INTERACTIVE EVOLUTIONARY APPROACH TO CHARACTER ANIMATION

M Nordin Zakaria
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, Malaysia
nordinz@fsktm.upm.edu.my

ABSTRACT

We describe in this paper an approach to the animation of articulated figure using a combination of forward kinematics, neural net and genetic algorithms. A set of kinematics motion data for an articulated model is evolved to generate a set of varied motion that performs according to an objective function set by the user. Neural net is used to constrain the search space of possible motions. User interaction with the process ensures a certain style evolves from the myriad of possibilities. We show how our technique makes it possible to create libraries of reusable motions from a small number of example motions.

KEYWORDS: *Computer Animation*, *Genetic Algorithm*

1. INTRODUCTION

This paper is concerned with semi-automatic generations of kinematics-based computer animation of articulated figures. The focus is on character animation as an arts form, rather than as a physics or biomechanics based simulation. Character animation is perceived as a creative, non-routine [Rosen97] design process where knowledge about the relationships between the requirements for a design and the form required to satisfy those requirements is intuitive rather than methodological. The animator's own vision and feel for the character and its 'illusion of life' sets the requirements. There is no clear mapping from requirement to exact form of the animation. The animator creates in passes, engaging in trial and error, and refining the current best set of solutions. The task requires an evolutionary act. The idea of using genetic algorithm to generate character animation is hence appealing.

Genetic algorithm can be used to explore the space of possible animations of a character model. While genetic algorithm has been used for the generation of controllers in physically modeled articulated figure [Sims94], our approach relies on kinematics data rather than a dynamic model. A set of kinematics motion data for an articulated model is evolved to generate a set of varied motion that performs according to an objective function set by the user. Neural net is used to constrain the search space of possible motions. User interaction with the process ensures a certain style evolve from the myriad of possibilities. The idea is that instead of the animator tediously toiling over the kinematics or physics-based details of possible character motions, the animator plays a reactive or supervisory role while the computer does the exploration. The user needs only supply a small set of initial motions. A wide range of motions is then evolved from the much smaller set of motions supplied by the user. As in [Ventr95], emphasis is on “optimization of expressivity”. The animator is in the optimization loop to aesthetically influence the evolution of motion.

2. CURRENT APPROACHES

Motion control of articulated figure is by nature tedious. Animation of articulated model has commonly been done using keyframing. High quality results are the result of highly developed human skill and artistry. Figures are posed painstakingly for every keyframe using either forward or inverse kinematic. In the bigger industry, motion capture is commonly used for familiar figures such as human and animals. Motion capture relies on kinematics knowledge of motion and locomotion [Boulc90]. Relying on motion of real actors, the technique can produce startlingly convincing results. The cost, however, is prohibitive and much work is involved in getting the right motion for a character. Using keyframing or motion capture, the end result sought is a library of reusable motions for a specific character. The set forms a basis set of motions from which additional motion can be warped [Witkn95] or interpolated [Wiley97, Rose98].

In much of the computer animation research community, the holy grail of research is often implied to be the complete simulation of the physical world complete with cognitive-inspired modeling of minds and emotions that drives motion and reaction of intelligent or lifelike articulated figures. In contrast to kinematic-based approach, dynamic approaches [Raibt91, Hodgn95, Grzes98] rely on knowledge about the physics and biomechanics of the motion. The goal is to leave the issue of motion control to the laws of forces. An arm moves due to forces acting on it, a character falls due to the pull of gravity. The problem then becomes that of specification of forces. While animation of character falling due to gravity is relatively trivial using dynamics package such as SD/FAST [SD/FAST] and MathEngine {MathEngine], putting the character to work on motions even as seemingly simple as walking has drawn intensive research, testifying to the difficulty of the approach. Moreover, expecting the animator to draw upon intimate knowledge of forces to animate a character seems hardly a practical idea. A quick fix is frequently in the form of higher-level user interface that makes animation as easy as providing high-level specification such as "walk up the stairs". While automation leads to an easy life for anyone who intends to produce a quick animation, it leads to lesser control by the animator over the animation. This may be undesirable especially for the highly skilled animator working on top-notch animation - where the slightest of details set apart the great from the also ran. The animation may be enough for purposes such as ergonomic study, as evident in the Jack Human Modelling and Simulation System [Jack], but it may not be appealing enough to find its way into the entertainment world.

The crux of our approach lies in genetic algorithm. Genetic algorithm, a class of optimization algorithm, offers a technique for the automatic generation of new motion specifications, while offering limited user control by means of fitness function. In genetic algorithm, a Darwinian "survival of the fittest" approach is employed to search for optima in a large multidimensional space. In computer graphics and animation, genetic algorithm has been used for synthesis of controllers for physically based articulated figures, and generations of interesting images [Sims93] and designs [Rosens97, Jacks99]. The evolutionary approach allows a more diverse area of the solution space to be traversed compared with other methods; furthermore its probabilistic selection method directs the random generative process towards meaningful and satisfactory solutions

The approach in the literature closest to ours is that of Lim and Thalman in [Lim99]. Lim and Thalman's pro-active interactive evolution synthesize example motions from a single motion. The primary difference between their approach and ours is that in our approach, new motions evolve from a set that can consist of more than one motion. The newer motions may be the results of crossover of the original motions. As a result, a new motion may be a combination of old motions. For example, a kicking and punching action may result from a set that has a kick motion and a punch motion. There is no mention of the use of crossover operator in [Lim99]. Also, we use neural net to constrain the possible motions generated by the evolution.

3. APPROACH

Our approach draws upon a combination of forward kinematics, neural net and genetic algorithms. For purpose of discussion, we assume that the model is represented as a hierarchy of rigid links connected by joints. Each joint has up to 3 degrees of freedom (DOF). The variation of each DOF is represented as a discrete function $\theta(T)$ of time. The function may be visualized and manipulated on as a curve, as shown in Figure 1.

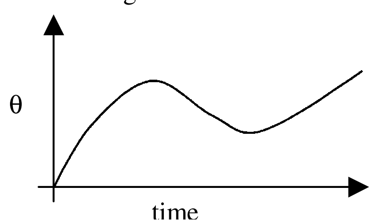
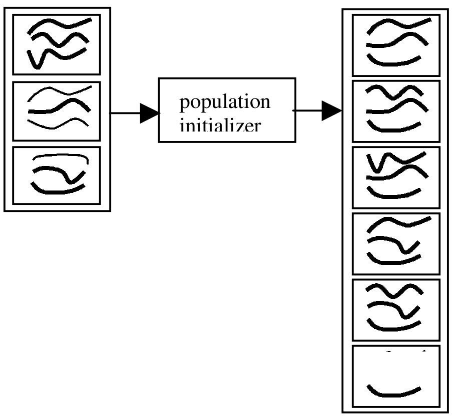


Figure 1: A DOF function

A set of DOF functions for a character model is specified for a defined duration. We shall refer to this set simply as a DOF set. Not every joint angle for each and every frame or time unit need be stored in a DOF set. Only the control points of the representative curve need be stored. The user needs to specify a collection of q DOF sets, D_1 ... D_q, each set corresponding to a distinct character motion. The set need not be too big. It only has to be sufficiently varied. The motion data may be handcrafted or it may be motion-captured. It may even have been simulated by dynamic based models. For a model with n joints, each of which has m DOFs, for each DOF set, there would be n*m motion curve to deal with. Let the ordered set of n*m curve be denoted by $s = <c_1\ c_2\ ...\ c_{n*m}>$. Each ordered set forms a genome in our genetic algorithm.

Among the DOF functions in a DOF set that the user has supplied, he or she marks one or more DOF function as 'interesting'. We denote the subsets of DOF functions in each DOF set D_i marked as interesting by d_i, i = 1..q. The genetic algorithm first creates its initial population by forming an ordered Cartesian product of d_i. Mutation, crossover and breeding operators are then performed on this population for a finite number of generations. Figure 2 illustrates the concept involved here.



Before evolution commences, our approach relies on the user providing knowledge to the system on the type of motions possible. The system is trained by neural net to recognize a motion when it 'sees' one. For example, the user can supply the system with a moderate training set of motions that correspond to walking, running, kicking, punching, and so on. The recognition capability is used to constrain the type of solutions produced by the genetic algorithm.

3.1. DETAILS OF THE GENETIC ALGORITHM

The three generic task in using a genetic algorithm, regardless of the exact form of the algorithm itself are

1. Defining a representation
2. Defining the genetic operators
3. Defining the objective function

3.1.1. DEFINING A REPRESENTATION

We seek a representation that is minimal but completely expressive. The representation should not contain information beyond that needed to represent a solution to the problem. The representation should be able to represent any solution to the problem. In our approach, each genome is an array, whereby each element of this array is a 2D array. The inner 2D array holds the discrete values of a DOF function. The first dimension of this 2D array holds the values of normalized time (range is from 0.0 to 1.0), while the second dimension holds the rotational values. Figure 3 illustrates an example gene, while Figure 4 illustrates an example genome.

0	0.1	0.3	0.5	0.7	0.75	0.86	1.0
10	20	15	12	9	8	20	10

Figure 3: An Example Gene

Note that the number of control points for each DOF function need not be the same. However, in actual implementation, we need to set the maximum possible number of control points.

One concern in genetic algorithm is how to design a representation such that it cannot represent infeasible solutions to a problem. In the animation problem, it is important that the character not be led to perform motions considered undesirable or impossible (such as rotating the leg 360 degrees about the x-axis). While, there is nothing in our representation to prevent such occurrence, the manner in which we deal with the issue is by using an evaluation function that assigns low score to undesirable or impossible motion.

ERROR: stackunderflow
OFFENDING COMMAND: setcolor

STACK:

0