joint accelerations considering the reference motion and the current state. This part itself consists of two feedback sections, local and global. The local feedback section simply computes the temporary desired accelerations by the local feedback controller at each joint, which are modified by the global feedback section to maintain the whole-body configuration. Given the desired joint accelerations, the optimization part then computes the solution of (34) that minimizes the error between the actual and desired joint accelerations. Both the controller and optimizer parts are designed to require only the current state and reference motion to enable online filtering.

C. Details

1) Controller: First, the local feedback section computes the temporary desired acceleration of the generalized coordinates $\ddot{\boldsymbol{\theta}}_G^{d0}$ by simple joint angle and velocity feedback

$$\ddot{\boldsymbol{\theta}}_G^{d0} = \ddot{\boldsymbol{\theta}}_G^{ref} + \boldsymbol{K}_D \left(\dot{\boldsymbol{\theta}}_G^{ref} - \dot{\boldsymbol{\theta}}_G\right) + \boldsymbol{K}_P \left(\boldsymbol{\theta}_G^{ref} - \boldsymbol{\theta}_G\right) \tag{39}$$

where $\boldsymbol{\theta}_G^{ref}$ is the generalized coordinates in reference motion, and $\boldsymbol{K}_D$ and $\boldsymbol{K}_P$ are gain matrices.

Then, in order to influence the configuration of the whole body, the global feedback section modifies the desired acceleration to feedback the position and orientation of a specified point $\boldsymbol{P}$ in the upper body. The desired acceleration of $\boldsymbol{P}$, $\ddot{\boldsymbol{\theta}}_P^d$, is computed by a similar feedback law as

$$\ddot{\boldsymbol{r}}_P^d = \ddot{\boldsymbol{r}}_P^{\mathrm{ref}} + \boldsymbol{K}_{DP} \left(\dot{\boldsymbol{r}}_P^{ref} - \dot{\boldsymbol{r}}_P\right) + \boldsymbol{K}_{PP} \left(\boldsymbol{r}_P^{\mathrm{ref}} - \boldsymbol{r}_P\right) \tag{40}$$

where $\boldsymbol{r}_P^{\mathrm{ref}}$ is the position and orientation of $\boldsymbol{P}$ in the reference motion, which can be obtained by the forward kinematics computation, $\boldsymbol{K}_{DP}$ and $\boldsymbol{K}_{PP}$ are gain matrices, and $\boldsymbol{r}_P$ is the current position and orientation of $\boldsymbol{P}$. The temporary desired acceleration of the generalized coordinates $\ddot{\boldsymbol{\theta}}_G^{d0}$ is modified into $\ddot{\boldsymbol{\theta}}_G^d$ so that the desired acceleration of $\boldsymbol{P}$, $\ddot{\boldsymbol{r}}_P^d$, is realized by

$$\ddot{\boldsymbol{\theta}}_G^d = \ddot{\boldsymbol{\theta}}_G^{d0} + \Delta\ddot{\boldsymbol{\theta}}_G^d \tag{41}$$

$$\Delta\ddot{\boldsymbol{\theta}}_G^d = \boldsymbol{J}_P^\sharp \left(\ddot{\boldsymbol{r}}_P^d - \ddot{\boldsymbol{r}}_P^{d0}\right) \tag{42}$$

where $\ddot{\boldsymbol{r}}_P^{d0} \triangleq \boldsymbol{J}_P \ddot{\theta}_G^{d0} + \dot{\boldsymbol{J}}_P \dot{\theta}_G$, $\boldsymbol{J}_P \triangleq \partial \boldsymbol{J}_P / \partial \boldsymbol{\theta}_G$, and $\boldsymbol{J}_P^\sharp$ is the weighted pseudoinverse of $\boldsymbol{J}_P$.

2) Optimization: Solutions of (34) represent all the feasible combinations of $\ddot{\boldsymbol{\theta}}_G$, τ_C, and τ_J. The task of the optimization part is to find the optimal solution of (34) where the generalized accelerations become as close as possible to the desired accelerations. The optimized accelerations are integrated to derive the joint angle data.

First, we derive the weighted least-square solution of (35) and the null space of $\boldsymbol{W}$, regardless of the desired acceleration, by

$$\boldsymbol{x} = \boldsymbol{W}^\sharp \boldsymbol{u} + (\boldsymbol{E} - \boldsymbol{W}^\sharp \boldsymbol{W})\boldsymbol{y} \tag{43}$$

where $\boldsymbol{W}^\sharp$ is the pseudoinverse of $\boldsymbol{W}$, $\boldsymbol{y}$ is an arbitrary vector, and $\boldsymbol{E}$ is the identity matrix of the appropriate size. Taking the upper rows of (43) corresponding to the generalized accelerations, we obtain

$$\ddot{\boldsymbol{\theta}}_G = \ddot{\boldsymbol{\theta}}_G^0 + \boldsymbol{V}_G \boldsymbol{y} \tag{44}$$

where $\ddot{\boldsymbol{\theta}}_G^0$ is the generalized acceleration in the least-square solution of (43).

Next, we determine the arbitrary vector $\boldsymbol{y}$ to minimize the acceleration error by

$$\boldsymbol{y} = \boldsymbol{V}_G^* \left(\ddot{\boldsymbol{\theta}}_G^d - \ddot{\boldsymbol{\theta}}_G^0\right) \tag{45}$$

where $\boldsymbol{V}_G^*$ is the singularity-robust inverse [39] of $\boldsymbol{V}_G$.

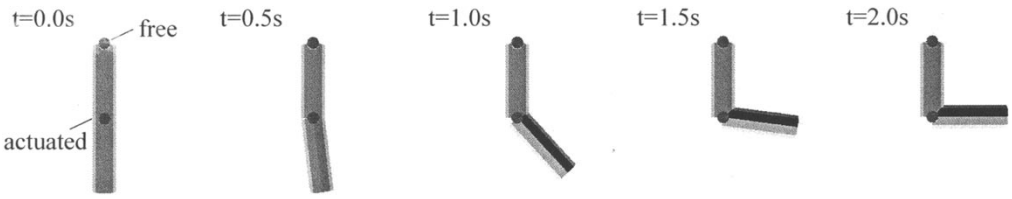

Fig. 5. Reference motion for the 2-DOF arm.

Fig. 6. Output of the dynamics filter for the 2-DOF arm example.

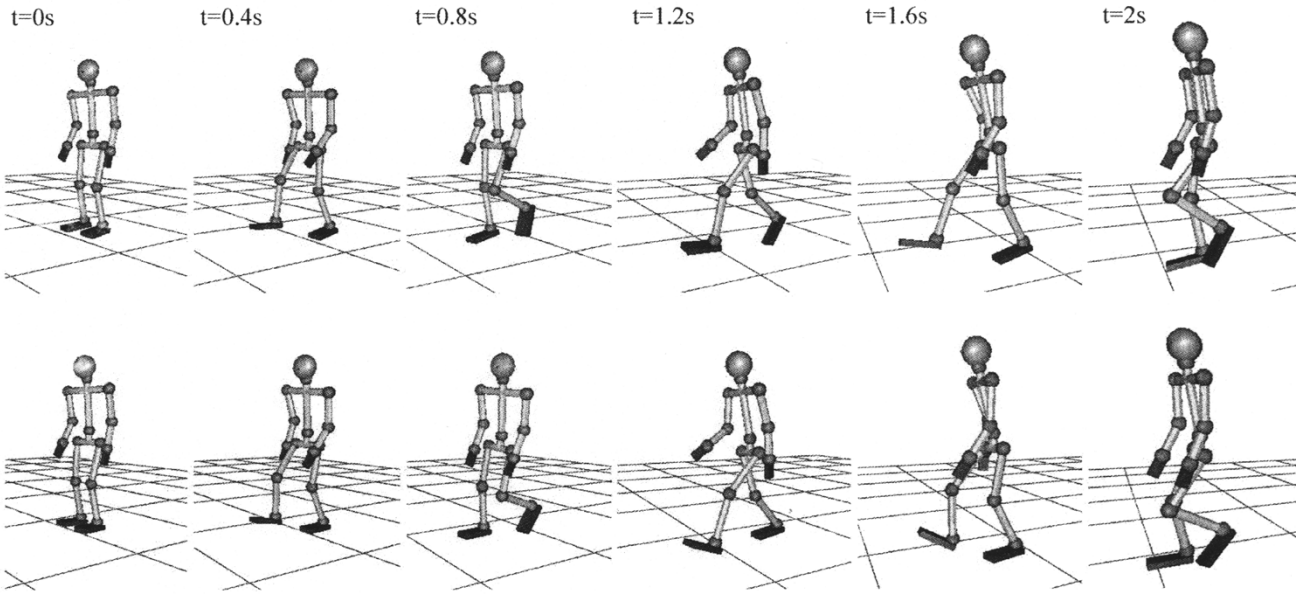

Fig. 7. Captured (above) and filtered (below) walking motions.

Finally, substituting $\boldsymbol{y}$ into (43), we get the optimized solution of $\boldsymbol{x}$. Because $\boldsymbol{x}$ includes the generalized acceleration, joint torques, and constraint forces, the optimization part plays three roles at the same time: 1) computation of optimized motion; 2) computation of joint torques to realize the computed acceleration; and 3) dynamics simulation of the result.

D. Applications

We used a 28-DOF skeleton model (7-DOF legs, 4-DOF arms, a 3-DOF waist joint, and a 3-DOF neck joint) in the following examples, except for the 2-DOF arm in the first example. In all the examples, the physically consistent accelerations were computed every 2 ms. The current implementation of the filter takes 70 to 80 ms per frame on an Alpha 21 264 500 MHz processor for the 28-DOF model.

The additional control point $\boldsymbol{P}$ was taken at the neck and its position and orientation were computed offline, although online

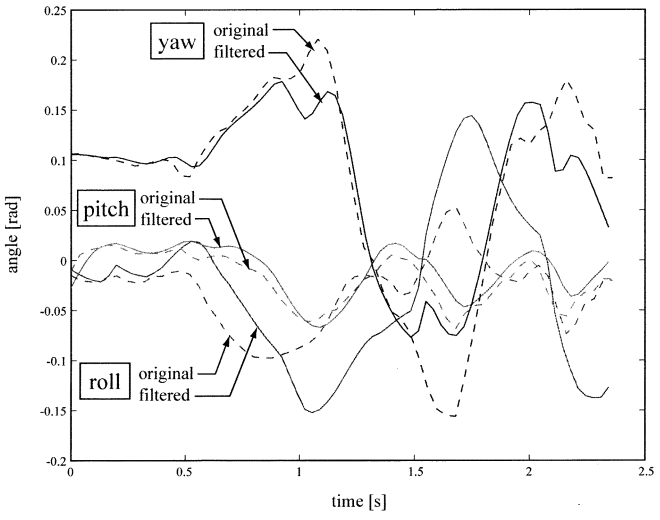

Fig. 8. Numerical comparison of roll, pitch, and yaw angles of the body.

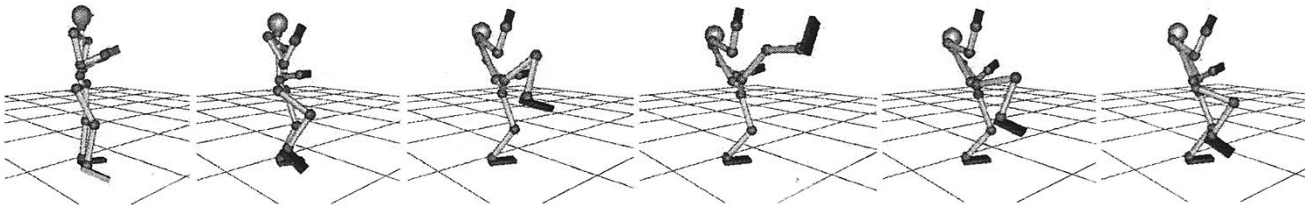

Fig. 9. Karate kick generated by the dynamics filter.

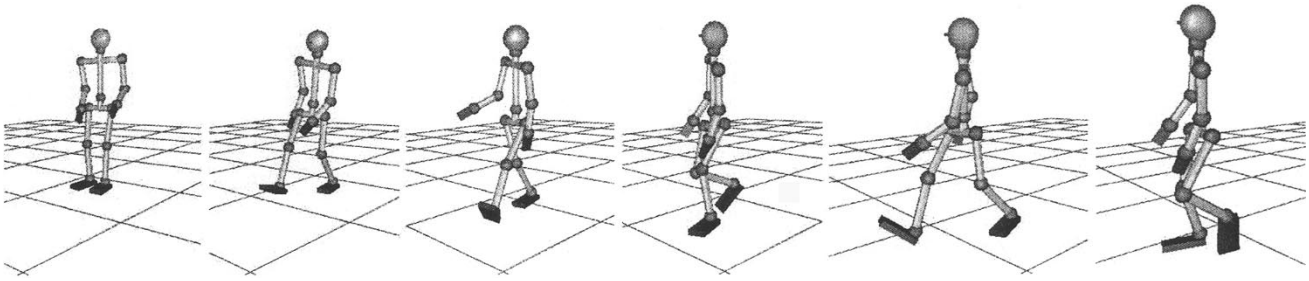

Fig. 10. Walk on a down slope.

computation could be realized by an easy improvement of the implementation. In the examples with motion capture data, the original data was taken 60 frames per second and interpolated to be used as the reference joint position, velocity, and accelerations.

Video clips of the motions are also available online at http://www.ynl.t.u-tokyo.ac.jp/publications/archives/video/.

2-DOF Arm: We applied the dynamics filter to a simple 2-DOF underactuated arm to demonstrate the function of the dynamics filter. The root joint of the arm is free and the other joint is actuated. We selected this mechanism because the dynamics filter is effective for underactuated mechanisms whose motion may not be dynamically feasible with any choice of actuator inputs. We used the reference motion in Fig. 5 and obtained the output shown in Fig. 6. The reference motion was physically inconsistent because of the free root joint, but it was corrected by the dynamics filter by changing the trajectory of the root joint.

Filtering Raw Motion Capture Data: Fig. 7 compares the captured (above) and filtered (below) walking motions. The roll, pitch, and yaw angles of the body of the original and filtered motions are compared in Fig. 8.

This method is applicable to any motion as shown in Fig. 9, which implies that we do not need to prepare different filters for different motions.

Filtering into a Different Environment: We applied the same walking motion as in the previous example to a down slope. The result is shown in Fig. 10. No modification on captured data was made except for the position of the body and the neck, which was modified to maintain the same height from the ground as in the original.

Filtering Kinematically Synthesized Motion: The dynamics filter accepts not only raw captured data but also kinematically synthesized motions. A walking motion with a 30° turn was generated (Fig. 11) from the reference motion obtained by smoothly connecting two walking motions heading in different directions. The reference motion does not take into account any dynamic effect such as centrifugal forces.

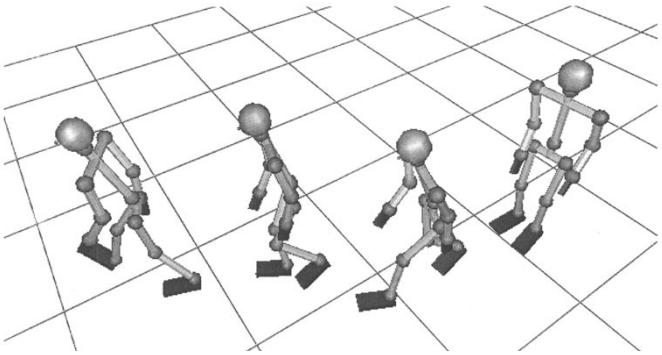

Fig. 11. Motion generated from a kinematically combined reference motion.

Interactive Motion Generation: Fig. 12 demonstrates the interactivity of our approach, where the figure is controlled to keep standing by the dynamics filter and reacts to the force applied by the user.

E. Discussion

The original and filtered walking motions showed good correspondence in the motions presented here, although we observed small latencies in some DOFs such as the roll angle of the body (Fig. 8). It turned out that the actual accelerations of those DOFs were much more different from the desired ones than the others, probably because they had to be modified by the optimizer to compensate for the physical consistency or the difference in the environment. Their angles will not follow the original trajectory until the feedback terms in (39) become large enough. In fact, the width of the motion of the roll angle of the body is much larger in the filtered motion than in the original one.

The examples in Figs. 10 and 11 showed the possibility of applying the dynamics filter to adapt motion capture data to the environment different from the one where the motion was actually performed, and to synthesize physically consistent motions from a motion capture data library. In these examples, we could generate valuable motions by quite simple kinematic modifications of the original walking motion. For more practical appli-

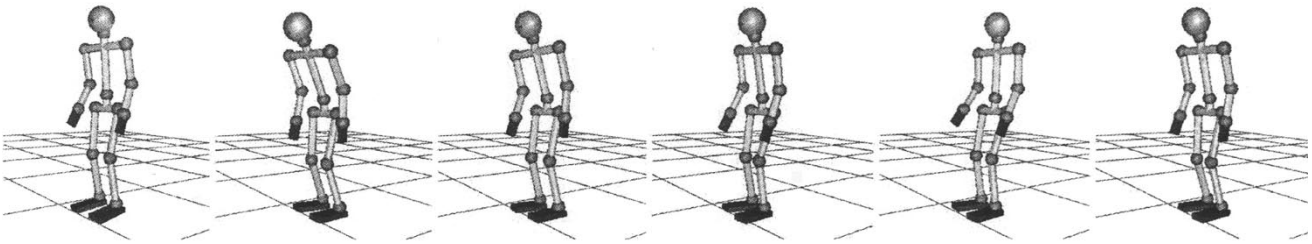

Fig. 12. Example of interactive motion generation: push a standing figure.

cations, however, careful kinematic preprocessing would be required to adjust the feet placements and constraints.

The problem of the dynamics filter is that it is very difficult to tune the parameters (feedback gains, weights for weighted pseudoinverses) and we need to find a set of parameters for each behavior. We will then encounter a problem in generating intermediate motions just as in using task-specified controllers. The encouraging fact, however, is that it is rather easier to interpolate the parameters than to interpolate the controllers to get intermediate behaviors.

Unfortunately, we could not find a systematic method for tuning the parameters. If the reference motion is physically consistent, larger gains would work because the figure only has to follow the reference motion as precisely as possible. Otherwise, we have to select the appropriate gains according to the importance of the trajectory of each joint in order not to overfit the joints to the original trajectory and to allow flexible interactions. We would also have to use smaller weights for less important joints so that the SR-inverse can allow enough deviation from the desired accelerations to compensate for the physical inconsistency.

The main contribution of the dynamics filter is that it provides a general framework for online generation of physically consistent motions using an efficient algorithm for dynamics simulation of human figures. In our current implementation, it is not guaranteed that the resulting motion would always become close to the reference. The result will be far from the original if we give extremely inconsistent motions as reference. A possible solution to this problem is to apply some task-specific controllers so that the figure will not fall down. For the standing human figure in Fig. 12, for example, we could design a controller that allows the human figure to step out if the force is too large to resist. Designing controllers for robotics systems is a common problem in various field in robotics, and a number of controllers are proposed. In the current implementation of the dynamics filter, we adopted a simple feedback controller consisting of a feedback loop in joint space and another in the Cartesian space of a selected link. The results presented in this paper proved that, even with the simple controllers we used, the dynamics filter can generate physically consistent motions on line using reference motions from various sources. It should also be possible to apply more sophisticated controllers proposed by many researchers, which will be included in our future work.

VIII. Conclusion

The conclusions of this paper are summarized as follows.

1) The concept of a dynamics filter is proposed. A dynamics filter is a motion generator that creates a physically consistent motion from any reference motion that may be physically infeasible for the model.
2) A method for simulating collisions and contact was presented. The method is based on our previous work on structure-varying kinematic chains, which was extended to handle unilateral constraints by a fast trial-and-error procedure.
3) The implementation of an online dynamics filter was described and proved the potential ability of the dynamics filter in interactive motion generation by a number of motions generated from motion capture data.

References

[1] Y. Nakamura and K. Yamane, "Dynamics computation of structure-varying kinematic chains and its application to human figures," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 124–134, Apr. 2000.

[2] J. Yamaguchi, A. Takanishi, and I. Kato, "Development of a biped walking robot compensating for three-axis moment by trunk motion," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robotics and Systems*, 1993, pp. 561–566.

[3] J. Park and Y. Rhee, "ZMP trajectory generation for reduced trunk motions of biped robots," in *Proc. 1998 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, vol. 1, 1998, pp. 90–95.

[4] Q. Huang, S. Kajita, N. Koyachi, K. Kaneko, K. Yokoi, H. Arai, K. Komoriya, and K. Tanie, "A high stability, smooth walking pattern for a biped robot," in *Proc. Int. Conf. Robotics and Automation*, 1999, pp. 65–71.

[5] S. Kagami, F. Kanehiro, Y. Tamiya, M. Inaba, and H. Inoue, "Autobalancer: an online dynamic balance compensation scheme for humanoid robots," in *Proc. Int. Workshop Algorithmic Foundation of Robotics*, Hanover, NH, Mar. 2000, pp. 329–340.

[6] J. Park and K. Kim, "Biped robot walking using gravity-compensated inverted pendulum mode and computed torque control," in *Proc. IEEE Int. Conf. Robotics and Automation*, Leuven, Belgium, May 1998, pp. 3528–3533.

[7] C. Chew and G. Pratt, "A general control architecture for dynamic bipedal walking," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 3990–3996.

[8] J. Hu, J. Pratt, and G. Pratt, "Adaptive dynamic control of a biped walking robot with radial basis function neural networks," in *Proc. Int. Conf. Robotics and Automation*, 1998, pp. 400–405.

[9] Y. Fujimoto, S. Obata, and A. Kawamura, "Robust biped walking with active interaction control between foot and ground," in *Proc. Int. Conf. Robotics and Automation*, 1998, pp. 2030–2035.

[10] Q. Huang, K. Kaneko, K. Yokoi, S. Kajita, and T. Kotoku, "Balance control of a biped robot combining offline pattern with real-time modification," in *Proc. Int. Conf. Robotics and Automation*, 2000, pp. 3346–3352.

[11] M. van de Panne, "Toward agile animated characters," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 682–687.

[12] W. Wooten and J. Hodgins, "Animation of human diving," *Computer Graphics Forum*, vol. 15, no. 1, pp. 3–13, 1996.

[13] D. Brogan, R. Metoyer, and J. Hodgins, "Dynamically simulated characters in virtual environments," *IEEE Comput. Graph. Appl.*, vol. 18, pp. 58–69, May 1998.

[14] W. Wooten and J. Hodgins, "Simulating leaping, tumbling, landing, and balancing humans," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 656–662.
[15] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien, "Animating human athletics," in *Proc. ACM SIGGRAPH '95*, 1995, pp. 71–78.
[16] A. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *Proc. SIGGRAPH 2001*, Los Angeles, CA, Aug. 2001, pp. 251–260.
[17] Z. Popović, "Editing dynamic properties of captured human motion," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 670–675.
[18] N. Pollard and F. Behmaram-Mosavat, "Force-based motion editing for locomotion tasks," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 663–669.
[19] V. Zordan and J. Hodgins, "Motion capture-driven simulations that hit and react," in *Proc. ACM SIGGRAPH Symp. Computer Animation*, San Antonio, TX, July 2002, pp. 89–96.
[20] A. DasGupta and Y. Nakamura, "Making feasible walking motion of humanoid robots from human motion captured data," in *Proc. Int. Conf. Robotics and Automation*, 1999, pp. 1044–1049.
[21] S. Tak, O. Song, and H. Ko, "Motion balance filtering," *Eurographics 2000, Computer Graphics Forum*, vol. 19, no. 3, pp. 437–446, 2000.
[22] M. Gleicher, "Retargetting motion to new characters," in *Proc. SIGGRAPH '98*, 1998, pp. 33–42.
[23] J. Lee and S. Shin, "A hierarchical approach to interactive motion editing for human-like figures," in *Proc. SIGGRAPH'99*, 1999, pp. 39–48.
[24] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and adverbs: multidimentional motion interpolation," *IEEE Comput. Graph. Appl.*, vol. 18, pp. 32–40, May 1998.
[25] C. Rose, B. Guenter, B. Bodenheimer, and M. Cohen, "Efficient generation of motion transitions using spacetime constraints," in *Proc. SIGGRAPH'96*, 1996, pp. 147–154.
[26] M. Walker and D. Orin, "Efficient dynamic computer simulation of robot manipulators," *ASME J. Dynam. Syst., Meas., Control*, vol. 104, pp. 205–211, 1982.
[27] Y. Nakamura and M. Ghodoussi, "Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 294–302, June 1989.
[28] D. Baraff, "Linear-time dynamics using Lagrange multipliers," in *Proc. SIGGRAPH*, 1996, pp. 137–146.
[29] C. Lubich, U. Nowak, U. Poehle, and C. Engstler. (1992) MEXX—Numerical software for the integration of constrained mechanical multibody systems. [Online]. Available: ftp://na.uni-tuebingen.de/pub/engstler/mexx/SC-92-12.ps.gz
[30] K. Lilly and D. Orin, "Efficient O(n) recursive computation of the operational space inertia matrix," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 1384–1391, Sept./Oct. 1993.
[31] K. Chang and O. Khatib, "Efficient algorithm for extended operational space inertia matrix," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1999, pp. 350–355.
[32] D. Marhefka and D. Orin, "Simulation of contact using a nonlinear damping model," in *Proc. 1996 IEEE Int. Conf. Robotics and Automation*, 1996, pp. 1662–1668.
[33] K. Hunt and F. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *ASME J. Appl. Mech.*, vol. 42, pp. 440–445, June 1975.
[34] P. Lötstedt, "Numerical simulation of time-dependent contact friction problems in rigid body mechanics," *SIAM J. Sci. Statist. Comput.*, vol. 5, no. 2, pp. 370–393, 1984.
[35] F. Pfeiffer and C. Glocker, *Multibody Dynamics With Unilateral Contacts*. New York: Wiley, 1996.
[36] D. Stewart and J. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with Coulomb friction," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, May 2000, pp. 162–169.
[37] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proc. SIGGRAPH '94*, Orlando, FL, July 1994, pp. 23–34.
[38] B. V. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, Univ. of California, Berkeley, CA, 1996.
[39] Y. Nakamura and H. Hanafusa, "Inverse kinematics solutions with singularity robustness for robot manipulator control," *J. Dynam. Syst., Meas., Contr.*, vol. 108, pp. 163–171, 1986.

Katsu Yamane (M'00) received the B.S., M.S., and Ph.D. degrees in mechanical engineering from University of Tokyo, Tokyo, Japan, in 1997, 1999, and 2002, respectively.

He was a Research Fellow of the Japan Society for the Promotion of Science from 2000 to 2002, and a Postdoctoral Fellow at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA from 2002 to 2003. Since 2003, he has been an Assistant Professor at the Department of Mechano-Informatics, University of Tokyo. His research interests include dynamics simulation of human figures, control of humanoid robots, and physically-based animation in computer graphics.

Dr. Yamane received an Excellent Paper Award from the Robotics Society of Japan in 2000 and King-Sun Fu Memorial Best Transactions Paper Award, IEEE TRANSACTIONS OF ROBOTICS AND AUTOMATION in 2001. He is a member of the Robotics Society of Japan and ACM SIGGRAPH.

Yoshihiko Nakamura (M'87) received the B.S., M.S., and Ph.D. degrees from Kyoto University, Kyoto, Japan, in precision engineering in 1977, 1978, and 1985, respectively.

He was an Assistant Professor at the Automation Research Laboratory, Kyoto University, from 1982 to 1987. He joined the Department of Mechanical and Environmental Engineering, University of California, Santa Barbara, in 1987 as an Assistant Professor, and became an Associate Professor in 1990. Since 1991, he has been with Department of Mechano-Informatics, University of Tokyo, Tokyo, Japan, and is currently a Professor. His fields of research include redundancy in robotic mechanisms, nonholonomy of robotic mechanisms, kinematics and dynamics algorithms for computer graphics, nonlinear dynamics and brain-like information processing, and robotic systems for medical applications. He is currently the Principal Investigator of "Brain-like Information Processing for Humanoid Robots" under the CREST project of the Japan Science and Technology Corporation. He is the author of the textbook, *Advanced Robotics: Redundancy and Optimization* (Reading, MA: Addison-Wesley).

Dr. Nakamura received Excellent Paper Awards from the Society of Instrument and Control Engineers (SICE) in 1985 and from the Robotics Society of Japan in 1996 and 2000. He was also a recipient of King-Sun Fu Memorial Best Transactions Paper Award, IEEE TRANSACTION OF ROBOTICS AND AUTOMATION in 2001 and 2002. He is a member of the ASME, the SICE, the Robotics Society of Japan, the Japan Society of Mechanical Engineers, the Institute of Systems, Control, and Information Engineers, and the Japan Society of Computer-Aided Surgery.