The Dancing Genome Project: Generation of a Human-Computer Choreography using a Genetic Algorithm

François-Joseph Lapointe
Département de sciences biologiques
Université de Montréal
C.P 6128, Succ. Centre-ville
Montréal, Québec, Canada, H3C 3J7
lapoinf@biol.umontreal.ca

Martine Époque
Département de danse
Université du Québec à Montréal
C.P 8888, Succ. Centre-ville
Montréal, Québec, Canada, H3C 3P8
epoque.martine@uqam.ca

ABSTRACT

In this paper, we present an interactive genetic algorithm for the generation of human-computer choreography, using motion capture technology. First, we introduce the four steps of the algorithm to (1) define a movement vocabulary, (2) initialize movement sequences, (3) generate mutants, and (4) select mutant sequences to create a choreography. Then, we show how this approach is implemented in real time to create interaction among dancers. Finally, we run simulations to assess the convergence rate of the algorithm, before generating a simple duet for actual and virtual dancers.

Categories and Subject Descriptors

J.5 [**Arts and Humanities**]: Performing arts (dance)

General Terms

Algorithms, Experimentation.

Keywords

Dance, genetic algorithm, human-computer interaction, motion capture software, virtual choreography.

1. INTRODUCTION

Recent advances in computer technology, motion capture sensors and cognitive science have revolutionized the ways choreographers design and create dances [2, 3, 7]. Indeed, it is now common for human performers to share the stage with virtual dancers. Merce Cunningham was one of the first choreographers to use computers to create dances with *Life Forms* [5, 10]. Since then, numerous animation softwares have been used to design choreographic pieces using computers [8]. The *LIFEanimation* software has been developed in Montréal by the LARTech (*www.lartech.uqam.ca*) as a tool to help choreographers generate virtual dances using motion capture technology (Figure 1). It was used by Martine Époque and Denis Poulin to create *Tabula Rasa*, a choreography for twenty-two dancers and one virtual dancer, which was shown at various locations around the world. In *Tabula Rasa*, the movements of the virtual dancers are choreographed beforehand so as to match those executed by the actual dancers on stage. In the present paper, we introduce an interactive genetic algorithm to be used jointly with *LIFEanimation*, thus allowing for the real-time generation of human-computer choreography.

The following sections will present the different steps of the algorithm for generating and selecting choreographic mutants in order to evolve the movements of actual and virtual dancers. The implementation of the algorithm for real-time performances will then be detailed. Computer simulations will also be performed to validate the parameters of the genetic model and this approach will be used to generate a simple duet for virtual and human dancers.

Figure 1. A pair of virtual dancers generated by the *LIFEanimation* software (LARTech).

2. THE *CHOREOGENETICS* ALGORITHM

In a previous paper, we introduced a genetic algorithm (GA) to transform movement sequences through mutations and selection [11]. Here, we propose the first application of this *Choreogenetics* approach for real-time multimedia performances. The different steps of the genetic algorithm are detailed in the next sections, before implementing interaction among virtual and human dancers.

2.1 Defining the vocabulary

The starting point of any dance lies in the movements that define the vocabulary and the style of a particular choreographer. The first step of the algorithm thus requires characterizing well-defined movements, which can be interpreted by actual and virtual dancers alike. This vocabulary may be fixed in advance or be modified in real time to allow for complex human-computer interactions. With no loss of generality, let us say that only four movements are defined to begin with: (1) run, (2) jump, (3) turn, and (4) fall. These movements are performed by actual dancers and coded with the *LIFEanimation* software to create a virtual vocabulary.

2.2 Initializing the movement sequences

Once the vocabulary is defined, sequences of movements (phrases) can be initialized by sampling at random a number of movements with replacement from the vocabulary. For example, the sequence 1 3 4 2 1 (i.e. run, turn, fall, jump, run) may be generated for the actual dancer. Likewise, the sequence 4 4 3 1 2 1 (i.e., fall, fall, turn, run, jump, run) may be generated for the virtual dancer. These sequences thus represent the original phrases of the choreography that will be performed by the human and the computer. As shown here, such initial sequences may differ with respect to the order of movements, as well as the number of movements sampled from the same vocabulary, but two distinct vocabularies could also be used to generate more complex choreographies.

2.3 Generating the mutant sequences

The algorithm generates choreographic variation of movement sequences through different types of random genetic operations. Whereas simple mutations only apply to one sequence at a time, complex mutations involve the combination of two sequences with one another:

- A substitution consists of replacing one movement (or more) by new movements sampled from the vocabulary to create a mutant sequence (e.g., 1 3 4 2 1 => 1 3 4 4 1).

- An insertion consists of adding one movement (or more) to the sequence to create a longer mutant sequence (e.g., 1 3 4 2 1 =>1 3 3 4 2 1).

- A deletion consists of removing one movement (or more) from the sequence to create a shorter mutant sequence (e.g., 1 3 4 2 1 => 1 2 1).

- A repetition selects one movement (or more) from the sequence and repeats it a given number of times to create a longer mutant sequence (e.g., 1 3 4 2 1 => 1 3 4 3 4 2 1).

- A translocation selects one movement (or more) from the sequence and inserts it at a different position to create a mutant sequence (e.g., 1 3 4 2 1 => 4 2 1 1 3).

- An inversion selects a number of movements (more than one) from the sequence and inverts the order to create a mutant sequence (e.g., 1 3 4 2 1 => 4 3 1 2 1).

- An horizontal transfer is a complex mutation that copies a number of movements from one sequence and inserts them into a second sequence to create a mutant sequence (e.g., 1 3 4 2 1 + 4 4 3 1 2 1 => 1 4 4 3 3 4 2 1).

- An hybridization event (or crossover) is a complex mutation that crosses movements from a pair of sequences to create a hybrid mutant sequence (e.g., 1 3 4 2 1 x 4 4 3 1 2 1 => 1 4 4 3 2 1 2 1).

At each iteration of the algorithm, a fixed number of mutants (e.g., 1000) are generated from each sequence, and mutants are compared with one another to determine which will be the next sequence in the evolution of the choreography.

2.4 Selecting the mutant sequences

The evolution of movement sequences is ensured by selecting at each iteration of the algorithm the mutants that are more similar to one another, gradually creating in the process unison among the dancers. To do so, a Levenstein distance [12] is computed between the actual and virtual sequences to select the best matching pairs of movement sequences. This distance counts the fewest number of mutations required to transform on sequence into another through substitutions, insertions and deletions. As an example, the distance between 1 3 4 2 1 and 4 4 3 1 2 1 equals 3, because two substitutions and one insertion are required to transform the first sequence into the second one. Namely, the smaller is the value of the Levenstein distance, the more similar are the sequences under comparison. Once the “best” mutant representing each dancer is selected, these new sequences are used as input for the next iteration of the algorithm.

3. IMPLEMENTING INTERACTION

The previous sections described in details the various steps of the *Choreogenetics* algorithm to generate and select mutant sequence of movements. In this section, we show how the interaction among virtual and actual dancers is implemented for real-time performances. This interactive genetic algorithm (IGA) proceeds as follows (see Fig. 2):

1- To begin with, initial sequences are generated for the actual and virtual dancers (see section 2.2) by sampling movements at random for a common vocabulary, or from different vocabularies (see section 2.1).

2- One sequence is transmitted in real time to the actual dancer to be performed on stage, while the other sequence performed by the virtual dancer is projected on a screen.

3- A sliding window is then used to record into memory the actual and virtual sequences for comparison purposes. (the number of movements to save at each iteration of the algorithm is fixed *a priori* by the user).

4- The movement sequences of the actual and virtual dancers are compared with one another to assess convergence. If the two sequences are different, they are fed back to the genetic algorithm to be mutated (see section 2.3) and the best mutant (the sequence closest to the other sequence) is selected (see section 2.4) in turn for both the virtual and actual dancers. If the two sequences are identical, the algorithm stops (goto step 6).

5- The steps 2 to 4 are repeated iteratively until the two sequences converge to the same movements.

6- The choreographic process is terminated when the Levenstein distance between the two sequences equals zero. This unique final sequence is performed in unison by the virtual and actual dancers to end the choreography.

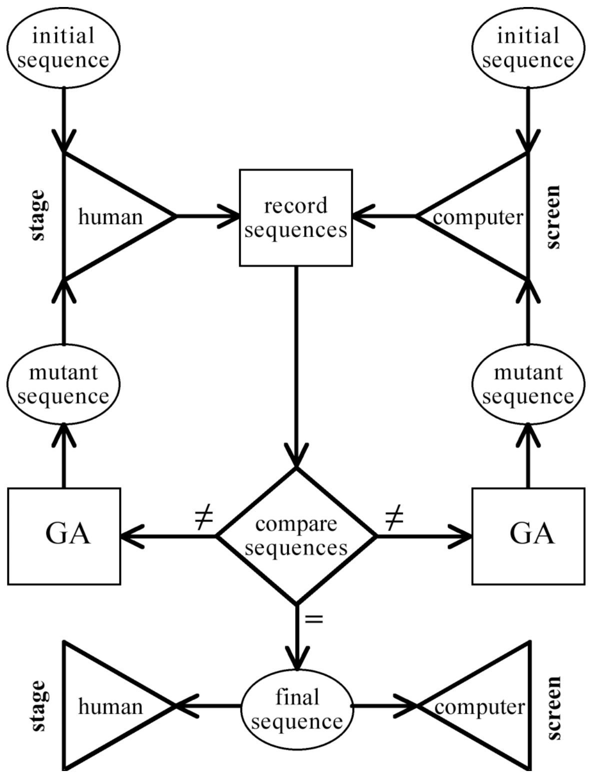

Figure 2. Flowchart of the interactive genetic algorithm (IGA) designed to generate duets for human and virtual dancers. See text for more details.

4. VALIDATION OF THE ALGORITHM

Depending on the number of mutants generated at each iteration, the lengths of the initial sequences (i.e., the number of movements), the complexity of the vocabulary, and the relative probabilities of the different types of mutation (simple and complex), the algorithm may converge more or less rapidly to a unique sequence to be performed at the same time by the virtual and actual dancers.

To demonstrate the convergence rate of the interactive genetic algorithm, we have generated random input choreographic sequences containing from 10 to 20 movements (L), and then ran simulations allowing for 10, 100 or 1000 mutants (m) at each iteration, with all types of mutations being equally likely. In all cases, the vocabulary (v) was set to four movements (1, 2, 3, 4), and for each combination of parameters, 100 replicates were generated.

Table 1. Average number of iterations required for a pair of random initial sequences of the same length (L) to convergence to a unique final sequence, as a function of the number of mutants (m) generated per iteration. The number of movements of the vocabulary (v) is set to four.

	L = 10	**L = 15**	**L = 20**
m = 10	78.3	173.7	398.9
m = 100	9.9	29.5	40.8
m= 1000	5.7	6.9	10.8

Table 1 shows the results of these simulations. It presents the average number of iterations required for both initial sequences to converge to a unique final sequence. These results show that a faster convergence is obtained for shorter sequences (L = 10) and a larger number of mutants (m = 1000). However, the convergence rate is also affected by the number of movements (v) in the vocabulary. When eight movements are used in the simulations, the numbers of iterations required to reach a common sequence are larger (Table 2), on average, than when only four movements are authorized (Table 1).

Table 2. Average number of iterations required for a pair of random initial sequences of the same length (L) to convergence to a unique final sequence, as a function of the number of mutants (m) generated per iteration. The number of movements of the vocabulary (v) is set to eight.

	L = 10	**L = 15**	**L = 20**
m = 10	167.0	497.5	1045.4
m = 100	21.8	48.8	107.5
m= 1000	7.1	12.5	16.2

Under optimal conditions (i.e., short sequences and a large number of mutants per generation), a pair of random initial movement sequences can converge to the same final sequence in as few as 5 iterations (5.7 iterations on average; Table 1).

Figure 3 presents a duet for actual and virtual dancers that was generated by the evolution of movement sequences with an optimal model (v = 4, L = 10, m = 1000), using the interactive genetic algorithm described in the previous sections (Fig. 2). With time, the sequences are shown to converge to the same movements, with Levenstein distances decreasing from t0 to t5.

	Actual dancer	**Virtual dancer**	
t0:	4124322142	3144313444	d=6
	substitution	**substitution**	
t1:	$\underline{1411331}142$	$3144313\underline{331}$	d=6
	translocation	**substitution**	
t2:	$1\underline{142}411331$	$314431333\underline{4}$	d=5
	insertion	**substitution**	
t3:	$114241133\underline{3}1$	$3144\underline{1}13334$	d=3
	substitution	**insertion**	
t4:	$1142411333\underline{4}$	$3\underline{1}144113334$	d=2
	deletion	**deletion**	
t5:	114_4113334	_1144113334	d=0

Figure 3. Convergence of random initial sequences towards a unique final sequence. At each iteration (t), the types of mutations affecting the original sequences are in bold, and the movements transformed by the mutations are underlined. The Levenstein distances (d) between the corresponding sequences are also provided to assess convergence. The algorithm stops when the final sequences are identical (d = 0).

5. DISCUSSION

The generation of movement sequences through random permutations and substitutions has been used time and time again by choreographers. Namely, Merce Cunningham used the *I Ching* [14] to derive choreographies that were performed to the random music of John Cage. As such, the generation of mutant sequences with an interactive genetic algorithm represents another way of creating random dances. However, there exists a fundamental difference between both approaches. Whereas all possible movements permutations are equally likely under a strictly random model (e.g., the *I Ching*), a genetic algorithm relies on an objective function to sort the sequences according to a selection criterion. In other words, the selection phase (section 2.4) replaces the choreographer by choosing only the "best" mutants among all possible movement sequences. In the present application, these "best" mutants are those that closely match the movements performed by the other dancer (virtual or human). Thus, the duets generated by the algorithm are not entirely random; otherwise they would never produce unison among the dancers. Still, these computer-generated choreographies are not predetermined either. The end points of the dance are not fixed in time and space, although convergence is always guaranteed regardless of the input sequences and the movement vocabularies.

It is one thing to be able to create human-computer duets, but it is an entirely different thing to compose choreographies of artistic value. The mere succession of movements performed at the same time by virtual and actual dancers does not make a piece of art. It is the interaction among dancers and the conjunction codes that enable one movement to flow into the next that allows for the generation of aesthetic dances. There exist many different ways of merging two movements together (e.g., interpolation and fusion), and we have used some of the improvisation methods proposed by William Forsythe [6] to do so with actual dancers. The same approach does not always apply to virtual dancers, however, unless all possible combinations of movements are captured beforehand (not possible for large vocabularies). Interestingly, *LIFEanimation* has a BLEND function that can be used to create transitions from one movement to the next. Obviously, these algorithmic transitions will be different from those adopted by human dancers, but it is of interest to compare the corresponding movement sequences. Future experiments will assess the conditions under which humans and computers would make the same choices.

Several studies have applied evolutionary computation in arts [13], but it is in the musical field that genetic algorithms have been the most popular [1, 4, 9]. The present paper introduces an application of genetic algorithms in dance. We have successfully used the *Choreogenetics* approach to generate interactions among actual dancers in the past. The proposed implementation is the first use of an interactive genetic algorithm to create a human-computer choreography for real-time performance environments.

6. ACKNOWLEDGMENTS

The author wishes to thank Mathieu Campeau and Frédérick Gravel for their experimentations with the algorithm, and Daniel Léveillé for providing a movement vocabulary. This work is supported by a grant from the Social Sciences and Humanities Research Council of Canada to Martine Époque.

7. REFERENCES

[1] Biles, J. A. GenJam in perspective: a tentative taxonomy for GA music and art systems. *Leonardo*, 36, 1 (2003), 43-45.

[2] Bradley, E., and Stuart, J. Using chaos to generate variations on movement sequences. *Chaos*, 8 (1998), 800-807.

[3] Bret, M., Tramus, M.-H., and Berthos, A. Interacting with an intelligent dancing figure : artistic experiments at the crossroads between art and cognitive science. *Leonardo*, 38, (2005), 46-53.

[4] Burton, A. R., and Vladimirova, T. Generation of musical sequences with genetic techniques. *Comput. Music J.*, 23 (1999), 59-73.

[5] Copeland, R. *Merce Cunningham: The Modernizing of Modern Dance*. Routledge, New York, NY, 2004.

[6] Forsythe, W. *Improvisation Technologies. A Tool for the Analytical Dance Eye*. ZKM Digital Arts Edition, Karlsruhe, Germany, 2003. [CD-ROM]

[7] Hagendoorn, I. Cognitive dance improvisation: how study of the motor system can inspire dance (and vice versa). *Leonardo*, 36, 3(2003), 221-227.

[8] Herbison-Evans, D., Green, R. D., and Butt, A. Computer animation with NUDES in dance and physical education. *Aust. Comput. Sci. Comm.*, 4, 1 (1982), 324-331.

[9] Johnson, C. G. Exploring sound-space with interactive genetic algorithms. *Leonardo*, 36, 1, (2003), 51-54

[10] Lansdowne, J. The computer and choreography. *IEEE Computer,* 11 (1978), 19-30.

[11] Lapointe, F.-J. *Choreogenetics*: the generation of choreographic mutants through genetic mutations and selection. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)* (Washington D. C., USA, June 25-29, 2005). ACM Press, New York, NY, 2005, 366-369.

[12] Levenstein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Report*, 10, 8 (1966), 707-710.

[13] Romero Cardlada, J. J., and Johnson, C. Genetic algorithms in visual art and music. *Leonardo*, 35, 2 (2002), 175-184.

[14] Yan, J.F. *DNA and the I Ching –The Tao of Life*. North Atlantic Books, Berkeley, CA, 1991.