CVGIP: GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 55, No. 6, November, pp. 507–521, 1993

Physically Based Modeling and Control of Turning

MICHIEL VAN DE PANNE,*,† EUGENE FIUME,‡ AND ZVONKO VRANESIC*

*Department of Electrical Engineering and †Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

Received June 23, 1992; revised February 1, 1993; accepted July 15, 1993

Turning plays an important role in activities such as skiing and bicycling. The act of turning requires steering, which performs two functions: changing the direction of motion and ensuring that balance is maintained. We analyze and control turning motions for a simple physical model. The physically based motion of the simple model is then used as a basis for the motion of more complex display models. A phase-diagram description of periodic turning motions (as in slalom skiing) is presented. The phase diagram is used to construct a control algorithm parameterized in terms of the frequency, sharpness, and heading of the turns. A second method of control allows an animator to draw an arbitrary path for the turning figure to follow while avoiding obstacles. A finite-time optimization is used to find the best physically feasible motion that closely follows the desired path. Examples of alpine skiing, snowboarding, bicycling, and telemark skiing are given. © 1993 Academic Press, Inc.

1. INTRODUCTION

The turning motion of a bicycle is a graceful manoeuver that is somewhat more complicated than it looks. A turn involves controlling both direction and balance. Similar motions exist in sports such as skiing or surfing. The goal of our work is to provide a suitable physically based model for this class of motions, and more importantly, to come up with general control techniques for turning motions. These control techniques should be easy and intuitive for an animator to apply.

Physically based animation is becoming a popular technique for creating realistic motion. It involves performing simulations based upon the laws of physics. There is a distinction, however, between simulations of passive and active systems. Motion in the former is determined only by the initial conditions, whereas motion in the latter is determined by the initial conditions and the control that is subsequently applied. The work presented here falls in the latter category.

Given that a motion can be influenced by some control variable, we must tackle the interesting (but difficult) problem of how an animator can manipulate the control variable to produce a desired motion. Furthermore, control variables are not always easily comprehended by animators. A large portion of this paper addresses the design of controllers that accept parameters intuitive to the animator and produce the control necessary to achieve the desired motion.

The next section briefly discusses the relevant previous work. The physical model we use and its equations of motion are given in Section 3. We present a phase diagram useful for analyzing and synthesizing periodic turning motions in Section 4. A parameterized controller for the model is given in Section 5. A path-following controller is described in Section 6. Examples of how to map display models onto the physical model and a discussion close the paper.

2. BACKGROUND

The difficulty of creating a suitable controller increases greatly with the complexity of an object. As a result, it is often useful to work with a simplified physical model. This is evident in the work of Raibert and Hodgins [1], who model and control the locomotion of various types of running figures. Their work involves assuming the role of various body parts. This reduces the number of control variables and makes them largely independent of each other, thereby greatly simplifying the control problem. The controllers generated are hand-tuned for each creature, but they are parameterized at a high level.

Bruderlin and Calvert [2] use a mix of dynamic and kinematic methods for the animation of walking. The stance leg and upper body are treated together as an inverted double pendulum. The dynamics obtained from the simplified physical model produces a natural locomotion pattern which is visually upgraded by adding kinematic "cosmetics." The appropriate kinematic motions are determined from experimentally-derived principles for walking, such as the determinants of gait. This work is similar to ours in that we shall also make use of a simplified physical model and amplify its motion to control a more complex display model.

† E-mail: van@dgp.toronto.edu.

McKenna and Zeltzer [3] also simplify the control problem by making use of modular and hierarchical decomposition of the control system where possible. The simulated six-legged insect is controlled by low-level parameters such as the stepping speed, the time between stepping of adjacent legs on the same side of the body, and the oscillator frequency.

Optimization techniques have been used with some success [4–6]. These typically minimize the control energy spent to get from the current state to the desired state. Calculating optimal solutions is generally a time-consuming proposition. A shortfall of these techniques is that they do not present an intuitive interface to an animator.

Miller produced convincing snake and worm animations [7]. The creatures move in a realistic way and can be controlled by specifying the desired speed and direction. The control is achieved by passing periodic oscillations of the appropriate magnitude and phase down the body of the snake or worm. This method of control is specific to the tubular structure of snakes and worms.

A large variety of work has been performed in the field of biomechanics. Work in this field usually involves analysis rather than synthesis, however. Sodeyama *et al.* [8] present an experimental study on the displacement of a skier's center of gravity during a ski turn. Little work has been done on a mechanical analysis of turning motions. Some qualitative analysis of turns can be found in popular ski magazines. Lee and Kunii [9] use filmed skiing motions to recreate animated skiing motions.

3. A PHYSICAL MODEL FOR TURNING

In order to understand our choice of simplified physical model, it will be helpful to begin with some simple intuition on how turning works for bicyclists, skateboarders, and other turning figures. It is clear that it is necessary to "lean into" a turn in order to remain properly balanced throughout the turn. What is not so clear is whether the turning occurs as a result of leaning, or if leaning occurs as a result of turning. We shall argue for the latter.

We shall use the cyclist in Fig. 1 to illustrate that the lean angle of a cyclist can not be controlled directly, and therefore that leaning occurs as a result of turning. In order for the cyclist in Fig. 1 to attain a leaning position, we require a torque acting around the z-axis. No mechanism for applying such a torque exists. It is possible to effect small changes in the lean angle using body motion alone, as demonstrated by skilled cyclists who can remain balanced while stationary. It becomes much easier to retain one's balance when moving forward on a bicycle, even if only moving slowly. At low speeds, the effect of gyroscopic forces are minimal, and we can conclude that it is the ability to turn that allows for easy balancing and control of the lean angle. A lean is thus obtained by moving the support out from underneath the body, which is easily done by making the supporting "foot" follow a curved path while the body is moving forward.

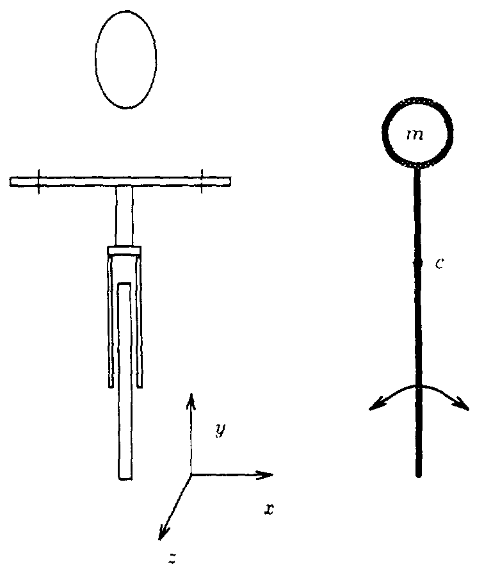


FIG. 1. A cyclist and a simplified model.

As an aside, it is worthwhile to explain how one can ride a bicycle without having hands on the handlebars, in which case we have no direct control over the steering. One can exert indirect control over the steering by bending sideways at the hips, shown as point c on the simplified model in Fig. 1. Bending in this fashion will not directly change the lean angle of the mass with respect to the support, but it will tilt the bicycle, which forms the lower part of the supporting "leg". Bicycles are constructed such that when they are tilted the front wheel turns, even when stationary. This mechanism allows for indirect control of steering by bending sideways at the hips.

All the figures we shall deal with have the ability to control the curvature of the supporting "foot." Bicyclists can turn their front wheels. Skiers place their skis on edge, which causes them to bend into an arc shape because of the *sidecut*. Snowboards work in a similar way. Skateboards turn because of the *trucks* that cause the wheels to twist when the board is tilted. Telemark skiers, loosely equivalent to going downhill on cross-country skis, separate their skis and place the front ski at an angle to the rear ski in order to turn. Photo 1 illustrates these configurations.

In order to make the control algorithm as simple and as general as possible, our physical model of the turning figure is the inverted pendulum shown in Fig. 2. The body has a known mass m and moment of inertia about the z-axis, I_v. The body moves forward along its z-axis with a velocity V_{cm} over a planar surface. The center of mass is

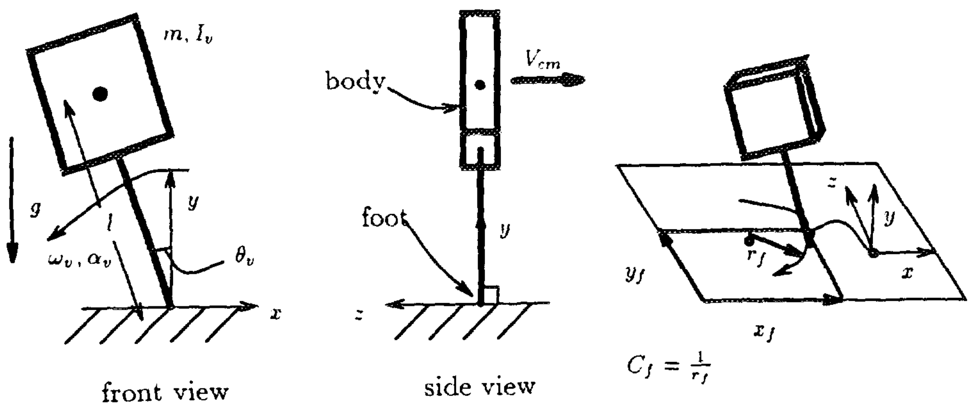


FIG. 2. The physical model used for turn synthesis.

located at a height l above the foot. The origin of the coordinate system is fixed to the "foot" of our model, with the y-axis perpendicular to the ground and the negative z-axis aligned with the direction of motion.

The moment of inertia about the x-axis is ignored because we assume the leg to be always vertical in the yz-plane. This is sufficient for the turning figures we shall consider; it is a simple matter for bicyclists and skiers to avoid falling forward or backward when moving across smooth terrain. The moment of inertia about the y-axis is ignored because rotation about the y-axis is caused by the nonholonomic constraint placed on the motion by the curvature of the foot, and not some direct torque about the y-axis. A constant velocity assumption is not crucial in our development of the equations of motion. It is, however, necessary to being able to obtain our parametric controller in an analytic form in Section 5. The model freely rotates about the z-axis of its foot. The model allows the curvature of its foot in the xz-plane, given by C_f, to be controlled. That is, the model can be steered. C_f is defined as $1/r_f$, where r_f is the instantaneous radius of curvature.

Because of the simplicity of the physical model, a mapping relation is necessary to relate the state of the physical model to the more complex *display* model. The mapping relation allows for the kinematic coordination of various body parts with that of the physical model, even though they are not explicitly included in the physical model. In this way subtle motions such as appropriate twists of the upper body can be generated. These motions would be difficult and prohibitively expensive to synthesize in a fully dynamic solution to the same motion. For example, optimal control techniques such as dynamic programming have an exponential complexity with respect to the size of the state vector.

The problem of using a more complex model is one of calculating the control to generate the appropriate motion. If one were to use an optimization technique to generate the necessary control, not only is the optimization itself difficult to perform for complex systems, but it is not clear what the optimization function should be. Our approach makes use of a physical model to generate the dominant part of the motion. Kinematic relationships are used to generate the motion of parts whose movements are less dependent on constraints imposed by physics and can also be used to capture more aesthetic or irregular aspects of the motion. A simple physical model also allows for a more general control solution because, as is the case here, a group of diverse sports can make use of the same simplified physical model, and hence the same controller.

Example mappings of several display models onto the physical model are shown in Photo 1. The mapping procedure itself is discussed in greater detail in Section 7.

The equations of motion of the turning body are derived in Appendix A. We are interested in the relationship between the curvature of the foot, C_f, and the angular acceleration about the z-axis, α_v. Because C_f is controllable, we shall define the forward dynamics as $\alpha_v = f(C_f)$ and the inverse dynamics as $C_f = f^{-1}(\alpha_v)$. The inverse dynamics is governed by the quadratic equation

$$0 = aC_f^2 + bC_f + k, \qquad (1)$$

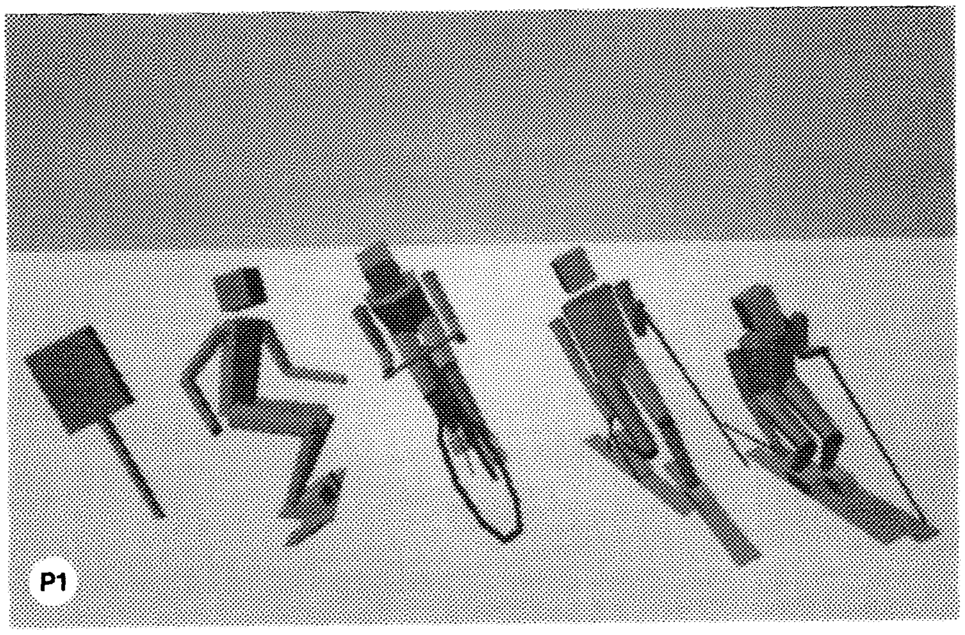
where C_f is the curvature of the foot, and

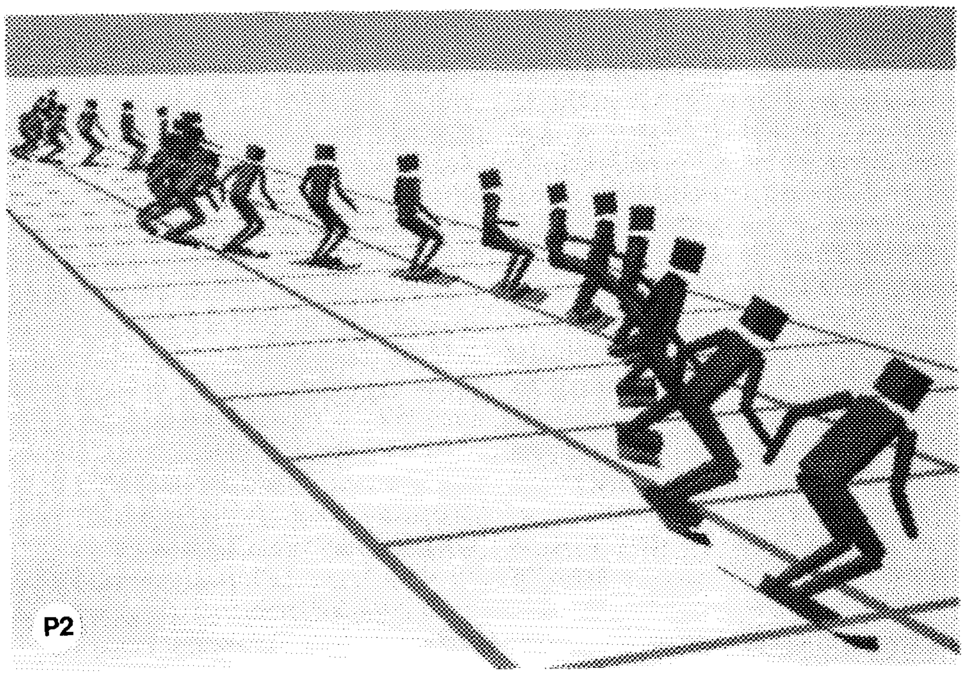
$$a = l^2 \sin^2 \theta_v k + lV_{cm}^2 \sin \theta_v \cos \theta_v$$

$$b = -2l \sin \theta_v k - V_{cm}^2 \cos \theta_v$$

$$k = g \sin \theta_v - \frac{\alpha_v}{ml}(I_v + l^2 m).$$

In the expressions for a and b, k is a quantity useful in simplifying the equation. The negative root of the quadratic is used when solving for C_f, as seen in Appendix

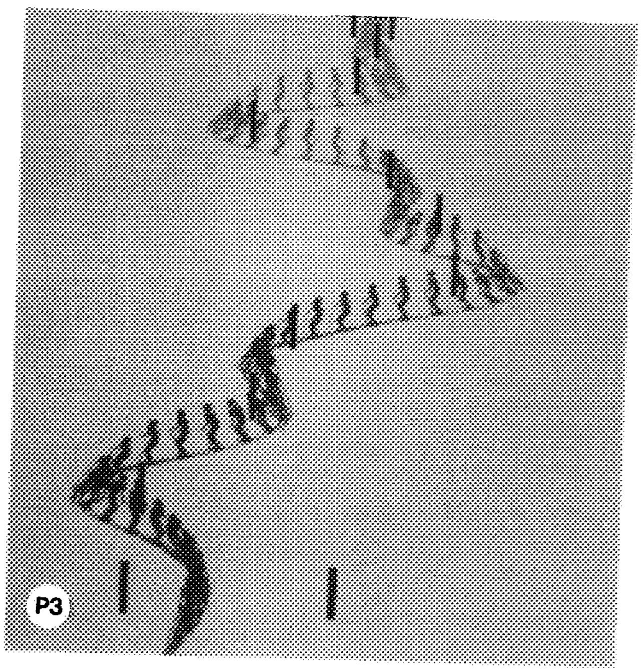


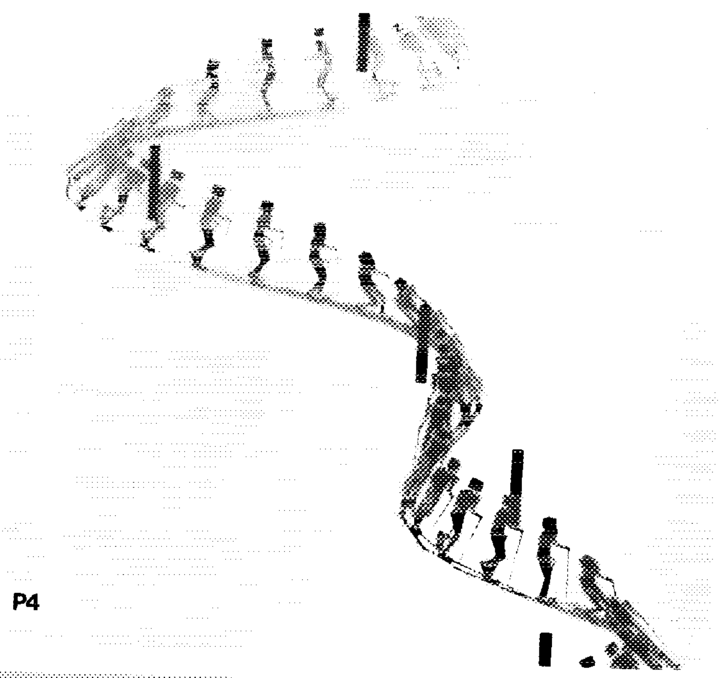


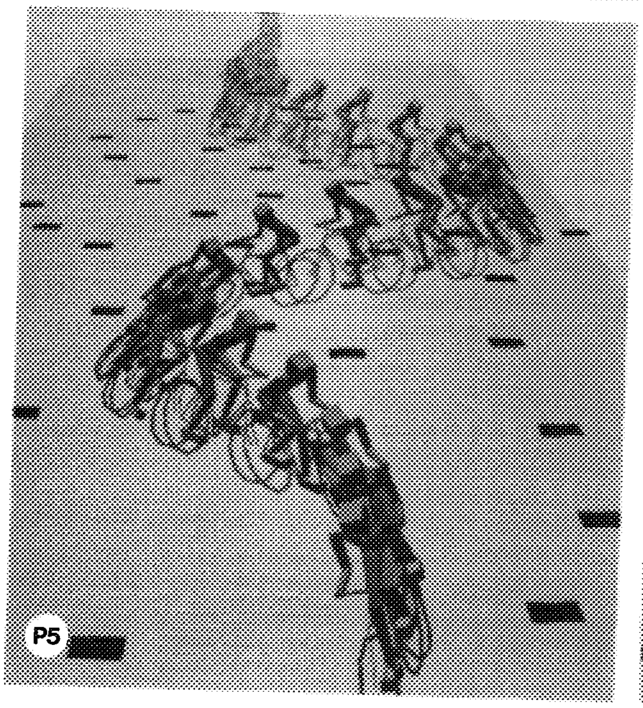
Photos 1 and 2

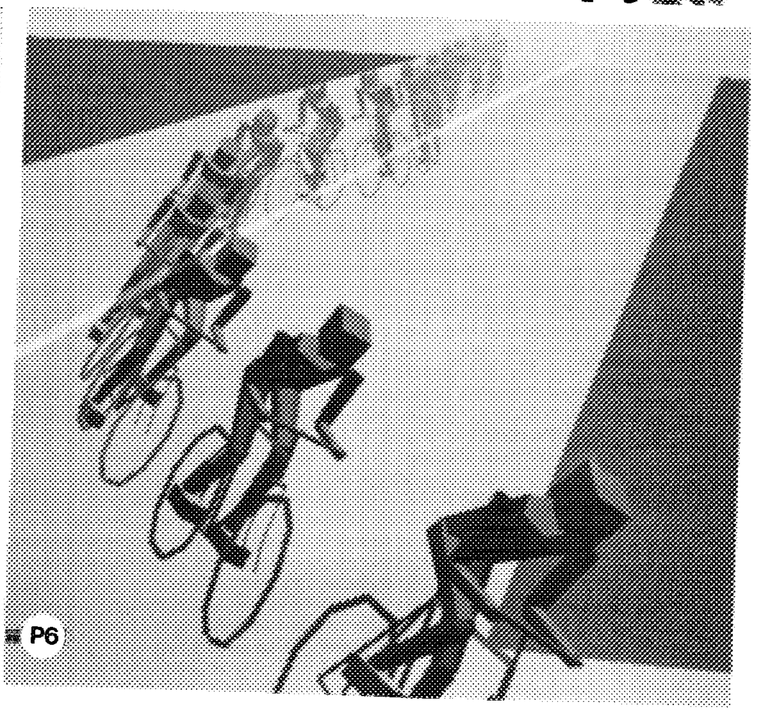
A. We can use Eq. (1) to calculate the curvature of the foot needed to achieve a given angular acceleration of the body, a_v.

Because it is unrealistic for the curvature of a ski or the steering angle of a bicycle to be unlimited in value, it is necessary to place upper and lower bounds on C_f. As we shall see later, we shall be calculating a desired value of α_v, then using Eq. (1) to solve for the corresponding C_f, bounding the value of C_f to a reasonable one if necessary, and lastly recalculating the α_v corresponding to the new, bounded C_f. This process is analogous to kinematically planning a motion, using inverse dynamics to calculate the forces required to produce the motion, limiting the forces to being reasonable ones, and then









Photos 3–6

recalculating the resulting motion. An expression that calculates the forward dynamics can be obtained by rearranging Eq. (1),

$$\alpha_v = \frac{lm(g \sin \theta_v - k)}{I_v + l^2 m}, \tag{2}$$

where k, the same quantity as in Eq. (1), is calculated as

$$k = \frac{C_f V_{cm}^2 \cos \theta_v}{1 - C_f l \sin \theta_v}.$$

Once α_v and C_f are known, all that remains for a simulation of the motion of the system is a numeric integration method to obtain new values for ω_v, θ_v, θ_{dir} (the direction of travel), and (x_f, y_f) (the position of the foot). Euler integration is used. x_f and y_f are integrated using the current speed and direction. θ_v and ω_v are integrated using the known value of α_v. The current direction, θ_{dir} is integrated knowing the current curvature and speed.

4. PHASE DIAGRAMS FOR PERIODIC TURNS

It is readily observable that slalom-type turns result in an oscillation of the body about the vertical. This means that θ_v in our model should vary sinusoidally. We shall use this as one basis for generating physically based turning motions. By controlling the turns of the foot of our model so as to generate body oscillations about the vertical axis, we generate realistic slalom turns as a desirable side-effect. It is thus easier to analyze and synthesize turns in terms of the oscillation of the body about the vertical axis rather than in terms of the path of the foot. In this section we analyze how this oscillation can be described using a phase diagram and how to use the phase diagram for analysis and synthesis of turns. In the following section we shall describe how to use the phase diagram for generating conveniently parameterized turn-controllers.

We shall begin by assuming that θ_v varies sinusoidally over time, as shown in Eq. (3). θ_{max} defines the amplitude of the oscillation, and thus represents the maximum angle of lean. Let ω_0 be the frequency of oscillation, or alternatively, the turning frequency. Then

$$\theta_v = \theta_{max} \sin(\omega_0 t). \tag{3}$$

We can take the derivative with respect to time of the expression in Eq. (3) to obtain expressions for the angular velocity and angular acceleration about the foot, as given by

$$\omega_v = \omega_0 \theta_{max} \cos(\omega_0 t), \tag{4}$$

$$\alpha_v = -\omega_0^2 \theta_{max} \sin(\omega_0 t). \tag{5}$$

We now have an expression for calculating α_v, which we can use as a controlling function using the inverse dynamics equation. Thus, we can control the curvature of the foot so as to create a desired sinusoidal body oscillation of a given amplitude and frequency. Note that although the body is made to oscillate sinusoidally, the path taken by the foot to achieve this is determined by the inverse dynamics and is thus not a perfect sinusoid, although it is naturally smooth and periodic.

The portion of the state of the system represented by θ_v and ω_v moves clockwise along an elliptical trajectory as shown in Fig. 3. We shall call this the *phase diagram*

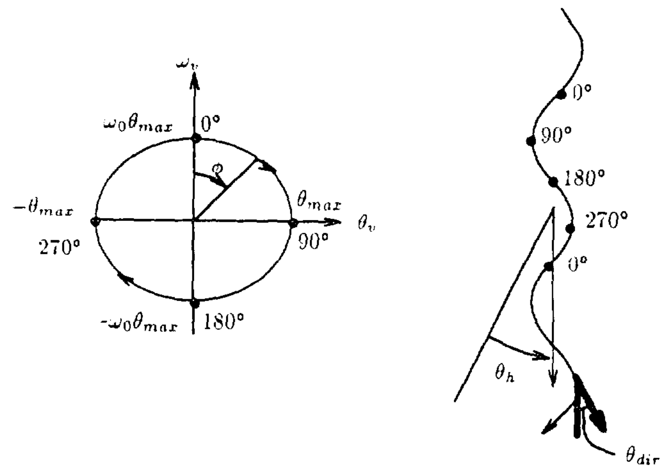


FIG. 3. The phases of a turn.

of a turn. The ellipse is defined by the constants θ_{max}, the maximum lean angle, and ω_0, the turning frequency.

The current *phase* of the body is given by $\phi = \omega_0 t$. It can also be calculated directly as follows: $\phi = \omega_0 t = \arctan(y/x)$ where $y = \theta_v/\theta_{max}$ and $x = \omega_v/\omega_0\theta_{max}$. The phase is dependent on both the state of the body as well as θ_{max} and ω_0, which define its planned motion. The phase contains useful information for the mapping of display models onto the physical model.

The simulation and open-loop control cycle using Eq. (5) is described by the pseudocode in Fig. 4. F_{leg} represents the compressive force experienced by the leg and is useful in the display mapping process, as is the phase, ϕ.

The correspondence of various points in a turn to the phase diagram is illustrated in Fig. 3. The shape of the path taken by the foot is not necessarily sinusoidal in shape, as will be further explained in the next section. The general direction of the turning figure is given by its θ_h, which defines its heading with respect to a fixed axis.

Figure 5 illustrates how the various parameters of the model vary over a complete period for a typical turn. The amplitudes have been scaled as necessary for this figure.

```
choose($\omega_0$, $\theta_{max}$)
while (TRUE) {
        $\alpha_v = -\omega_0^2\theta_{max}\sin(\omega_0 t)$
        $C_f = f(\alpha_v)$ (as per equation 1)
        numerical integration of $x_f, y_f, \theta_{dir}, \theta_v, \omega_v$
        $F_{leg} = \sqrt{F_x^2 + F_y^2}$
        $\phi = f(\theta_v, \omega_v, \theta_{max}, \omega_0)$
        display($x_f, y_f, \theta_{dir}, \theta_v, \omega_v, C_f, F_{leg}, \phi$)
        $t = t + \Delta t$
}
```

FIG. 4. Pseudocode for open-loop control of turn.

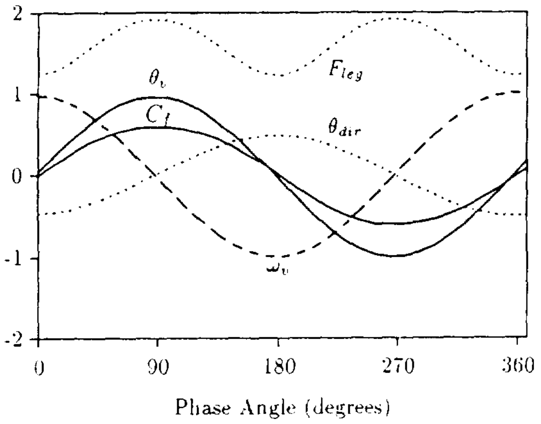


FIG. 5. A typical turn.

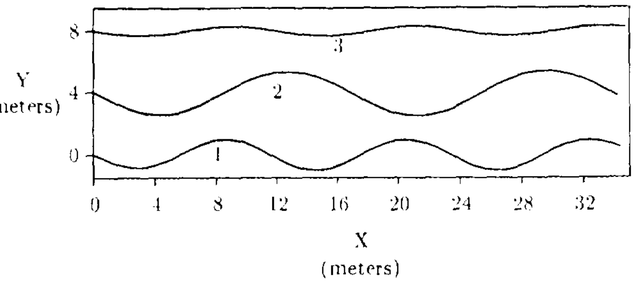


FIG. 6. Turning examples.

This figure also gives an insight as to why turning sports can seem effortless. The angle of body lean, θ_v, and the curvature of the foot, C_f, must be in phase to cause the appropriate turning motion. This happens naturally in turning sports because of the construction of skis, surfboards, and skateboards. Tilting any of these devices causes a curved motion proportional to the angle of tilt. Because θ_v and C_f are always in phase, the motion requires little effort to sustain, although it may in practice be difficult to initially achieve. This means that on a skateboard or snowboard, one need use very little ankle motion to tilt the board because the tilt angle between one's body and the ground is very close to giving the board the desired tilt angle already.

Figure 6 shows several tracks left by the "foot" of the physical model for various values of ω_0 and $\theta_{\max}$. Phase ellipse 1 has $\omega_0 = 2\pi/2.0$, $\theta_{\max} = 23°$; phase ellipse 2 has $\omega_0 = 2\pi/3.0$, $\theta_{\max} = 23°$; and phase ellipse 3 has $\omega_0 = 2\pi/2.0$, $\theta_{\max} = 7°$.

To summarize, the control variable C_f is calculated by determining the curvature necessary over time to oscillate the body in a sinusoidal fashion.

5. PARAMETERIZED CONTROL

In this section we look at how to move from one phase ellipse to another and how to control the general direction of motion for slalom-type turns. The controller that we will generate has three parameters: ω_0, $\theta_{\max}$, and θ_h. The first two define the phase ellipse, and the last defines the general direction of motion. Using the parameterized controller requires specifying the values of these three parameters over time.

5.1. *Control of Turn Frequency and Sharpness*

We shall first present a method of taking the system from any current state onto a desired phase ellipse. The control has the purpose of allowing the turning body to move from one phase ellipse to another, as well as closed-loop control once on a phase ellipse. A simple solution can be obtained if we look at the sets of phase ellipses that share the same major and minor axes, as shown in Fig. 7. These ellipses can be used to take the system onto target phase ellipses, shown in bold.

The ellipses in Fig. 7a can be used for states in quadrants 1 and 3 to bring the turning figure onto the target ellipse defined by ω_0 and $\theta_{\max}$. Similarly, the ellipses in Fig. 7b can be used for states in quadrants 2 and 4. We thus need to calculate the values of ω_0' and $\theta_{\max}'$ defining the controlling phase ellipse that will bring the state of the turning figure onto the desired ellipse.

For quadrants 1 and 3, $\theta_{\max}' = \theta_{\max}$, and $\omega_0' = \omega_v/\theta_{\max} \cos\phi$. For quadrants 2 and 4, $\omega_0' = \omega_0/\beta$ and $\theta_{\max}' = \beta\theta_{\max}$ where the value of β can be calculated from Eq. (3) as $\beta = \theta_v/\theta_{\max} \sin\phi$.

Once the controlling ellipse is known, Eq. (5) is used with these newly calculated values to determine α_v, and hence the desired control value C_f, as before. The controlling ellipse is recalculated at every time step because $\theta_{\max}$ and ω_0 are changing functions of time specified by the animator.

Although Fig. 7 implies that the desired ellipse can always be reached within 90° of phase, bounds placed on acceptable values of C_f and also dC_f/dt usually lengthen the time required to reach the desired phase ellipse. These bounds reflect the physical bounds on the curvature and rate of change of curvature of skis and wheels.

The preceding solution does not work equally well in all regions of the $\omega_v\theta_v$ plane. States having $|\theta_v| > \theta_{\max}$ or $|\omega_v| > \omega_0\theta_{\max}$ do not lie on any of the sets of ellipses shown in Fig. 7. For these regions one can directly assign values of α_v that drive the system towards the desired ellipse: $\alpha_v = \alpha_{v_{\max}}$ for quadrants 2 and 3, and $\alpha_v = -\alpha_{v_{\max}}$ for quadrants 1 and 4.

The origin, representing a perfectly balanced vertical state, is a singularity where there are also no proper controlling phase ellipses. A fixed arbitrary value is assigned to α_v for a small region around the origin.

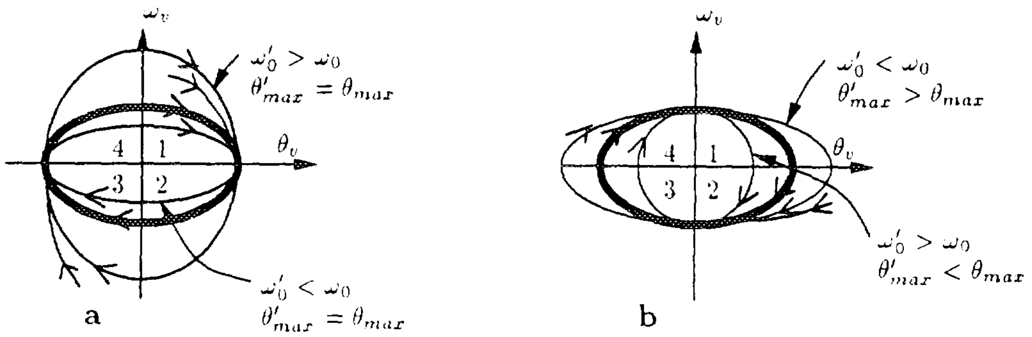


FIG. 7. Sets of phase ellipses sharing axes with a desired phase ellipse.

5.2. *Directional Control*

The last useful parameter is the direction. Figure 8 shows an example of a change in direction of $\Delta\theta_h$. We shall use θ_h to represent the desired direction that the animator specifies as a function of time. It indicates the direction with respect to some fixed axis.

In Fig. 8 we see that a change in direction to the left is performed using the short right-hand turn found at $x = 26m$. Similarly, a small change in direction to the right can be effected by making a sharper, longer right-hand turn or alternatively, making a shallow, shorter left-hand turn. This method is consistent with our experience, and we shall use it to perform our directional control.

Let the instantaneous direction be given by the function $\theta_{dir}(\omega_0 t)$, which is modulated by foot turning, as specified by C_f. In our control scheme thus far, we use the turning of the foot to control the body lean (e.g., θ_v and ω_v), and not to purposely change direction. The maximum change in direction over a left turn is given by

$$\Delta\theta_{dir} = \theta_{dir}(180°) - \theta_{dir}(0°). \tag{6}$$

In order to effect a change of heading of $\Delta\theta_h$ in a single turn, as shown in Fig. 8, we require

$$\Delta\theta'_{dir} = \Delta\theta_{dir} + \Delta\theta_h. \tag{7}$$

The modified turn should be continuous with respect to the turns performed before and those that must be resumed afterward. We shall assume here that the turns before and after the change in direction are the same, and are specified by ω_0 and θ_{max}. Figure 9 shows the phase diagram of a modified turn. As with the ellipses of Fig. 7b, the modified turn is defined by a number β. $\beta > 1$ results in a sharper, longer turn than the original. Similarly, $\beta < 1$ results in a shallower, shorter turn than the original.

$\Delta\theta_{dir}$ is a function of ω_0 and θ_{max}. Rewriting Eq. (7) to reflect this gives

$$\Delta\theta_{dir}\,(\beta\theta_{max}, \omega_0/\beta) = \Delta\theta_{dir}\,(\theta_{max}, \omega_0) + \Delta\theta_h. \tag{8}$$

The only unknown variable in this equation is β, which specifies how to modify the turn such that the appropriate change in direction is made.

Obtaining the function $\Delta\theta_{dir}(\theta_{max}, \omega_0)$ in an analytical form is difficult. Using first-order approximations for sin and cos, a simplified expression for the curvature can be obtained from Eq (1). It can be verified in Fig. 5 that the curvature does indeed approximate a sine function, as indicated by Eq. (9). $C_f(t)$ can be integrated to obtain $\theta_{dir}(t)$, as shown in Eq. (10). The velocity of the foot of the model, V_f, is assumed to be constant in order to be able to evaluate this integral:

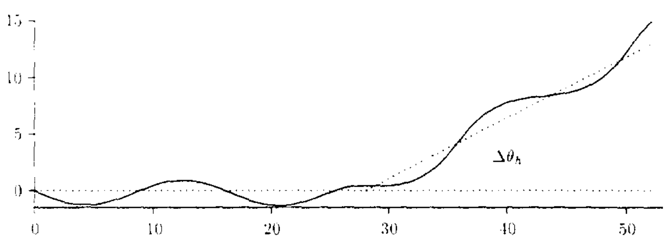


FIG. 8. An example of direction control.

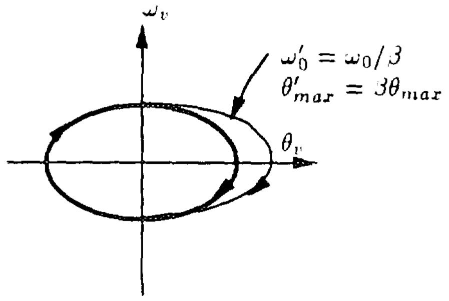

ω_v
$\omega_0' = \omega_0/\beta$
$\theta'_{max} = \beta\theta_{max}$
θ_v

FIG. 9. Phase diagram of a modified turn to change direction.

$$C_f = \frac{\theta_{\max}\sin(\omega_0 t)}{V_{cm}^2}\left(g + \frac{\omega_0^2}{lm}(I_v + l^2 m)\right) \tag{9}$$

$$\theta_{\mathrm{dir}}(t) = V_f \int C_f(t)dt \tag{10}$$

Equation (8) can now be solved directly for β, giving the analytic expression that we seek:

$$\beta = \sqrt{1 + \frac{\Delta\theta_h V_f \omega_0}{2g\theta_{\max}}}. \tag{11}$$

By inspection, we can see that to change direction to the left during a left-hand turn ($\Delta\theta_h > 0$), a value of $\beta > 1$ is required. This is the intuitive result expected, namely that a longer ($\omega_0' = \omega_0/\beta$) and sharper ($\theta_{\max}' = \beta\theta_{\max}$) turn is required. To change direction during a right-hand turn, β should be calculated using

$$\beta = \sqrt{1 - \frac{\Delta\theta_h V_f \omega_0}{2g\theta_{\max}}}. \tag{12}$$

During sharp turns ($\omega_0 > 2\pi/2.0$ and $\theta_{\max} > 10°$), the approximations made during the above derivation cease to hold, and it is necessary to build a two-dimensional table storing $\Delta\theta_{\mathrm{dir}}$ for various values of ω_0 and $\theta_{\max}$. Equation (8) can then be solved using Newton–Raphson iteration for the value of β.

In applying the directional control, a modified turn is calculated every time the state of the turning figure passes through the 0° or 180° points. The phase-ellipse control then uses the phase ellipse of the modified turn as being the target phase ellipse. Because of our assumption that the phase ellipses before and after the modified turn are the same, large instantaneous changes to the ω_0 and $\theta_{\max}$ parameters should be avoided.

Some examples of results obtained using the parameterized control are shown in Figure 10. Track 1 shows a set of figure-8 tracks created by two figures turning out of phase and is similar to those seen in skiing. Track 2 shows turns of gradually decreasing frequency. Track 3 shows turns of increasing sharpness. Lastly, track 4 shows the direction being controlled. It should be noted that the directional control is important for tracks 2 and 3 as well, because the process of moving onto a new phase ellipse would normally result in an unwanted change of direction. Photo 2 shows an animation sequence with a snowboarder mapped onto the physical model. A videotape of our animation results is also available [10].

The parameterized controller is computationally inexpensive and can be used in real time. The bounds placed on C_f, the foot curvature of the turning figure will depend largely on the activity being simulated. A bicycle front

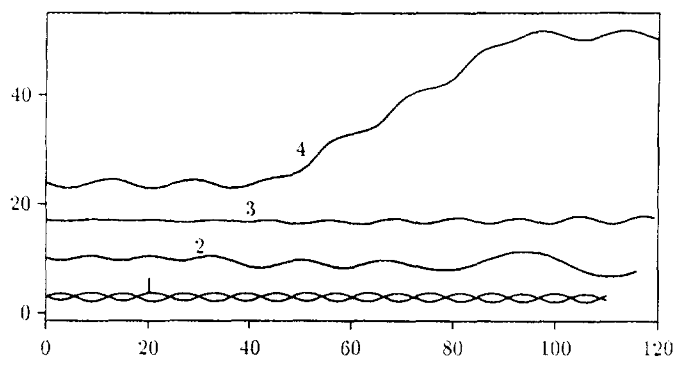


FIG. 10. Tracks for turning using parameterized control.

wheel can be made to change direction very quickly and turn sharply as well. A ski or a surfboard, however, can neither change direction as quickly, nor turn as sharply.

6. PATH-FOLLOWING CONTROL

An alternative method of control is to have the animator draw a path for the turning body to follow. It is impossible, however, for a turning body to exactly follow an arbitrarily path. If the foot is made to exactly follow a drawn path, this fixes the curvature, which leaves no freedom to control the body balance.

Our solution is to treat the motion synthesis as a constrained finite-time optimization problem. The optimization function to be minimized is the sum of the distances from points on the track of the turning body to the closest point on the desired path. The optimization problem is solved using a branch-and-bound search. During the search, any solutions resulting in collisions with obstacles are also eliminated.

The problem is specified by the animator in a graphical fashion, as shown in Fig. 11. The desired path and any obstacles are drawn using an interactive editor. We choose to use a piecewise cubic Hermite spline to draw the desired path, but a series of connected straight-line segments could be used equally well. The bold vectors in the figure represent the tangent vectors at the endpoints of each spline segment, assuming continuity of the curve and its tangents at the endpoints. A minimum tolerance

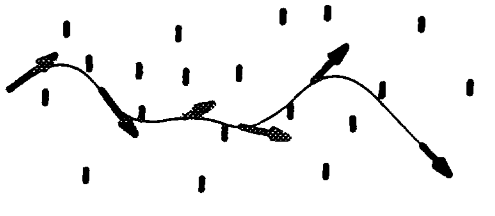


FIG. 11. User specification of path to follow.

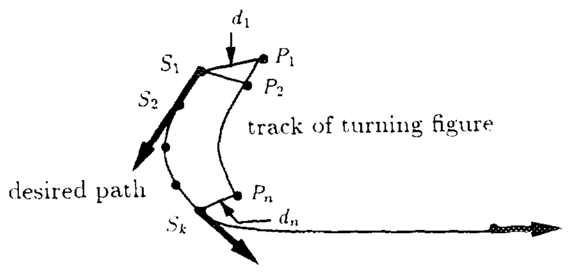


FIG. 12. Evaluation of the optimization function.

minseg	spline segment to begin search for shortest-distance points
x, y	coordinates of the foot
θ_{dir}	direction
θ_v, ω_v	state of lean
fopt	value of optimization function
stage	number of stage
C_f	foot curvature
dC_f/dt	value of dC_f/dt taken at stage 1

FIG. 13. Node information.

can be specified for avoiding obstacles. The speed can be made to vary as a function of the terrain if desired. The desired path exerts its effect only through the optimization function, which is simple to calculate, as we shall now see.

Before describing the optimization algorithm in detail, we shall describe how the optimization function is evaluated. Let d_i be the shortest distance from the position of the foot at time step i to the spline path. Instead of explicitly solving for the shortest distance, we approximate it by sampling spline segments at regular intervals and taking the minimum distance to one of the samples. Thus we have $d_i = \min_j \|P_i - S_j\|_2$. P_i is the position of the foot at time step i and S_j is point sample j of the spline segments.

The value of the optimization function is determined by sampling the position of the track of the turning body and summing over all n such samples, namely $f_{\text{opt}} = \sum_{i=1}^{n} d_i$. Figure 12 illustrates the points and distances involved. Typically 12 sample points per spline segment and a time-step of 0.1 s for sampling the track has produced good results.

6.1. *The Optimization Algorithm*

For the parametric controller, we calculate α_v and use Eq. (1) to calculate C_f. Here, however, we shall be determining C_f directly. The algorithm uses a branch-and-bound search to explore how C_f should be controlled by planning ahead for a fixed interval of time. This fixed interval of time (typically 3 s) is divided into N discrete stages, where the control variable is maintained constant during each stage. We shall use dC_f/dt as our control variable because we wish to have C_f be C^0 continuous.

The branch-and-bound technique determines the optimal values of C_f to be used during the next N stages. Only the value for the first stage is actually used, however. When the end of the first stage is reached, the branch-and-bound process is repeated. In this way each control decision is based upon a plan looking N stages into the future.

The branch-and-bound search works as follows. In the first stage, the effects of several values of dC_f/dt are simulated for t_{stage} seconds, beginning at the current state. In the second stage, the same process is repeated for each of the states produced by the first stage. This process repeats until the desired maximum number of stages have been evaluated. Let N be the maximum number of stages. Without bounding, the branching process will result in q^N possible ending states, where q is the number of values of dC_f/dt tried at each stage. Fortunately, we shall see that many branches can be clipped or bounded at an early stage.

Let the starting state and ending states of stages be called nodes. Each node contains the information shown in Fig. 13. The pseudocode for the branch-and-bound algorithm as it pertains to the path-following problem is shown in Fig. 14. The algorithm is used once every t_{stage} seconds to calculate the required control for the next stage. The nodes are kept in an evaluation queue sorted by their f_{opt} values. The algorithm always expands the node having the minimum value of f_{opt} first. Its children are placed in the node queue if they are not clipped. Nodes are clipped if their f_{opt} value exceeds that of the current upper bound, found by keeping the current best (lowest) f_{opt} value of all completed paths.

Nodes can also be clipped as a result of a collision with an obstacle or the body falling over (a bound is placed on θ_v for this purpose). Obstacle avoidance is thus obtained by checking the track segments for collisions as they are produced. A collision is said to have taken place if a produced track segment intersects any obstacles.

```
enqueue(current state)
while (nodes in queue)
        dequeue(node)
        calculate inputs to be applied
        for each input
            end_node = simulate(node,input,$t_{stage}$);
            if (hit obstacle) continue
            if (fell over) continue
            if (exceeded bound) continue
            if (at last stage)
              update bound if necessary
              continue
            enqueue(end_node)
return( $dC_f/dt$ taken at stage 1 of best path )
```

FIG. 14. Pseudocode for branch-and-bound algorithm.

Typical values used for the algorithm are as follows: $t_{\text{stage}} = 0.3$ s, N = 10 to 12 stages. θ_v bound = 30°, and q = 5 to 7 values of dC_f/dt. The clipping is generally quite effective in restricting the number of nodes that need to be searched.

The algorithm uses various values of dC_f/dt to expand the nodes in the tree and thus produces an optimal value of dC_f/dt to use. Using the derivative of the curvature as a control variable allows for C^0 continuity of the curvature and will thus produce smooth motions. In generating values of dC_f/dt to apply, bounds are placed on dC_f/dt and C_f to ensure realism.

6.2. *Calculating the Phase*

The current phase, ϕ, of a motion is very useful in creating proper motions for the display model. Because the process of determining the phase requires information about the future motion of the turning figure, we produce the phase in a post-processing pass after the optimal path has been determined. Using an estimate of ω_0 and $\theta_{\max}$ as defined for the parametric controller, we can calculate the phase as before.

Estimates of ω_0 and $\theta_{\max}$ can be obtained by noting the points at which the state of the turning body enters and exits the current phase quadrant. This is illustrated in Fig. 15. The magnitude of the intercept along the θ_v-axis serves as an estimate of $\theta_{\max}$. Similarly, the magnitude of the intercept along the ω_v-axis serves as an estimate of $\theta_{\max}\omega_0$, which thus allows us to estimate ω_0.

6.3. *Results*

Various examples of the use of the path-following algorithm are shown in Figs. 16–18. Figure 16 shows the results of an automatically synthesized tree-skiing path. The solid line is the spline path to be followed. The dotted path is calculated using $N = 10$ stages. The dashed line shows the result for $N = 12$ stages. In this case planning ahead an extra two stages considerably changes the motion by choosing a different route through the trees.

The example in Fig. 17 shows how the path-following algorithm can be used to solve for a path through a slalom course. The poles are drawn as lines in order to restrict passage to along one side only. Some additional obstacles are added to speed up the solution. The solid line is given as the path to follow, and the dotted line is the calculated solution. The solid line is given as the path to follow, and the dotted line is the calculated solution. Photo 3 shows an animation sequence of a skier mapped onto a path-following solution for a slalom course. Photo 4 shows a telemark skier passing through a portion of the same slalom course and making use of the same solution.

An example of a bicyclist going around a corner is shown in Fig. 18. The edges of the road are drawn as obstacles. Additional obstacles are drawn blocking the bottom and left exits from the intersection in order to aid in restricting the size of the search tree. Photo 5 shows

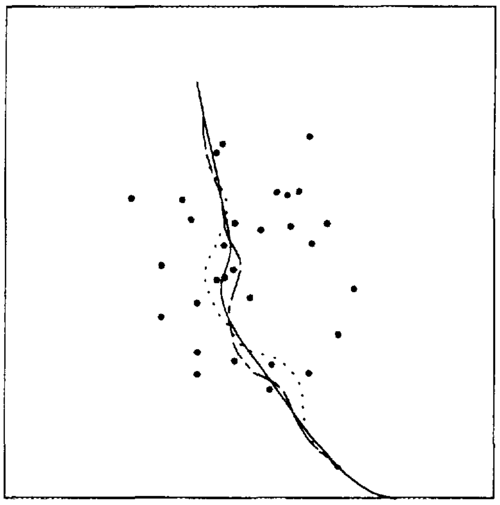


FIG. 16. Tree skiing.

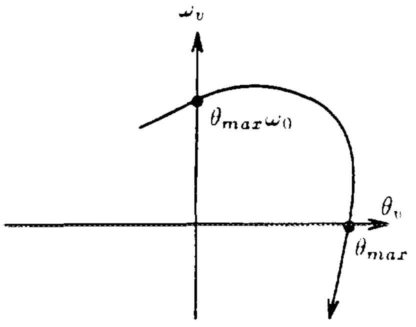


FIG. 15. Estimating $\theta_{\max}$ and ω_0 from a phase diagram.

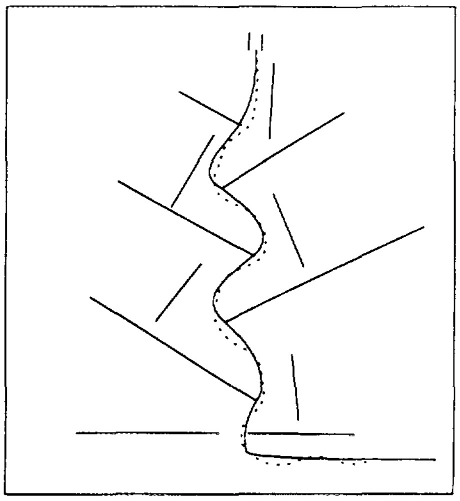


FIG. 17. The slalom race.

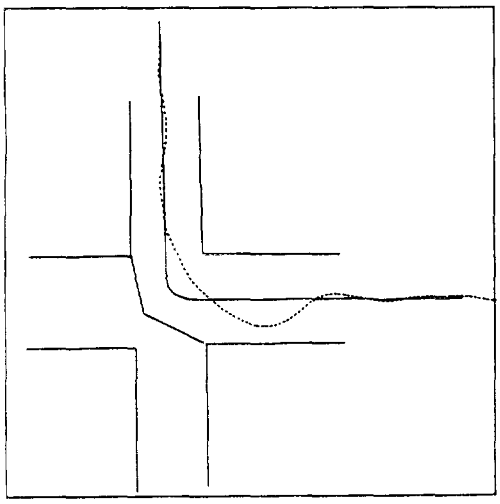


FIG. 18. Bicycling around a corner.

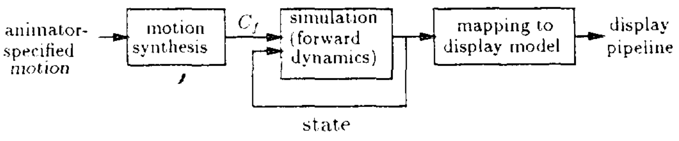


FIG. 19. Control, simulation, mapping, and display.

a bicyclist weaving through a series of potholes on a road. Photo 6 shows an animation sequence from the high-speed bicycle turn shown in Fig. 18.

7. MAPPING A DISPLAY MODEL ONTO THE PHYSICAL MODEL

Many body parts are not directly represented in the physical model. The realistic motion of many of these parts would be difficult to synthesize using a physical model because of the lack of obvious criteria for their synthesis. The motion of these parts is nevertheless coordinated with the main motion of the body. We shall make use of variables associated with the simplified physical model to drive these motions. This represents a type of *motion amplification* by defining the relationships that coordinate the motion of all the body parts with that of the simplified physical model. The most useful variable in this regard is ϕ, the current phase of the turn. The force exerted on the leg of the physical model and several other variables are also used in the mapping.

Figure 19 shows at what point the mapping process takes place. The input to the mapping process consists of the variables shown in Fig. 20. The creation of a suitable mapping procedure typically involves some experimentation, especially if biomechanical data from real motions is not readily available. We shall now briefly describe how the mapping was performed for several figures.

Alpine skiers clearly have an up-and-down motion during turns. This is achieved in the display model by making the legs act as a spring. The force on the leg of our simplified model varies approximately ±30% from the static force during typical turns. Figure 5 shows how the force on the leg varies during a typical turn. A suitable spring constant k_s can be chosen based upon how much up-and-down motion is desired. Given the force at any instant and k_s, the leg length is determined and inverse kinematics is used to calculate the bend of the knees. The bend at the waist is directly correlated to the bend at the knees, resulting in a coordinated motion. A linear relationship is assumed between the curvature of the ski and the angle at which the ski is tilted. The constant to use is easily estimated from a plot like the one in Fig. 5. The tilt of the outside, weight-bearing ski is made to be slightly larger than that of the inside ski. When the tilt angle of the body, θ_v, and that of the ski do not correspond, we need a way of tilting the ski on edge with respect to the body. This is accomplished by twisting the leg to the inside or outside. The pole planting motions of the arms are coordinated with the phase so that the pole plant occurs before the actual turn, at $\phi = 30°$ and $210°$. A twisting motion of the upper body is made to be in phase with C_f.

The snowboarder is similar in many respects to the skier. The bend at the knees and waist are calculated based upon the leg force, as with the skier. The upper body is made to twist in phase with C_f. Tilting the snowboard with respect to the body is easily accomplished by bending at the ankles. The arms are moved to lead the twisting motion of the upper body slightly.

The bicyclist requires only a simple calculation to determine how the front wheel should be turned to achieve the given curvature C_f. Inverse kinematics is used to keep the hands on the handlebars. The pedals move as a function of the distance travelled. Inverse kinematics is used to keep the feet on the pedals.

Turning for a telemark skier is achieved by separating the skis and placing them at an angle with respect to each other. Inverse kinematics defines the position of the legs. The remaining motions are similar to that of the skier.

x_f, y_f	foot position
θ_{dir}	current direction
θ_v	lean-angle of body
C_f	foot curvature
F	force exerted on leg
ϕ	phase angle

FIG. 20. Input parameters for the mapping relation.

8. DISCUSSION

The path-following algorithm provides a method of specifying the motion in terms of the desired path of the foot. This provides a complementary solution to the parameterized solution, in which motion was specified using parameters related to the body motion. The two methods of control, namely a parameterized controller and an optimization-based controller, each have their uses. The tradeoffs between these two methods of control seen as used here are probably typical of their use in general.

The motion amplification method of using a simple dynamic model to drive a more complex display model represents a type of hierarchical control. One could further use the final kinematic positions produced by the display mapping as the set-points for controllers in a detailed and complete physical model. We have not yet experimented with such a scheme, but we believe it is feasible. It is not entirely clear what additional gains in the quality of the motion would be attained in return for the additional complexity. One of the general limitations of the kinematic display model mapping is that the motion can seem too perfect. This would be especially evident with more frequent and spurious interaction with other objects or phenomena in the environment.

Many subtle effects that occur in various turning sports are not included in our model. We have described the basic physics that these sports have in common and a common method of control. Further detailed analysis of turning motions is possible, but the results then become specific to the given sport. This is also another source of difficulty when considering the possibility of proceeding to a more detailed and complete physical model. Simulating the transmission of forces from the snow to a skier through a flexing, twisting ski is a difficult simulation problem. A full dynamic simulation of a bicyclist is probably easier to perform, but will not yield as many benefits over the kinematic display mapping. More complete dynamic simulation would likely yield benefits for turning figures moving quickly over bumpy and variable terrain. Such terrain would likely require controllers with some additional complexity over the ones presented here, however.

Modeling and control of turning motions across nonplanar terrain is a formidable challenge. Skiing and snowboarding can make good use of changes in terrain to facilitate turning. The problem here is one similar to the path-following problem in that any decision must be based not only upon the current state of the figure, but also the state of the environment. Experienced practitioners of these sports can rapidly make generalizations about how terrain will affect a planned motion. It is difficult as of yet, however, to efficiently synthesize and store these generalizations.

The branch-and-bound method used to perform the optimization might be useful in general as a control-synthesis technique. It is especially useful for systems with a small number of controllable inputs and a large number of degrees of freedom. In these cases it is more convenient to explore possible motions in the "input space" of the system rather than its state space.

In the future, it will useful to develop ways of automating the generation of parameterized controllers. Turning is such a common and reasonably-simple motion that it is worthwhile to analytically determine a parameterized controller, as we have done here. In general, however, the synthesis of such a controller is still a laborious task that is preferably automated.

APPENDIX A

The following is a brief summary of the equations of motion of the turning model.

A free-body diagram of the forces and moments acting on the body is shown in Fig. 21. The equations of motion of the body are given below. F_x and F_y refer to the horizontal and vertical components of the force exerted on the foot by the ground. I_v is the moment of inertia of the body about its centre of mass. α_v is the angular acceleration of the body and leg about the z-axis. a_{cm_x} and a_{cm_y} are the horizontal and vertical components, respectively, of the linear acceleration of the body. These terms are related as follows:

$$T = F_x l \cos\theta_v + F_y l \sin\theta_v = I_v \alpha_v \tag{13}$$

$$F_x = m a_{\mathrm{cm}_x} \tag{14}$$

$$F_y = m a_{\mathrm{cm}_y} + mg. \tag{15}$$

Equation (13) equates the sum of the angular moments in the xy-plane to the change in angular momentum, and Eq. (14) and (15) equate the sum of the forces to the change in linear momentum.

The acceleration of the center of mass can be expressed

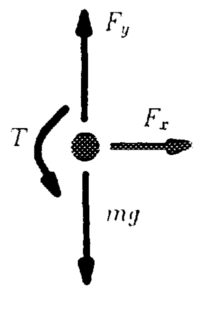


FIG. 21. Free body diagram.

relative to the acceleration of the foot, a_f, as in Eq. (16) below. Here, $a_{cm/f}$ is the acceleration of the center of mass with respect to the foot, and a_c is the coriolis force, which is zero in our case. The orthogonal decomposition of Eq. (16) is given in Eq. (17) and (18). ω_v is the angular velocity in the xy-plane. V_f is the forward velocity of the foot. r_f is the radius of curvature of the foot in the xz-plane, and represents our control variable. The last term in Eq. (17) is due to rotation of the body in the xz-plane:

$$a_{cm} = a_f + a_{cm/f} + a_c \tag{16}$$

$$a_{cm_x} = a_{f_x} - l\cos\theta_v\alpha_v + l\sin\theta_v\omega_v^2 + l\sin\theta_v\frac{V_f^2}{r_f^2} \tag{17}$$

$$a_{cm_y} = a_{f_y} - l\sin\theta_v\,\alpha_v - l\cos\theta_v\omega_v^2. \tag{18}$$

Substituting Eq. (17) and (18) into Eq. (14) and (15), and then using the results in Eq. (13), gives

$$I_v\alpha_v = mll_c(a_{f_x} + l_s\frac{V_f^2}{r_f^2} - l_c\alpha_v + l_s\omega_v^2) + ml_s(a_{f_y} - l_s\alpha_v - l_c\omega_v^2 + g), \tag{19}$$

where $l_c = l\cos\theta_v$ and $l_s = l\sin\theta_v$.

Assuming that the foot is travelling on a planar surface, we can write $a_{f_x} = -v_f^2/r_f$ and $a_{f_y} = 0$. Substituting these into Eq. (19) and combining terms gives the following result:

$$\frac{I_v\alpha_v}{ml} = -\frac{V_f^2}{r_f}\cos\theta_v + l\sin\theta_v\cos\theta_v\frac{V_f^2}{r_f^2} - l\alpha_v + g\sin\theta_v. \tag{20}$$

Finally, because we assume that the forward velocity, V_{cm}, is known, we can relate the forward velocity of the foot to that of the center of mass as follows:

$$V_f = V_{cm}\frac{r_f}{r_f - l\sin\theta_v}. \tag{21}$$

Using the result of Eq. (21) and (20), we can arrange the terms to form a quadratic equation in terms of the curvature of the foot, C_f, where $C_f = 1/r_f$. This result is Eq. (1).

$$0 = aC_f^2 + bC_f + c, \tag{22}$$

where

$$a = l^2\sin^2\theta_v k + lV_{cm}^2\sin\theta_v\cos\theta_v$$

$$b = -2l\sin\theta_v k - V_{cm}^2\cos\theta_v$$

$$c = k$$

$$k = g\sin\theta_v - \frac{\alpha_v}{ml}(I_v + l^2m).$$

In the expressions for a, b, and c, k is a constant useful in simplifying the equation. The negative root of Eq. (1) is used, as it can be shown that this leads to the correct result of zero curvature necessary to obtain zero angular acceleration when the body is already vertical. The equation allows us to calculate C_f give α_v and vice versa.

APPENDIX B: LIST OF SYMBOLS

θ_v	Angle of model with respect to vertical
ω_v	Time derivative of θ_v
α_v	Time derivative of ω_v
g	Gravity
r_f	Radius of turn for the foot
C_f	Curvature of turn for the foot, $1/r_f$
l	Height of center of mass
I_v	Moment of inertia about z axis
m	Mass of body
ω_0	Frequency of oscillation
θ_{max}	Amplitude of oscillation
ϕ	Phase angle of turn
x_f, y_f	Foot location in world
β	Turn modifier
θ_h	Desired direction
θ_{dir}	Instantaneous direction
a_{cm}	Accleration of center of mass
T	Sum of moments about center of mass
V_f	Velocity of foot
V_{cm}	Velocity of center of mass

REFERENCES

1. M. H. Raibert and J. K. Hodgins, Animation of dynamic legged locomotion, in *Siggraph 1991, Proc. ACM* **25**(4), 1991, 349–358.
2. A. Bruderlin and T. W. Calvert, Goal-directed dynamic animation of human walking, in *Siggraph 1989, Proc. ACM* **23**(4), 1989, 233–342.
3. M. McKenna and D. Zeltzer, Dynamic simulation of autonomous legged locomotion, in *Siggraph 1990, Proc. ACM* **24**(4), 1990, 29–38.
4. L. S. Brotman and A. N. Netravali, Motion interpolation by optimal control, in *Siggraph 1988, Proc. ACM* **22**(4), Aug. 1988, 309–315.
5. A. Witkin and M. Kass, Spacetime constraints in *Siggraph 1988, Proc. ACM* **22**(4), Aug. 1988, 159–168.
6. M. van de Panne, E. Fiume, and Z. Vranesic, Reusable motion synthesis using state-space controllers, in *Siggraph 1990, Proc. ACM* **24**(4), 1990, 225–234.
7. G. Miller, Goal-directed animation of tubular articulated figures or how snakes play golf, in *Making Them Move*, (Norman L. Badler, Brian A. Barsky, and David Zeltzer, Ed.), Morgan Kaufmann, San Mateo, California, 1991.

8. H. Sodeyama *et al.*, Study of the displacement of a skier's center of gravity during a ski turn, in *International Series on Biomechanics, Proceedings of the 5th International Conference of Biomechanics*, Univ. Park Press, 1976.
9. M. W. Lee and T. L. Kunii, Animation design: A database-oriented animation design method with a video image analysis capability, in *Proceedings of Computer Animation '89*, pp. 97–112, 1989.
10. M. van de Panne, *Physically-Based Turning Animations*, Dynamic Graphics Project, Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4, 1992.