Entertainment Computing 5 (2014) 285–294

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

Planar character animation using genetic algorithms and GPU parallel computing ☆

Benjamin Kenwright

Edinburgh Napier University, United Kingdom

ARTICLE INFO

Article history:
Received 8 March 2014
Revised 23 September 2014
Accepted 24 September 2014
Available online 5 November 2014

Keywords:
Animation
Control
Interactive
Graphical processing unit (GPU)
Computer games
Genetic

ABSTRACT

The emergence of evolving search techniques (e.g., genetic algorithms) has paved the way for innovative character animation solutions. For example, generating human movements without key-frame data. Instead character animations can be created using biologically inspired algorithms in conjunction with physics-based systems. While the development of highly parallel processors, such as the graphical processing unit (GPU), has opened the door to performance accelerated techniques allowing us to solve complex physical simulations in reasonable time frames. The combined acceleration techniques in conjunction with sophisticated planning and control methodologies enable us to synthesize ever more realistic characters that go beyond pre-recorded ragdolls towards more self-driven problem solving avatars. While traditional data-driven applications of physics within interactive environments have largely been confined to producing puppets and rocks, we explore a constrained autonomous procedural approach. The core difficulty is that simulating an animated character is easy, while controlling one is more complex. Since the control problem is not confined to human type models, e.g., creatures with multiple legs, such as dogs and spiders, ideally there would be a way of producing motions for arbitrary physically simulated agents. This paper focuses on evolutionary genetic algorithms, compared to the traditional data-driven approach. We demonstrate generic evolutionary techniques that emulate physically-plausible and life-like animations for a wide range of articulated creatures in dynamic environments. We help address the computational bottleneck of the genetic algorithms by applying the method to a massively parallel computational environments, such as, the graphical processing unit (GPU).

1. Introduction

1.1. Preface

How do creatures move in the real-world? For example, how do we as humans move? Can you explain it? While we are able to simulate dynamic rigid body worlds (e.g., by means of unintelligent solid objects moving due to forces and collisions) in interactive virtual environments, such as video-games and training simulations, the control of complex physics-based character systems is an active area of research. The traditional approach of creating character animations in interactive worlds is through data-driven solutions (i.e., pre-recorded animations) that are blended and applied to rigid body structures. Alternatively, a number of approximate procedural approaches are available, that include using low-dimensional systems (e.g., inverted pendulum or optimization search algorithms). These low-dimensional models allow the animation to run in real-time while mimicking the fundamental dynamic principles of real-world characters (e.g., balanced stepping logic). Hence, creating animated character animations using procedural techniques is highly challenging, interesting, and important.

1.2. Motivation

Static poses (i.e., motion-less poses) can be used to control proportional-derivative controllers. However, in dynamic environments, where the character is in motion (e.g., running and stepping), the timing information might not correlate between the aim and target. For example, if a character is taking a step, the character's foot will miss the target, and in interactive environments, such as games, this can be unacceptable. Hence, we need to constantly create poses on-the-fly that can accomplish diverse tasks while being able to adapt to the environment around them

☆ This paper has been recommended for acceptance by Ryohei Nakatsu. (Where Ryohei Nakatsu is the name of the Handling Editor).

E-mail address: b.kenwright@napier.ac.uk

http://dx.doi.org/10.1016/j.entcom.2014.09.003
1875-9521/

(e.g., changing terrain types, such as, slopes, bridges, and potholes).

1.3. Challenges

The key challenges that make it difficult to reproduce physically-plausible life-like movements in "real-time" that mimic real-world creatures are given below. We also point out where an evolutionary genetic algorithm sits in relation to each problem.

- *Realism*: A particular character model gives rise to a large set of possible motions with different styles. While the physics-based system enforces the control laws, the genetic algorithm has the challenging task of searching for poses that re-construct and reproduce the intricate and agile movement we observe in nature.
- *High dimensions*: Characters have a relatively high number of degrees-of-freedom, making the search for the appropriate control parameters hard. The search space can be vast and non-continuous while containing multiple solutions. Although *single threaded* genetic algorithms can cope with large search spaces, the stringent demands make it impossible for interactive applications or even performed in reasonable time-frames.
- *Underactuation*: Dynamically simulated characters are difficult to control because they have no direct control over their global position and orientation. Even staying upright is a challenge for large disturbances. In order to succeed, the genetic algorithm control law must plan ahead to determine actions that can stabilize the body (e.g., not just a static moment in time, but will have to the run simulation forwards to predict how the movement will evolve based on the physical constraints) [9].
- *Contacts*: Characters are restricted to move within a certain region of its three-dimensional environment, and these constraints are difficult to maintain in real-time control systems. Furthermore, frequent ground contacts create a highly discontinuous search space rendering most continuous controller synthesizing methods ineffective at planning over longer time horizons.

1.4. Contribution

The contributions of this paper are: (1) We introduce and demonstrate a procedural approach for synthesizing the evolution of artificial-life animations without data (e.g., key-frames) through a genetic based search algorithm (i.e., an evolving survival of the best approach). (2) We create physics-based character motions that adapt and explore complex environments (e.g., we go beyond flat terrain and include stairs and slopes). (3) We extend the technique to non-biped type avatars (e.g., creatures with multiple legs, such as cats and snakes). (4) We explore performance improvements through exploitation of massively parallel execution environments, such as the graphical processing unit (GPU) for evolving populations. (5) We discuss practical aspects, such as fitness functions and controller models, for generating the joint control torques of the articulated character.

2. Related work

Evolving genetic algorithms have been used across multiple disciplines to solve a wide variety of problems (e.g., robotics, biomechanics, and computer animation). While there is a plethora of exciting and interesting research in the area of procedural animation, we review and compare essential research that is specifically related and has inspired this work.

2.1. Where did it all start?

The idea of using a population of solutions to solve practical engineering optimization problems was considered several times during the 1950's and 1960's [5]. The concept of a Genetic Algorithm (GA) was coined by John Holland and his colleagues and students at the University of Michigan in the 1970s [12]. However, GA was originally popularized by one of John Holland's students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation in 1980s [7].

2.2. Space–time constraints

Space–time constraints, originally proposed by Witkin and Kass [17], allows an artist to specify how an animated figure should move but not how to do it. This approach is very appealing in animation. Witkin and Kass [17] original space–time constraint method generated kinematic motions that satisfied both high level goals (e.g. jump from here to there) while appearing physically plausible. Furthermore, the resulting motions exhibited many of the principles of traditional animation (e.g., squash, stretch, follow-through, and anticipation). Space–time constraints use sequential quadratic programming (SQP) to optimize the kinematic positions of an articulated figure, using energy consumption as an objective function that constraint the solution in order to ensure that the motion is physically plausible. Unfortunately, this method generates solutions that depend on an initial guess given to the optimizer, which can unfortunately converge on a local minimum that creates a kinematic motions that appears unnatural. In addition, it is by no means clear that energy is the best criteria to optimize.

2.3. Automation

Other researchers proposed methods of automatically generating walking motion using search and optimization techniques [16,10]. Both of which have demonstrated impressive results; yielding physically plausible motions and automatically finding a number of walking methods. However, the resulting motion had high specialization (i.e. they made walking gaits, not general motions), but low specificity (i.e. they both simply walked forward, rather than having more specific instructions, like, walk to position X). These methods were optimizing structures of fixed complexity (a network of fixed topology and a stimulus response table, respectively). We believe that this is a disadvantage compared to a representation which may change its own complexity. We chose a different representation, namely a mathematical description (computer program) describing how the joint forces vary with time and changes in the state of the simulation. We believe that this representation offers many advantages, and is appropriately optimized using the genetic algorithm technique.

2.4. Genetic algorithms

Karl Sims [13] used a genetic-like approach for evolving procedural textures, where the fitness metric was human evaluation. Later, Karl Sims also described the use of evolutionary programming to design entire creatures [14], in which even the creature topology was evolved, for example, walking and swimming. In contrast, our work deals with figures of fixed topology and geometric structure, as one would expect for character animation.