Applying Evolutionary Computation Operators for Automatic Human Motion Generation in Computer Animation and Video Games

Luis de la Vega-Hazas[1], Francisco Calatayud[1], Andrés Iglesias[2,3,†]

[1]BINARYBOX STUDIOS, Calle Juan XXIII, 1, 39001, Santander, SPAIN
[2]Toho University, 2-2-1 Miyama, 274-8510, Funabashi, JAPAN
[3]Universidad de Cantabria, Avda. de los Castros, s/n, 39005, Santander, SPAIN
[†]Corresponding Author: `iglesias@unican.es`
`http://personales.unican.es/iglesias`

Abstract. This paper presents an evolutionary computation scheme for automatic human motion generation in computer animation and video games. Given a set of identical physics-driven skeletons seated on the ground as an initial pose, the method applies forces on selected bones to obtain a stable pose with all skeletons standing. Those forces are modulated by a set of evolutionary operators (selection, reproduction, and mutation) to make the digital characters learn to stand up by themselves. An illustrative example is discussed in detail to show the performance of this approach. This method can readily be extended to other skeleton configurations and other interesting motions with little modification.

Keywords: artificial intelligence, evolutionary computation, computer animation, automatic motion generation, skeletal model, virtual actors

1 Introduction

Nowadays, artificial intelligence (AI) techniques are increasingly used in computer animation and video games as powerful tools to improve the performance of current development frameworks, simplify and speed up the graphical animation and game creation pipelines by task automation and support for digital production, and enhance the quality and realism of the final product. Sophisticated AI and machine learning techniques are currently being applied for behavioral animation of the NPCs (non-player characters) in video games [1–3]. Crowd simulation in computer movies can now be generated automatically taking advantage of swarm intelligence and other AI techniques [4]. New AI-assisted software tools are now available for automatic generation of terrains, buildings, cities, characters, and assets. Expert systems are applied to provide goal-directed specifications for cameras and lights. Machine learning and reasoning can be applied to the automatic generation of cinematic scripts for computer movies. And the list of applications is continuously growing and expanding to meet the current needs of computer animation and video games industries [5].

Human animation is one of the fields that can mostly benefit from recent advances in AI. Roughly, three approaches are mainly applied in the field: manual generation, physics-based, and data-driven. Human motion in computer animation is typically carried out by professional animators by using controllers and other software tools for the manipulation of joints, constraints, deformers and so on. This process relies on a number of shape and motion parameters, which are determined by the animator in a rather manual way. As expected, the process becomes error-prone and time-consuming, and demands a lot of expertise from the animators for realistic and believable motion. Physics-based approaches capture the natural aspect of a particular motion by applying a set of physical laws describing its dynamics. These methods are slow and very demanding in terms of CPU and memory storage, as they require to solve huge systems of equations. Also, in many cases the obtained solutions lead to unnatural trajectories and unrealistic motions. In data-driven techniques (e.g., motion capture), the motion of real human actors is tracked and recorded with cameras or other devices. This technology is very popular in the entertainment industry for movies and video games to get more realistic human movements (take, for instance, the acclaimed blockbuster *Avatar*). However, it requires specialized hardware for motion capture, which is costly and difficult to use. Furthermore, it is based on the records of specific actors, so it cannot be extended to any human character.

In conclusion, neither of the previous approaches is actually well suited for automatic motion generation. This is the motivation of our current work in the field. In particular, in this paper we raise the following questions:

Can a digital character learn to make a particular motion autonomously (i.e., without human intervention)? If so, how?

Obviously, these questions are too ambitious and general to be answered here. In this paper we focus on a particular motion: to stand up. Basically, we look for a motion pattern where the digital character is lying on the floor (initial pose) and at the end is standing, getting firmly into a stable upright position on his/her feet (final pose). Stated as such, this is a typical problem in inverse kinematics. It is also a difficult one, since it is highly multimodal: there might be several (even infinite) solutions for the motion pattern from the initial to the final pose. As a starting point, in this paper we try to elucidate whether it is possible to obtain one of such solutions *automatically and autonomously.* To this purpose, we rely on *evolutionary computation* (EC), a subfield of artificial intelligence focused on algorithms for global optimization inspired by biological evolution. Generally speaking, EC 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 make these methods ideal tools for dealing with problems subjected to uncertainty, noise, and with little (or none) information about the problem, which is actually our case here.

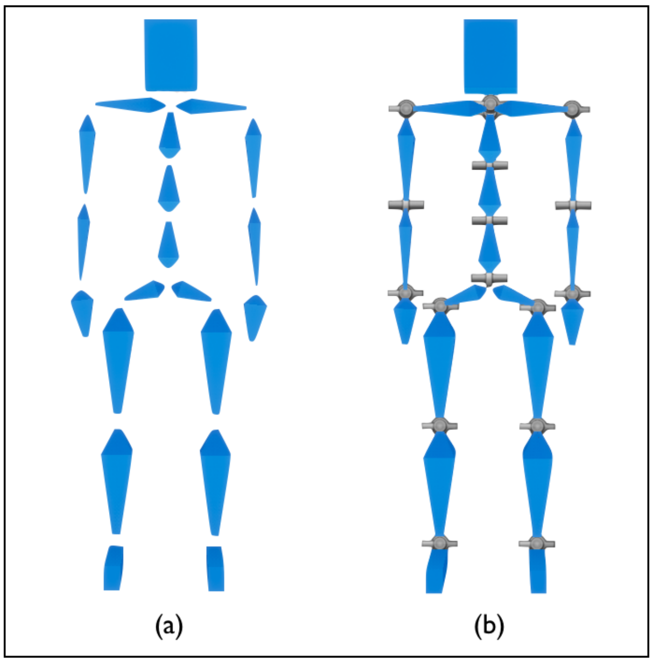

Fig. 1. Skeletal model used in this paper: (left) bones; (right) joints.

This paper presents an evolutionary computation scheme for automatic human motion generation in computer animation. Our approach applies forces on a given population of physics-driven human skeletons. 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. In Section 2 we describe briefly the skeletal model used in this paper. Then, our evolutionary computation approach is presented through an in-depth discussion of its main evolutionary operators in Section 3. Our experimental results are reported in Section 4.

2 The Skeletal Model

In this paper we consider a collection of η human virtual actors $\{\mathcal{H}_i\}_{i=1,\ldots,\eta}$. The physical structure of each virtual character $\mathcal{H}_i$ is described by a skeletal model $\mathcal{S}_i$, which provides the geometric rigid structure of the virtual body much like its real-world counterpart actually does. The skeleton $\mathcal{S}_i$ consists of two components: a collection of bones and a collection of joints. For each $\mathcal{S}_i$ the set of bones, denoted as $\mathcal{B}_i$, consists of λ_i bones $\mathcal{B}_i = \{\mathbf{b}^i_j\}_{j=1,\ldots,\lambda_i}$, while the set of joints $\mathcal{T}_i$ consists of all joints $\{\Xi^i_{k,l}\}_{k,l}$ connecting bones $\mathbf{b}^i_k$ and $\mathbf{b}^i_l$. 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

Table 1. List of bones and joints of the skeleton 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	

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 $\mathbf{b}_1^i$, is located in a central part of the body, generally the spine. All other bones $\mathbf{b}_j^i$ $(j > 1)$ are children of this root bone, either directly (first level) or indirectly (higher levels) via other intermediate bones $\mathbf{b}_k^i$, 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 $\mathbf{b}_j^i$ we can define two types of sequences of bones: the forward (or outer) sequences, denoted by $\{\phi_p(\mathbf{b}_j^i)\}_p$ and given by all bones that are children of $\mathbf{b}_j^i$ and connected by joints continuously, and the backward (or inner) sequences, denoted by $\{\varphi_q(\mathbf{b}_j^i)\}_q$ and given by all bones connecting $\mathbf{b}_j^i$ with the root bone through joints in a continuous way. These sequences are important because affecting a bone $\mathbf{b}_j^i$ (for instance, by applying a force F) also affects all of its children, i.e. all bones in any forward sequence $\phi_p(\mathbf{b}_j^i)$, while the bones in the backward sequence $\varphi_q(\mathbf{b}_j^i)$ are not affected by F. Although this is somehow opposed to what happens in real life, this assumption simplifies the model and reduces its computational load.

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) and labelled as (a) and (b), 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 1. 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,

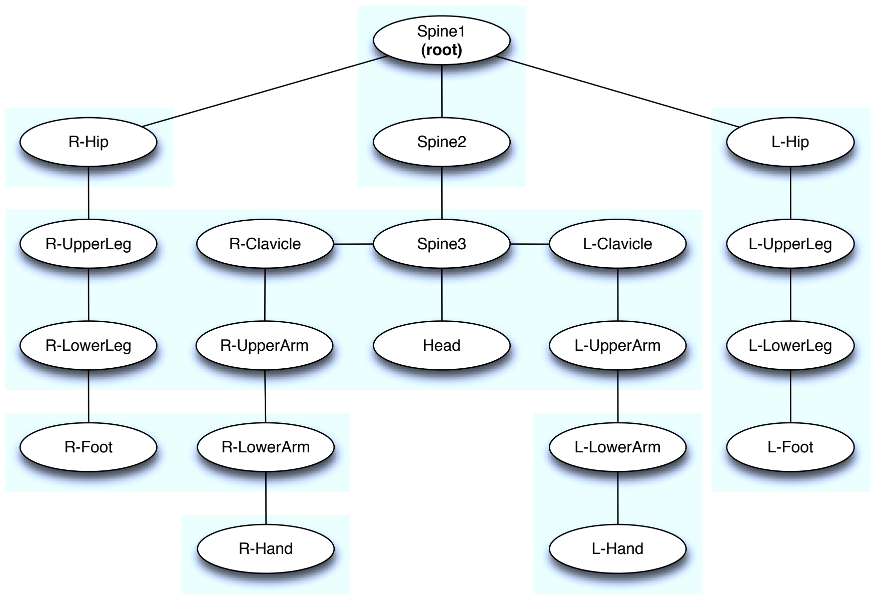

Fig. 2. Skeleton represented by a top-down hierarchical tree.

there are five different sequences of connected bones going down until terminal nodes (feet, hands and head) are found.

3 The Method

Our method is a population-based evolutionary computation approach that operates on an initial population of skeletons $\{\mathcal{S}_i\}_{i=1,\ldots,\eta}$ 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 generation g, a force vector ${}^{[g]}\mathcal{F} = \left\{{}^{[g]}\mathcal{F}_1, \ldots, {}^{[g]}\mathcal{F}_\eta\right\}$ is applied, where ${}^{[g]}(.)$ is used to indicate the generation and each force ${}^{[g]}\mathcal{F}_i$ is applied on skeleton ${}^{[g]}\mathcal{S}_i$. At its turn, ${}^{[g]}\mathcal{F}_i = \left\{{}^{[g]}\mathcal{F}_{i,j}\right\}_{j=1,\ldots,\lambda_i}$, where force ${}^{[g]}\mathcal{F}_{i,j}(t)$ is applied on bone $\mathbf{b}_j^i$ at instance time τ of generation g. This force takes the form of an impulsive force consisting of a burst of χ forces $\{\xi_{i,j}^\kappa\}_{\kappa=1,\ldots,\chi}$ applied sequentially at times $\tau + \kappa\Delta t$. Mathematically, it means that:

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

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: ${}^{[g]}\mathbf{\Psi}(\mathcal{S}_i) = {}^{[g]}\mathbf{\Psi}_1(\mathcal{S}_i) + {}^{[g]}\mathbf{\Psi}_2(\mathcal{S}_i) + {}^{[g]}\mathbf{\Psi}_3(\mathcal{S}_i)$, 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;
(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:

$$ {}^{[g]}\mathbf{\Psi}_1^i(\mathcal{S}_i) = \mathbf{\Theta}_H \left({}^{[g]}\mathbf{b}_{11}^i \cup {}^{[g]}\mathbf{b}_{19}^i, B_1 \right) \tag{2} $$

$$ {}^{[g]}\mathbf{\Psi}_2^i(\mathcal{S}_i) = d_2 \left(\gamma \left({}^{[g]}\mathbf{b}_8^i \cup {}^{[g]}\mathbf{b}_{16}^i \right), \gamma(B_2) \right) \tag{3} $$

$$ {}^{[g]}\mathbf{\Psi}_3^i(\mathcal{S}_i) = d_2 \left(\gamma \left({}^{[g]}\mathbf{b}_{20}^i \right), \gamma(B_3) \right) \tag{4} $$

where $\mathbf{\Theta}_H$ 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 generation g, denoted as ${}^{[g]}S^*$ and ${}^{[g]}S_*$ 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

Table 2. Results of iterations 9–17 for the example in Figure 4.

g	${}^{[g]}S^{*}$	$\bowtie({}^{[g]}S_{*})$	$\odot\left(\left\{{}^{[9]}S_i\right\}_i\right)$
7	9	6	1,2,3,4,5,7,8,10
8	7	6,2,	1,3,4,5,8,10
9	5	6,2,8	1,,3,4,7,9,10
10	3	6,2,8,10	1,4,5,7,9
11	3	6,2,8,10,9	1,4,5,7
12	7	6,2,8,10,9,4	1,3,5
13	4	6,2,8,10,9,4,1	3,5,7
14	1	6,2,8,10,9,4,1,3	5,7
15	1	6,2,8,10,9,4,1,3,5	7
16	1	6,2,8,10,9,4,1,3,5	7
17	**1**	**6,2,8,10,9,4,1,3,5**	**7**

according to the Darwinian survival of the fittest. In our method, we consider *elitism* so that the best solutions from the current generation are transferred directly to the next generation 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 ${}^{[g]}\mathcal{F}_{i,j}$. This procedure is repeated iteratively until ${}^{[g]}\mathbf{\Psi}(\mathcal{S}_i) = 0$, $\forall i$, meaning that all skeletons in our population hold our constraints according to Eqs. (3)-(4).

4 Experimental Results

Figure 4 shows an illustrative example of our experimental results. In this experiment, we consider a set of 10 skeletons $\{\mathcal{S}_k\}_{k=1,\ldots,10}$, placed in a row and numbered from left to right. In this example the method is executed for 17 iterations, shown in sequence from top to bottom and 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. As shown in the figure, they still fail to get any

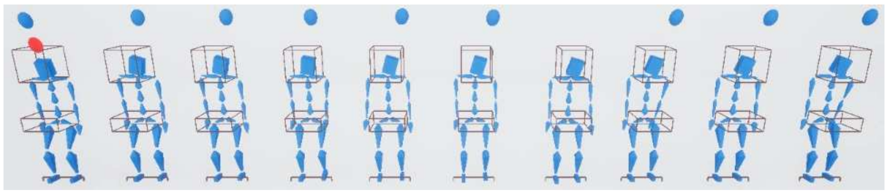

Fig. 3. Final results at iteration $g = 17$.

skeleton upright for the 7 first iterations. At $g = 8$ the first standing skeleton is obtained for $i = 9$; it becomes the best ${}^{[8]}S^* = \odot\left(\{{}^{[8]}S_i\}_i\right)$, marked by a red balloon above its head. After ranking, the worst solution (corresponding to $i = 6$ and marked by a blue balloon above the head) is replaced by a clone of the best, $\bowtie({}^{[8]}S_9)$ and then mutated $\otimes\left[\bowtie\left({}^{[8]}S_9\right)\right]$. Then, the process is restarted again for $g = 9$ and so on. The full process after iteration $g = 7$ is summarized in Table 2. The table reports (in columns): the iteration number g, the index of the global best for this iteration $\bowtie({}^{[g]}S_*)$, the indices of the skeletons to be replaced by clones of the best and then mutated $\bowtie({}^{[g]}S_*)$, and the indices of the skeletons passed to the next iteration without further modification (except the best). Final results of this simulation example (reached for generation $g = 17$) are highlighted in bold. The corresponding graphical output for this iteration is depicted in Figure 3. It shows that all skeletons are properly standing after 17 iterations. Although only one example is discussed here, we performed several executions for this problem, all successful in less than 50 iterations.

Acknowledgements
Research supported by projects: *PDE-GIR* (EU Horizon 2020 research and innovation program, MSCA-RISE grant agreement No 778035), *TIN2017-89275-R* (Agencia Estatal de Investigación and EU Funds FEDER, AEI/FEDER-UE), and *JU12* (SODERCAN and EU Funds FEDER, SODERCAN/FEDER-UE).

References

1. Díaz, 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** pp. 343–353 (2017).
2. Iglesias, A., Luengo, F.: New goal selection scheme for behavioral animation of intelligent virtual agents. *IEICE Transactions on Information and Systems*, Vol. E88-D, Issue 5, 865–871 (2005).
3. Iglesias, A., Luengo, F.: AI framework for decision modeling in behavioral animation of virtual avatars. *LNCS*, **4488**, 89–96 (2007).
4. Schwab, B.: *AI Game Engine Programming (2nd. edition).* Course Technology, Boston, MA (2009).
5. 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).

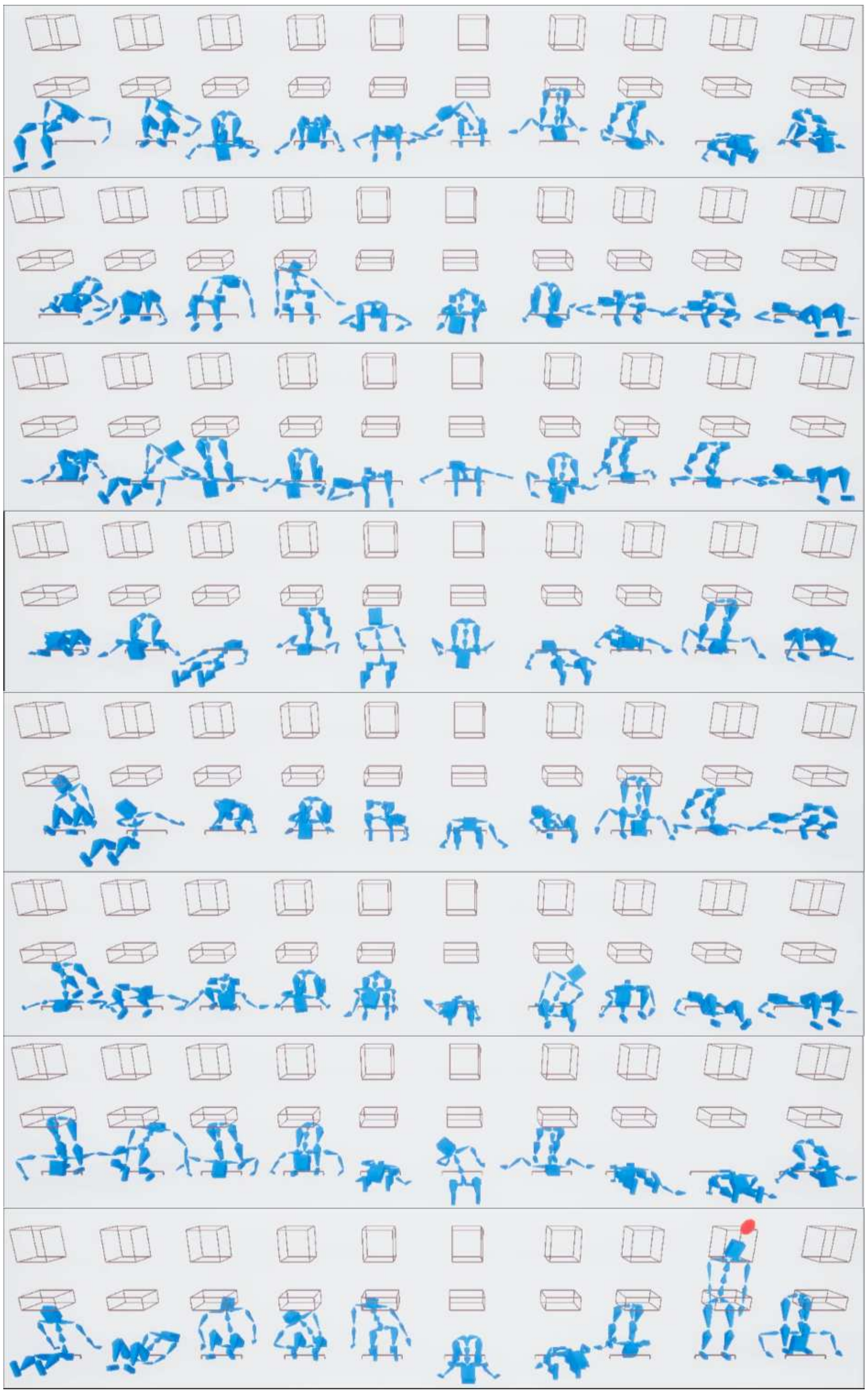

Fig. 4. Example of execution for 10 skeletons: (top-bottom) iterations 1–8

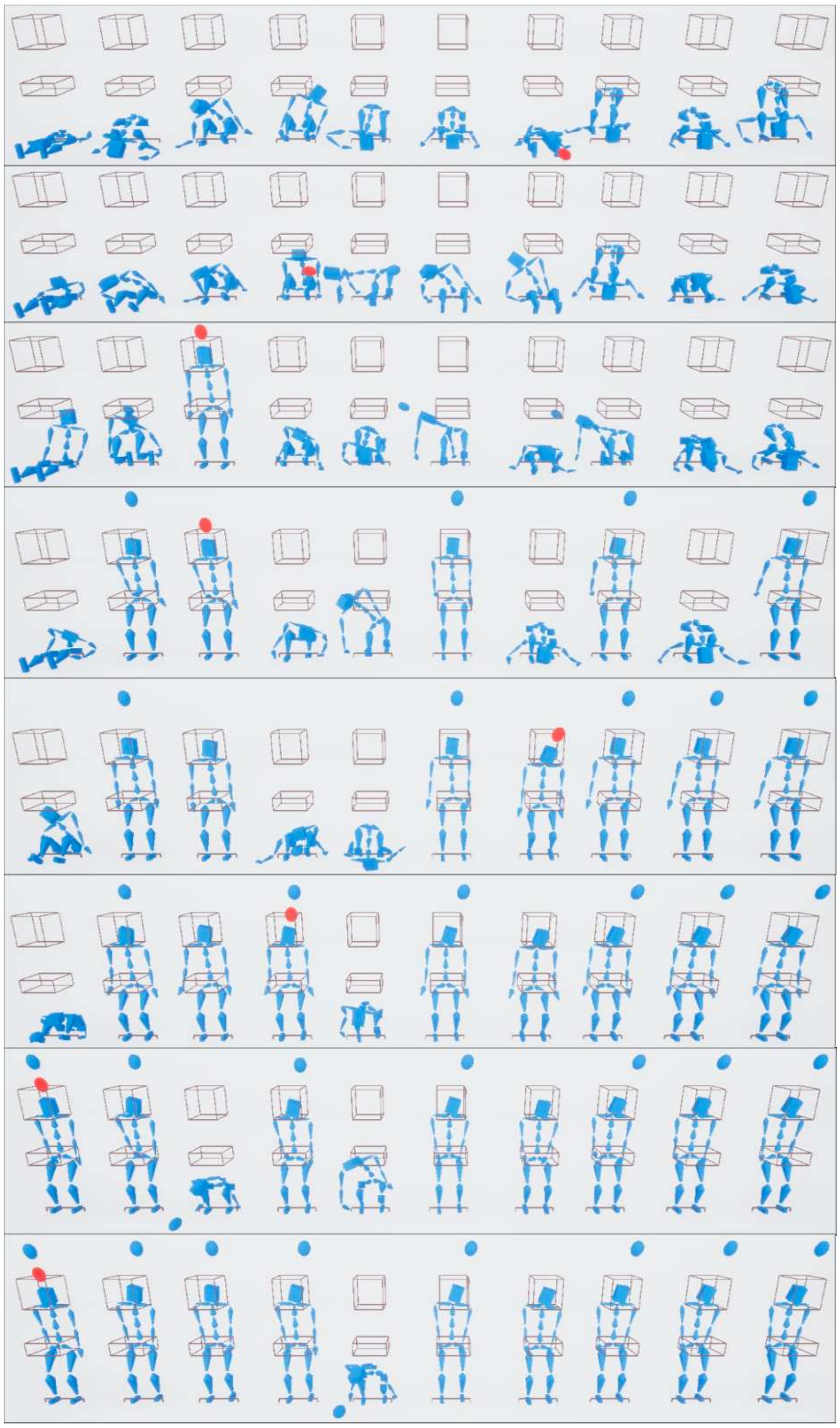

Fig. 4. *(cont'd)* Example of execution for 10 skeletons: (top-bottom) iterations 9–16