Force-Based Evolutionary Computation Approach for Automatic Skeletal Motion Learning in Human Animation

Francisco Calatayud[1], Luis de la Vega-Hazas[1], Andrés Iglesias[2,3,†]
[1]BINARYBOX STUDIOS, Calle Juan XXIII, 1, 39001, Santander, SPAIN
[2]Department of Information Science, Faculty of Sciences, Narashino Campus
Toho University, 2-2-1 Miyama, 274-8510, Funabashi, JAPAN
[3]Dpt. of Applied Mathematics & Comp. Sci., E.T.S.I. Caminos, Canales y Puertos
Universidad de Cantabria, Avda. de los Castros, s/n, 39005, Santander, SPAIN
[†]Corresponding author. Email: iglesias@unican.es
Website: http://personales.unican.es/iglesias

***Abstract*—The realistic animation of human characters is a hot topic of research in computer graphics, with remarkable applications in computer animation and video games. A current trend is the application of powerful evolutionary computation techniques to meet the needs of increasing sophistication in the field. These techniques are inspired by the principles and mechanisms of biological evolution, such as selection, mutation, recombination, crossover, and so on. A great advantage of such methods is that they do not make any assumption on the problem to be solved. In this paper we present a new evolutionary computation approach for automatic skeletal motion learning in human animation. This approach is intended to generate automatically a sequence of motions on a human skeleton leading to plausible and realistic movements of the human body. This sequence is obtained autonomously (i.e., without human intervention) through an iterative intelligent process where the digital characters learn the optimal values of forces applied to selected bones according to the desired motion routines for proper movement. An illustrative example is discussed in detail to show the performance of this approach. This method can readily be adapted/extended to other skeleton configurations and interesting motions with only minor modifications.**

***Index Terms*—artificial intelligence, evolutionary computation. machine learning, computer animation, automatic motion iteration, skeletal model, virtual actors, human motion**

I. Introduction

A. Motivation

One of the major issues in computer animation and video games nowadays is the realistic animation of human characters. Research in this field is increasingly moving towards sophisticated digital actors evolving in complex virtual 3D worlds and with the ability to be conscious of their environment and other digital actors and interact with them in an intelligent and plausible way (from the standpoint of a human observer). To this aim, artificial intelligence (AI) techniques are increasingly seen as valuable and powerful tools in these fields for many different purposes. Sophisticated AI techniques (expert systems, decision trees, machine learning) are applied for behavioral animation of the NPCs (non-player characters) in computer animation [11], [12] and video games [5], [10]. Crowd simulation in computer movies can be generated automatically through swarm intelligence and other AI techniques [18]. New AI-assisted software tools are now available for automatic iteration of terrains, buildings, cities, characters, and assets. The list of applications is growing to meet the current needs of computer animation and video games industries [22].

A critical challenge in this regard concerns the animation of human motion. We are so used to see it in real life that our sensory systems are extremely well trained to capture even its most subtle details. Sometimes, we distinguish familiar people (relatives, friends, acquaintances) at a relatively far distance by simply observing their body motion patterns. This remarkable feature has been advantageously used in traditional animation for many years. Think for instance on the acclaimed animation techniques from Disney, where the movement of animal characters and inert objects was recreated from identifiable human motion patterns to create hilarious effects in animation movies and cartoons.

Unfortunately, describing such motion patterns in a machine-understandable way is more complex that you might think at first sight. Among the solutions devised to address this problem, those based on artificial intelligence techniques are being increasingly used owing to their good performance and suitability for task automation. A remarkable step forward in this field is given by *evolutionary computation* (EC), a subfield of artificial intelligence focused on algorithms for optimization inspired by biological evolution [6]. Generally speaking, evolutionary computation algorithms are population-based metaheuristic methods inspired by the principles and mechanisms of biological evolution, such as selection, mutation, recombination, crossover, and so on. A great advantage of such methods is that they do not make any assumption on the problem to be solved such as the functional structure of the objective function or the underlying fitness landscape. This feature makes them ideal tools for dealing with problems subjected to uncertainty, noise, and where little (or none at all)

information about the problem is available.

B. Aims and structure of the paper

In this context, the present paper introduces a new evolutionary computation approach for automatic skeletal motion learning in human animation. The ultimate goal of our proposal at large is to develop an artificial intelligence engine for human motion that can be seamlessly embedded into software packages for computer animation and video games. Ideally, this engine (currently at initial stages of the process) should be able to generate *automatically* and *autonomously* (i.e., without human intervention) a sequence of motions on a human skeleton leading to plausible and realistic movements of the human body.

Obviously, this objective is too ambitious and general to be fully addressed here. Yet, this paper is a significant first step in this process. It focuses on one of the major components of our ongoing AI engine: the evolutionary computation kernel. In particular, in this paper we introduce our evolutionary computation approach, which is described in detail in Section IV. Its performance is then discussed through its application to an illustrative and interesting problem: given an initial and a final pose, we seek to apply our EC approach to allow a physics-driven skeleton learn autonomously how to reach the final pose from the initial one. In our example, we consider a set of identical physics-driven skeletons seated on the ground (initial pose). The method applies forces on selected bones to obtain a stable final pose where all skeletons are standing, getting firmly into a stable upright position on their feet (final pose). Such forces are modulated by a set of evolutionary operators carefully chosen so as to make the digital characters learn to stand up by themselves.

The structure of this paper is as follows: in Sect. II we describe briefly some previous work in the field. Our skeletal model is explained in detail in Sect. III. Then, our evolutionary computation approach is presented through an in-depth discussion of its main evolutionary operators in Sect. IV. Our experimental results are reported in Section V. The paper closes with the main conclusions and some hints about future work in the field.

II. Previous Work

A. EC for computer modeling and animation

Evolutionary computation became popular to the computer graphics community during the 90s, thanks to the seminal work of Karl Sims by applying genetic algorithms for the creation and modeling of new creatures both in shape and in behavior [19], [20]. About the same period, application of evolutionary design principles was reported for urban planning and architecture [1], [7], [21]. It was followed in the next decade by the "modeling by examples" paradigm, a data-driven approach for constructing new 3D models by assembling parts from previously existing ones [8]. The already constructed objects are stored in a large database of 3D meshes. The design task is fully controlled by the user, who selects the parts of interest from the 3D meshes through intelligent scissoring, and composes all pieces together in different ways to form new objects. Other approaches to support creative discovery in 3D modeling can be found in [2], [3] and the very nice references within. We also remark the excellent contribution on evolutionary design in [23].

Regarding human animation, genetic algorithms have been used for optimizing character gaze behavior animation based on the preference of viewers about where and how to gaze without the need to manually set the gaze control parameters [13]. Kernel-based approaches such as radial basis functions (RBF) and Gaussian processes (GP) have been proposed for blending different types of locomotion [14]–[17]. Neural networks are applied for character control in [9] through a real-time character control mechanism using a novel neural network architecture called phase-functioned neural network. The system can automatically produce motions where the character adapts to different geometric environments such as walking and running over rough terrain, climbing over large rocks, jumping over obstacles, and crouching under low ceilings.

B. EC for video games

Advanced AI has also been applied to video games. In *The Division's* game the player motion is automated, leaving the player extra time to focus on more important actions such as shooting. In *The Witness*, the AI helps the player solve some nasty problems such as getting caught on edges or tapped in walls. The *AI Director* of the two *Left 4 Dead* games has been used to created more realistic player experiences, such as the simulation of crowds where all characters behave differently. In other cases, the AI is used to learn from the player, like in *Forza* series, where a neural network was used to watch the human player in action and learn by imitation.

Regarding the evolutionary approaches for video games, *City Conquest* uses a genetic algorithm during the design process to identify dominant strategies and evolve the game design, even before it is released. In *Darwin's Nightmare* game, evolutionary computation is applied to drive the exploration of a large combinatorial space defining the behavior and appearance of enemy crafts. Also, particle swarm optimization has been applied to the problems of pathfinding and action planning in video games [4].

III. The Skeletal Model

In this paper, the human character animation is performed at the skeleton level. Therefore, it is convenient to introduce some basic definitions and notation for a proper representation and better understanding of the whole process. We remark however that the basis and notation given here do not necessarily match those commonly found in anatomical studies and medical treatises or even in other computer animation settings. In general, the definitions in this paper are simplified versions of the real-world models, as we focus only in some specific aspects required for our purposes while neglecting others for the sake of simplicity. In this sense, all statements onwards apply only to our simplified skeletal model.

A. Basic definitions and notation

In this paper we consider a collection of η human virtual actors $\{\mathcal{A}_i\}_{i=1,...,\eta}$. The physical structure of each virtual character $\mathcal{A}_i$ is described by a skeletal model $\mathbf{\Lambda}_i$, which provides the geometric rigid structure of the virtual body much like its real-world counterpart actually does. The skeleton $\mathbf{\Lambda}_i$ consists of two components: a collection of bones and a collection of joints. For each $\mathbf{\Lambda}_i$ the set of bones, denoted as $\mathbf{\Theta}_i$, consists of λ_i bones $\mathbf{\Theta}_i = \{\beta_j^i\}_{j=1,...,\lambda_i}$, while the set of joints $\mathbf{\Xi}_i$ consists of all joints $\{\xi_{k,l}^i\}_{k,l}$ connecting bones β_k^i and β_l^i. The joints provide a natural representation for the constraints of different parts of the virtual body by a proper selection of the kind of deformations allowed and their degrees of freedom (DOFs). In this sense, they play a quite similar role to the real-world human body joints in which they are originally based on.

To form a skeleton, the bones must be connected. A natural way to do so is to use a hierarchy, where each bone belongs to a parent and can have one or several children connected to it. For human body representation, we use a hierarchical tree in which the primary bone, called the root bone and represented onwards as β_1^i, is located in a central part of the body, generally the spine. All other bones β_j^i $(j > 1)$ are children of this root bone, either directly (first level) or indirectly (higher levels) via other intermediate bones β_k^i, which are children of the root bone (and possibly of other bones as well) and parents of this bone (and possibly of others too). In this way, for any given bone β_j^i we can define two types of sequences of bones: the forward (or outer) sequences, denoted by $\{\phi_F(\beta_j^i)\}_p$ and given by all bones that are children of β_j^i and connected by joints continuously, and the backward (or inner) sequences, denoted by $\{\varphi_B(\beta_j^i)\}_q$ and given by all bones connecting β_j^i with the root bone through joints in a continuous way. These sequences are important because affecting a bone β_j^i (for instance, by applying a force $\mathcal{F}$) also affects all of its children, i.e. all bones in any forward sequence $\phi_p(\beta_j^i)$, while the bones in the backward sequence $\varphi_q(\beta_j^i)$ are not affected by $\mathcal{F}$. Although this is somehow opposed to what happens in real life, this assumption simplifies the model and reduces its computational load.

B. Our prototypical skeletal model

Figure 1 shows the prototypical skeletal model for human actors used in this paper. The figure is organized in two parts, displaying the set of bones (left) and the set of joints (right), respectively. As the reader can see, we consider a very basic and simple (yet good enough for the purposes of this paper) skeleton, comprised of 20 interconnected bones and 16 joints, fully reported in Table I. The initial 'L' and 'R' letters followed with a hyphen stand for left and right, respectively. Figure 2 shows the top-down hierarchical tree of all bones of our skeleton model. It consists of a undirected graph where nodes of the tree represent the bones, which are connected by edges. As shown in the figures, the tree starts with the root bone at the top of the hierarchy. From it, there are five different sequences of connected bones going down until terminal nodes (feet, hands and head) are found.

TABLE I: List of bones and joints of the skeletal model used in this paper.

Bones		*Joints*	
1. Spine1	11. L-Foot	1. Spine1	11. R-Shoulder
2. Spine2	12. R-Clavicle	2. Spine2	12. R-Elbow
3. Spine3	13. R-UpperArm	3. Spine3	13. R-Wrist
4. L-Clavicle	14. R-LowerArm	4. Neck	14. R-Hip
5. L-UpperArm	15. R-Hand	5. L-Shoulder	15. R-Knee
6. L-LowerArm	16. R-Hip	6. L-Elbow	16. R-Ankle
7. L-Hand	17. R-UpperLeg	7. L-Wrist	
8. L-Hip	18. R-LowerLeg	8. L-Hip	
9. L-UpperLeg	19. R-Foot	9. L-Knee	
10. L-LowerLeg	20. Head	10. L-Ankle	

IV. The Method

Our method is a population-based evolutionary computation approach that operates on an initial population of skeletons $\{\mathbf{\Lambda}_i\}_{i=1,...,\eta}$ all lying on the floor and then moving to get seated. This initial movement is identical for all skeletons and has been pre-processed and then stored, so it is not affected by our approach. The method starts when all skeletons are seated on the floor and works in an iterative fashion. At initial time of each iteration ν, a force vector ${}^{[\nu]}\mathcal{F} = \left\{{}^{[\nu]}\mathcal{F}_1, \ldots, {}^{[\nu]}\mathcal{F}_\eta\right\}$ is applied, where ${}^{[\nu]}(.)$ is used to indicate the iteration and each force ${}^{[\nu]}\mathcal{F}_i$ is applied on skeleton ${}^{[\nu]}\mathbf{\Lambda}_i$. At its turn, ${}^{[\nu]}\mathcal{F}_i = \left\{{}^{[\nu]}\mathcal{F}_{i,j}\right\}_{j=1,...,\lambda_i}$, where force ${}^{[\nu]}\mathcal{F}_{i,j}(t)$ is applied on bone β_j^i at instance time τ of iteration ν. This force takes the form of an impulsive force consisting of a burst of χ forces $\{\zeta_{i,j}^\kappa\}_{\kappa=1,...,\chi}$ applied sequentially at times $\tau+\kappa\Delta t$. Mathematically, it means that:

$$ {}^{[\nu]}\mathcal{F}_{i,j}(t) = \sum_{\kappa=1}^{\chi} {}^{[\nu]}\zeta_{i,j}^\kappa \delta_{\mathbf{T}}(\tau + \kappa\Delta t) \qquad (1) $$

where $\delta(.)$ is the Dirac delta function and $\mathbf{T}$ is the vector of time impulses of the force. For simplicity, in this paper we assume that $\lambda_i = \lambda$, $\forall i$. We also assume that all forces in our method are exerted vertically upwards except for gravity, which works in the opposite direction. As a result of the action of these forces, the skeletons move for a while until reaching a stable position (in our case, until all skeletons keep fully static for 5 seconds). Once the steady-state is attained, the new skeleton configurations are evaluated and ranked according to a given fitness function. In this work, the fitness function $\mathbf{\Psi}$ is defined as the sum of three terms:

$$ {}^{[\nu]}\mathbf{\Psi}(\mathbf{\Lambda}_i) = {}^{[\nu]}\mathbf{\Psi}_1(\mathbf{\Lambda}_i) + {}^{[\nu]}\mathbf{\Psi}_2(\mathbf{\Lambda}_i) + {}^{[\nu]}\mathbf{\Psi}_3(\mathbf{\Lambda}_i) $$

where each term evaluates a specific feature of the skeleton configuration. Assuming that the body is resting but stretched, for a skeleton to be standing it is required that:

(1) the feet are firmly standing on the ground;
(2) the hip should lie on the imaginary vertical axis Y that goes upwards from the feet to the head and at a distance from the ground given by the length of the skeleton from the foot base to the hip;

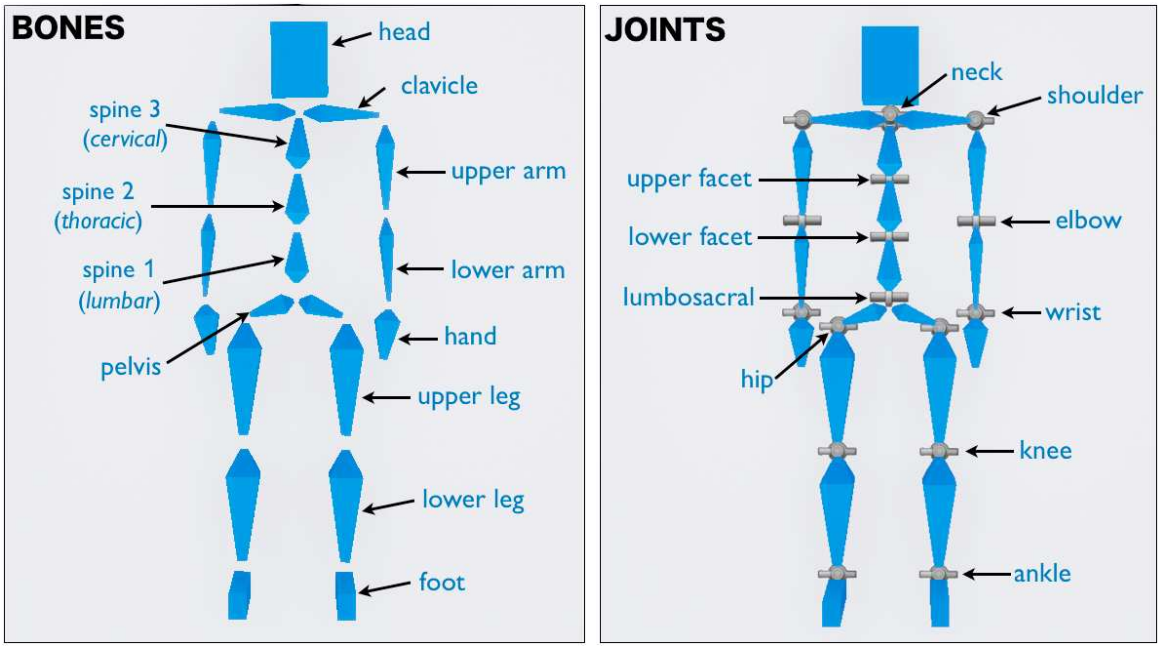

Fig. 1: Skeletal model used in this paper: (left) bones; (right) joints.

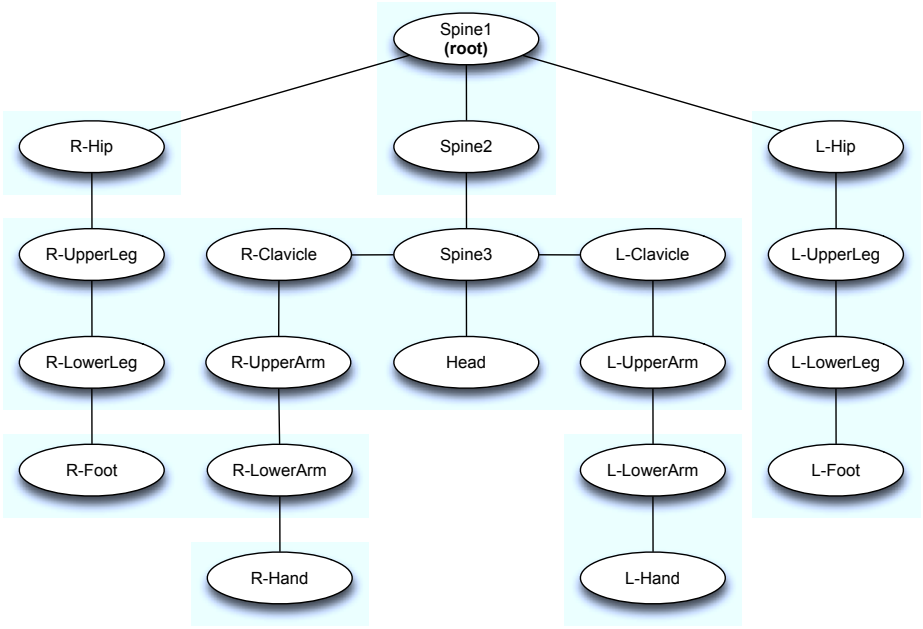

Fig. 2: Skeleton represented by a top-down hierarchical tree.

(3) the axes of the head and the body are aligned so the center of the head lies on the vertical axis Y and with the same orientation, and the distance from the ground to the top of the head corresponds to the body height.

Functions $\mathbf{\Psi}_i$ are associated with the three conditions, respectively. We remark that conditions (1)-(3) are ideal and, hence, very difficult to replicate accurately. They are also unreasonable, as we expect different characters to stand up differently, similar to how human beings actually do. For these reasons, we allow a threshold error given by three imaginary boxes $\{B_i\}_i$. B_1 is a 2D box on the ground marking the available area for feet placement, providing some flexibility on constraint (1) by allowing the feet to be placed freely within this area as long as they are stepping on the ground. B_2 and B_3 are volumetric boxes intended to provide some flexibility on the position and stance by allowing lateral and

vertical displacements of the hip and the head respectively, provided that such bones still keep inside their respective boxes. Mathematically, these functions are defined as:

$$ {}^{[\nu]}\boldsymbol{\Psi}_1^i(\boldsymbol{\Lambda}_i) = \mathcal{H}\left({}^{[\nu]}\beta_{11}^i \cup {}^{[\nu]}\beta_{19}^i, B_1\right) \quad (2) $$

$$ {}^{[\nu]}\boldsymbol{\Psi}_2^i(\boldsymbol{\Lambda}_i) = d_2\left(\gamma\left({}^{[\nu]}\beta_{8}^i \cup {}^{[\nu]}\beta_{16}^i\right), \gamma(B_2)\right) \quad (3) $$

$$ {}^{[\nu]}\boldsymbol{\Psi}_3^i(\boldsymbol{\Lambda}_i) = d_2\left(\gamma\left({}^{[\nu]}\beta_{20}^i\right), \gamma(B_3)\right) \quad (4) $$

where $\mathcal{H}$ is the Hausdorff distance between sets, d_2 is the Euclidean distance, γ computes the 3D geometrical center of a set, and $\cup$ is the set union operator.

The ranked skeletons are sorted in increasing order and stored in a list L, whose first and last elements correspond to the best and worst values in current iteration ν, denoted as ${}^{[\nu]}\boldsymbol{\Lambda}^*$ and ${}^{[\nu]}\boldsymbol{\Lambda}_*$ respectively. Improvement over time is achieved by applying three evolutionary operators: selection, reproduction, and mutation. Similar to genetic algorithms and other evolutionary methods, the selection operator, denoted as $\odot$, is used to promote the best individuals according to the Darwinian survival of the fittest. In our method, we consider *elitism* so that the best solutions from the current iteration are transferred directly to the next iteration without further modification. We also consider a monoparental reproduction operator denoted as $\bowtie$ and based on *cloning*, where the best individual is cloned (i.e., copied identically) so that further operators are applied on the clones while preserving the original individual for elitism. The clones undergo mutation under the action of an mutation operator $\otimes$ that applies Gaussian white noise additive perturbations on ${}^{[\nu]}\mathcal{F}_{i,j}$. This procedure is repeated iteratively until ${}^{[\nu]}\boldsymbol{\Psi}(\boldsymbol{\Lambda}_i) = 0$, $\forall i$, meaning that all skeletons in our population hold our constraints according to Eqs. (3)-(4).

V. Experimental Results

Figure 3 shows an illustrative example of our experimental results. In this experiment, we consider a set of 10 skeletons $\{\boldsymbol{\Lambda}_k\}_{k=1,\ldots,10}$, placed in a row and numbered from left to right. In this example the method is executed for 14 iterations, shown in sequence from top to bottom in Fig. 3. For each iteration, we show the final configuration of the skeletons along with the three boxes B_i described in previous section. As mentioned above, our starting point consists of the skeletons seated on the ground (initial pose). Then, we consider an initial population of forces ${}^{[0]}\mathcal{F}_i$ applied on our skeletons. Since the skeletons are already seated, forces are applied only on the bones in the trunk (active bones) to capture better the real physics of the process. This means that ${}^{[0]}\mathcal{F}_i = \left\{{}^{[0]}\mathcal{F}_{i,j}\right\}_{j\in\mathcal{A}_B}$ where $\mathcal{A}_B$ is the list of indices for the active bones, given by: $\mathcal{A}_B = [1..7]\cup[12..15]$, where $[a..b]$ means all integer numbers between a and b (including a, b if they are integers). The forces are initially chosen randomly according to a uniform distribution within a feasible interval $[\alpha_j, \beta_j]$ specific for each bone. Then, our method is applied iteratively so that the force values improve over time according to our fitness function.

TABLE II: Execution results for the example in Figure 3.

ν	${}^{[\nu]}\boldsymbol{\Lambda}^*$	$\bowtie\left({}^{[\nu]}\boldsymbol{\Lambda}_*\right)$	$\odot\left(\left\{{}^{[\nu]}\boldsymbol{\Lambda}_i\right\}_i\right)$
1	1	4	2,3,5,6,7,8,9,10
2	4	9	1,2,3,5,6,7,8,10
3	9	4,10	1,2,3,5,6,7,8
4	10	2,4,9	1,3,5,6,7,8
5	10	2,3,4,9	1,5,6,7,8
6	10	2,3,4,6,9	1,5,7,8
7	10	2,3,4,5,6,9	1,7,8
8	7	2,3,4,5,6,8,9,10	1,7
9	9	2,3,4,5,6,7,8,10	1
10	9	2,3,4,5,6,7,8,10	1
11	1	2,3,4,5,6,7,8,9,10	–
12	4	2,3,5,6,7,8,9,10	1
13	9	1,2,3,4,5,6,7,8,10	–
14	**6**	**1,2,3,4,5,7,8,9,10**	–

We have applied a color code to the skeletons in Fig. 3 for easier identification and better understanding of the iterative process. At each iteration ν, the best individual of the population, ${}^{[\nu]}\boldsymbol{\Lambda}^*$, is displayed in red. The corresponding indices are indicated in second column of Table II. Similarly, the individuals with the worst fitness value are displayed in green. They are then subjected to cloning and mutation and included in the list $\bowtie\left({}^{[\nu]}\boldsymbol{\Lambda}_*\right)$ (third column of Table II). Finally, the rest of individuals are selected for elitism and transferred to the next iteration without further mutation. They are displayed in blue, and form the set $\odot\left(\left\{{}^{[\nu]}\boldsymbol{\Lambda}_i\right\}_i\right)$ (fourth column of Table II).

As shown in the figure, at the initial iterations almost all skeletons fail to get the final pose. In this particular example, the first skeleton successfully reaches the target pose and it becomes the global best (in red). We remark, however, that this is not usually the case. Instead, some initial iterations are generally required to get the first successful result (about 90% of cases). After ranking, the worst solution (corresponding to $i = 4$ and displayed in green) is replaced by a clone of the best, $\bowtie\left({}^{[1]}\boldsymbol{\Lambda}_1\right)$ and then mutated $\otimes\left[\bowtie\left({}^{[1]}\boldsymbol{\Lambda}_1\right)\right]$. Then, the process is restarted again for $\nu = 2$ and so on. The full process is summarized in Table II. The table reports (in columns): the iteration number ν, the index of the global best for this iteration ${}^{[\nu]}\boldsymbol{\Lambda}^*$, the indices of the skeletons to be replaced by clones of the best and then mutated $\bowtie\left({}^{[\nu]}\boldsymbol{\Lambda}_*\right)$, and the indices of the skeletons passed to the next iteration without further modification (except the best), $\odot\left(\left\{{}^{[\nu]}\boldsymbol{\Lambda}_i\right\}_i\right)$. As the reader can see, the number of cloned and mutated individuals (in green) increases with the iterations, as this is the driving force for enhancement. Simultaneously, the number of individuals selected for elitism (in blue) decreases, as a result of the other individuals in the population are improving, so those selected are no longer good enough for elitism and must be mutated to further improve their fitness. The application of these evolutionary operators leads to improvement over the time until all skeletons finally reach the target pose. Final results of this simulation example (reached for iteration $\nu = 14$) are highlighted in bold in Table II. The corresponding graphical

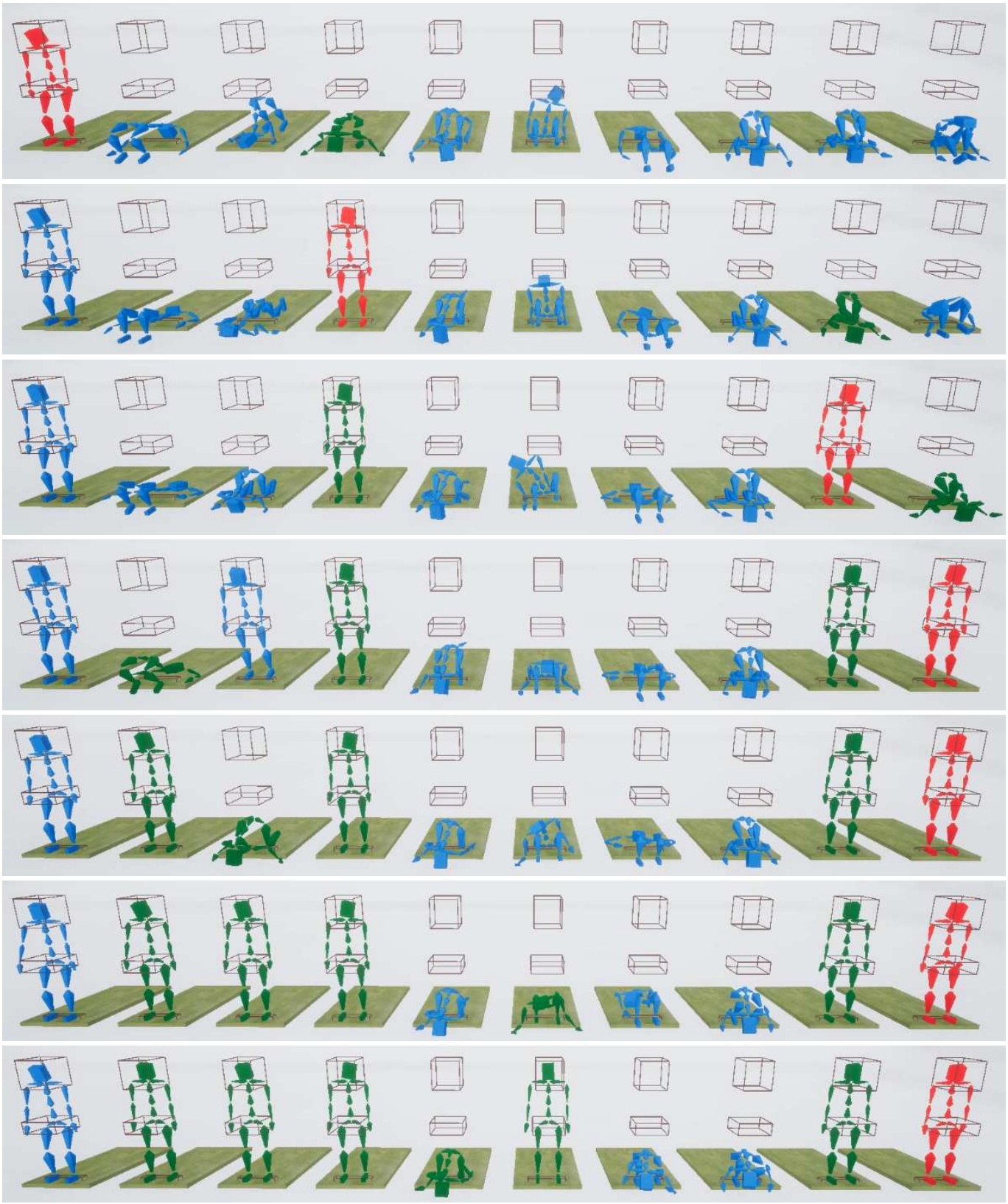

Fig. 3: Example of execution for 10 skeletons: (top-bottom) iterations 1–7

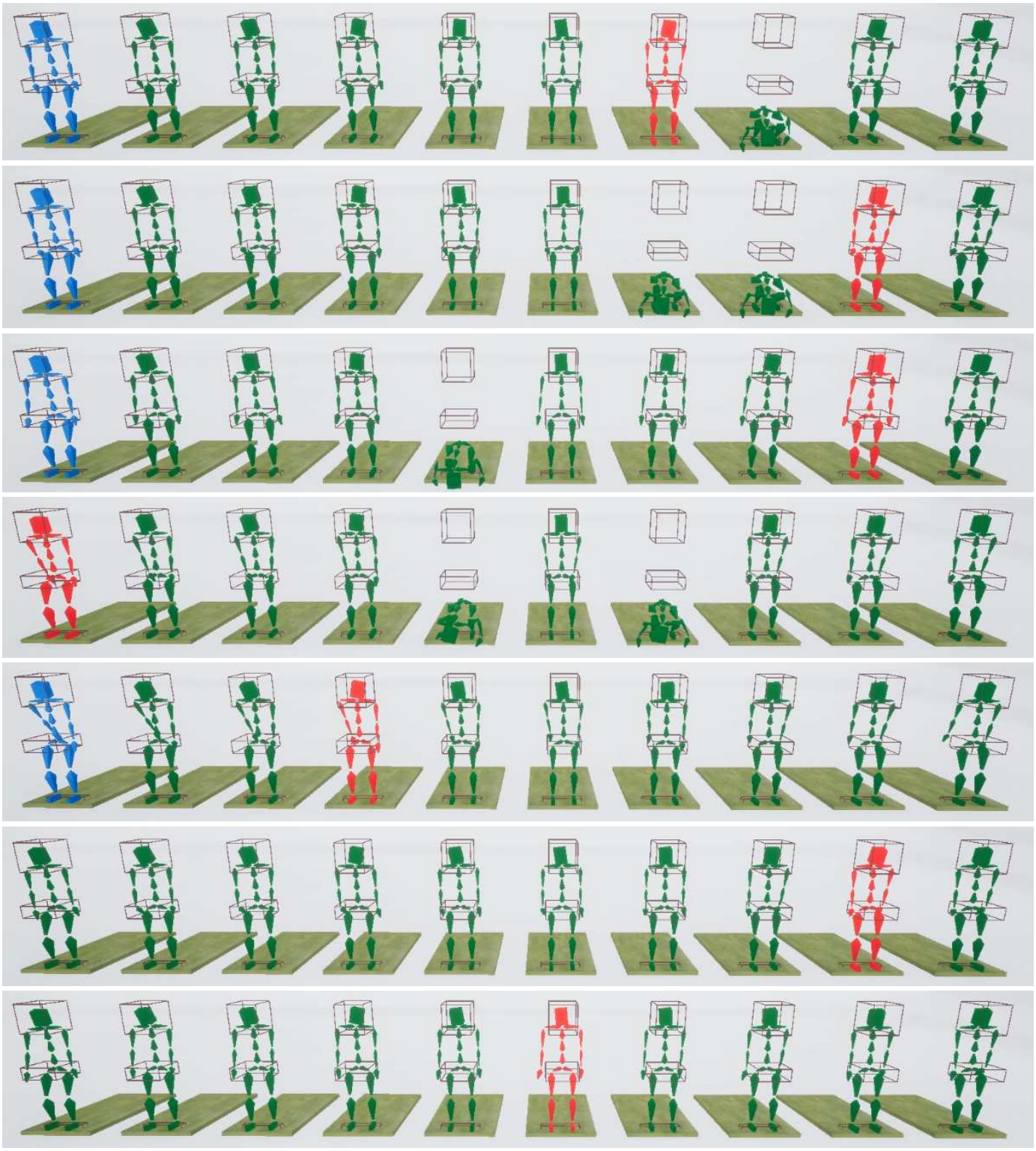

Fig. 3: *(cont'd)* Example of execution for 10 skeletons: (top-bottom) iterations 8–14

output for this iteration is depicted as the final picture in Figure 3. It shows that all skeletons are properly standing after 14 iterations. Although only one example is discussed here, we performed several executions for this problem, all successful in less than 40 iterations.

Regarding the implementation issues, all the computational

work in this paper has been performed on a personal PC with a 2.6 GHz Intel Core i7 processor and 8 GB of RAM. The simulations have been carried out by using the powerful game engine *Unreal Engine 4*, by *Epic Games*. This engine is particularly adequate for our purposes, as it provides a number of useful tools and features for different tasks. For instance, it uses the *PhysX 3.3* physics engine to drive the physical simulation calculations, compute all collisions and all other physics-related tasks. It also provides a nice graphical output for animation. Most of the source code has been implemented by the authors by using the visual scripting system *Blueprint*, although some source code has been implemented by direct coding in the Unreal scripting language *UnrealScript* and C++.

VI. Conclusions and Future Work

In this work we introduced a new evolutionary computation approach for automatic skeletal motion learning in human animation. The approach is designed to allow the digital characters to learn several motion routines automatically and autonomously, without any user intervention. The procedure consists of applying forces on selected bones to trigger movement. The bones are chosen according to the particular targeted motion. At initial stages, random forces are applied; then, they are modulated by our EC approach. The most successful forces are preserved for the next iterations (elitism) through a selection operator according to the Darwinian principle of *survival of the fittest*. To widen the search space seeking for more promising values, we also apply two addtional evolutionary operators: reproduction and mutation. The former is a simple monoparental cloning, while the latter is performed by perturbing the best solutions locally, aiming at finding the global best while avoiding getting stuck in local minima.

This evolutionary strategy pays off. As shown in the example discussed, the method is able to reach the final pose in just a few iterations. In this paper only one example is discussed because of limitations of space. However, we have carried out more than one hundred independent simulations for this particular problem and found the method to be very stable in all cases. In our simulations, we always got a suitable solution in less than 40 iterations. Regarding the computation times, the whole process (including rendering and animation) can be performed in real-time. This is not surprising, however, as we are working with a small population of simple skeletons.

As mentioned above, this work is just one step in the whole process of developing an artificial intelligence engine for human motion to be ultimately embedded into software packages for computer animation and video games. As such, this work can be improved in many different ways. We plan to extend this approach to other (more challenging) motion routines, where probably more bones should participate in the motion process. We also plan to consider skeletons with a larger number of bones, particularly for motions involving the hands and feet. On the other hand, the selection of bones for motion is now performed manually by the user. We plan to develop an intelligent system to make this choice automatically through artificial intelligence techniques.

Acknowledgment

This research work has received funding from the project PDE-GIR of the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 778035, the Spanish Ministry of Economy and Competitiveness (Computer Science National Program) under grant #TIN2017-89275-R of the Agencia Estatal de Investigación and European Funds FEDER (AEI/FEDER, UE), and the project #JU12, supported by public body SODERCAN and European Funds FEDER (SODERCAN/FEDER UE).

References

[1] Bentley, P.J.: *Evolutionary Design by Computers*. Morgan Kaufman Publishers (1999).

[2] Chaudhuri, S., Koltun, V.: 2010. Data-driven suggestions for creativity support in 3D modeling. *ACM Trans. on Graphics* (Proc. of SIGGRAPH Asia) **29**(6), 183:1–9 (2010).

[3] Chaudhuri, S., Kalogerakis, E., Guibas, L., Koltun, V.: Probabilistic reasoning for assembly-based 3d modeling. *ACM Trans. on Graphics* (Proc. of SIGGRAPH) **30**, 35:1–10 (2011).

[4] Diaz, G., Iglesias, A.: Swarm intelligence scheme for pathfinding and action planning of non-player characters on a last-generation video game. *Advances in Intelligent Systems and Computing*, **514**, 343–353 (2017).

[5] Diaz, G., Iglesias, A.: Intelligent behavioral design of non-player characters in a FPS video game through PSO. *LNCS*, **10385**, 246–254 (2017).

[6] Engelbrecht, A.P.: *Fundamentals of Computational Swarm Intelligence*. John Wiley and Sons, Chichester, England (2005).

[7] Frazer, J.: *An Evolutionary Architecture*. Architectural Association Publications (1995).

[8] Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz,, S., Dobkin, D.: Modeling by example. *ACM Trans. on Graphics* (Proc. of SIGGRAPH) **23**(3), 652–663 (2004).

[9] Holde, D., Komura, T., Saito, J.: Phase-functioned neural networks for character control. *ACM Transactions on Graphics* (ACM SIGGRAPH 2017), **36**(4), Article No. 42 (2017).

[10] Jacopin, E.: Game AI planning analytics: evaluation and comparison of the AI planning in three first-person shooters. In: *Proc. of Tenth Annual AAAI Conf. on Art. Intell. & Interactive Digital Entertainment - AIIDE 2014*. AAAI Press, Palo Alto, CA, pp. 119–124 (2014).

[11] Iglesias, A., Luengo, F.: New goal selection scheme for behavioral animation of intelligent virtual agents . *IEICE Transactions on Information and Systems*, **E88-D**(5), 865–871 (2005).

[12] Iglesias, A., Luengo, F.: AI framework for decision modeling in behavioral animation of virtual avatars. *LNCS*, **4488**, 89–96 (2007).

[13] Mori, H., Toyama, F., Shoji, K.: Optimization of character gaze behavior animation using an interactive genetic algorithm. *Int. Journal of Asia Digital Art & Design*, **21** (1-4), 25–31 (2017).

[14] Mukai, T.: Motion rings for interactive gait synthesis. In: *Proc. of I3D*. 125–132 (2011).

[15] Mukai, T., Kuriyama, S.:. Geostatistical motion interpolation. *ACM Trans on Graphics*, **24**(3), 1062–1070 (2005).

[16] Park, S.I., Shin, H.J., Shin, S.Y.: On-line locomotion iteration based on motion blending. In: *Proc. SCA*. 105–111 (2002).

[17] Rose, C., Cohen, M.F., Bodenheimer. B.: Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, **18**(5), 32–40 (1998).

[18] Schwab, B.: *AI Game Engine Programming* (2nd. edition). Course Technology, Boston, MA (2009).

[19] Sims, K.: Artificial evolution for computer graphics. In: *Proc.of ACM SIGGRAPH*, 319–328 (1991).

[20] Sims, K.: 1994. Evolving virtual creatures. In: *Proc. of ACM SIGGRAPH*, 15–22 (1994).

[21] Soddu, C., Colabella, E.: Recreating the citys identity with a morphogenetic urban design. In: *Proc. of Int. Conf. on Making Cities Livable*, 5–9 (1995).

[22] Woodcock, S.: Game AI: The state of the industry 2000–2001: It's not just art, it's engineering. *Game Developer*, August 2001, 36–44 (2001).

[23] Xu, K., Zhang, H., Daniel Cohen-Or, D., Chen, B.: Fit and diverse: set evolution for inspiring 3D shape galleries. *ACM Transactions on Graphics*, **31**(4), Article No. 57 (2012).