

# Motion blending for real-time animation while accounting for the environment

Stéphane Ménardais  
SIAMES-IRISA,  
France  
smenarda@irisa.fr

Franck Multon  
Université Rennes2,  
France  
multon\_f@uhb.fr

Richard Kulpa  
Université Rennes2,  
France  
kulpa\_r@uhb.fr

Bruno Arnaldi  
SIAMES-IRISA,  
France  
arnaldi@irisa.fr

## Abstract

*Using motion capture systems to animate human-like figures still remains difficult when the movements are complex or need to be adapted to geometric constraints. We propose a new method to blend several captured movements while adapting the trajectories to new skeletons and to unknown environments. For each body part (considered as resources), a priority is defined for each movement (considered as consumers). The trajectories applied to the skeleton consist of a weighted sum of the motions trajectories. A new technique to compute the weights is proposed. Finally, the system adapts the resulting trajectories to the synthetic skeleton and to the environment. The results enabled to animate up to one hundred actors in interactive environments.*

## 1. Introduction

Motion capture systems have been widely used to create realistic synthetic motions. To increase the capabilities of human-like figures, captured motions can be merged and blended by ensuring time-correspondence between the original sequences [5,6,7]. To this end, a cost metric for selecting transitions between motion segments has been introduced [8,13]. Motion graphs [1,6,8] have been used to represent transitions between poses of the captured motion data. A pose of a motion graph is linked to another pose of a different motion graph if the latter pose can be derived from the previous one. This technique was especially studied to simulate human locomotion according to planned foot-prints [4], on uneven terrains [10] and accounting for rapid change of directions [12]. In general, these works specifically address the problem of synchronizing several motions together in order to blend them. However, in all these works, the problem is to search the sequence that solves a set of preliminary imposed constraints. In real-time animation, external events or

user commands may change these constraints at any time. Consequently, with these methods, each modification has to be linked to a new search process that can not be compatible with real-time animation.

Other works propose to add dynamics to improve the realism of the resulting blended motions [2,11]. Nevertheless, these techniques also require optimizing the motion on all the sequence and can hardly be extended to interactive animation. Moreover, in these approaches, computational cost is directly linked to the number of recorded motions.

An alternative is to use elementary motions that can be blended by tuning weights in real-time [3]. The resulting motion is a weighted sum of all the activated motions. In addition, each action has a different priority depending on the considered body parts.

In this paper, we especially focus on this blending problem. We first present a new method to describe postures that enables us to easily adapt the data to a new skeleton and to the environment. Then, we propose a more intuitive and controllable algorithm to evaluate the blending weights.

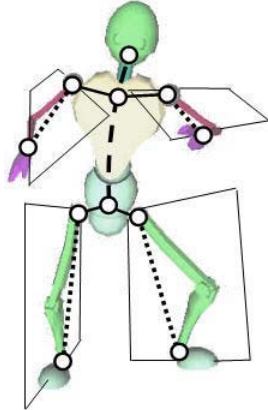
## 2. Elementary actions

In order to adapt the motion to the environment in real-time, the skeleton is subdivided into several sub-chains (see figure 1).

The limbs are represented separately by storing the position of the shoulder (resp. hip) relatively to the root, the wrist (resp. ankle) relatively to the shoulder (resp. hip). These data are normalized by the distance between the two extremities. The orientation of the plane passing through shoulder, elbow and wrist (resp. hip, knee and ankle) is also stored.

The trunk is given by the spline parameters that go through the control points placed at the pelvis, the neck and the head. This representation allows one to adapt the motion to any kind of trunk subdivision.

All the above parameters denoted  $Param_i(t)$  (where  $i$  is the number of the parameter) defining a posture at time  $t$  are gathered into a structure called  $Param(t)=\{Param_i(t)\}$ . A set of constraints (such as ensuring foot contact) is also defined in  $Param(t)$ . At each time step any constraint can be activated or not. This choice ensures foot contact only during contact phases. Resolving these constraints is performed geometrically by the real-time animation engine at each time step [9].



**Figure 1: description of the normalized skeleton used for motion adaptation.**

A module provides the system with data required for motion blending for each elementary action. To this end, the elementary action  $A^j$  (where  $j$  states for the action number) is associated to a priority value  $P_i^j(t)$  for each body part or constraint  $i$ . For example, high priority values for the legs and low values for the upper-body are defined for walking. This priority changes continuously depending on time according to the user's command during the real-time animation.

In the following, the number of the action (denoted by the symbol  $j$ ) is placed as an exponent and the number of the parameter (denoted by the symbol  $i$ ) is placed as an index. The state  $S^j(t)$  of an action  $A^j$  depends directly on the events that occur in real-time animation. Its range goes from 0 if it is stopped to 1 if it is fully active.  $S^j(t)$  is the same for all the parameters  $i$  since an action is activated/deactivated for all the degrees of freedom in the same way.

### 3. Computation of weights

Each action  $A^j$  provides a description  $Param^j=\{Param_i^j(t)\}$  for each time  $t$  and parameter  $i$ . Finally, the posture  $Param=\{Param_i\}$  that is used to animate the skeleton is:

$$(1) \forall i, Param_i(t) = \sum_j k_i^j Param_i^j(t)$$

where  $k_i^j$  is the weight associated to the  $j^{th}$  action for the  $i^{th}$  parameter. The sum of all the  $k_i^j$  for each action  $i$  is equal to 1. The relationship between the states, priorities and  $k_i^j$  must follow other logical rules:

1. an inactive action ( $S^j=0$ ) generates a null weight value,
2. the variation of each weight value depending on time is continue ( $C^1$  continuity),
3. if two actions  $A^1$  and  $A^2$  are running ( $S^1=S^2 \neq 0$ ) and the priority on the  $i^{th}$  parameter for  $A^1$ ,  $P_i^1$ , is greater than the priority of  $A^2$ , then the weight of  $A^1$  is also greater than the one of  $A^2$ . Thus

$$(4) \left. \begin{array}{l} S^1 = S^2 \\ \forall i, P_i^1 > P_i^2 \end{array} \right\} \Rightarrow \forall i, k_i^1 > k_i^2$$

4. if two actions  $A^1$  and  $A^2$  have the same priority on the  $i^{th}$  parameter and the state for  $A^1$  is greater than the state of  $A^2$ , then the weight of  $A^1$  is also greater than the one of  $A^2$ . That is

$$(5) \left. \begin{array}{l} S^1 > S^2 \\ \forall i, P_i^1 = P_i^2 \end{array} \right\} \Rightarrow \forall i, k_i^1 > k_i^2$$

5. if two actions  $A^1$  and  $A^2$  have the same priority on the  $i^{th}$  parameter and the same state, then the weight of  $A^1$  and  $A^2$  are equal.

$$(6) \left. \begin{array}{l} S^1 = S^2 \\ \forall i, P_i^1 = P_i^2 \end{array} \right\} \Rightarrow \forall i, k_i^1 = k_i^2$$

In order to verify all the above priorities, we defined the relation below:

$$(7) \forall i, \forall j, k_i^j = \frac{S^j \times (P_i^j + \varepsilon)}{\sum_c S^c \times (P_i^c + \varepsilon)}$$

where  $\varepsilon$  is a positive real number. We also designed a neutral action  $A^0$  which state  $S^0$  is always equal to 1 and which priority for each parameter is always null.  $\varepsilon$  is used to ensure that the denominator is not null only if  $A^0$  is active.

Let us consider now the case of two co-occurring actions: locomotion and a grasping task. The grasping action is programmed such that the arm has to reach a goal at a given position; but the arm is also influenced by the locomotion. To overcome this limitation, we have introduced a constant threshold value  $\Delta P$ : if  $P^2 > P^1 + \Delta P$  then action  $A^1$  is supposed negligible and is not considered for blending. Let  $\Phi(P^1, P^2)$  be a function that returns a real value between 0 and 1 that states for the relative importance of  $P^1$  compared to  $P^2$ . If  $P^1 \geq P^2$  ( $P^1$  is more important than  $P^2$ ) then  $\Phi(P^1, P^2) = 1$ . On the

other hand, if  $P^1 < P^2 - \Delta P$  ( $P^1$  is negligible compared to  $P^2$ ),  $\Phi(P^1, P^2) = 0$ . For the intermediate cases  $P^2 - \Delta P < P^1 < P^2$  ( $P^1$  is less important than  $P^2$  but is not negligible),  $\Phi(P^1, P^2)$  is given by the following equation:

$$(8) \Phi(P^1, P^2) = \exp\left(\ln(\varepsilon) \times \left(\frac{P^2 - P^1}{\Delta P}\right)^2\right)$$

Hence, for each action  $A^i$ , it is possible to define the set  $I^i$  of actions that are not negligible compared to it. A new rule is then introduced: if  $A^i$  is active ( $S^i=1$ ) and all the other actions are negligible compared to  $A^i$  its weight is equal to 1.

To account for all the above properties, we propose a recursive process to compute  $Param_i(t)$ . This recursive process starts with the action that has the higher priority  $A^n$  since we are sure that it has an influence on the resulting motion. Then, we compute recursively the effect of the other action  $j$ ,  $F(j, i)$  (in the decreasing order of their priority) until a negligible action is encountered or until all the actions have been processed:

$$(10) F(j, i) = S^j \times G(j, i) + (1 - S^j) \times F(j-1, i)$$

with

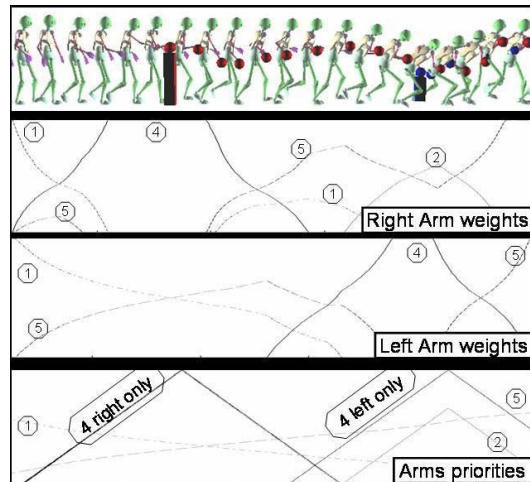
$$G(j, i) = \frac{\sum_{p \in I^j} (S^p \times \Phi(P^p, P^i)) \times Param_p}{\sum_{p \in I^j} S^p \times \Phi(P^p, P^i)}$$

where  $F(0, i)$  is the  $i^{\text{th}}$  parameter of the neutral action  $A^0$ , always active ( $S^0=1$ ) and which priorities are null for all the parameters ( $P_i^0=0$ ). Let us take the walking  $A^1$  and grasping  $A^2$  example. In this example  $S^1=S^2=1$  all along the sequence so that walking and grasping are running concurrently. Only the actions that are not negligible compared to the current action  $A^i$ , ( $p \in I^i$ ), are considered. By defining a very high priority  $P_i^2$  (where  $i$  stands for the hand-position constraint),  $P_i^1$  is considered as negligible. Consequently, the position of the hand is completely constrained by the grasping movement.

## 4. Results

The example of figure 2 deals with the grasping of two objects while running and walking. Five different motions are considered, assuming no synchronization problem. The first one,  $M^1$ , is a natural walk. The second one  $M^2$  is a crouching motion. The third one,  $M^3$ , is a special walk. The fourth one,  $M^4$ , is a grasping motion and is associated to a hand-position constraint. The last one,  $M^5$ , is a run.

The top figure shows the resulting animation. The two images below depict respectively the weights and the priorities applied to the arms for each motion. The method also ensures that the constraint's weight increases in a continuous way until the sphere is reached. When no grasping is performed, the arms were driven by the other active motions. The two figures at the bottom indicate the time-dependant priority on the grasping arms. This example was calculated in real-time at 60Hz on a P3 450MHz.



**Figure 2: resulting sequence involving 5 motions (1: normal walk, 2: crouching, 3: silent walk, 4: grasping and 5: running). The user interactively tuned the priority of the grasping task with the corresponding hand-positioning constraint, for the two arms. The priorities and weights are shown for the two arms. Several kinds of locomotion were also blended together.**

## 5. Conclusion

The blending process presented in this paper is part of a global project where the blended movements are constrained to fit the skeleton dimensions and the environment while ensuring synchronization. This method is inspired by Boulic et al.'s work [3] where movements are stored on a stack to model priorities. The higher the position of an action is in the stack, the higher it has an influence on the resulting blended motion. To increase the influence of an action  $A^2$  stored in the second position in the stack requires to set the first movement priority to 0 or to stop and restart  $A^2$  (that is to move it at the top level). In our system, the user has only to tune priorities of an action to give it more or less importance without

taking the other actions into account. So far, this method has been applied to our skeleton description  $Param(t)$  but it could be also applied to quaternions.

As with several other automatic systems based on kinematics, our system suffers for some limitations. Indeed, nothing ensures that the weighted sum of two motions that both respect dynamics also respects dynamics. To take this kind of constraints into account, it could be interesting to control the blending process by using inverse kinetics or dynamics. Nevertheless, it is still difficult to define generic dynamical laws that could work for every kind of motion. In the present version of our system, if the synchronization between motions is possible, the system can blend them. However, blending two movements that are totally dynamically incompatibles is the responsibility of the user.

The low computation cost of this method enabled us to animate a hundred of walkers in a virtual environment. Moreover, as we are able to blend movements and adapt the results to the skeleton and the environment, only a tiny database of motions is enough to face a lot of situations.

## Acknowledgements

This work was partially founded by the French research program, RIAM within the AVA-MOTION project initiated by Dæsign corp. We thank C. Pelachaud for her help and corrections.

## References

1. Arıkan, O., and Forsyth, D. Interactive motion generation from examples. *ACM Transactions on Graphics* 21,3, 2002, 483-490.
2. Ashraf, G., and Wong, K.C. Constrained Framespace Interpolation. In *Proceedings of Computer Animation 2001*, IEEE Computer Society Press, 61-72, 2001
3. Boulic, R., Becheiraz, P., Emering, L., and Thalmann, D. Integration of Motion Control Techniques for Virtual Humans and Avatars Realtime Animation. In *Proceedings of ACM International Symposium VRST*, ACM Press / ACM SIGGRAPH, 111-118, 1997
4. Choi, M.G., Lee, J., and Shin, S.Y. Planning Biped Locomotion using Motion Capture Data and Probabilistic Roadmaps. *ACM Transactions on Graphics*, 22, 2, 182-203, 2001
5. Gleicher, M., Shin, H. J., Kovar, L., and Jepsen, A. Snap Together Motion: Assembling Run-Time Animation. In *Proceedings of Symposium on Interactive 3D Graphics*, 2003
6. Kovar, L., Gleicher, M., and Pighin, F. Motion Graphs *ACM Transactions on Graphics*, 21, 3, 473-482, 2002
7. Kovar, L., and Gleicher, M. Flexible Automatic Motion Blending with Registration Curves. In *Proceedings of ACM Symposium on Computer Animation*, ACM Press / ACM SIGGRAPH, 214 – 224, 2003
8. Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., and Pollard, NS. Interactive Control of Avatars Animated with Human Motion Data. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, J.F. Hughes, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 491-500, 2002
9. Ménardais, S. Fusion et adaptation temps réel de mouvements acquis pour l'animation d'humanoïdes synthétiques. *PhD Thesis*, University of Rennes 1, 2003.
10. Sun, H.C., and Metaxas, D.N. Automating gait generation. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 261-269, 2001.
11. Tak, S., Song, O.Y., and Ko, O.Y. Spacetime sweeping: an interactive dynamic constraints solver. In *Proceedings of Computer animation 2002*, IEEE Computer Society Press, 261-270, 2002.
12. Tsumura, T., Yoshizuka, T., Nojirino, T., and Noma, T. T4 : A motion capture based goal directed realtime responsive locomotion engine. In *Proceedings of Computer Animation 2001*, IEEE Computer Society Press, 52-60, 2001
13. Wang, J., and Bodenheimer, B. An evaluation of a Cost Metric for Selecting Transitions between Motion Segments. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, 232-238, 2003