Published online 10 September 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1546

RESEARCH ARTICLE

Natural preparation behavior synthesis

Hubert P. H. Shum[1]*, Ludovic Hoyet[2], Edmond S. L. Ho[3], Taku Komura[4] and Franck Multon[5]

[1] Northumbria University
[2] Trinity College Dublin
[3] Hong Kong Baptist University
[4] University of Edinburgh
[5] University Rennes 2

ABSTRACT

Humans adjust their movements in advance to prepare for the forthcoming action, resulting in efficient and smooth transitions. However, traditional computer animation approaches such as motion graphs simply concatenate a series of actions without taking into account the following one. In this paper, we propose a new method to produce preparation behaviors using reinforcement learning. As an offline process, the system learns the optimal way to approach a target and to prepare for interaction. A scalar value called the *level of preparation* is introduced, which represents the degree of transition from the initial action to the interacting action. To synthesize the movements of preparation, we propose a customized motion blending scheme based on the level of preparation, which is followed by an optimization framework that adjusts the posture to keep the balance. During runtime, the trained controller drives the character to move to a target with the appropriate level of preparation, resulting in a humanlike behavior. We create scenes in which the character has to move in a complex environment and to interact with objects, such as crawling under and jumping over obstacles while walking. The method is useful not only for computer animation but also for real-time applications such as computer games, in which the characters need to accomplish a series of tasks in a given environment. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
preparation behavior; motion synthesis; reinforcement learning; motion blending; posture optimization

Supporting information may be found in the online version of this article.

***Correspondence**
Hubert P. H. Shum, Northumbria University.
E-mail: hubert.shum@northumbria.ac.uk

1. INTRODUCTION

In an environment where humans have to quickly conduct one motion after another, they adapt their behaviors and prepare for the next motion while approaching the target. This allows smooth transitions between actions, as well as a quick activation of the next action, especially when the person needs to interact with another person or the environment. For example, when a human notices a forthcoming obstacle that he or she needs to crawl under, he or she usually slows down and lowers his or her body before reaching the obstacle. It is not likely that one would run in full speed, stop absurdly, and start crawling.

Traditional kinematic motion synthesis approaches such as motion graphs [1,2] do not alter the movements whatever the next motion is. As a result, awkward behaviors, such as running to an obstacle and suddenly stopping to crawl, may appear. A naive solution would be to capture different levels of preparation movements for all pairs of movements and to select the appropriate transition motion according to the action sequence. However, the size of the motion database increases in the square order with respect to the number of actions.

In this paper, we introduce a scalar parameter called the *level of preparation*, that adjusts the kinematics of the movements and the speed of the motion during the transition. We design a customized motion blending scheme that utilizes the value to generate a smooth preparation movement. To ensure that the synthesized posture is physically plausible, we introduce a postural optimization framework to adjust the center of mass of the character. Given the two actions and the corresponding level of preparation, our method can synthesize a series of movements with the style of preparation added, while following the original motion context.

We further propose a unified controller based on reinforcement learning that controls a character to approach to a target with the proper preparation behavior in real time.

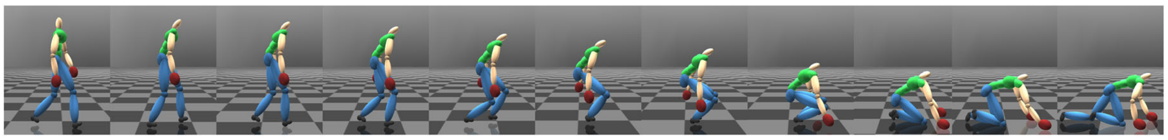


Figure 1. A motion sequence created by using our method. The character prepares for the crawling motion by lowering its pelvis and crouches its torso while walking.

During the training stage, our system learns the optimal motion to approach to the target and, more importantly, the level of preparation that should be applied to create realistic preparation behaviors. At runtime, the trained controller anticipates how much the preparation behavior may affect the movement and computes the time series of the level of preparation. As a result, an optimal transition motion that appears natural and humanlike can be synthesized.

We present experiments for various target actions, such as punching, crawling, kicking, and jumping and show that our system can produce realistic preparation behaviours in real time. Figure 1 shows an example of preparation behaviour synthesized by our system. The proposed algorithm is computationally efficient, making it suitable for applications such as interactive animation creation and computer games.

The rest of the paper is organized as follows. We first review previous work in Section 2. We then explain how we prepare the captured motion in Section 4. Our algorithm involves two major parts. Section 5 explains how to synthesize a preparation movement, and Section 6 explains how to create a controller to control a character with preparation behavior. We present experimental results in Section 7 and conclude the paper in Section 8.

2. RELATED WORKS

Our idea to adapt motions toward the subsequent motion is related to motion blending, motion-style translation, and spacetime motion editing. We first compare our method with such methodologies. Then, we evaluate algorithms involving data-driven approaches, which perform suboptimally in our problem. Finally, we discuss long horizon techniques to intelligently control the characters by predicting the future.

Motion blending is a traditional technique to synthesize new motions by interpolating existing motion data. Motion warping [3] edits a motion by inserting an intermediate posture as an offset. Boulic *et al.* [4] blended several motions simultaneously by changing the weights for different degrees of freedom. Ménardais *et al.* [5] proposed a method to compute the appropriate weights for blending different motions based on intuitive input by the user. Shum *et al*. [6] proposed to blend motions considering the angular momentum profile. In these methods, the change of weights has to be hand tuned during synthesis to create a realistic motion. We need an automatic method to compute the weights in order to control agents intelligently.

Motion style translation extracts movement style based on two motions, which are a plain motion without any style and a style-added motion. Style can be extracted b using Hidden Markov Models [7] or linear time-invariant models [8], then injected to other plain motions in order to synthesize new style-added motions. A major problem is the need of capturing styled motions. As movements with different degrees of preparation appear visually dissimilar, these approaches require capturing a large amount of motions. Moreover, it is difficult to ask humans to precisely conduct preparation movement with small variations on the degree of preparation. On the contrary, our method computes the styled motion and minimizes the amount of labor work.

Spacetime optimization [9] is a method for optimizing a motion when the objectives of the motion conversion are known. It has been used to satisfy kinematic constraints [10], to maintain kinematics similarity in the original motion while retargeting on a different character [11], and to convert manually designed motions to physically plausible motions [12,13]. The major problem is its computational cost: each motion computation requires an optimization of hundreds of parameters, which can hardly be done in real time, especially when synthesizing motions in a dynamic environment.

With **data-driven motion synthesis**, all motions including those preparing for the target motions are captured to compose data structures such as motion graphs [1,2] or finite state machines [14,15]. One can also interpolate/blend motions to synthesize intermediate motions [16–18]. However, such methods are data intensive, making it unsuitable for applications such as 3D computer games. In our approach, we represent the style of the motion by a single posture in the target motion. As a result, we can enhance the reusability of the individual motions and reduce the total amount of data to be precaptured.

Approaches involving **future prediction** predict upcoming situations and select the most appropriate motion. Support vector machine has been used to select the appropriate motion and to avoid external disturbances [19]. Min-max search can control a character to intelligently fight with another character [20,21]. Reinforcement learning produces control systems for characters to fight [22], to avoid obstacles [23], to control pedestrians [24], and to prepare for user controls [25]. In these works, the optimal sequence of captured motion clips are selected and concatenated. Thus, they can only reproduce the captured sequence of motions. In this research, we also apply reinforcement learning to learn the optimal motions for approaching a target while preparing for the upcoming event. The major difference with previous approaches is that we include an additional dimension in the action space, which is

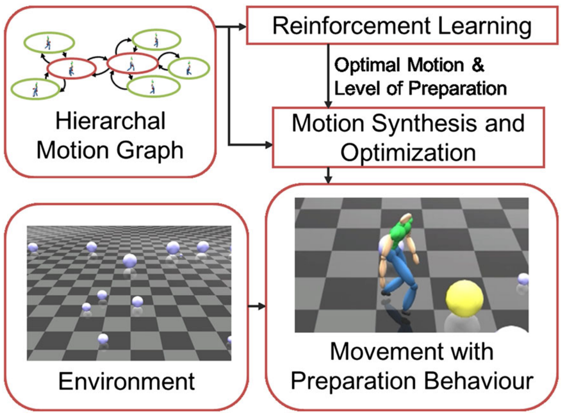


Figure 2. The overview of our method.

called the level of preparation. This allows us to adjust a basic motion for synthesizing a large variety of preparation behaviors.

3. METHODOLOGY OVERVIEW

In this paper, we synthesize the behavior to approach a target object/character while preparing for the subsequent action to interact with it. The outline of our approach is shown in Figure 2. A hierarchical motion graph is used as the motion database. An offline reinforcement learning is used to train a controller that selects the optimal motion and level of preparation for a specific action to perform. On the basis of a given environment and the information from the controller, we synthesize movements with the a correct preparation behavior.

We have two contributions in this paper:

- We propose a customized motion blending and optimization algorithm to synthesize preparation movements from a given level of preparation. The level of preparation is a scalar parameter used to describe the movement style and movement trajectories when preparing for a target action. The method can synthesize preparatory movements without capturing additional motions.
- To control a character to reach a target with the proper preparation behavior, we propose a controller based on reinforcement learning. Unlike traditional reinforcement learning approaches, our action space consists of both motions and the level of preparation, enabling it to synthesize a large variety of movements.

4. MOTION DATABASE

In this section, we explain how we organize the motion database, as well as the extraction of the reference postures in the motions, which contain the preparation characteristics.

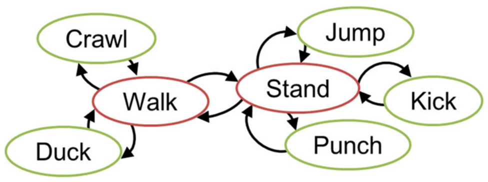


Figure 3. A hierarchical motion graph that includes cyclic locomotion as core nodes and target motions to interact with the environment.

4.1. Hierarchical Motion Graph

We construct a hierarchical motion graph structure [14,15,26] that includes cyclic locomotion as core nodes, such as standing and walking, and target motions to interact with the environment, such as crawling, jumping, and attacking. Edges in the motion graph represent valid transitions between motions with no kinematics artifacts. Figure 3 shows an example of the motion graph. Notice that while the character can conducts a specific target motion from the locomotion, the behavior for switching to the target motion can be unnatural. One example is walking in full speed and then ducking suddenly. The overall movement is kinematically continuous, but the sudden transition between two movements may not appear realistic.

4.2. Reference Posture Extraction

Here, we explain how we extract a reference posture from each target motion, which is used to produce a transition motion.

Let $\mathcal{L}$ be a locomotion, and $\mathcal{T}$ be a target motion. We scan $\mathcal{T}$ to find a posture that represents the style of the preparation behavior semiautomatically. Assuming $\mathcal{T}$ is connected to $\mathcal{L}$ in the motion graph, it can be divided into three major parts: (i) a kinematically smooth transition from the ending posture of $\mathcal{L}$; (ii) a duration that involves that target movement; and (iii) a transition to the starting posture of $\mathcal{L}$ or another locomotion. Figure 4 shows an example with a jumping motion. In our system, the user identifies the first frame of the target movement, $\mathcal{T}_t$. Our system then automatically scans from $\mathcal{T}_t$ toward the first frame of $\mathcal{T}$, which is denoted as $\mathcal{T}_0$, and monitors the following conditions:

1. The center of mass of the body is within the supporting area.
2. The contact foot pattern is the same as that of $\mathcal{T}_0$.
3. The sum of squares of the joint position difference of the legs with respect to $\mathcal{T}_0$ is within a predefined threshold.

Condition 1 ensures us to obtain a stable posture. Conditions 2 and 3 ensure the reference posture does not significantly affect the edge connectivity in the motion graph when synthesizing the preparation movement in Section 5.

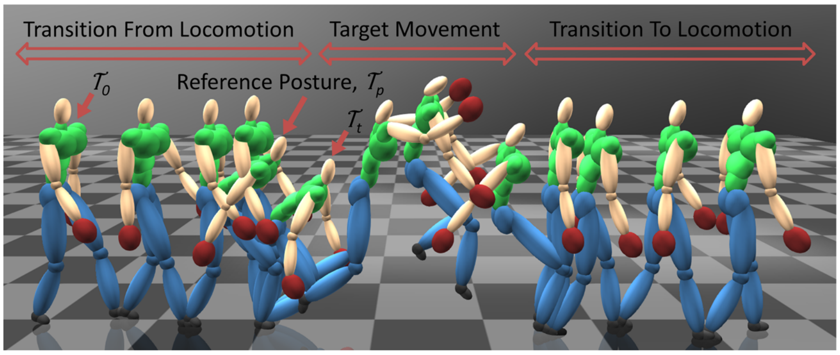


Figure 4. Finding the reference posture in a typical target motion that consists of three parts.

The first frame found that satisfies all the conditions is used as the reference posture of the target motion, $\mathcal{T}_p$. If we cannot find such a frame, $\mathcal{T}_0$ is used as the reference posture.

5. MOVEMENT SYNTHESIS

In this section, we explain how we synthesize preparation movements, which are movements with added styles such that the following target action can be launched smoothly. We observe that preparation movements are spatially offsetted and temporally scaled. To facilitate efficient style control, we introduce a single scalar parameter $\alpha \in \mathbb{R}$ called the level of preparation to control the aforementioned two characteristics of the preparation movement.

5.1. Movement Style Synthesis

Here, we explain our customized motion blending algorithm to create the preparation movement style.

Given two movements, previous techniques of parametric motion blending use a linear blending weight to interpolate the movement of all the degrees of freedom. However, we observe that when humans shift their motions from one to another, the joint movements are not linearly warped altogether. For example, when we prepare for actions such as punching or crawling while walking, we usually adapt the arms and torso quickly in the beginning stage of the preparation movement, whereas the joints of the lower body are adapted gradually. This is because of the dominant role of the arms in the target motion, whereas the gait movement does not affect the target motion heavily. Such an observation leads us to separately blend the joint movements during motion blending.

The blending weight of joint i is calculated as follows:

$$w_i = \sqrt[C_i]{1-(1-\alpha)^{C_i}} \tag{1}$$

where α is the level of preparation given by a high-level controller that will be explained in Section 6, and C_i is a constant individually set for each joint. The blended joint angle of joint i is calculated as follows:

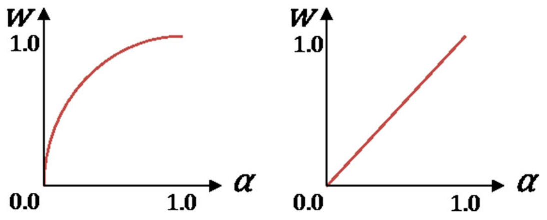


Figure 5. The blending curves for (left) the upper body with C_i set to 0.5 and (right) the lower body with C_i set to 1.0.

$$\boldsymbol{q}_i = (1-w_i)\boldsymbol{q}_i^{\mathcal{L}} + w_i \boldsymbol{q}_i^{\mathcal{T}_p} \tag{2}$$

where $\boldsymbol{q}_i^{\mathcal{L}}$ and $\boldsymbol{q}_i^{\mathcal{T}_p}$ represent the Euler joint angles in the locomotion $\mathcal{L}$ and reference posture $\mathcal{T}_p$, respectively. When $\alpha = 0$, the character is not prepared and the blending weights are zero: the posture is the same as those in $\mathcal{L}$. When $\alpha = 1$, the character is fully prepared and the weights become one: the resultant posture becomes the reference posture $\mathcal{T}_p$.

The constant C_i in Equation 1 represents the blending behavior of the joint, because it affects the curvature of the blending curve as shown in Figure 5. Although it is possible to design a suitable C_i for each joint, we observe that for our motion database, the human joints can be roughly classified into the upper body and the lower body. In our system, we suggest the user to set $C_i = 1.0$ for the lower body joints including the root and the legs (Figure 5 left), and $C_i = 0.5$ for the upper body joints (Figure 5 right). For other combinations of locomotion and target actions, the user can further edit C_i for specific joints to obtain satisfactory transitions.

An example of using a traditional blending method [3] to prepare for a boxing motion is shown in Figure 6 (top). Notice that the arms are gradually raised while increasing the level of preparation because a single blending parameter is used. The third and fourth postures appear unnatural, as the character raises its arm horizontally while walking.

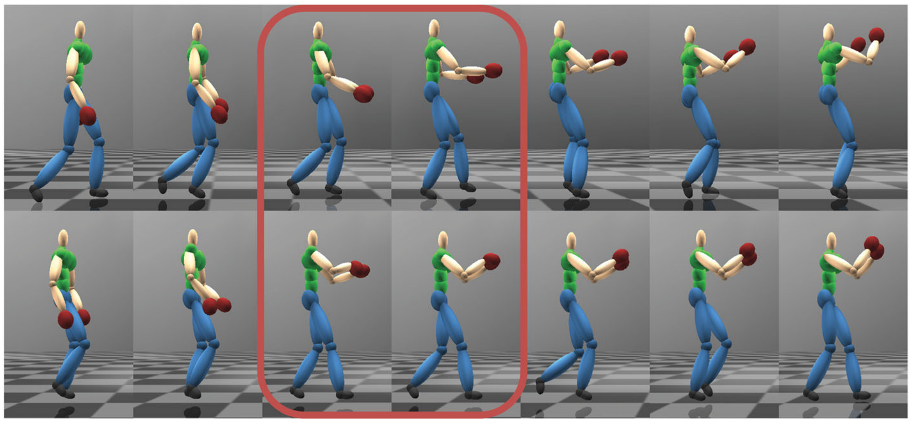


Figure 6. A walking motion with the preparation behavior to boxing created by (top) traditional blending methods and (bottom) our method.

On the other hand, with our approach, the arms quickly switch to the boxing style when the level of preparation increases, whereas the lower body is blended with a longer transition period (Figure 6 bottom).

5.2. Movement Trajectory Adjustment

Here, we explain how we adjust the position of the root in the blended motion, which affects the movement trajectory of the character. Notice that the root orientation is scaled in the same way as other joints with Equation 2.

The displacement of the root in the horizontal plane (x–z plane) depends on the lower body movement. Although the best solution is to calculate the horizontal displacement based on the stride of the legs, in the interest of computational costs, we approximate it by the blending weight of the root:

$$\boldsymbol{p_r}.x = (1 - w_r)\boldsymbol{p_r^{\mathcal{L}}}.x \tag{3}$$

$$\boldsymbol{p_r}.z = (1 - w_r)\boldsymbol{p_r^{\mathcal{L}}}.z \tag{4}$$

where w_r is the blending weight of the root calculated by Equation 1, $\boldsymbol{p_r^{\mathcal{L}}}.x$ and $\boldsymbol{p_r^{\mathcal{L}}}.z$ represent the horizontal root displacement of the locomotion $\mathcal{L}$.

The root displacement in the vertical direction (y direction) is calculated on the basis of the difference of the height of the feet before and after the blending. The average difference of the left and right feet height is computed and subtracted from the root height:

$$\boldsymbol{p_r}.y = \boldsymbol{p_r^{\mathcal{L}}}.y - \frac{1}{2}(\Delta y_l^{\mathcal{L}} + \Delta y_r^{\mathcal{L}}) \tag{5}$$

where $\boldsymbol{p_r^{\mathcal{L}}}.y$ is the root height in $\mathcal{L}$, and $\Delta y_l^{\mathcal{L}}$, $\Delta y_r^{\mathcal{L}}$ are the height of the left and right feet relative to the root caused by blending the joint angles.

Finally, we use inverse kinematics to adjust the location of the feet. We detect the foot contact information from the original locomotion data, which is kept the same in the blended motion. When a foot is supposed to support the body, its location is kept the same on the ground throughout the supporting phase. In our implementation, we apply inverse kinematics based on a particle system to solve for the posture [27].

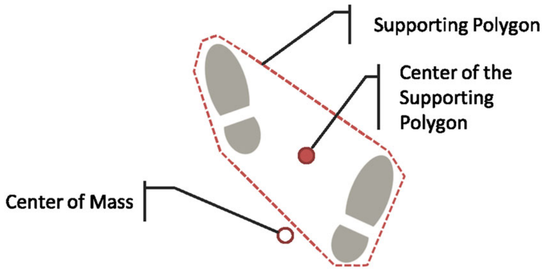


Figure 7. An example of the COM and the center of the supporting polygon during a double supporting stage.

5.3. Balancing the Character

Here, we explain about our optimization scheme that adjusts the posture of the character such that the center of mass stays within the supporting polygon of the body. We use a cost function that is composed of three different terms to evaluate each posture, which are described in the succeeding texts.

First, the **balance term** evaluates how dynamically stable the posture is. It is based on the distance between the center of mass (COM) and the center of the supporting polygon, as illustrated in Figure 7:

$$C_b = ||\boldsymbol{p_{com}} - \boldsymbol{p_{cos}}|| \tag{6}$$

where $\boldsymbol{p_{com}}$ is the 2D floor projection of the COM, and $\boldsymbol{p_{cos}}$ is the 2D COS on the floor. The COM is calculated as the weighted average of body part centroids, and the weights are set according to the estimated mass as described in [28].

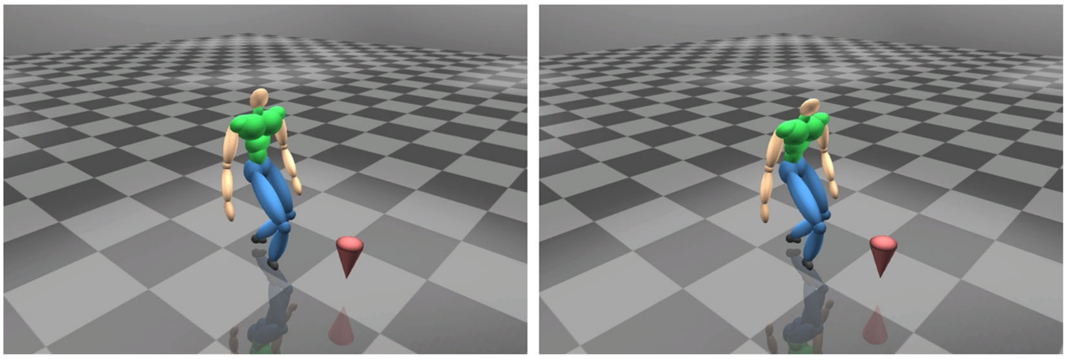


Figure 8. The synthesized preparation movement (left) without and (right) with posture optimization.

The **postural style term** is used to minimize the positional difference between the posture being optimized and the posture computed by blending:

$$C_s = \sum_{i=0}^{i_{total}} \frac{\left\|\boldsymbol{p}_i^{op}(t) - \boldsymbol{p}_i(t)\right\|}{i_{total}} \tag{7}$$

where i_{total} is the total number of body parts in a character, and $\boldsymbol{p}_i^{op}(t)$ and $\boldsymbol{p}_i(t)$ are the 3D center of body part i under the current time step t in the optimized posture and the original posture, respectively.

Finally, the **smoothness term** maintains the smoothness of the movement in the time domain. It considers the change of displacement vector for each body part across the previous two frames:

$$C_m = \sum_{i=0}^{i_{total}} \frac{\left\|\left(\boldsymbol{p}_i^{op}(t) - \boldsymbol{p}_i^{op}(t-1)\right) - \left(\boldsymbol{p}_i^{op}(t-1) - \boldsymbol{p}_i^{op}(t-2)\right)\right\|}{i_{total}} \tag{8}$$

$$= \sum_{i=0}^{i_{total}} \frac{\left\|\boldsymbol{p}_i^{op}(t) - 2\boldsymbol{p}_i^{op}(t-1) + \boldsymbol{p}_i^{op}(t-2)\right\|}{i_{total}} \tag{9}$$

where $\boldsymbol{p}_i^{op}(t)$, $\boldsymbol{p}_i^{op}(t-1)$, and $\boldsymbol{p}_i^{op}(t-2)$ are the 3D center of body part i in the optimized posture of the current frame, last frame, and the second to last frame, respectively.

The overall cost of an optimized posture is evaluated as the weighted sum of the three terms:

$$C = w_b C_b + w_s C_s + w_m C_m \tag{10}$$

where w_b, w_s, and w_m are weights set as 2.0, 1.0, and 0.25, respectively. A general principle to tune the weight is to use the smallest possible w_m that can create smooth movement, as the term may constraint the optimization process. Besides, w_d should be larger than w_s because the major objective of the optimization is to adjust the COM.

We apply gradient descent [29] in the Euler joint angle space to optimize for a posture that minimizes Equation 10 at runtime. The computation is terminated either if a local optimal solution is found or if the number of steps reach a predefined threshold, which is set to 10 in our system. By adjusting the maximum number of steps, we can adjust the trade-off between the postural quality and the runtime computational cost. We do not apply postural optimization if α is smaller than a threshold, which is set to 0.3 in our system, because it is likely that the blended movement already looks natural enough.

Because optimizing the lower body would easily lead to unnatural behavior such as foot sliding, we only adjust the joints of the upper body. Although this strategy limits the degrees of freedom that we can manipulate, we find that in most situations, adjusting the upper body significantly improves the naturalness of the motion.

Figure 8 demonstrates the effect of the postural adjustment scheme: the COM of the blended posture is slightly offsetted from the supporting polygon, whereas the upper body including the spine and the neck leans forward in the processed posture, as can be seen in real human motions to ensure balance.

6. BEHAVIOR CONTROLLER

In this section, we explain how we apply reinforcement learning to train a unified controller, which controls the motion that the character should perform to move toward a target, and the level of preparation that should be applied.

We define the term preparation behavior as the high-level behavior to approach a target and to prepare to interact. The behavior involves multiple motions with a change of level of preparation throughout the series of movement.

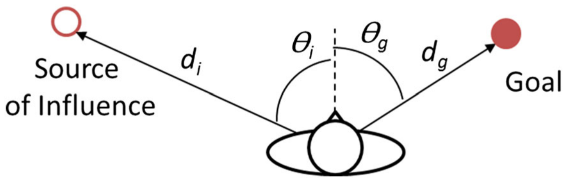


Figure 9. The separated goal and source of influence in our state representation.

6.1. State Space and Action Space

Here, we first explain the *states* and *actions* designed in our system.

We assume the character has a goal to reach, and there is a source of influence that affects the preparation behavior. The goal of movement and source of influence are usually identical, such as approaching a ball and preparing to kick it. The advantage of separating them into two variables is that it allows more general definition and thus a wider range of synthesis. For example, we can simulate a character walking in one direction, while preparing to avoid a nearby obstacle that blocks the way.

We create a state space $\mathbb{S}$ in which every state is defined as follows:

$$s = \{\alpha, Next(m), d_g, \theta_g, d_i, \theta_i\}, s \in \mathbb{S} \tag{11}$$

where α is the level of preparation in the current time instance, m is the last action performed, $Next(m)$ is the set of available actions to be performed after m according to the motion graph, d_g and θ_g are the distance and relative orientation between the character and the goal, respectively, and d_i and θ_i are those with respect to the source of influence (Figure 9). Except from $Next(m)$, the parameters in the state are continuous numerical values.

We create an action space $\mathbb{A}$, in which each action is defined as follows:

$$a = \{m, \alpha\}, a \in \mathbb{A} \tag{12}$$

where m is the action to be performed, and α is the corresponding level of preparation. Notice that unlike traditional reinforcement learning [30], the action in our system is not simply a movement. Instead, it involves the level of preparation, α, which is quantized during training and is used to adjust the movement. Because using discrete values of α in simulations could result in discrete behaviors, we apply a Gaussian filter on α to smooth the value over time.

6.2. Reward Function

Here, we explain about the *reward function*, which evaluates the actions in a given state.

Let s_t be the current state and s_{t+1} be the next state after performing the chosen action a_{t+1}. α in the action is a feedback to the state whenever an action is selected:

$$s_{t+1}.\alpha \leftarrow a_{t+1}.\alpha \tag{13}$$

The reward function that evaluates a_{t+1} consists of three terms. The first term evaluates how much the character faces the target. It is defined as the absolute orientation between the target and the goal:

$$f_\theta = 1 - \frac{|s_{t+1}.\theta_g|}{\theta_{norm}} \tag{14}$$

where $s_{t+1}.\theta_g$ denotes the angle with respect to the goal for the state s_{t+1}, and θ_{norm} is a constant to normalize angle values and is set as 180°. The second term evaluates how much the character approaches the goal. It is defined as the difference in distance toward the goal:

$$f_d = \frac{d_{norm} + s_t.d_g - s_{t+1}.d_g}{2 \times d_{norm}} \tag{15}$$

where $s_{t+1}.d_g$ and $s_t.d_g$ denote the distance to the goal for the two states, and d_{norm} is a constant to normalize distance values and is set as 1 m. The last term evaluates how steadily the level of preparation changes. It is defined as the absolute value of the difference in α:

$$f_\alpha = 1 - |s_{t+1}.\alpha - s_t.\alpha|^{C_\alpha}. \tag{16}$$

where C_α is a constant, and $s_{t+1}.\alpha$ and $s_t.\alpha$ denote the level of preparation of the two states. This term is used to penalize sudden changes of preparation level. C_α is a power operator and is used to magnify the difference between small and large changes, which is set to 2.0 in our system. During training, when a character reaches the source of influence, α is forced to be 1.0. Therefore, the trained system increases α gradually before reaching the source of influence to minimize this term.

Finally, the reward function is defined as follows:

$$r_t = w_\theta f_\theta + w_d f_d + w_\alpha f_\alpha \tag{17}$$

where w_θ, w_d, and w_α are constant weights, which are empirically set to 0.1, 0.3, and 0.6 respectively. The expected range of r_t is [0.0, 1.0].

6.3. State-Action-Reward-State-Action Learning

Here, we explain the concept of *return* and how we use the SARSA (State-Action-Reward-State-Action) approach to solve for it.

The return represents the long-term benefit of launching a series of actions based on a *control policy*, which tells the action to be performed in a given state:

$$R = \sum_{t=0}^{n} \gamma^t r_t \tag{18}$$

where γ is called a *discount value* that reduces the influence of future states because of their uncertainties, and n is the number of steps until the character reaches the terminal state.

To find an *optimal policy* that maximizes R in Equation 18 at every state, we apply the SARSA learning which is a standard scheme in reinforcement learning. Here, we briefly explain the SARSA framework. The reader is referred to [31,32] for further details.

Using SARSA, the system learns the *state-action value* $Q(s,a)$, which is the estimated return of launching an action a at state s, for all possible state-action combinations. A control policy is defined as the optimal choice of action based on the set of the state action values for all states. The objective is to learn the optimal policy by converging $Q(s,a)$ to the maximum R in Equation 18.

The training is divided into *episodes*. At each episode, we randomly pick a state $s \in \mathbb{S}$, allow the character to perform an action $a \in \mathbb{A}$ based on the current policy, and observe the reward r. The character keeps launching actions until the terminal state, for which we define as reaching the goal position. SARSA represents Equation 18 as follows:

$$\begin{aligned} Q(s_t,a_t) \leftarrow Q(s_t,a_t)+ \\ \lambda(r_{t+1}+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)) \end{aligned} \tag{19}$$

where λ is the *rate of learning*, which is set to 0.7 in our experiments, and γ refers to the discount value as shown in Equation 18, which is set to 0.3.

The control policy is updated whenever a state action value is changed. Training is conducted by using a large number of episodes until the policy becomes optimal (i.e. the policy cannot be improved further). The algorithm converges provided that all state-action pairs are visited infinite number of times. In practice, to reduce the convergence time, we keep track of the number of state action values updated per episode. If the number drops below 2% of total number of states, we terminate the training and assume the policy to be optimal.

6.4. Maintaining Exploration

It is important to maintain exploration during the training process in order to obtain a globally optimal policy. We apply two strategies for this:

- **Exploring starts:** We attach a flag for each action in all the states to indicate whether if the actions have been selected or not. During training, higher priority is given to the actions that have not been selected. This ensures uniform exploration in the state action space during the early stage of training.
- **ϵ-greedy policy:** When selecting an action, there is an ϵ chance that the system randomly selects an action, instead of picking the one with the best state-action value. This ensures exploration throughout the training. We set ϵ to 0.1 in our system.

7. EXPERIMENTS

In this section, we explain the experiments we conducted to evaluate our system.

7.1. System Setup

All experiments are performed BY using a computer with an Intel Core i7-2630QM CPU, 8GB of RAM, and a GeForce GTX 560M graphic card.

We set up a motion database with 78 motions, including walking, crawling, ducking, jumping, punching, and kicking. The cyclic walking motions including stepping to different directions are used as the core nodes in the hierarchical motion graph, whereas the rest are used as target motions. The created graph contains four core nodes.

We quantize the state and action spaces for the SARSA learning. The sampling steps and number of samples for each variable are shown in Table I, which resulted in 19 200 states. The training stage took around 2 h (around 20 million training steps), whereas the runtime system performed in real-time. Figure 10 shows the average return per state during the training process.

7.2. Preparation Control

An interactive 3D application prototype is presented to show the effectiveness of our system. Notice that our preparation control system does not involve launching the target motions. Although it is possible to define procedural rules that control the character launching such motions, we prefer an interactive system where the user can indicate the timing of launching the motions for better flexibility.

In the first experiment, we prepare an interface for the user to select a ball in a given environment, which is used as both the goal location and the source of influence. The system controls the character to approach the ball with the appropriate preparation behavior. The user then indicates when a kicking motion is launched to interact with the ball. Notice that when the character approaches the ball, it lowers its COM and slows down to prepare for the kicking motion (Figure 11 top).

The second experiment involves interacting with non-player characters that are controlled by a procedural controller to move toward and to attack the user-controlled character. The user uses the keyboard to indicate the goal, while using the mouse to launch target motions. We synthesized preparation behavior for all characters in the scene, and the closest opponent is considered as the source of

Table I. The sampling steps and number of samples in the state space.

	d_i	θ_i	d_g	θ_g	α
Steps	40 cm	90°	40 cm	45°	0.2
Number	5	4	5	8	5

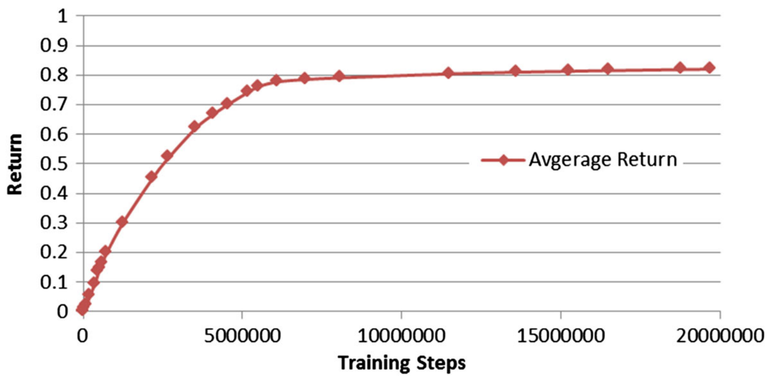


Figure 10. Average return per state against the number of training steps.

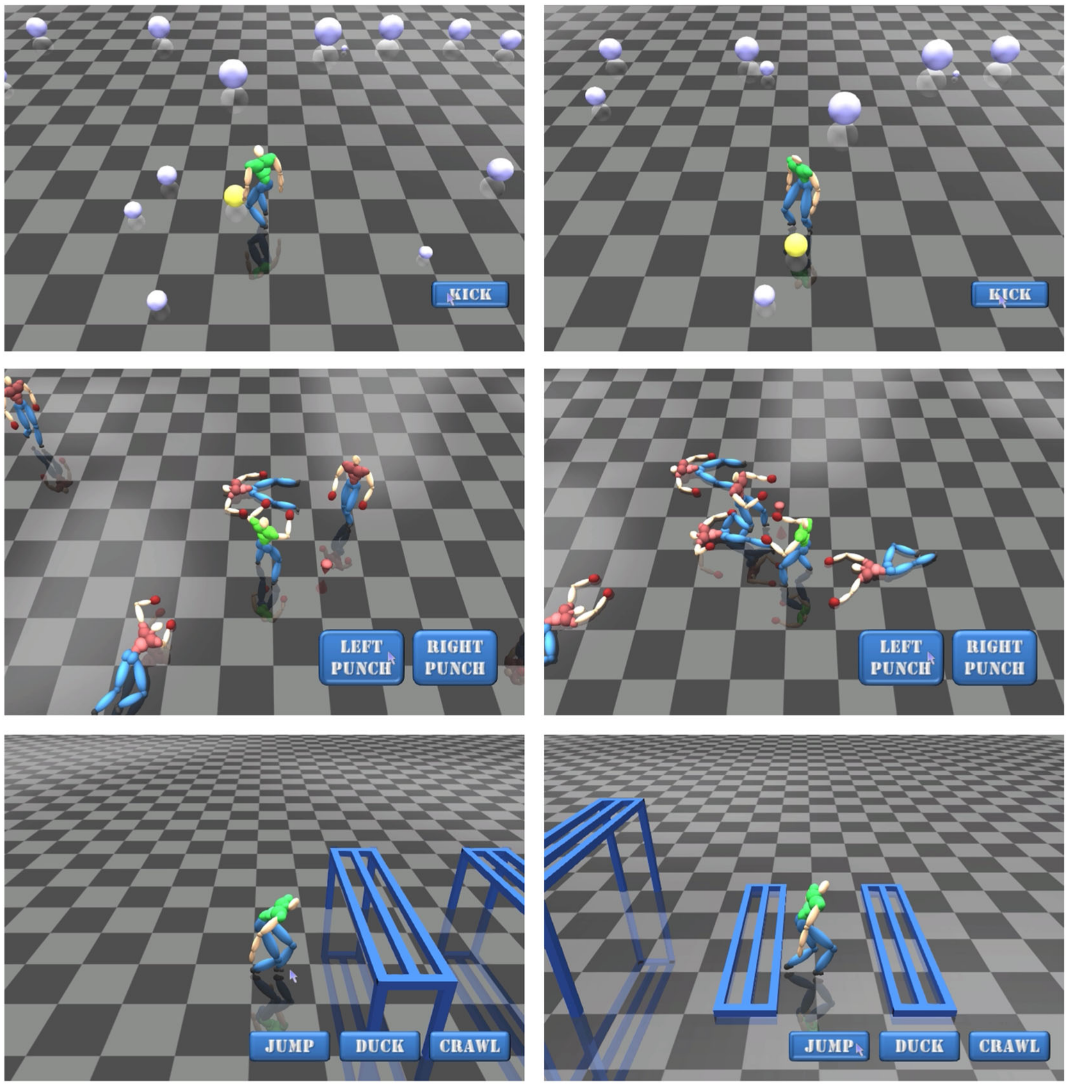


Figure 11. Preparation behaviors created by our system for different target motions.

influence. As a result, they raise their arms, slow down, and prepare the punch whenever an opponent is nearby. Notice that the left punch and the right punch motions have the same reference posture $\mathcal{T}_p$ and thus having the same preparation styles (Figure 11 middle).

In the third experiment, the character has to navigate through a complex environment with different obstacles. Again, the user controls the goal and launches the target motions. The source of influence is set as the closest obstacle in the facing direction, and the preparation style is based on the corresponding target motion to interact with the obstacle. The system creates different levels of preparation behaviors to perform crawling, ducking, and jumping. Notice that with our system, the character maintains a preparation style when getting through two nearby obstacles, generating a humanlike behavior (Figure 11 bottom).

8. CONCLUSION AND DISCUSSIONS

In this paper, we proposed a method to synthesize preparation behavior for the next motion to be launched. By using our method, we can generate realistic behaviors to prepare for the next action as humans do. Our system produces convincing results in real-time, making it suitable for real-time applications such as games.

We observe that in console games nowadays, because of the lack of preparation movement, character movement is usually unrealistic. Because the character does not consider the next motion, it is common that a character performs a target motion, runs in full speed, and performs a target motion again. In our system, the character maintains the preparation style if there are remaining targets nearby.

A possible solution for preparation behavior synthesis is the data-driven approach [16,17]. For every pair of locomotion and subsequent action, one can capture different levels of preparatory motions. However, this approach is not preferred because of the large amount of motion that has to be captured. Also, it is difficult for a human to exhibit preparatory behavior properly with small variations of level of preparation, because the movements are usually performed subconsciously in daily life.

The major challenge of creating preparation behavior is that it affects the movement of the locomotion. The character needs to approach the target quickly, while adjusting movement trajectories to prepare for the target motion in advance. We defined the reward function and return and conducted reinforcement learning to solve the multimodal problem as a whole.

A feature of our design is that the state space representation in Equation 11 contains the locomotion only without the target motion. This design allows us to reuse a trained system for different target motions, as long as the blending curve of the lower body remains unchanged. If the blending curves for two target motions are different, the lower body, hence the movement trajectory, in the synthesized motion will be different. As a result, a controller has to be trained for each motion.

At the current stage, our system cannot synthesize the preparation behavior when there are multiple potential target motions, unless they have the same reference posture and blending curve. As a future direction, we plan to control the character intelligently in a dynamic environment where arbitrary events can happen in a probabilistic manner. We could apply Bayesian models to let the characters predict what kind of behaviors could benefit them the most under different environments.

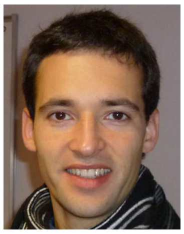
REFERENCES

1. Lee J, Chai J, Reitsma PSA, Hodgins JK, Pollard NS. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics* 2002; **21**(3): 491–500.
2. Kovar L, Gleicher M, Pighin FH. Motion graphs. *ACM Trans. Graph.* 2002; **21**(3): 473–482.
3. Witkin A, Popovic Z. Motion warping. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95. ACM, New York, NY, USA, 1995; 105–108.
4. Boulic R, Bécheiraz P, Emering L, Thalmann D. Integration of motion control techniques for virtual human and avatar real-time animation. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '97. ACM, New York, NY, USA, 1997; 111–118.
5. Ménardais S, Multon F, Kulpa R, Arnaldi B. Motion blending for real-time animation while accounting for the environment, In *Computer Graphics International*, Crete, Greece, 2004June; 156–159.
6. Shum HPH, Komura T, Yadav P. Angular momentum guided motion concatenation. *Computer Animation and Virtual Worlds* 2009; **20**(2-3): 385–394.
7. Brand M, Hertzmann A. Style machines. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000; 183–192.
8. Hsu E, Pulli K, Popović J. Style translation for human motion. *ACM Trans. Graph.* 2005; **24**(3): 1082–1089.
9. Witkin A, Kass M. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, New York, NY, USA, 1988; 159–168.
10. Gleicher M, Litwinowicz P. Constraint-based motion adaptation. *The Journal of Visualization and Computer Animation* 1998; **9**(2): 65–94.
11. Gleicher M. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on*

Computer Graphics and Interactive Techniques, SIGGRAPH '98. ACM, New York, NY, USA, 1998; 33–42.
12. Liu CK, Popović Z. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, New York, NY, USA, 2002; 408–416.
13. Fang AC, Pollard NS. Efficient synthesis of physically valid human motion. *ACM Trans. Graph.* 2003; **22**(3): 417–426.
14. Lau M, Kuffner JJ. Behavior planning for character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New York, NY, USA, 2005; 271–280.
15. Kwon T, Shin SY. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New York, NY, USA, 2005; 29–38.
16. Kovar L, Gleicher M. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.* 2004; **23**(3): 559–568.
17. Mukai T, Kuriyama S. Geostatistical motion interpolation. *ACM Trans. Graph.* 2005; **24**(3): 1062–1070.
18. Safonova A, Hodgins JK. Construction and optimal search of interpolated motion graphs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. ACM, New York, NY, USA, 2007; 1–11. Article No. 106.
19. Zordan V, Macchietto A, Medin J, Soriano M, Wu C-C, Metoyer R, Rose R. Anticipation from example. In *VRST '07: Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*. ACM, New York, NY, USA, 2007; 81–84.
20. Shum HPH, Komura T, Yamazaki S. Simulating interactions of avatars in high dimensional state space. In *I3d '08: Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*. ACM, New York, NY, USA, 2008; 131–138.
21. Shum HPH, Komura T, Yamazaki S. Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics* 2012; **18**(5): 741–752.
22. Julian Gold Thore Graepel, Herbrich R. Learning to fight, In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004. (Available from: http://research.microsoft.com/pubs/65639/graehergol04.pdf).
23. Ikemoto L, Arikan O, Forsyth D. Learning to move autonomously in a hostile world. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*. ACM, New York, NY, USA, 2005; Article No. 46.
24. Treuille A, Lee Y, Popović Z. Near-optimal character animation with continuous control. *ACM Trans. Graph.* 2007; **26**(3): 1–7. Article No. 7.
25. McCann J, Pollard N. Responsive characters from motion fragments. *ACM Trans. Graph.* 2007; **26**(3): 1–7. Article No. 6.
26. Shin HJ, Oh HS. Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006; 291–298.
27. Jakobsen T. Advanced character physics, In *Game Developers Conference Proceedings*, 2001. (Available from: http://www.gamasutra.com/resource_guide/20030121/jacobson_pfv.htm).
28. Armstrong HG. Anthropometry and mass distribution for human analogues. volume 1. military male aviators, 1988.
29. Yin K, Coros S, Beaudoin P, van de Panne M. Continuation methods for adapting simulated skills. *ACM Trans. Graph.* 2008; **27**(3).
30. Lee J, Lee KH. Precomputing avatar behavior from human motion data. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004; 79–87.
31. Rummery GA, Niranjan M. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.
32. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

AUTHORS' BIOGRAPHIES

Hubert P. H. Shum is a Senior Lecturer (Assistant Professor) at the Northumbria University, Newcastle upon Tyne, UK. Before joining the university, he worked as a lecturer at the University of Worcester, Worcester, UK, and a post-doctoral researcher at RIKEN, Wako, Japan. He received his PhD degree from the University of Edinburgh, Edinburgh, UK, as well as his Master and Bachelor degrees from the City University of Hong Kong, Hong Kong, China. His research interests include character animation, machine learning, physical simulations, and human computer interaction.



Ludovic Hoyet is a Research Fellow in the Trinity College Dublin (Ireland) since 2011. He received his PhD in 2010 in Computer Sciences from the French Institut National des Sciences Appliquées (INSA). His research interests include biological human motion analysis, synthesis, and perceptual evaluations, in particular in the case of physically based animations.

Edmond S. L. Ho received the PhD degree from the University of Edinburgh, Edinburgh, Scotland in 2011.He is currently a Research Assistant Professor at the Department of Computer Science, Hong Kong Baptist University. His current research interests include character animation, robotics, and human activity understanding.

Taku Komura is a Reader (Associate Professor) at the Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh. As the head of the Computer Animation and Robotics group, his research has focused on data-driven character animation, physically-based character animation, crowd simulation, cloth animation, and anatomy-based modelling and robotics. Recently, his main research interests have been in indexing and animating complex close interactions, which include character-character interactions and character-object interactions.

Franck Multon is a Professor in University Rennes2 in France. He is performing his research in biomechanics (M2S Lab) and in character simulation (MimeTIC/Inria group). His research interests are biomechanics and character simulation, as well as interaction between real and virtual humans. He defended his PhD in 1998 in INRIA Rennes on motion control of virtual humans. Since 1999, he was Assistant Professor in University Rennes2, has defended his “authorization to supervise research” in 2006, and has been hired as full Professor in 2008.