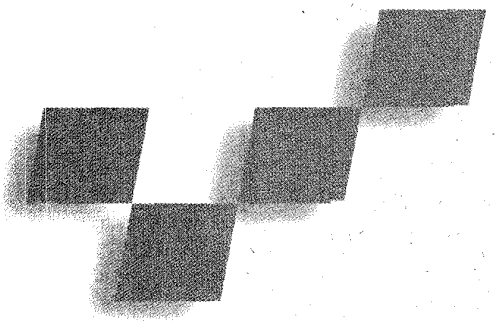


Animating Human Locomotion with Inverse Dynamics

Hyeongseok Ko
Seoul National University

Norman I. Badler
University of Pennsylvania

Locomotion is a major component of human activity, and there have been many attempts to reveal its principles through the application of physics and dynamics. Both computer graphics and robotics continue such efforts, but many problems remain unsolved, even in characterizing the simplest case: linear, forward, rhythmic walking.

This technique generates dynamically sound walking motion for any human gait, figure scale, or motion path by maintaining balance and keeping joint stress within the torque given by empirical strength data.

Since all mechanical linkage systems are subject to forces and physical laws, the computation of dynamics seems a promising approach to human motion problems. Forward dynamics has proved useful in predicting the motion of nonliving objects. For example, you can simulate a swinging chain by defining the initial state and then just integrating the effect of gravity or other forces acting on the system.

Unfortunately, the theory is less successful in animating the movements of "self-actuated" systems, namely, living creatures. Because the major force components—the internal muscular forces and torques—are not known a priori over time, you cannot use forward dynamics to predict how the human body will walk. Nor is there any known physical law to predict how that walk will change if an external force acts on the model. Accordingly, it is not easy to guess the joint torque patterns that will drive the model to take a step. Even if it takes a step, the result is unlikely to resemble a human walking pattern.

The alternative to the apparently intractable problem of specifying the joint torque patterns in advance is to use inverse dynamics to analyze the torques and forces required for the given motion. Such an analysis can show, for example, that the motion induces excessive torque, that the system is out of balance at a certain point, or that the step length is too great. In this article, we present a method of using an inverse dynamics computation to dynamically balance the resulting walking motion and to maintain the joint torques within a moderate range imposed by human strength limits. This method corrects or predicts a motion as indicated by the inverse dynamics analysis.

Dynamic correctness is a sufficient condition for realistic motion of nonliving objects. In animating a self-actuated system, however, visual realism is another important, separate criterion for determining the success of a technique. Dynamic correctness is not a sufficient condition for this visual realism. An animation of dynamically balanced walking that is also "comfortable" in the sense of avoiding strength violations can still look quite different from normal human walking. In this article, a visually realistic and dynamically sound animation of human locomotion is obtained using an effective combination of kinematic and dynamic techniques.

Related work

Boulic et al.[1] and Ko and Badler[2-4] attempted kinematic generalization of empirical walking data to generate locomotion along a curved path and intermittent, nonrhythmic stepping in any direction (forward, backward, lateral, clockwise, and counterclockwise). Their kinematic generations, however, do not handle the important case of a load or force attached to the body.

Bruderlin and Calvert[5] used a combination of kinematic considerations and dynamic motion control for goal-directed animation of human walking. For biped running, Girard[6] computed the impulses at each liftoff that drive the center of mass along the given path. He also added banking, which is a function of the velocity and curvature of running, for dynamic stability.

In robotics, many researchers have built actual bipedal walking robots,[7] but walking stability has not been easy to achieve. The gait pattern for walking robots is predesigned off line, and robot makers are not usually concerned with achieving human-like realism in stepping. Rather, they focus on the stability of the whole system.

Overview of the Speedy system

In conventional analysis systems like the Dynamic Analysis and Design System from Computer Aided Design

0272-17-16/96/$5.00 © 1996 IEEE

Software (Iowa City, Iowa), the analysis and correction phases are temporally disjoint: A given motion is first analyzed (for example, to obtain the joint torques) and the required correction is then computed. This works fine for mechanical linkages in non-self-actuated systems; but in self-actuated systems, the correction can make the analysis result incorrect and thus necessitate analysis of the new motion to ensure its dynamic soundness. Thus, generating a stable gait can involve many iterations.

To avoid the problem, we built a real-time inverse dynamics package and real-time motion corrector. These are used in a single-phase dynamic motion generator called Speedy. It calls the analyzer and corrector alternately for each frame to generate dynamically sound motion in real time.

The Speedy system controls human locomotion so that balance is maintained and joint stress is kept within the available torque given by empirical strength data. The strength data can be used to simulate comfortable walking or fatigued walking according to a scaled set of available torques.

Figure 1 gives an overview of the Speedy system. A kinematic locomotion generator ("Klog") spawns walking motions, which are then analyzed and dynamically modified to meet balance and stress constraints by the Dyncontrol module.

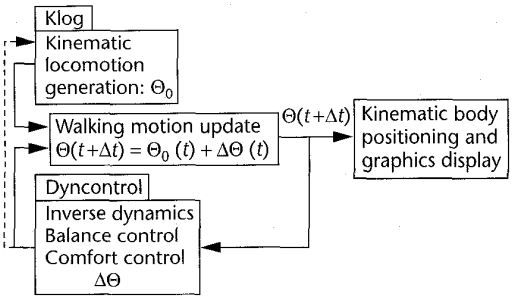
We use $\Theta(t)$ to denote all the joint angles and the position of the figure base in world coordinates at frame t. To control inverse dynamics requires an underlying motion Θ_0 and some mechanism for modifying it over time. We use the normal gait pattern with no load for Θ_0. The modification is done through the control parameters $\Delta\Theta$ predicted through inverse dynamics. Conceptually, we can write the modification as

$$\Theta(t+\Delta t) = \Theta_0(t) + \Delta\Theta(t) \quad \textbf{(1)}$$

At each frame, real-time inverse dynamics provide balance information and exerted force and torque at every joint in the model. The balance control unit computes the control parameters $\Delta\Theta(t)$ that should be added to the normal gait to retain balance. The comfort control unit compares the currently *exerted* joint torque with the *available* torque. If it finds a strength violation, it updates some parts of $\Delta\Theta(t)$ to reduce the exerted torque. If the update is on the movement of the figure base (foot), it is delayed until the beginning of the next step (dotted arrow in Figure 1), and affects Klog in generating the next step. The updated pose $\Theta(t + \Delta t)$ is used to position the body kinematically for the next frame.

Kinematic locomotion generation

For the walking animation, biomechanical data from straight, level, forward steps is generalized to the motion of an arbitrary anthropometrically scaled human figure.[2] The Klog generalization, which produces realistic locomotion animation in real time, can take steps along any curved path,[3] including intermittent nonrhythmic steps in any direction or turning toward any orientation.[4] A walk can be produced simply by specifying the goal location, a path, or a walk direction. Without a path, a linear path is assumed.



1 Overview of the locomotion control.

The primary parameters driving the input to the Klog consist of the next footprint and the designation of the stepping foot. This footprint-driven approach provides high-fidelity control of human locomotion. The primary parameters can be supplied from many sources:[4] reactive or nonreactive path planning systems, virtual reality applications, or interactively constructed spline curves, and so forth. (Note that there are a few other ways of specifying locomotion, such as the path of the center of mass or pelvis. However, these paths do not have exact control of the footprints, so the agent can fail, for example, to step over a pit on the way. In comfort control, the locomotion path is sometimes given as the primary parameter, which allows taking smaller steps to resolve a strength violation. Also, in balance control the locomotion path is sometimes the primary parameter to allow narrowing of the lateral step width and thus to reduce body swaying.)

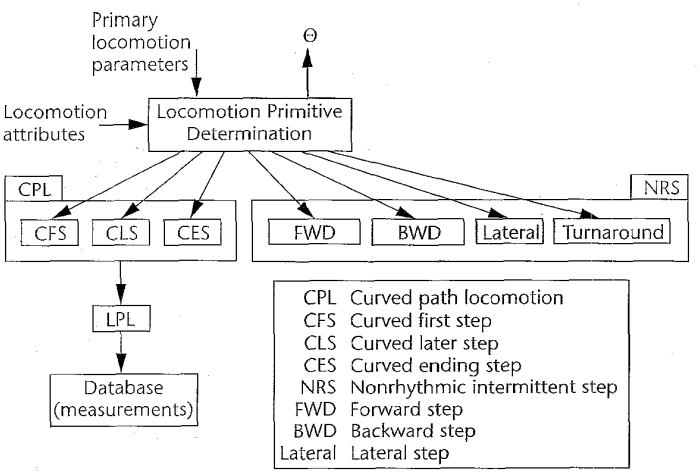
Klog is *goal driven* in that the resulting motion achieves the goal footprint very accurately. Even though some other aspects of the walking motion may be altered by later application of Dyncontrol, the goal achievement is not affected.

Klog has two phases. During the first phase it precomputes the kinematic motion assuming no load or external force is attached. This computation is purely kinematic, and there is no interaction with Dyncontrol. In the second phase, Klog actually generates the frames by putting in the effect of the 12 control parameters that modify the Klog result so that the dynamic balance and motion comfort can be achieved.

Figure 2 (next page) shows the architecture of the Klog subsystem. Klog receives the primary locomotion parameters and locomotion attributes. The default values for the attributes are those for normal nonloaded locomotion. Based on the step distance, direction, and previous history of stepping, Klog decides which of the locomotion primitives should be used. Locomotion primitives can be divided into several groups:

- Curved path locomotion (CPL) handles the rhythmic case and consists of curved first step (CFS), curved later step (CLS), and curved ending step (CES).
- Nonrhythmic intermittent stepping (NRS) consists of forward step (FWD), backward step (BWD), lateral step (Lateral), and Turnaround.

FWD is similar to CPL but more static in the motion, especially in the foot angle variation. Whenever a new



2 Structure of Klog.

originally recorded motion. The database can have multiple prototypes, and LPL can render different walking styles to the individual agents in the animation scene.

Torso flexion and pelvic rotation/translation have been parameterized to generate different styles of walking. The torso can be flexed or twisted in any direction rhythmically. For the pelvis, the position as well as the orientation relative to normal walking can be controlled over the gait cycle. Combining both torso flexion and pelvic rotation/translation produces minor stylistic or major changes in walking.

The Speedy system provides default values for setting these parameters. Thus the user perceives Klog as a high-level, goal-oriented locomotion system. The parameters are also modifiable through user specification or program control. Some of those parameters, namely the control parameters, are used for balance and comfort control.

Table 1. Classification of control parameters.

	Micro Parameters	Meta Parameters
Balance control	t^y_{pelvis}	Lateral step width
	t^z_{pelvis}	
	r^x_{torso}	
	r^z_{torso}	
Comfort control	t^x_{pelvis}	Step length
		Foot angle
Other purpose	r^x_{pelvis}	
	r^y_{pelvis}	
	r^z_{pelvis}	
	r^y_{torso}	

The control parameters are categorized into two sets: micro parameters and meta parameters (Table 1). Micro parameters μ are related to pelvis and torso motion. They specify the relative pelvis and torso rotation and/or translation compared with the normal gait. The pelvis has six degrees of freedom: three for translation ($\mathbf{t}_{pelvis}$) and three for the rotation ($\mathbf{r}_{pelvis}$). The torso can bend and twist in any direction, represented by a 3-vector $\mathbf{r}_{torso}$. Thus

$$\mu = (\mathbf{t}_{pelvis}, \mathbf{r}_{pelvis}, \mathbf{r}_{torso}) \quad (2)$$

Our conventions associate x, y, and z directions with forward, lateral (right), and down. For example, with $\mathbf{t}_{pelvis} = (-10, 0, 10)$, the hip is both lowered and displaced backwards by 10 centimeters. If we further set $\mathbf{r}_{torso} = (0, -30, 0)$, the torso will be bent forward by 30 degrees, and the overall motion will look like a crouched walk. Note that the balance control will adjust the gait as necessary for these torso and pelvis configurations, since all body segments have an appropriate mass and moment of inertia.

Meta parameters are related to the motion of the figure base: step length, foot angle, and lateral step width relative to the normal gait. In Figure 1, the dotted arrow shows control of meta parameters, which are modified at the end of each step to generate the next kinematic step.

As summarized in Table 1, balance control deals mostly with the micro parameters except for the lateral step width, and comfort control deals mostly with the meta parameters except $\mathbf{t}^z_{pelvis}$. The micro parameters are updated every Δt, and the meta parameters are updated at the end of every walking step.

locomotion primitive is added, such as running or crawling, we augment the Locomotion Primitive Determination module appropriately.

Our CPL[3] is obtained by generalizing a linear path locomotion (LPL). The LPL currently in use is called the *underlying LPL*. The implementation of CPL does not depend on the underlying LPL, and we make no specific assumption on the underlying LPL. We can therefore generalize actual walk data and most of the preexisting straight-line locomotion systems into curved-path locomotion systems by applying our algorithm. We are very cautious not to lose the realistic result of LPL. Thus in following the curved path, the original motion of LPL is retained as much as possible. When the curve happens to be a line, the result of the generalization algorithm is the same as the result of the original linear path algorithm. Our curved path generalization adds only a constant time to the straight-line walking.

LPL in turn is obtained by a kinematic generalization technique.[2] The motion characteristics are extracted from a prototype set of measured data on human walking, then applied to generate the walking step of an arbitrary anthropometrically scaled human figure in stepping an arbitrary step along a straight path. Even with the differences in the bodies and step lengths, the resulting motion of our LPL is very realistic and resembles the style of the

Real-time inverse dynamics

Speedy performs dynamic force and torque analysis at the joints of the human body model. It applies standard

robotics techniques, specifically, Denavit-Hartenberg notation (DH-notation) and Newton-Euler dynamics (recursive method).[8] Loads or general 3D forces can be attached to the body segments. Extra work is necessary to solve the closed-loop problem in the lower limbs. The inverse dynamics algorithm in the "Algorithms" sidebar summarizes the following discussion.

Dynamic model

DH-notation is a kinematic notational convention. Once a linked system is represented in this form, its kinematics are computable in a systematic way.[8] DH-notation is designed for a system with single-degree-of-freedom joints. We model a human body as several rigid links L_i ($i = 1, \ldots, s$) connected by joints J_j (see Ko[3]) and decompose multiple-DOF joints into several single-DOF joints. For example, the hip joint is formed by cascading three revolute joints that are mutually perpendicular at the home position (standing upright). Thus some L_i are null links with zero mass and length.

Newton-Euler method of computing torques

The problem of computing the joint torque is well defined, and systematic and efficient methods exist for serial link mechanisms. Of the two popular formulations (Newton-Euler and Lagrangian), we adapted Newton-Euler dynamics. The Newton-Euler dynamics algorithm[8] costs $O(n)$ where n is the number of DOFs in the system. An algorithm's complexity becomes an important factor when the model has many degrees of freedom. Linearity of the algorithm is the minimum requirement for real-time dynamics computation of such a complex model.

The Newton-Euler method works by computing, at each time t, the positional and angular acceleration of every link propagating from the base of the figure to the end-effectors (outward iteration). During the inward iteration (from the end-effectors to the base), the force and torque at the previous joint are propagated for the computation at the current joint. The mass and inertia of each link is considered during this phase of the computation. For mathematical details of the method, see Craig.[8]

Closed-loop problem

We have a closed loop (in the mechanism sense) during the human figure's double-stance phase. At this point, the lower limbs form a loop with the supporting plane. The difficulty in handling closed loops comes from indeterminacy. For example, if an object is held with both hands, inverse dynamics cannot determine the joint force and torque along an arm from the given motion alone. Some counteracting forces can exist without being detected in the kinematic profile of the motion itself. The indeterminacy is not caused by our selection of the dynamic computation methodology. Rather, it is a generic property of the problem itself. Kumar[9] studied closed-loop problems in cases such as multilegged vehicles and multifingered grippers.

For biped locomotion, we adopted a simple approximate solution. In propagating the force and torque from the pelvis to the two thighs during the inward (Newton-Euler method) iteration, the force is distributed according to the percentage of body support on each leg. If a percent of the upper body weight is supported by the left leg and b percent is supported by the right leg (this can be judged by computing the center of mass and looking at its projection within the figure's support polygon), the left hip gets a percent of the force and torque from the pelvis and the right hip gets b percent. (Details are presented in the sidebar "Resolving the closed loop at the lower limbs," p. 55.)

During locomotion, the dynamic computation should switch back and forth, from nonloop to loop conditions, as the loop opens or closes. Accordingly, there are three different states:

- left foot off the ground, right leg supporting (LORS),

Algorithms

Three algorithms discussed in this article are summarized in the following steps.

Inverse dynamics

At each frame,

(1) Compute Θ and $\dot{\Theta}$.
(2) Compute the mass and global inertia of the links including the effect of the attached load.
(3) Update the external force and moment (not necessary if interactive update is not allowed).
(4) Add the compensation torque at the waist for the skeleton model (see "Discussion").
(5) Compute the global position of the center of mass of each link.
(6) Compute the global position of the local origin of each link.
(7) Compute the global direction of the joint axes.
(8) Do outward iteration (see Craig[8]).
(9) Do inward iteration (see Craig[8]). The balance vector is computed here.
(10) If the current frame is within the double stance phase, the joint moments at the hips are computed by the closed loop algorithm in the sidebar "Resolving the closed loop at the lower limbs."
(11) Compute the joint torques τ_i (see Craig[8]).

Balance control

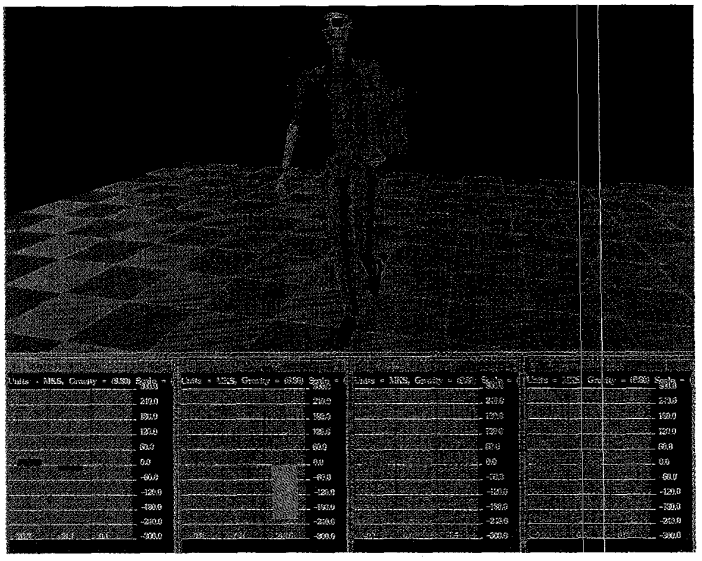
At each frame after the inverse dynamics algorithm,

(1) Get the balance vector $\mathbf{b}$ from inverse dynamics.
(2) Compute $\mathbf{b}^{\perp}$ by rotating $\mathbf{b}$ 90 degrees clockwise.
(3) Translate the pelvis by $\alpha\mathbf{b}^{\perp}$.
(4) Bend the torso by $\beta|\sin(\pi t/T)|\mathbf{b}$, where T is the duration of a step.
(5) If the lateral swaying is too large, signal Klog to reduce the lateral step width in the next step.

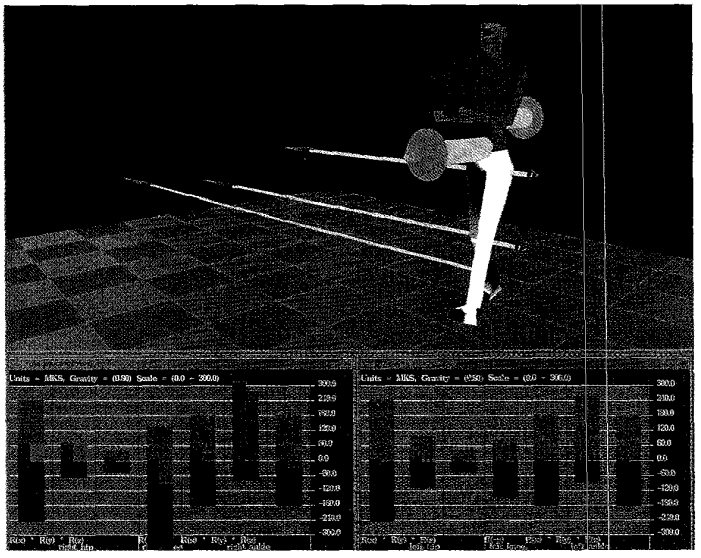
Comfort control

At each frame after the inverse dynamics and balance control algorithms,

(1) Compute the dynamic strength $\gamma_i (i = 1, \ldots, s)$ by Equation 5.
(2) Check for any i such that $\tau_i > \gamma_i$ (strength violation).
(3) If there exists any strength violation then
- Reduce the knee angle by increasing $\mathbf{t}'_{\text{pelvis}}$.
- Signal Klog to take an appropriate step length and foot angle in the next step.



3 Visualizations for balance control.



4 Visualizations for comfort control.

- left leg supporting, right off (LSRO), and
- both supporting (LSRS).

The closed-loop problem solution may differ slightly from what is actually happening because in practice the assumptions made in formulating the solution are not true throughout the duration of the motion. During locomotion, balance is relatively stable in the double-stance phase. Moreover, the weight and torque from the upper body is distributed between the two legs in this phase. Thus each leg gets less stress than in a single-support phase. Therefore, a small discrepancy between the computation and an actual walk does not cause a drastic change in the balance and comfort control.

Approximation for real-time computation

The acceleration computation at t needs the states at $t-\Delta t$, t, and $t+\Delta t$. But in real-time computing, the state at $t+\Delta t$ is not available. As discussed in the overview of the Speedy system, if the computation must alternate between the analysis and motion correction, it cannot be delayed even by Δt. Thus we use the states at $t-2\Delta t$, $t-\Delta t$, and t for the approximate acceleration. Even though this may create considerable differences during sharp motion changes, it appears to be quite tolerable for simulating locomotion.

Visualizations

The loading force is portrayed by a cube that swells or shrinks according to the mass (Figures 3 and 4). Any external force is shown by a thick arrow with a blue head, which also lengthens or contracts in proportion to the magnitude (Figure 4). Interactive changes to either force are allowed during the motion.

Three kinds of data graphs can be displayed: balance, reaction force (Figure 3), and available-versus-required torque (Figure 4). The two windows at the bottom corners of Figure 3 are the balance displays—the right one is for the left leg and the left one is for the right leg. (The apparent left-right reversal is because we like to view the figure walking toward us; thus its left is our right. The window appears in a visually compatible position.) The red bars show the extent of imbalance along the x, y, and z axes (subject to the conventions given above in the overview). The first bar indicates lateral imbalance, and the second indicates longitudinal imbalance. The third bar indicates the amount of twisting reaction torque from the ground to the foot sole.

The two middle windows in Figure 3 are the reaction force graphs. The light blue bar shows the reaction forces from the ground in the x, y, and z directions. As expected, most of the reaction force is in the z direction. Nonzero values in x or y are the reaction forces from the ground that prevent sliding. They cannot be greater than the maximum friction force.

Joints on the body display are colored (Figure 4) so that the torques can be portrayed. Specifically, as the torque increases, the color changes from white to blue. If the torque exceeds the strength limit, the color is set to red. The thin arrow (with red head) coming out of the joint indicates the magnitude of the torque.

The two bar graphs in Figure 4 show the available-versus-required torque panels of the right and left legs. A bar corresponds to a DOF. There are seven DOFs in a

panel: three for the hip, one for the knee, and three for the ankle. The purple bar shows the available torque (strength). Each DOF can rotate in two directions, positive (flexion) and negative (extension). The strengths in these two directions differ. They are called the positive and negative strengths, respectively. The light purple bar shows the positive strength, and the dark purple bar shows the negative strength.

The blue bar shows the dynamic required torque for the motion. The green bar shows the static torque considering each frame as a static case. The blue (dynamic torque) and green (static torque) bars can grow or shrink within the purple bar (strength). If the blue or green bar exceeds the purple bar, that part is colored red, indicating a strength violation. At the same time, the joint of the human figure on the display turns red. The available-versus-required torque panels can be created for other parts of the body as well.

Balance control

Balance control is activated at each frame after the inverse dynamics computation. The balance control algorithm in the "Algorithms" sidebar summarizes the following discussion.

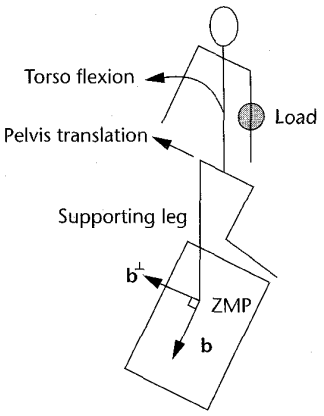
The zero-moment point and balance

Static balance can be achieved by keeping the projection of the center of mass within the figure's support polygon, even if only one leg provides support. In locomotion, however, we need to consider dynamic balance because the inertia effect is not negligible.

Consider a single support phase. Even though the body seems to be supported by the whole foot, we can find a point on the sole at which the moment is zero. This idea comes from the analogy of walking "on tiptoe." There can be no exerted torque at the toe. Similarly, during walking there is a point called the zero-moment point (ZMP),[10] where the exerted torque should be zero. ZMP is not a fixed point; it translates from the heel to the tip of the toe during support. We create a fake joint at ZMP that connects the foot to the world. (Note that ZMP is approximated by a monotonically advancing function from the heel to the tip of the toe throughout the support duration.[10])

Balance vector and balance control

Normally the moment at ZMP should stay zero all the time. If the result of inverse dynamics indicates a nonzero value, it means such a torque should have been exerted for the motion. We interpret it as the measure of imbalance at that moment. We call the torque at the ZMP the *balance vector* **b**. Thus, for example, if the balance vector is heading forward (Figure 5), a left-to-right torque (by the right-hand rule) should have been exerted to prevent lateral right-to-left collapse. Therefore, moving the pelvis to the perpendicular direction (right side) of **b** and bending the torso left-to-right will help maintain balance. (The pelvis translation and torso bending can be achieved by exerting appropriate torques, that is, forward dynamics.) In this work, however, we have direct control of the positional translation and angular rotation.) If we let $\mathbf{b}^{\perp}$ be the vector obtained



5 The balance vector.

Resolving the closed loop at the lower limbs

Let f^{PT} and τ^{PT} be the force and torque exerted on the upper body by the pelvis (see Figure A). Let f^{LP} and f^{RP} be the reaction forces acting on the pelvis from the left and right thighs, respectively. Let f^{ext} be the external force on the pelvis. Let τ^{LP} and τ^{RP} be the torque exerted from the left and right thighs to the pelvis, respectively. Let τ^{ext} be the external torque acting on the pelvis.

When an apparent force F^p (equal to the product $m \cdot v^{\cdot}_{pelvis}$ where m and $v^{\cdot}_{pelvis}$ are the mass and acceleration of the pelvis) is acting on the pelvis, we have the following relation between the forces:

$$f^{ext} + f^{PT} + f^{LP} + f^{RP} = F^p$$

Let $f = f^{LP} + f^{RP}$, then

$$f = -f^{ext} + f^{PT} + F^p$$

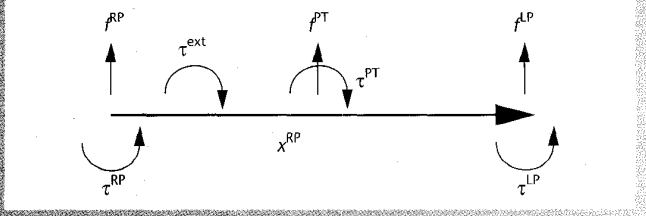
We distribute f according the relative distances a_l and a_r of the projection of the center of mass from the ankles. Thus $f^{LP} = a_l \cdot f$ and $f^{RP} = a_r \cdot f$, where $a_l + a_r = 1$, $a_l \geq 0$, $a_r \geq 0$.

Let $\tau_{f^{RP}}$ be the moment generated by f^{RP}, which is given by $(1/2)x^{RL} \times f^{RP}$, where x^{RL} is the vector from the right hip to the left hip. Let $\tau_{f^{LP}}$ be the moment generated by f^{LP}, which is given by $x^{RL} \times f^{LP}$. Let τ_p be the apparent moment on the pelvis and τ_{ext} be the external moment on the pelvis. We have

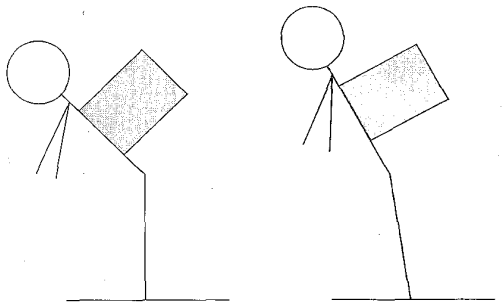
$$\tau^{ext} + \tau^{LP} + \tau^{RP} = \tau^{PT} + \tau_{f^{RP}} + \tau_{f^{LP}} = \tau_p$$

Similarly, τ^{LP} and τ^{RP} are given by $a_l \cdot \tau$ and $a_r \cdot \tau$, where

$$\tau = -\tau^{ext} + \tau^{PT} - \tau_{f^{RP}} - \tau_{f^{LP}} + \tau_p$$



A The pelvis diagram.



6 Knapsack example with upper body balance (a) at the waist and (b) not at the waist.

by rotating **b** by 90 degrees clockwise, the pelvis displacement is given by

$$\Delta\text{pelvis} = \alpha\mathbf{b} \quad (3)$$

and the torso bending is given by

$$\Delta\text{torso} = \beta\mathbf{b} \quad (4)$$

Determining the value of α and β is not an easy problem. If we use values that are too large, the adjustment will overshoot the correct balance; if too small, the adjustment will be too slow to achieve balance in time. Also, we must determine the ratio between them. These issues will be addressed under "Comfort control."

During the double-stance phase, the balance vector is approximated by the weighted average of the balance vectors at the two feet according to the relative position of the center of mass projection. We use smaller values for α and β, since the double-stance phase is relatively more stable than the single-stance phase, especially in the longitudinal direction. Thus a small discrepancy resulting from this approximation does not significantly affect the whole balance control.

Lateral swaying

Balancing creates lateral swaying. The swaying amplitude increases as the load gets heavier, the step gets slower, or the lateral width of the step gets wider. Swaying consumes energy (as kinetic energy). It can be reduced by having laterally narrower steps. Thus, if a step caused too much swaying, the lateral step width is reduced by a certain amount in the next step. As the load gets heavier, we use a narrower lateral step width. In real human walks, excessive loads can even cause a negative lateral step width.

Comfort control

Comfort control detects strength violation by comparing the result of the inverse dynamics and the strength data. If the Speedy system detects a strength violation, it activates the motion strategy that will reduce torque at the joint. The comfort control algorithm in the "Algorithms" sidebar summarizes this discussion.

Strength data

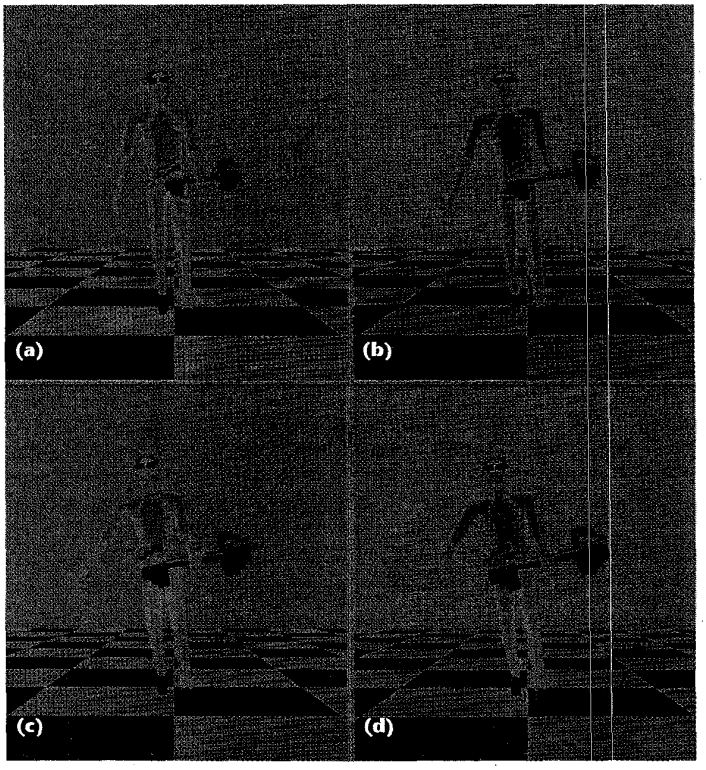
Obviously there is a limit to the torque that can be exerted at a joint. For each rotation axis, we have two limits: one for extension and the other for flexion. The body torque that tries to counteract the external flexional torque is called *extensional torque* and has a negative value. The *flexional torque* is similarly defined and has a positive value. Thus the torque at each joint is limited to the extreme values of the extensional and flexional torques. These upper and lower bounds are determined from strength data.

Pandya et al.[11] collected strength data for several human subjects. Strength varies with each individual. Moreover, it depends on joint angles and their angular velocity. It is approximated by a second-degree polynomial:

$$y = f_0(\dot{\theta}) + f_1(\dot{\theta})\theta + f_2(\dot{\theta})\theta^2 \quad (5)$$

where θ is the joint angle, and f_0, f_1, and f_2 are the functions of $\dot{\theta}$.

Lee et al.[12] proposed an animation technique that considers comfort at the joint during an end-effector motion. The end-effector proceeds



7 Comfort control when a 40-kg load is attached: (a) first step, (b) second step, (c) sixth step, and (d) final step.

Table 2. Summary of approximations in Dyncontrol.

	Approximation	Ideal
Acceleration	$\frac{\frac{\Theta(t) - \Theta(t - \Delta t)}{\Delta t} - \frac{\Theta(t - \Delta t) - \Theta(t - 2\Delta t)}{\Delta t}}{\Delta t}$	$\frac{\frac{\Theta(t + \Delta t) - \Theta(t)}{\Delta t} - \frac{\Theta(t) - \Theta(t - \Delta t)}{\Delta t}}{\Delta t}$
Double-stance inverse dynamics	See sidebar "Resolving the closed-loop"	Not uniquely determined
Double-stance balance vector	See discussion "Balance vector and balance control"	Not defined

for Δt along a suggested direction (for example, straight path to the goal) unless it causes a strength violation. If that occurs, a cone of possible directions is computed, and the original direction is projected within the cone. For example, if the object in a lifting task is heavy, the hand trajectory is altered to move it closer to the body to reduce the required torque. However, because locomotion moves the figure base as well as the end-effectors, a more complex strategy is required.

Comfort control implementation

If the system is balanced on a point, the overall moment around that point is zero by the definition of dynamic balance. For example, if a knapsack is attached at the back, the upper body alone can achieve a balance by bending the torso forward an appropriate amount (see Figure 6a). In this case the torque at the waist is zero, and the torques at the joints along the lower limbs will be smaller than they are when the upper body balance is not at the waist (Figure 6b). This observation leads us to balance by torso flexion only and suggests a relatively large value for β. But too much torso bending can lead to excessive instantaneous torque at the waist at the moment of heel strike, thus suggesting a relatively small value for β. The conflict can be resolved by changing β periodically. β is minimized at heelstrikes and maximized in the middle of the step.

We used $\alpha = 0.02$ (constant) and $\beta = 0.018$ (the maximum) in producing animations. Note that the values are not compatible by themselves since one is positional (cm) and the other is angular (degrees). The pelvis translation was limited to 5 cm, 10 cm, and 2 cm in the x, y, and z directions, respectively. Thus if there is excessive external force in the lateral direction, the pelvis is displaced by 10 cm and the rest of the imbalance is resolved by torso bending.

Comfort control manages the step length, foot angle, and knee angle parameters. A shorter step together with a smaller foot angle variation induces less torque at the hip, thus less torque at other joints along the lower limbs. A smaller knee angle helps reduce torque at the knee.

After each step, if any strength violation is detected, the step length and foot angle are reduced by a certain amount. The knee angle is indirectly controlled through the pelvis height $\mathbf{t}^z_{\text{pelvis}}$. Thus the pelvis is raised a certain amount during the stance phase, creating a more or less stiff walking pattern. The simulation in Figure 7 shows the decrease of the step length, foot angle, and knee angle until the problem (red-colored joint) disappears.

Table 3. Summary of control parameters in Dyncontrol.

Parameter	Interactive or Automatic	Nominal Value
$\mathbf{t}^x_{\text{pelvis}}$	Automatic	
$\mathbf{t}^y_{\text{pelvis}}$	Automatic	
$\mathbf{r}^x_{\text{torso}}$	Automatic	
$\mathbf{r}^y_{\text{torso}}$	Automatic	
$\mathbf{t}^z_{\text{pelvis}}$	Automatic	
Lateral step width	Automatic	
Step length	Automatic	
Foot angle	Automatic	
α	Interactive	0.020
β	Interactive	0.018

Discussion

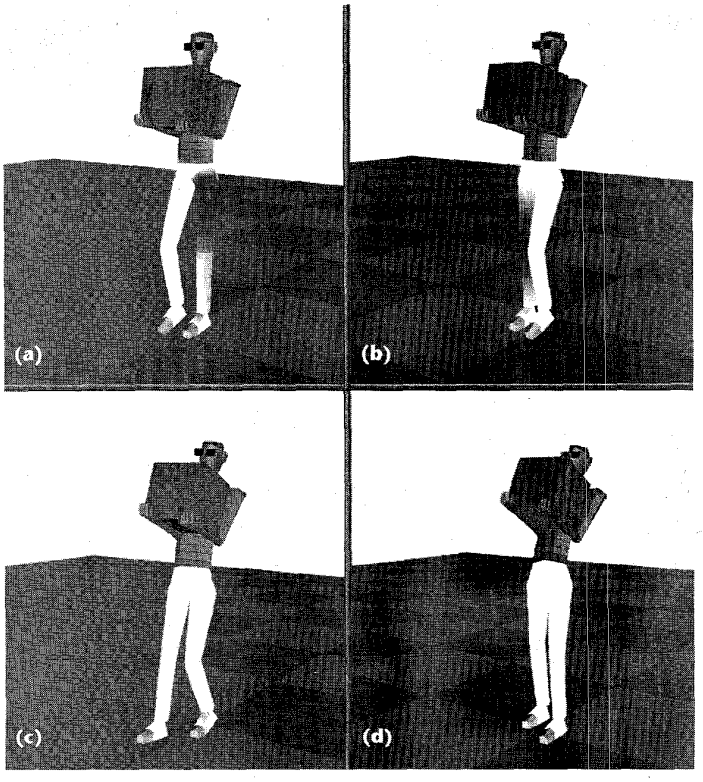
In the dynamic simulation, variable loads and/or a 3D force can be attached to any points of the body. If a load is attached, the Speedy system recomputes the mass and moment of inertia of the attached link. We assume the external force is acting on the center of mass. If not, we translate it to the center of mass by adding an extra moment on the link.

Approximations are used in three places: computation of acceleration, closed-loop inverse dynamics, and balance vector in the double-stance phase. As pointed out earlier, the small discrepancies do not materially affect the final analysis result or locomotion prediction. Table 2 summarizes all the approximations used and compares them with an ideal computation.

We used a 17-segment torso model with 17 flat discs connected along the spine. This skeletal model creates a torque at the waist to the forward direction. A more complex abdominal strength model may solve the problem, but it would require more complex computations. Instead, we just added a counteracting torque backward to compensate.

Note that the above approximations and compensation are handled automatically by the program. The only things the animator may need to adjust are the values of α and β to produce a dynamically balanced and comfortable walking animation under an arbitrary load and external force.

Table 3 summarizes all the control variables used in balance and comfort control. It shows whether they are controlled interactively or automatically by programs. In



8 Balance control under loads of (a) 15 kg, (b) 30 kg, (c) 45 kg, and (d) 60 kg.

9 Jack carrying a German shepherd in a hurricane.

the case of manual control, a nominal value is specified.

Our decision regarding comfort control is based on the comparison between the required and the available torques. The joint (compression) forces should be also considered for a dynamically safe motion. A joint force is the force exerted at the joint from the above link. In a situation that involves a great impact, the joint force becomes more important. When landing from a high jump, stiff straight legs induce zero torques but huge forces at the moment of the impact. Normally, impact forces are not great enough to cause problems in locomotion. We may later refine our decisions on strength violations if we acquire data on impact endurance.

Speedy is implemented on Silicon Graphics workstations. On an SGI Crimson it draws 20 frames per second for the curved path walking alone. With balance and comfort control, it draws 10 frames per second. The speed is reduced to seven frames per second if the human figure's 2,000 polygon faces are shaded. (Note that the duration between frames is fixed—for example, 1/30 second—thus the result does not depend on nonuniform machine speed.) Faster workstations such as the SGI Onyx bring this to real time.

We performed several experiments using the Speedy system. Figure 7 shows the result of the comfort control algorithm when the human figure is carrying a 40-kilogram load. The initial strength violation is resolved at the sixth and following steps. Figure 8 shows the gait changes due to balance control under different loads.

Figure 9 is a snapshot from a short movie in which a person carries a dog in a strong wind. The inflexible chain was added to show the direction and magnitude of the wind. Its direction shows the direction away from the wind, and the (world coordinate) chain angle shows the magnitude of the wind. The dog's motion does not affect the chain. The experiment produces realistic human locomotion and postural adjustment in the presence of significant loads and changing external forces.

The Speedy system thus implements an efficient real-time technique for human locomotion animation using balance and comfort control, inverse dynamics, and strength data. It modifies the walk in real time when a load or a 3D external force is applied and includes several visualization techniques to display the dynamics computation result. ■

Acknowledgments

This research is partially supported by ARO DAAL03-89-C-0031 and the US Army Research Laboratory (Aberdeen); DMSO DAAH04-94-G-0402; US Air Force through Hughes Missile Systems F33615-91-C-0001; ARO DAAH04-95-1-0023; ARPA DAAH04-94-G-0362; and National Science Foundation CISE CDA88-22719.

References

1. R. Boulic, N. Magnenat-Thalmann, and D. Thalmann, "A Global Human Walking Model with Real-Time Kinematic Personification," *The Visual Computer*, Vol. 6, No. 6, 1990, pp. 344-358.
2. H. Ko and N.I. Badler, "Straight-Line Walking Animation Based on Kinematic Generalization that Preserves the Original Characteristics," *Proc. of Graphics Interface 93*, Morgan Kaufmann, San Francisco, 1993, pp. 9-16.
3. H. Ko, *Kinematic and Dynamic Techniques for Analyzing, Predicting, and Animating Human Locomotion*, doctoral dissertation, MS-CIS-94-31, Univ. of Pennsylvania, Dept. of Computer and Info. Science, Philadelphia, Penn., 1994.
4. N.I. Badler, C.B. Phillips, and B.L. Webber, *Simulating Humans: Computer Graphics and Animation and Control*, Oxford Univ. Press, New York, 1993.
5. A. Bruderlin and T.W. Calvert, "Goal-Directed, Dynamic Animation of Human Walking," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, July 1989, pp. 233-242.
6. M. Girard, "Interactive Design of 3D Computer-Animated Legged Animal Motion," *IEEE CG&A*, Vol. 7, No. 6, Jun. 1987, pp. 39-51.
7. S. Kajita, K. Tani, and A. Kobayashi, "Dynamic Walk Control of a Biped Robot Along the Potential Energy-Conserving Orbit," *Proc. IEEE Int'l Workshop on Intelligent Robotics and Systems*, IEEE, New York, 1990, pp. 789-794.
8. J.J. Craig, *Introduction to Robotics, Mechanics, and Control*, 2nd edition, Addison-Wesley, Reading, Mass., 1989.
9. V.R. Kumar and K.J. Waldron, "Force Distribution in Closed Kinematic Chains," *IEEE J. Robotics and Automation*, Vol. 4, No. 6, 1988, pp. 657-664.
10. M. Vukobratovic, *Biped Locomotion*, Springer-Verlag, Berlin, 1990.
11. A.K. Pandya et al., "The Validation of a Human Force Model to Predict Dynamic Forces Resulting from Multijoint Motions," Tech. Report 3206, NASA, Houston, Texas, 1992.
12. P. Lee et al., "Strength Guided Motion," *Computer Graphics* (Proc. Siggraph), Vol. 24, No. 4, Aug. 1990, pp. 253-262.

Hyeongseok Ko *is a faculty member of the School of Electrical Engineering, Seoul National University. His research interests include computer animation, human locomotion, virtual reality, and human factor analysis. He received BA and MS degrees in computer science from Seoul National University in 1985 and 1987, and a PhD in computer science from the University of Pennsylvania in 1994.*

Norman I. Badler *is a professor of computer and information science at the University of Pennsylvania and director of the Center for Human Modeling and Simulation. His research focuses on human figure modeling, manipulation, and animation. He is the originator of the Jack software system. Badler received a BA in creative studies mathematics from the University of California at Santa Barbara in 1970, and an MS in mathematics in 1971 and PhD in computer science in 1975, both from the University of Toronto. He is coeditor of* Graphical Models and Image Processing *and coauthor of* Simulating Humans *(Oxford University Press, 1993).*

Readers may contact Hyeongseok Ko at the School of Electrical Engineering, Seoul National University, San 56-1, Shinrim-Dong, Kwanak-Ku, Seoul, 151-742, Korea, e-mail ko@kdb.snu.ac.kr.