[12] K. Fukunaga and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, pages 750–753, 1975.

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[14] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the Americal Statistics Association*, 68:361–368, 1973.

❑

Synthesis of Realistic Motion for Legged Creatures

V.V.KAMAT
Goa University, Goa
email: vvkamat@unigoa.ernet.in

Introduction

Since the first silent film flickered to life, mankind has been fascinated with motion pictures. No other contemporary message communication medium has the same visual impact as the movie or the video. To this day, 3D animated characters or motion video is the visual element in computer based presentations that can draw gasps from a crowd in a show or exhibition, or hold the student's attention and interact in a computer aided tutoring system. While digital video relies entirely on capturing and playing back the visual dynamics existing in the real world, animation is concerned with the creation and playback of the visual dynamics of an artificially created world within the realms of the digital computer.

In recent years computer animation has been experiencing increased popularity, especially in the advertising and film industry. Today, the technology allows an artist or a scientist to visualize what was previously possible only in the realm of imagination. In no movie is this better illustrated than in a film such as *Jurassic park*, where realistically modelled 3D giant sized dinosaurs, complete with textured skin and limbs, walk, run or gallop at roaring speeds in a most realistic fashion.

The movement in animation is really just an illusion. Like in a movie, an animation is just a series of still images that are displayed in sequence at a fixed rate. Animations, or for that matter movies or video, are possible because of a biological phenomena - persistence of vision. The image on an eye's retina is retained for a short while, even after the object is moved out of view. If a series of images, slightly differing from

each other are shown in rapid succession, the eye/mind blends these successive images to result in the visual illusion of movement or change. Typically, movies are shot at a shutter rate of 24 frames or pictures per second. Ten minutes of animation would thus require as many as 14400 separate frames or pictures; each one differing only slightly from its predecessor.

Traditional computer animation

Animation can be considered to be of two kinds. 2D or cel-based animation and 3D or model based animation. To this day, 2D cel-based animation is the most common kind, where flat images are hand-drawn and painted one frame at a time. While this is very time consuming, cel animation has produced spectacular results. Any animated Disney film is a fine example of this. Increasingly, computer graphics techniques are being used for improving the sketching, inking and colouring process in cel animation [4].

In 3D animation, a mathematical model of the three dimensional object is created and realistically rendered in each frame using sophisticated image synthesis techniques. For inanimate rigid objects, animation would involve a series of transformations of object position and orientation. As a result for such objects modelling and rendering can be much more time consuming than synthesizing their motion.

On the other hand, animating the movement of legged creatures is very complex; as it has to be physically valid and also appear realistic - a dog like creature must move like a dog. In the rest of this paper we shall specifically address the techniques for this.

The traditional computer animation methods are focused on providing a rich set of geometric primitives to model shape and motion. To support animation, the models are parameterized into a hierarchy such that animating the model is then a matter of varying the parameter value over time. *Keyframe animation* is one such technique [3]. For a legged creature a frame in animation is basically a description of the particular position of the object at a particular instance in time. The position refers to the *degrees of freedom* (DOF), or generalized coordinates that define the object's position in space. Degrees of freedom are the minimal set of parameters that can specify the complete configuration of the object in space. For example, a single body in a plane has three DOFs. Two coordinates to define a reference point (x, y) on the body and one coordinate θ to define its orientation. With these three generalized coordinates, it can be translated and rotated to any configuration in a plane.

In a keyframe animation system, the animator need not describe each frame. Instead he describes a set of 'keyframes' from which the animation system can geometrically interpolate each DOF independently, to automatically obtain the frames in between. Keyframing is a powerful and still widely used technique in the animation industry. It allows the animator to have complete control over the depicted motion. A variety of techniques has been devised to interpolate motion between keyframes. Linear interpolation is very simple to implement but may lead to jerky motion (the velocity being discontinuous). There are methods based on spline fitting which make use of non-linear interpolation to generate smoother motion.

There are two major problems with keyframe animation systems. Firstly it puts a large burden on the animator to adjust too many parameters at very fine levels of detail. For example, consider the motion specification problem of an articulated object such as a human or an animal. For a reasonably detailed figure with 30 DOFs, a minute of animation with a keyframe every quarter of a second, would require approximately eight thousand values to be specified. This is perhaps an impossible task. Secondly, to generate very convincing looking motion, the animator requires to have a very good understanding of the motion and artist like skills, for resynthesizing the internalised motion. Therefore more often than not, even after many trials, synthesized motion tends to look unrealistic and puppet like.

Motion is a result of forces

In the late 1980's, researchers working in the field of computer animation were convinced that if the animation has to look realistic, the physics behind the motion has to be taken into account [1, 16, 7]. Moreover at that time researchers from

computer graphics were already using physics to model interaction of light with the environment to produce photorealistic effects. Initial results on incorporation of physics to produce life like motion were very encouraging. This is typically done by augmenting the traditional geometric model to include other physical characteristics that computers can use to compute motion. Some of these physical characteristics are mass of the body, its moment of inertia, and external forces such as gravity, frictional forces etc. The idea is to incorporate appropriate physical complexity and realism of the behaviour into the model itself, rather than requiring that it be imposed by the animator.

- Already, some of the early research into physically based modelling and animation is finding application in commercial software. For example, gravity, friction, and wind help animators create nearly automatic animation of characters and objects in Softimage's work-station based, 3D animation software program called *Actor*. Also using Knowledge Revolution's *Interactive Physics*, students and animators are able to simulate 2D objects and elements such as ropes, motors and pulleys that move according to the physical laws [12].

In computer animation, the objects of interest may vary from rigid body to flexible or deformable body, single body to articulated body, passive body to active body . A rigid body is one that does not change shape over time. On the other hand a deformable body changes shape. A passive object is an inanimate object which behaves according to the forces acting upon it but has no internal mechanism to bring about its own motion. On the other hand active bodies can bring about their own motion without external forces. Articulated bodies are composed of multiple rigid links that are connected to each other via joints. A simple articulated body with 5 DOF is shown in Figure 1. Many articulated bodies have actuators placed at some of its joints which make them active. The role of an actuator is similar to that of a muscle in a biological creature. For animation purposes, simulated actors resembling humans and animals are created out of articulated bodies and then automatically fleshed with deformable elements to give the effect of bones, flesh, skin. The ultimate goal of physically based animation is to be able to control and synthesize complex motions involving simulated actors. The role of

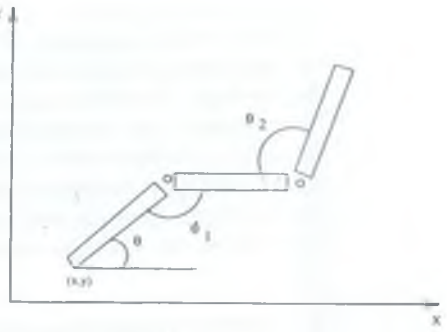

Figure 1: A planar articulated body with three links and two rotary joints

the animator should be that of a director rather than that of one who must implement every detail of animation. Just as the dance choreographer relies on the ability of the individual dancer , the animator should be able to rely on the abilities of the simulated actors involved in a scene.

We see living beings all around us, whether they are people, animals, birds or fishes. From the smallest of these creatures to the largest of them; they all exhibit tremendous skill in coordination and balance in moving about. Their movements seem amazingly simple, yet as we shall see later are very demanding to reproduce. In everyday life, we execute many tasks involving complex motion without feeling burdened or knowing how we do what we do. The goal of physically based animation is to be able to reproduce such type of motion in the simulated environment of a computer. How is this different from creating dinosaurs and animating them? A film like Jurassic Park is a marvel of collective effort by artists, animators, computer graphics specialists and experts in animal motion. Not only did they create make believe animals on the screen, they were also able to demonstrate how well computer graphics techniques can be integrated into live action. However, there is no single technique used in the creation of these sequences. Rather a number of different strategies/techniques have gone into making dinosaur scenes. These include measurement from physical models, realistic motion data capturing , rotoscoping and traditional animation, all very cleverly used [11].

On the other hand, in physically based modelling and animation, we are talking about creating mathematical models of the objects which

follow laws of physics. To create mathematical models of the complexity of dinosaur and simulating them is beyond the means of the current technology. Therefore most of the creatures that we consider are relatively simple but still rich enough to demonstrate a wide variety of interesting motions. Implementing such scenarios requires creating simulated actors and bestowing upon them the skills to move about in a natural way.

The potential application of physically based animation is not restricted to the entertainment industry, it has already found applications in diverse fields such as industrial design, medical imaging and education.

Physically based modelling and animation is relatively a young discipline [1]. It allows the user to model complex motion in a realistic fashion. Common elements in all physically based animation are classical dynamics, interbody interaction and motion control. In the following sections, we discuss each of these elements briefly in relation to articulated bodies. Animation techniques concerning deformable objects are not however discussed here.

Dynamics

Dynamics is concerned with the formulation of equations of motion and their numerical solution. Typically these are second order differential equations and are solved using some of the standard techniques in numerical methods. There are many methods to formulate equations of motion [17]. But all are based on Newton's second law force equals mass times acceleration $F = ma$. Netwton's law implies that if the position, velocity and all the forces acting on the body are known at time t, one can compute it's position and velocity at time $t + \Delta t$ by numerical integration. See Figure 2. For an articulated body, the equations of motion are non-linear and are gencrally quite complicated to be derived by hand. These equations are symbolically evaluated, to yield differential equations which can be written in the form:

$$Ax = b$$

[1]The field was first named in a course in the 1987 ACM SIGGRAPH annual conference.

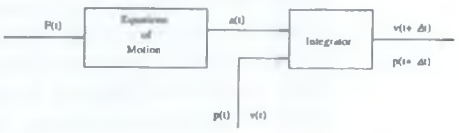

Figure 2: Numerical integration of equation of motion

where A and b are dependent on the internal torques produced by the actuators at the joints, external forces due to gravity, forces due to interaction with the environment, the physical properties of the links, and the state of the links. x represents the vector of unknown accelerations.

The accelerations are then numerically integrated to determine the new velocities and positions of the links.

The most straight-forward method of integration is using Euler technique. Given the state (position, velocity) of the system and the acceleration at time t, one can compute state at time $t + \Delta t$ by the simple formula given below:

$$p(t + \Delta t) = p(t) + v(t)\Delta t$$

$$v(t + \Delta t) = v(t) + a(t)\Delta t$$

Inter-body interaction

When several objects are moved about by dynamic simulation, there is a possibility that they will interpenetrate. In order to preserve the authenticity of simulation, it is essential that the computer detect collisions amongst objects and simulate object response appropriately. Collision detection is mainly a geometric problem involving the positional relationship of objects in the world. On the other hand, collision response is a dynamic problem, that involves predicting behaviour according to physical laws [9]. Collision detection involves checking at every time step t whether any two objects penetrate. This is computationally very expensive. The basic algorithm is $O(n^2)$ for n objects. A number of methods have been proposed to minimize the computational cost. A simple way to handle collision response is to introduce spring forces which prevent objects being penetrated. Thus whenever collision is detected, a very stiff spring is temporarily inserted

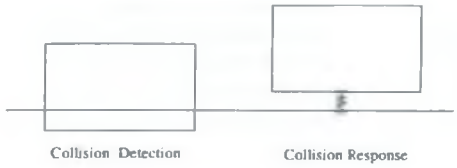

Figure 3: A collision and response

between the points of closest approach. See Figure 3. The spring law is usually K/d, that goes to infinity as the separation d of the two objects approaches zero. K is a spring constant controlling stiffness of the spring. The direction of the force is such that it pushes the two objects apart. The spring method is easy to understand but is computationally expensive. Stiffer springs mean stiffer equations of motion, which require smaller time steps for accurate numerical integration. A robust method of resolving collision is to use an analytical method [2]. The analytical method depends upon the conservation of momentum during collision. It may introduce discontinuities in the velocities when a collision occurs. This is typically solved by stopping the integrator at the time of collisions, resolving the collision by computing new angular and linear velocity for each body and then restarting the integrator with new initial conditions.

Motion control

Motion control is a fundamental problem in computer animation. One of the major tasks of the animator is to control the motion of the moving object according to the script of the animation. In physically based modelling, it means that proper torques have to be associated at the joints such that the desired motion is obtained.

Some of the earlier techniques required the animator to directly specify the torques, each as a function of time, then view the resulting animation and refine the torques iteratively to achieve the desired motion [1, 7]. This *explicit control* gives physically realistic trajectories, but the level of automation is low in terms of effort required to discover and refine acceptable motion. One of the important objectives of an animation system is to automate this task.

A common method of defining torques at the

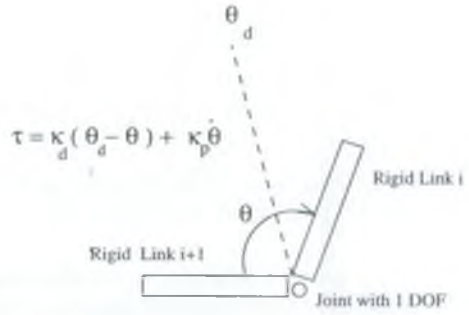

Figure 4: A planar joint with an actuator

joints is through the *proportional derivative* (PD) control law [10]. The control law acts like a simulated spring. The way it works is as follows. Whenever the current angle θ deviates from the rest angle θ_d of the spring , it applies torque on the two adjacent links according to the formula:

$$\tau = k_p(\theta_d - \theta) - k_d\dot{\theta}$$

Where k_p and k_d are spring and damping constants respectively. See Figure 4. The advantage of the PD controller is that the torque function gets automatically defined once the desired angle is specified. To execute a particular task, it is necessary to define a series of intermediate desired joint angles. Any coordinated task will involve a series of activation of desired angles for each active joint. In animation, it would mean that these desired intermediate angles need to be determined for executing a particular task. This is an inverse problem and does not have a unique solution. For example, if we have to reach for a cup of coffee, we can do so by moving our hand in many different ways. The problem is grossly underconstrained and can be solved by imposing additional constraints to define a unique solution.

There are two methods to solve this problem. In the first method we specify an approximate trajectory in terms of the position of the object in space and time [18]. This is very similar to specification of keyframes; except that we require them in fewer numbers. In addition, we specify desired initial and final states (position and velocity) and an objective function that would minimize or maximize certain criteria. The method uses the variational calculus technique which is very similar to gradient methods used for finding minima of an ordinary real-valued function. The method starts with the trajectory specified, and

Figure 5: A three state controller for generating hopping motion for a lamp

alters it slightly by moving certain points on the trajectory in some direction. Then a computation is made to determine whether the new trajectory is closer to a good trajectory, where good trajectory means laws of physics satisfied, low energy expenditure and starting and ending conditions are met. In case it is closer, it becomes the new trajectory and the entire step is repeated. Here the assumption is that animals and human beings move such that the energy expended is minimum. Although there is no proof for the argument, in theory, the results gathered to date do support the suggestion [5].

The second method tries to synthesize a controller which when actuated results into motion [14, 15]. See example in Figure 5. A controller is just a finite state machine (FSM). Each state of the controller determines its internal configuration.

Depending upon certain inputs, the FSM goes from one state to the next state (in the example shown in Figure 5, it is time). Associated with each state there is a set of control laws which specify the torques to be applied at the joints to bring about the motion. The quality of the generated motion depends upon the representation of the controller and the method of controller synthesis. Typically, the representation of the controller will determine the number of states in FSM and its topology. Associated with each state there are a set of control parameters. For example, the desired configuration to be reached in a particular state could be one such set of parameters. The values of these parameters are determined in the synthesis phase. In one synthesis method, to begin with, we choose the control parameter values arbitrarily. The resulting controller is then plugged into the simulator (refer to Figure 5) to generate motion. The output motion is evaluated to determine how good or how bad it is, the control parameters are altered if necessary and the simulation process is repeated. Evaluation is done either by visually inspecting the motion and then changing the control parameters manually or with the help of a fitness function and an automated search procedure, which alters the control parameters incrementally to the desired optimal state.

We have implemented and experimented with the controller based technique for animations with automated search technique at the National Centre of Software Technology [8]. We discuss in brief the basic idea behind our method. We define, what we call *motion features* as attributes to characterize motion. Mathematically speaking, one can characterize a motion by a feature vector $f = (f_1, f_2, ...f_n)$ where $f_1, f_2...f_n$ are the n individual features. The features are computable functions which when applied to a given motion return a set of numbers. For example, walking has features which are distinct from that of running or hopping. Motions which have similar features will cluster together in feature space. However, degree of separability among types of motion will strongly depend on the selected features and the task at hand. The features are chosen such that they are intuitive for the animator to specify. The animator specifies the desired motion with the help of set of features. Once the features are specified, a fitness function is composed and a search procedure is called. The task of the search procedure is to search through the space of controllers and locate motion having similar features. Since the search space is large and discontinuos (a small change in the control parameter causing large change in the resulting motion), we have chosen the *genetic algorithm* technique, which is well suited for this kind of situation [13].

Genetic algorithms are based on the evolutionary principle of "survival of the fittest". The basic algorithm which can be used for search is described below. Let $P(t) = x_1^t, ..., x_n^t$ be a population (set) of solutions, for iteration t. Each solution x_i^t is evaluated to give some measure of *fitness*. Then a new population is formed for iteration $t + 1$ by selecting the more fit individuals from amongst the current population. Some members of the current population undergo reproduction by means of *genetic* operators to form new solutions. There are unary transformations m_i (mutation types), which create new individual (solution) by a small change in a single individual, and higher order transformations c_j (crossover type), which create two new indi-

viduals by combining parts from two individuals. After a number of generations the algorithm converges; the best solution often represents the optimal solution. See Figure 6. Since the entire procedure is very compute intensive, we have parallelised the algorithm to run on a number of networked CPUs using the parallel virtual machine (PVM) system [6].

```
procedure genetic algorithm;
begin
        t :=0;
        initialize P(t);
        evaluate P(t);
        while not ( terminate condition) do
        begin
                t := t + 1;
                select P(t) from P(t-1);
                recombine P(t);
                mutate P(t);
                evaluate P(t);
        end;
end;
```

Figure 6: A simple genetic algorithm

The results of some of our experiments are shown in the Figures 7-9 below. We have experimented using our technique on a number of simple but representative articulated bodies. The simulations typically take about 6 to 7 hours time on four networked VAX 2000 workstations.

Figure 7: Mr. Luxo, a lamp like creature hopping

Figure 8: Mr. Pogo, a dog like creature walking

Figure 9: Mr. Walker, a human like creature walking

Conclusions

The price performance of digital system currently doubles about every 12 months or so. This has happened for the last few years and industry pundits see no immediate end in sight. It is therefore important that we consider approaches which today may look computationally impractical. In the area of virtual reality systems there is tremendous interest in multi-participant simulations - virtual worlds networked via the global internet and hyperlinked with the worldwide web. These virtual worlds must impose the same physical restraints as in the real world *i.e* gravity, picking up and putting down objects, restricting motion to walking around, climbing up/down stairs *etc.* A world without life (virtual or real) would be a dull place indeed! Physically based and automated search techniques like the genetic algorithm discussed above may today be very compute intensive but will surely come very soon within the reach of available computing power and thus are bound to have far reaching impacts on the simulation of virtual worlds.

Acknowledgement

This work was carried out at the Graphics & CAD Division of the National Centre for Software Technology (NCST), Bombay during my study leave period from the University. I am grateful to Dr. S. P. Mudur, for his continued intellectual, emotional, and technical support. I thank Atul Jain for his help in the implementation. Lastly, I thank Dr. S. Ramani, Director NCST for providing an excellent research environment and the Goa University authorities for their financial support.

References

[1] William Armstrong and Mark Green. The

dynamics of articulated rigid bodies for purpose of animation. *Visual Computer*, 1(4):231–240, 1985.

[2] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–231, 1989.

[3] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Communication of the ACM*, 19(10):564–569, 1976.

[4] E. Catmull. The problems of computer assisted animation. *Computer Graphics, SIGGRAPH'78*, pages 348–353, 1978.

[5] C. K. Chow and D. H. Jacobson. Studies of human locomotion via optimal programming. *Mathematical Bioscience*, 10:239–306, 1971.

[6] Geist et. al. *PVM 3 user's guide and reference manual.*

[7] Paul M. Issac and Michael F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics, SIGGRAPH'87*, 21:215–224, July 1987.

[8] V. V. Kamat. *Automatic motion synthesis of articulated figures for computer animation.* PhD thesis, Goa University, 1996. To be submitted.

[9] Mathew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, 1988.

[10] Mark Raibert and Jessica Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25:349–358, July 1991.

[11] B. Robertson. Dinosaur magic. *Computer Graphics World*, pages 44–52, September 1993.

[12] B. Robertson. Physical graphics. *Computer Graphics World*, pages 37–43, March 1993.

[13] M. Srinivas and L. M. Patnaik. Genetic algorithms: a survey. *IEEE Computer*, 27(6):17–26, 1994.

[14] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controller. *Computer Graphics SIGGRAPH'90*, 24:225–234, August 1990.

[15] Michiel van de Panne, Ryan Kim, and Eugene Fiume. Virtual wind-up toys for animation. In *Proceedings of Graphics Interface*, pages 208–215, 1994.

[16] Jane Wilhelms. Toward automatic motion control. *IEEE Computer Graphics and Applications*, 7(4):11–22, 1987.

[17] Jane Wilhelms. Dynamic experiences. In Norman I. Badler, Brian A. Barsky, and David Zeltzer, editors, *Making them move: mechanics, control and animation*, pages 265–279. Morgan Kaufmann, 1991.

[18] Andrew Witkin and Michael Kass. Spacetime constraints. *Computer Graphics*, 22:159–168, August 1988.

❑