An Evolutionary Programming Technique Using Motion Features For Animating Articulated Figures

V. V. Kamat[1], Atul Jain[2], S. P. Mudur[3]

The primary purpose of a computer animation system is to provide assistance to the human animator in synthesizing the movement of the animated character such that the resulting motion appears physically correct and natural. If the character is modelled as an articulated figure consisting of an appropriate number of links and joints, then it's movement could be specified by a trajectory -- time dependent values for the position of the figure. Hence all possible trajectories for a given set of goals to the character can be considered as forming a space of trajectories and the problem of synthesizing movement can be viewed as that of searching in this space for a trajectory satisfying all the physical requirements and the animator's goals. Unfortunately, this space is normally of very high dimension, and even for relatively simple characters and goals, the solution space is vast, multimodal and discontinuous, not lending itself very well to the use of conventional gradient based search techniques.

This paper presents an evolutionary programming technique that uses motion features to direct the search. While genetic algorithms for motion synthesis have been proposed by other researchers, the computation of motion features and their use in the definition of a fitness function for determining subsequent generations is novel. The method has been implemented and simulation experiments with characters of reasonable complexity have yielded very good results. In this paper, we describe in detail the evolutionary programming technique that we have devised, computation of motion features and fitness function formulation. We also present results from some of the experiments conducted using our implementation.

1 Animation as Trajectory Search

Over the past few years, computer animation has rapidly progressed from the production of simple transformation based movements of inanimate objects to the production of natural looking and highly sophisticated movement of simulated characters. The primary purpose of a computer animation system is to provide assistance to the human animator in synthesizing the movement of the character such that all the movements and actions appear to work in concert in order to suspend the disbelief of a character's existence. The realism of the movements of dinosaurs in the film *Jurassic Park* is a very good example of what is possible with computer animation techniques today.

Movement or motion is a dynamic phenomena. If the character to be animated is modelled as an articulated body composed of links that are connected to each other via joints, then the movement involves change in spatial configuration of the articulated body over time. The spatial configuration of a body is defined geometrically using *Degrees of freedom* (DOF), which constitutes the minimal set of parameters needed to completely specify the configuration of a body in space. An articulated body has DOF depending on the number of links, joint types *etc.* Consider for example the planar articulated body with 3 links and 2 rotary joints as shown in Figure 1. This simple body has 5 degrees of freedom. Human bodies are amongst the most complicated of articulated bodies, with about 200 degrees of freedom [20].

A universally accepted convention is to consider degrees of freedom as constituting a vector and use vector notation say $\mathbf{X}$ to denote the spatial configuration of a body. Hence the motion of the body over a time period T would be denoted by $\mathbf{X}(t)$, $0 \leq t \leq T$. This is also referred to as the trajectory.

[1]Goa University, Goa
[2]HCL Technologies, Noida
[3]National Centre for Software Technology, Mumbai

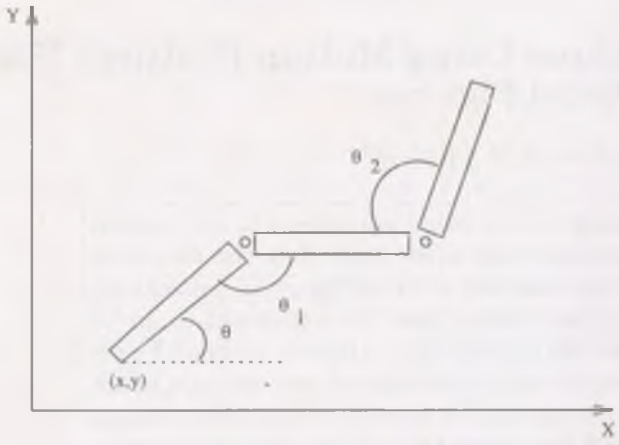

Figure 1: A planar articulated body with three links and two rotary joints

A rather simple definition of the motion synthesis problem is as follows:

> Given the geometry of an articulated body, say X, a desired type of movement say walk and a time period T, determine $X(t)$, $0 \leq t \leq T$ such that the resulting motion looks physically correct and natural.

The above definition hides the enormous underlying complexity in this problem. Certainly physics is involved; gravitational and other forces have to be considered. Animators for example, take years before they acquire the necessary skills to predict the spatial configurations of objects at any time for generating a specific movement.

There is another view to the motion synthesis problem. All possible trajectories for a given character for a particular goal specified by the animator, can be considered as forming a space of trajectories and the problem of synthesizing a particular movement can be viewed as that of searching for a suitable trajectory in that space. The trajectory that is finally selected must satisfy all the physical requirements and also the animators goals. For autonomous articulated figures however, the search problem gets extremely complex. Firstly the number of DOF is very large. As a result the trajectory space is of very large dimension. Secondly, there is always built in task level redundancy, *i.e* any behaviourial goal can be achieved in many different ways. Thus, for example, a cup of coffee might be reached while moving the hand along many different paths. Usually the search is cast as a non-linear constrained optimization problem. The physical laws, and physical and user specified constraints are to be satisfied while the animators goals are in the form of an objective function to be optimized. Unfortunately, the solution space is vast and even for relatively simple characters and goals, the space could be multimodal and discontinuous. As a result traditional local optima search techniques such as steepest descent *etc.*, do not work satisfactorily. Other more robust and global search methods need to be considered. Genetic algorithm based search techniques are in this category.

2 A Brief Review of Trajectory Search Techniques

Traditional Key-frame animation systems provided some minimal support in the interactive creation of a trajectory. In key-frame animation, the animator describes a set of "key-frames" from which the system geometrically interpolates in between frames to obtain the complete trajectory. With sufficient number of trials the animator can construct the desired trajectory. There are two major problems with key-frame animation. Firstly, it requires the animator to adjust too many parameters at very fine levels of detail. Secondly, to generate very convincing looking motion, the animator must have a very good understanding of the motion. In practice, however, even after many trials, key-frame synthesized motion tends to look unrealistic and puppet like.

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. Thus traditional geometric models were augmented to include physical characteristics such as mass of the body, its moment of inertia, external forces like gravity, friction *etc*. Interaction with other objects in the environment and resulting behaviour is also modelled and variety of collision detection and collision response techniques have evolved [9]. The idea is

to incorporate appropriate physical complexity and realistic behaviour into the model itself, rather than imposing it on the animator. Initial results on incorporation of physics to produce realistic movements were very encouraging [1, 18].

However, this is not without problems. The specification of forces and torques to produce desired motion is non-intuitive and certainly non-trivial. Further, once time varying forces/torques are specified the motion is completely determined and is autonomous, and not any more under the control of the animator. Thus incorporation of physics into the model results into loss of fine control over the generated motion.

Two approaches have evolved that partially overcome this problem. In the first approach the trajectory is discretized in space and time and then an iterative search is made in the trajectory space for a trajectory satisfying physical constraints, user specified constraints and optimizing an objective function [19]. The method uses the variational calculus technique which is very similar to the gradient methods used for finding the local minima of an ordinary real-valued function.

The second approach, tries to synthesize a controller which when executed results into motion [16, 13, 14]. A controller is just a finite state machine (FSM) with a set of rules that determines the torques to be applied at the joints in any given state. Controllers can be parameterized using fewer number of parameters and the search is made in the parameter space of controllers for a set of parameters which when plugged into a controller results into motion that satisfies the user specified constraints and optimizes the objective function. (See Figure 2).

Although both the methods differ fundamentally in technique, they suffer from the same source of difficulties mentioned earlier, namely the solution space for desired motion is vast, and multimodal *i.e* replete with solutions which are not so good. Secondly, due to the dynamic instability of the system the solution space is discontinuous which causes extreme sensitivity towards control parameters and initial conditions. This manifests itself in the form of drastic changes in the motion behaviour of the creature even with slight changes in the control parameters or in the initial condition.

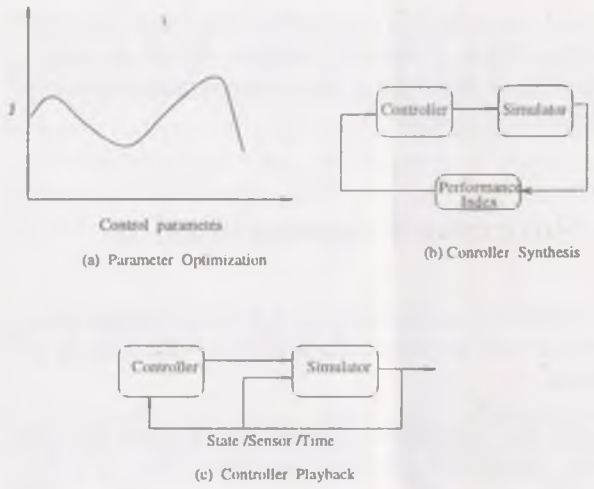

Figure 2: Controller synthesis and playback

Evolutionary based search algorithms basically being global and more robust are better suited for such situations. Application of evolutionary based search for motion planning and motion control is not new. Davidor [4] has used genetic algorithms for generating and optimizing a time-indexed sequence of configurations in robot trajectory planning. Fogel has used evolutionary programming to generate "bang bang" solution to classic cart-pole [2] problem in control. De Garis [5] uses genetic algorithms to synthesize weights in a neural network to produce the walking behaviour of a bipedal articulated figure. However, his underlying model is geometric and not physical. Lately researchers in computer animation have also started using evolutionary based techniques to synthesize motion sequences in animation [13, 15, 8]. Ngo and Marks [13] for example have used a massively parallel genetic algorithm to synthesize motion of articulated figures using what they call as "Banked stimulus response" controllers. A genetic algorithm is also used in controller synthesis in the recent work by [15] and by Gritz and Hahn [8]. While all the above research has shown the applicability and effectiveness of genetic algorithm based search techniques in animating articulated figures, the work reported in this paper defines new techniques which extend the use of such evolutionary algorithms for more goal oriented motion synthesis. We use motion features to characterize different types of motions and for making the specification of animator's goals sim-

ple and natural. These motion features are then used to define a fitness function which is instrumental in determining subsequent generations of trajectory solutions.

3 Solution Representation

The choice of representation for the solutions plays a crucial role in the success of the evolutionary algorithm.

Ideally, the representation should be such that the motion synthesis problem can be solved in a reasonable time, without sacrificing generality. This directly implies the following:

1. The representation should be as compact as possible so that the dimensionality of the parameter space is kept low.
2. A method is needed that smoothly alters the internal configuration from one time slice to another. This would reduce the number of distinct spatial configuration samples that need to be computed for the given time interval.

Compactness is achieved by having fairly powerful rule based controller representations that need a small number of states and hence a small number of parameters.

A common method of achieving smoothness is through the use of the *proportional derivative* (PD) control law. It consists of a spring/damper mechanism in which a torsional spring attached between two links applies a torque on the adjacent links according to the equation:

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

where k_p is the spring constant, k_v is the damper constant, θ and $\dot{\theta}$ are the current angle and the angular velocity respectively and θ_d is the rest angle (equilibrium position) of the spring. (See Figure 3) The values of k_p and k_v are typically chosen such that the mechanism is critically damped. A critically damped system is one that when disturbed will most rapidly return to equilibrium position.

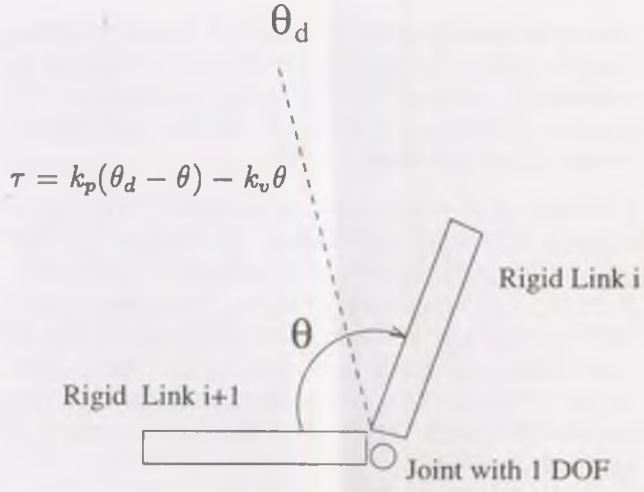

Figure 3: Actuator modelling spring damper mechanism

Such a system acts to control the angular position, in that, if the current angle is different from the equilibrium angle, the spring will apply a torque on the adjacent links so as to again be in equilibrium. For example, if the current limb joint angle is less than its desired angle, the mechanism will cause a positive torque to be applied to the joint to move it back toward the desired angle and vice versa. The velocity damping term reduces the torque applied to the joint once movement towards the desired angle is underway. By changing the desired angle at different instances of time, the mechanism can be actuated to bring about the motion.

The advantage of the PD controller is that the torque function gets automatically defined once the desired angle is specified To execute a particular movement of the joint, it is necessary to define a series of desired joint angles. The motion synthesis problem is then converted to that of synthesizing the function $\theta_d(t)$. If the articulated figure has m actuators, it amounts to synthesizing m functions of the type $\Theta_d(t) = (\theta_d^1(t), \theta_d^2(t) \ldots \theta_d^m(t))$. The simplest way to solve the problem is to choose a piece-wise continuous function. This function could be a constant, linearly varying or more complex with continuous basis functions such as splines [3], sinusoids or wavelets [11].

For the sake of simplicity, for our experimentation purposes, we have used piece-wise constant functions. To synthesize a motion sequence for duration T, we divide T into several phases or states. Each phase will be associated with a set of parameters such as

$$t, \theta_d^i, k_p^i, k_v^i$$

where

i — i^{th} joint actuator
θ_d^i — desired angle for the i^{th} joint
k_p^i — spring constant corresponding to the i^{th} joint
k_v^i — damping constant corresponding to the i^{th} joint
t — time duration of the phase.

If there are fifteen phases in a motion sequence, the number of unknowns to be determined are $15 \times 4 = 60$. We have mentioned earlier, that, as the simulation duration increases, the search space increases exponentially. When we consider periodic motion, such as walking, running, hopping, *etc.* one can reduce the search space. In periodic motion, after every period t_{per}, the motion is repeated to fill the simulation time T. Unknown parameters depend only on the number of phases in the period t_{per}. (See Figure 4.)

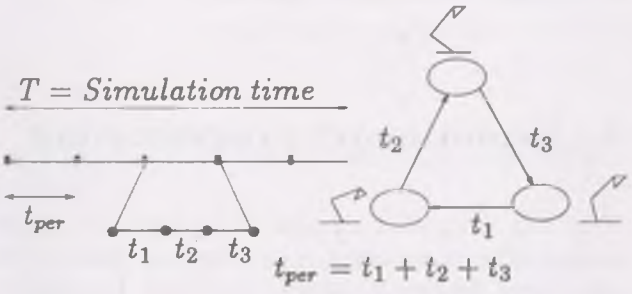

Figure 4: An illustration of the controller

In order to reduce the search space further, one can fix the values t, k_p^i, k_v^i a priori. The time period t could be either derived from a previously synthesized keyframed version of the motion or from live or video data of a creature similar in size and shape [17]. Values of spring constants can be estimated depending on the mass of the body. For heavier links, higher values of spring constants are required in order to protect the springs from a possible collapse (spring failure). However, values should not be so high that they would generate such high torques at the joints that unexpected motions are caused.

4 Motion features

In automatic motion synthesis for articulated figures, the animator essentially specifies the following:

1. the physical structure of the character (link lengths, mass, moment of inertia, range of angles for the joints *etc.*)
2. the joints where the actuators are to be placed which will control the character's internal configuration
3. criteria for evaluating the character's motion

Given this information, the computer animation system is expected to automatically generate a physically realistic motion satisfying the criteria specified by the animator. In most of the systems described in the literature so far, the criterion for evaluating the character's motion is kept rather simple, say, motion with minimal energy, or with maximum height or maximum distance traveled, *etc.* One of the main drawbacks of these methods is the difficulties associated with the specification of a suitable performance criterion. Even in the case when the animator wants to generate a slightly different motion for the character, a different motion evaluation criteria would have to be given and the synthesis process started again. The methods provide no clue whatsoever as to how to choose other criteria, even those which are slightly differing from the first. We have proposed the use of motion features to overcome this problem and these are briefly discussed below.

We define, what we call motion features as attributes to characterize motion. The basic idea here is to classify different types of motion according to features, and use that information to synthesize new

controllers that can generate motion with similar features. It is well known that, humans can effortlessly perceive the subtle details of the types of motions found in animals. This includes a wide variety of motions ranging from undulatory motion of worms, to the esthetic walk of human beings. The deep rooted structure underlying motion is so well understood by humans that we are often able to identify the gender of a person just by the style of the walk. Since in computer animation our objective is to generate natural looking motion, an obvious question to ask is, whether the information present in the perceived motion can be used in creation of natural looking motion. In this section we explore this idea in some detail. Our hypothesis is that, the information which humans use to identify, distinguish among, and classify motion patterns could be used in the search for suitable motion. The motivation for such an hypothesis can be found in [10, 12].

Mathematically speaking, a motion can characterized by a feature vector $\mathrm{f} = (f_1, f_2, \ldots f_n)$ where $f_1, f_2 \ldots f_n$ 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 to be chosen such that they are intuitive for the animator to specify and also have good discrimination properties. The animator specifies the desired motion with the help of a 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. (See Figure 5.)

Some of the features that we have built into our implementation and experimented with are :

- key poses ($\theta_{s,i}$), fully or partially specified
- external energy (E)
- horizontal distance traveled by the centre of mass (D)

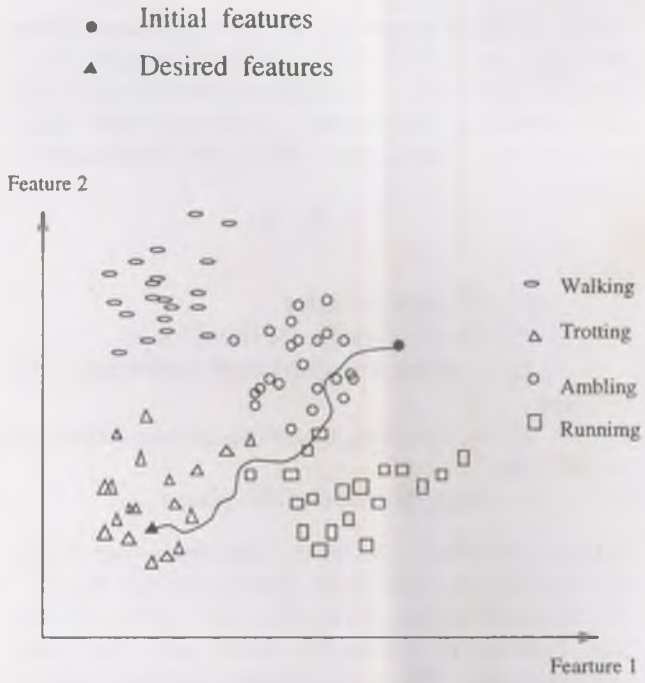

Figure 5: Search in feature space for desired motion

These features could either be extracted out of actual motion using motion capture devices or specified by the animator. The optimization function tries to match the features of the synthesized motion to that desired by the animator.

5 Evolutionary programming

Typically the goal of an optimization process is not only to achieve an optimal solution but also to find an optimal process to achieve that goal. In order to search effectively, it is necessary to specify a small but useful amount of additional information that can greatly reduce the synthesis or search time. The information we are providing is the following:

1. time interval between the phases
2. the number of phases in period t_{per}.
3. values for spring and damper constants

$$f = w_1 * \sum_{s=1}^{samples} \sum_{j=1}^{joints} (1 - (\theta^o_{(s,j)} - \theta_{(s,j)})) + w_2 * (1 - \frac{(E_o - E)}{E_{max}}) + w_3 * (1 - \frac{(D_o - D)}{D_{max}})$$

where,

w_1, w_2, w_3	are weights, assigned to the features depending on their relative, importance the value ranging between 0 and 1.
$\theta_{(s,j)}$	is the angle at joint j for sample,
E	is the external energy
D	is the horizontal distance travelled
E_{max}, D_{max}	are the maximum expected external energy and horizontal distance, respectively. These values are used for normalization of the two. quantities.
$\theta^o_{(s,j)}, E_o, D_o$	are the feature values obtained during feature extraction process.

Stochastic Population Hill Climbing (SPHC) Algorithm

The SPHC algorithm is an evolutionary programming algorithm [7] that can be distinguished from genetic algorithms, primarily by the fact that it uses only the mutation operator and does not use a crossover operator.

Like all genetic algorithms (SPHC) uses a population of solutions. Each solution in the population is perturbed randomly at each iteration, using the mutation operation with probability 1. The resulting solutions are compared with their original solutions and the better ones are kept for the next generation. Periodically, a re-seeding operator is applied which selects the top half of the population and copies them into the bottom half of the population, refocusing the search on the most promising solutions in the population. The full algorithm is shown in Figure 6.

```
Initialize population
Evaluate each solution in the population
for generation = 1 to number_of_generations
     for each individual solution in the
     population Randomly perturb the
     solution Evaluate new solution
     if ( new solution better than )
     old solution)
     then replace old solution with
     new solution
end for
if ( ( generation mod
re-seed_interval ) = 0- )
     then Rank order the population
     Replace bottom 50% of population
     with top 50%
end if
end for
```

Figure 6: Stochastic population hill climbing (SPHC)

Mutation Operator

The mutation operation is the backbone of our SPHC algorithm. In every iteration all solutions go through this operation. Since a slight change in parameters can change the motion drastically, it is necessary to apply the mutation operation with care. We have selected to mutate only one parameter at a time with only a small change in original value. The parameter to mutate is selected randomly with all the parameters having equal probability. This was found quite suitable through experiments, as it helps the algorithm to fully explore the region near to the existing solution. If we try to mutate more than one parameter at a time, the solution may jump from one region to another without exploring the current one. As the function is multimodal, it may actually be the case that optimal solution is in the vicinity of solution being mutated. Each time mutation is called either the selected parameter goes through a *creep* operation[4] or all its parameters are randomized from scratch. As there

[4]The creep operation is used here to modify the parameter by a very small factor

```
Randomly select any one of the phases
     to be modified
Randomly select the operation to be
     applied on the selected phase
if ( operation is creep operation )
     then
     Randomly select one of the
     creep operations operations
     and apply
else
     Generate randomized parameters
     for new phase
phase from scratch
```

Figure 7: Mutation operation

are three types of parameters to be modified, there are three possible creep operations, these are defined as follows:

1. The original time interval is multiplied by a randomly chosen factor close to unity (0.8 – 1.2).
2. One of the joint angles is selected randomly and changed by a randomly chosen amount between −10 degrees and 10 degrees.
3. One of the joint angles is selected randomly and multiplied by a randomly chosen factor close to unity (0.8 – 1.2).

6 Implementation and Results

For the sake of containing the computational efforts involved, we have considered only 2D planar bodies. Apart from gravity, the other external forces exerted on the articulated figure that we have considered are those due to its interaction with the ground. The ground has been modelled using a spring and damper mechanism which exerts forces on the articulated figure at the points of contact. Typically, the position and velocity of these contact points on the articulated figure are used to compute the external forces as follows:

$$F_x = -(m_x - p_x)k_p - v_x k_v$$
$$F_y = -(m_y - p_y)k_p - v_y k_v$$

Spring and damper constants chosen are $k_p = 10^5 N/m$ and $k_v = 10^3 N/m$. This creates a suitably stiff floor that functions effectively when used in a simulator. Since the equations of motion for our articulated bodies are too complex to be derived by hand, they have been compiled symbolically using the Newton-Euler recursive formulation. The equations of motion are solved for acceleration using LU decomposition method and integrated to get position/orientation using simple Euler method. The time step for integration has been chosen as small as 0.005 to overcome the stiffness problem. However, this results into long simulation times. In order to overcome this problem, we have parallelized the SPHC algorithm to run on a Parallel Virtual Machine (PVM) [6]. We have experimented using our technique on a number of simple but representative articulated bodies. It took about 6 to 7 hours of computation time to synthesize a controller on four networked VAX 2000 workstations running PVM.

We describe below the structure and motion behaviour from our experiments of two creatures named Luxo and Pogo.

6.1 The Luxo creature

Figure 8 shows the geometric structure of Luxo. It is a lamp like creature made of three links connected with two joints.

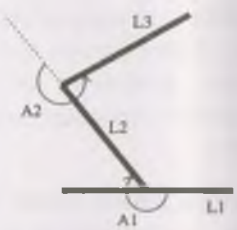

Figure 8: A luxo articulated body

Table 1 shows the allowable ranges of joint angles (in degrees) defining the internal configurations.

Joint	*Min*	*Max*
A1	-300	-240
A2	360	210

Table 1: Luxo Angles

Table 2 shows the physical properties of different links.

Link	*Mass*	*Inertia*	$cmass_x$	$cmass_y$
L1	0.15	0.00312	0.0	0.0
L2	0.10	0.00208	0.25	0.0
L3	0.30	0.00625	0.25	0.0

Table 2: Physical Properties of Luxo

The controller for Luxo has been designed using two phases. The parameter space is ten dimensional. The ratio k_p/k_v is approximately chosen as 0.1.

$$[\theta_1^1, \theta_2^1, k_{p1}^1, k_{p2}^1, t_1] \qquad [\theta_1^2, \theta_2^2, k_{p1}^2, k_{p2}^2, t_2]$$

Feature values have been chosen so as to synthesize hopping motion. The progress of the search algorithm in finding the hopping motion controller is shown in Figure 9. The motion has been synthesized for two different initial populations. In the first case the desired controller is found after 40 generations with a population size of 50. In the second case, more or less a similar controller was found after only 25 generations.

Figure 10 shows the hopping motion obtained for Luxo Figure 11 shows a phase diagram which plots the height of the centre of mass versus vertical velocity of centre of mass. The trajectories in the phase diagram show a periodic behaviour with trajectories being attracted towards an attractor cycle. Figures 12 and 13 show the variation of joint torques and the variation of joint angles with time.

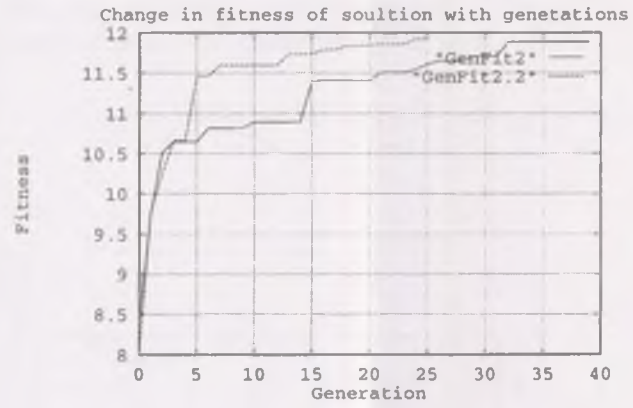

Figure 9: Synthesis of two different controllers for Luxo

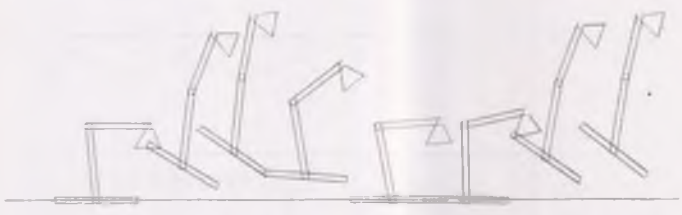

Figure 10: Mr. Luxo, a lamp like creature hopping

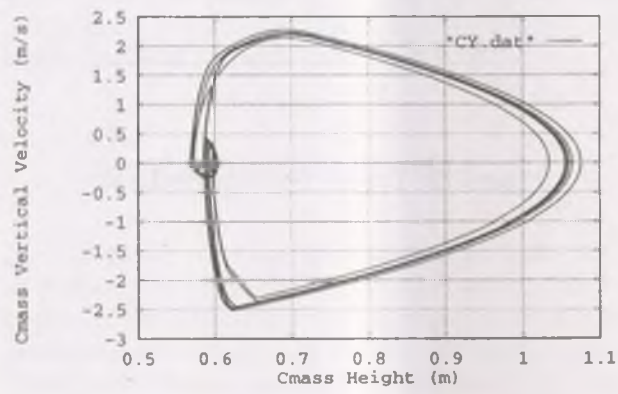

Figure 11: Height of centre of mass v/s velocity for Luxo

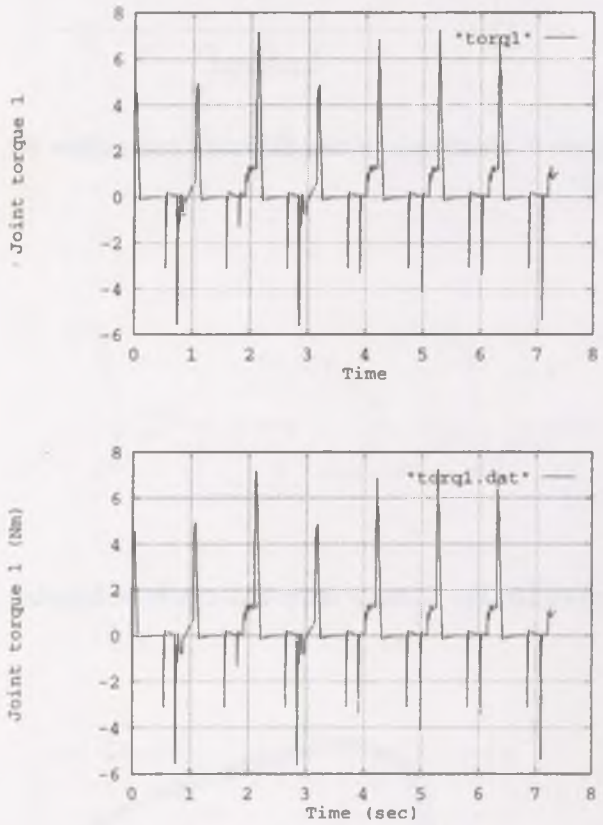

Figure 12: Torque v/s Time for Luxo

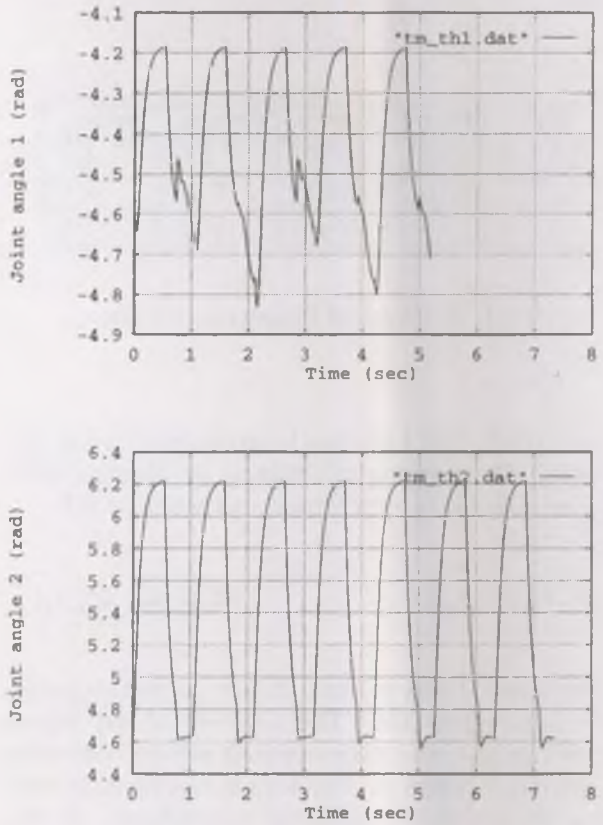

Figure 13: Joint Angles v/s Time for Luxo

6.2 The Pogo creature

The geometric structure of "Pogo" is shown in Figure 14. It is a dog like creature made of five links connected with four joints.

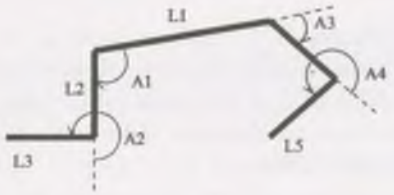

Figure 14: A pogo articulated body

Table 3 shows the allowable ranges of joint angles(in degrees) defining the internal configuration.

Joint	*Min*	*Max*
A1	-115	-45
A2	270	360
A3	-70	-40
A4	210	300

Table 3: Pogo Angles

Table 4 shows the physical properties of different links. 4.

Link	*Mass*	*Inertia*	$cmass_x$	$cmass_y$
L1	0.15	0.003123	0.25	0.0
L2	0.10	0.002082	0.125	0.0
L3	0.10	0.002082	0.125	0.0
L4	0.10	0.002082	0.125	0.0
L5	0.10	0.002082	0.125	0.0

Table 4: Physical Properties of Pogo

The controller for Pogo has been designed using two phases again. The parameter space is however eighteen dimensional. The ratio k_p/k_v is approximately chosen as 0.1. Feature values have been chosen so as to synthesize walking motion. The progress of the search algorithm in finding the walking motion controller is shown in Figure 15.

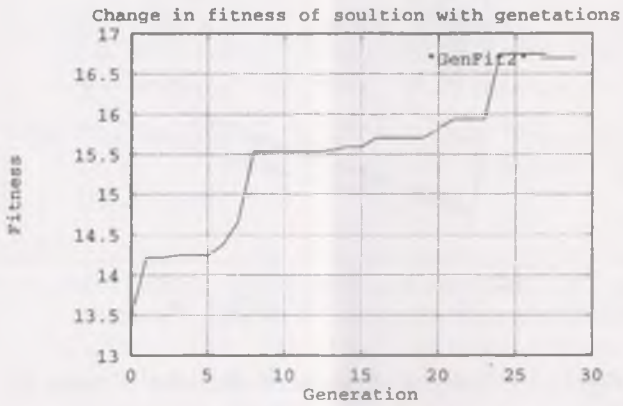

Figure 15: Synthesis of a controller for Pogo

The result of the simulations of Pogo are shown in the Figure 16 below.

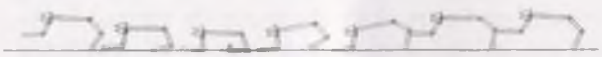

Figure 16: Mr. Pogo, a dog like creature walking

Figure 17 shows a phase diagram which plots height of centre of mass versus vertical velocity of centre of mass. The trajectories in the phase diagram once again show a periodic behaviour with trajectories being attracted towards an attractor cycle.

7 Concluding Remarks

In this paper we have first shown how the task of motion synthesis for an autonomous character in computer animation can be viewed as a constrained optimal search problem in trajectory space. Since the solution space does not admit to conventional gradient based local optima search techniques use of more robust and global optima search like the evolutionary programming technique is recommended. We have described in detail a novel

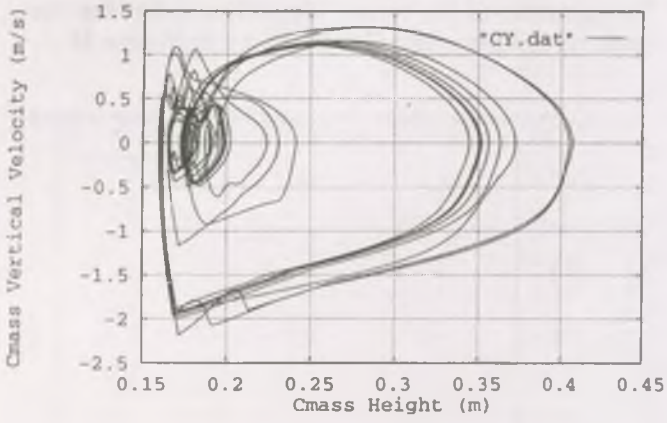

Figure 17: Height v/s velocity of centre of mass for Pogo

extension of the evolutionary programming technique using motion features. While we have used very simple motion features in our implementation and in our experimental simulation, the results obtained for complex movements like hopping and walking for reasonably complex virtual creatures are indeed very encouraging. Parallelizing the search algorithm has also shown that the method scales easily to perform well with increase in the number of processors. Certainly a more comprehensive repertoire of motion features have to be evolved and more experiments need to be carried out with different kinds of movements being specified and automatically synthesized.

8 Acknowledgements

The above work has been carried out in the Graphics and CAD Division of NCST Bombay while the first author was on a study leave from the university and the second author was a project student. The authors are very grateful to Dr. S. Ramani, Director, NCST for his generous support and encouragement. The first author would also like to express his sincere thanks towards Goa University authorities for the 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] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuron-like adaptive elements that can solve difficult learning control problem", *IEEE Transaction on Systems Man and Cybernetics, 13:834-846, 1983.*

[3] Micheal F. Cohen, "Interactive space time control for animation", *Computer Graphics, 26:293-302, July 1993.*

[4] Y. Davidor, "A genetic algorithm applied to robot trajectory generaiton." *In L. Davis, editor, Handbook of genetic algorithms*, pages 144-165. Van Nostrand Reinhold, 1991.

[5] H. de Garis, "Genetic programming: building artificial nervous system using genetically programmed neural network modules." *Proceedings of the seventh international conference on machine learning, pp 132-139, 1990.*

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

[7] A. Fukunaga, L. Hsu, P. Reiss, A. Shuman, J. Christenen, J. Marks, and J. T. Ngo. "Motion-synthesis techniques for 2d articulated figures". Technical Report TR-05-94, Harvard University, Center for Research in Computing Technology, 1994.

[8] L. Gritz and J. Hahn. "Genetic programming for articulated figure motion". *The Journal of Visualisation and Computer Animation*, 6(3):129-142, 1995.

[9] V. V. Kamat. "A survey of techniques for simulation of dynamic collision detection and response". *Computers and Graphics*, 17(4):379-385, 1993.

[10] J. A. S. Kelso and A. S. Pandya. "Dynamic pattern generation and recognition". In Norman I. Badler, Brian A. Barsky, and David Zeltzer, editors, *Making them move: mechanics, control and animation*, pages 171-190. Morgan Kaufmann, 1991.

[11] Zicheng Liu, Steven J. Gortler, and Micheal F. Cohen. "Hierarchical spacetime control". *Computer Graphics*, 28:35-42, 1994.

[12] T. McMahon. "Mechanics of locomotion". *The International Journal of Robotics Research*, 3(2):5-26, 1984.

[13] J. T. Ngo and J. Marks. "Physically realistic motion synthesis in animation". *Evolutionary Computation*, 1(3):235-268, 1993.

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

[15] Karl Sims. "Evolving virtual creatures". *Computer Graphics*, 28:15-22, July 1994.

[16] 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.

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

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

[19] Andrew Witkin and Micheal Kass. "Spacetime contraints". *Computer Graphics*, 22:159-168, August 1988.

[20] D. Zeltzer. "Task-level graphical simulation: abstraction, representation and control". In Norman I. Badler, Brian A. Barsky, and David Zeltzer, editors, *Making them move: mechanics, control and animation*, pages 171-190. Morgan Kaufmann, 1991.