Interactive animation of virtual humans based on
motion capture data

Franck Multon\textsuperscript{1,2}, Richard Kulpa\textsuperscript{1,2}, Ludovic Hoyet\textsuperscript{2}, Taku Komura\textsuperscript{3}

\textsuperscript{1} M2S Lab., University Rennes2, Av. Charles Tillon, 35044 Rennes, France
Tel. (+33)2 99 14 17 78 Fax. (+33)2 99 14 17 74
email: \{Franck.Multon, Richard Kulpa\}@uhb.fr

\textsuperscript{2} Bunraku Project, INRIA, Campus Universitaire de Beaulieu,
35042 Rennes, France
email: Ludovic.Hoyet@irisa.fr

\textsuperscript{3} IPAB JCMB, Edinburgh Univ, Kings Buildings,
Mayfield Rd, Edinburgh EH9 3JZ, UK
email: tkomura@inf.ed.ac.uk
Abstract

This paper addresses the problem of obtaining real-time parameterized simulations of human motion thanks to motion capture data. Contrary to motion graphs that are based on large database of motions, our method relies on per-frame adaptations of motion clips that are selected by a user. For the locomotion example, the user selects a motion that conveys a specific style and drives the character in a complex environment in real-time with only the information of stances that can be computed automatically.

The framework presented in this paper uses then sequentially several features such as morphological adaptation, kinematic constraints solving and dynamic correction of the motion. It automatically adapts the required motions to the size of the synthetic figure and to its environment (such as complex grounds and additional external forces). All these features are organized in layers in order to be easily combined together. It also simplifies the design of controllers such as those enabling displacements or grasping. Indeed, since all the features are embedded and organized in layers, creating a controller simply means defining the constraints (kinematic, dynamic...) and the framework ensures animating hundreds of humanoids with different morphologies in real-time (per-frame solver). It is particularly suitable for interactive applications such as video games and virtual reality where a user interacts in an unpredictable way.

Keywords: virtual human, real-time animation, kinematics, dynamics, real-time controller design
Adapting motion capture data to new situations rises three main problems: solving kinematic constraints, ensuring continuity and preserving the original style. Displacement maps [1] [2] [3] were introduced in order to solve these problems. It consists in solving constraints when needed and then filtering the resulting discontinuous motion. This approach can be used to retarget a motion to a character [4]. However, displacement maps require a complete knowledge of the sequence and the constraints in advance. When the user interacts with the virtual character, such constraints are not accurately known in advance as the user can modify the actions and the environment in an unpredictable way.

Real-time inverse kinematics [5] [6] [7] and kinetics [8] (ensuring the control of the position of the center of mass) can also be solved in real-time. In the specific case of footskate cleanup, this process is very fast [9]. For a whole-body pose control, it is very difficult to control more than one character in real-time since the computation resources must be shared by all the tasks processed by the animation engine. Moreover these techniques are only designed to solve kinematic and kinetic constraints in interactive time but cannot ensure that mechanical laws are verified (such as preserving the angular momentum constant during aerial phases or limiting joint torques).

To solve kinematic and dynamic constraints concurrently, several authors have proposed to optimize an objective function gathering all these constraints [10, 11, 12] [13] [14]. By nature, this process cannot be used in real-time animation as it requires a complete knowledge of the constraints in advance. Another solution consists in applying inverse dynamics and in optimizing the motion until the forces and torques reach acceptable values [15]. This
method has been extended in order to deal with dynamic stability [16] and to correct the angular momentum in aerial phases thanks to time warping [17] or real-time local adjustments [18]. Dynamic filters were introduced to adapt an incorrect motion frame by frame in order to satisfy the mechanical laws [19]. It has been applied to obtain a stable motion from the dynamic point of view [20].

In order to react to external perturbations, it is also possible to simulate the passive behavior of the virtual human’s mechanical system and to extract from a database of possible reactions the motion that best fits the simulation conditions [21, 22]. This process is based on a whole sequence and is difficult to apply in real-time for interactive applications. Moreover, only local perturbations such as pushes are considered. It cannot be used to compensate continuous external actions, such as pushing an heavy object.

In videogames, the problem is to control the motion of a character in real-time according to a set of parameters, such as footprints, global direction of the displacement, velocity. . . Parametric models of human-like locomotion have been widely proposed in past works based on biomechanical knowledge [23, 24, 25, 26]. Recently, this kind of model has been extended to dynamic simulation [27, 28]. These models are very efficient in order to obey complex orders provided by a user in real-time but it remains difficult to deal with various individual styles. Encoding style is very difficult and motion capture data are generally used as an alternative to solve this problem.

In this paper, we propose a framework to animate virtual humans in real-time thanks to motion capture data while taking kinematic and dynamic constraints into account. All
this framework is based on a morphology-independent representation of motion that adapts the motion to any humanoid morphology and speeds up the process of constraints solving. Thanks to this framework it is then possible to easily design a controller that is acting as a parametric model of displacements based on stylized motion capture data.

**Overview**

The whole framework is able to synchronize and blend several motion capture data in order to solve a complex task in real-time, as described in [29]. When all the desired motions are blended into a unique pose, this latter can be adapted to a set of constraints. This paper focuses on this part of the framework only. Let us consider \( q \) the set of data that is associated to a pose of the character. In this paper, these data are based on a morphology-independent representation of motion, as described in the following section. At time \( t \), \( q \) is associated to a set of kinematic and kinetic constraints (modeled as equality equations \( f_i(q) = c_i \)).

Before solving these constraints, \( q \) must be scaled to the virtual human’s morphology, as depicted in figure 1. The resulting vector \( q_s \) is not composed of joint angles any more but contains Cartesian and angular data. The inverse kinematics and kinetics solver is applied on \( q_s \) in order to find a pose that satisfies a compromise of all the constraints. At this step, the solver delivers joint angles that are compatible with the virtual character. However, the resulting motion doesn’t take external forces into account so that the angular and linear momentum may be incorrect from the mechanical point of view. The Dynamic Correction
module aims at modifying the resulting angles in order to ensure that external forces are taken into account. This module doesn’t compute the joint torques but consists of analyzing the mechanical constraints applied to the global system. Hence, during the aerial phase, the angular momentum should remain constant and the acceleration of the center of mass should equal gravity. During the contact phase, we assume that the character should counteract the external forces to preserve as much as possible the initial task (such as walking whereas the additional forces would lead to falling). To do so, the system can tune the orientation of groups of body segments. The goal is to modify the ground reaction force, the center of pressure, the position of the center of mass and the global orientation in order to counteract the new imposed external forces.

Morphology-independent representation of motion

The morphology-independent representation of motion was preliminary described in [30]. It is based on a Cartesian and angular normalized representation of a posture. In this representation, the human body is subdivided into kinematic subchains (see figure 2) that describe parts of the skeleton. Those kinematic chains are divided into three main types:

- the normalized segments that consist of only one body segment (such as the hands, the feet, the clavicle and the scapula). They are normalized by their length;

- the limbs with variable length that encode the limbs composed with two body segments each. In this representation, the intermediate joints (elbows and knees) are not
encoded given that their positions are directly linked to the characters anthropometric properties. To retrieve them, an analytical solution derived from IKAN [31] is used;

- and the spine that is modeled by a spline (as suggested in biomechanics) which advantage is that it can be easily subdivided into as many segments as wishes, depending on the character description. This spline is easily normalized by scaling the coefficients between 0 (close to pelvis) and 1 (close to the skull).

Given this data structure, every motion capture data or edited trajectory (whatever the size of the initial skeleton) can be used as an initial pose at each time step. This pose is then adapted to the new skeleton by simply multiplying all the normalized data by the dimensions of the new character.

**Motion adaptation**

This section addresses the problem of adapting a motion to new constraints, including kinematic constraints, external forces while correcting mechanically invalid poses.

**Solving kinematic constraints**

Once a set of parameters $q_s$ is scaled to the size of the virtual character, the system has to solve the equality constraints $f_i(q_s) = c_i$ to ensure realistic animation (e.g. the contact of the feet on the ground) and to verify the actions of the character (e.g. grasping an object).
Generally, this task is performed thanks to an inverse kinematics solver that consists in locally linearizing $f_i$ to inverse it. However, the computation cost is very high and this process sometimes leads to unrealistic poses. The main problem is to solve constraints expressed in the Cartesian frame whereas $q$ is a set of joint angles. With our representation based on relative Cartesian positions, the problem is simpler. Hence, for a group of body segment, it is possible to find an analytical solution for each constraint. For an arm, it is thus possible to determine very efficiently the new location of the wrist in the shoulder reference frame while preserving the initial orientation of the arm. The same kind of direct solution can be found for each group of body segments (limbs, trunk and head). In [32], we provide some details about this constraint solver that also enables to control the position of character’s center of mass.

The contact with the ground can be achieved thanks to a specific method in order to speed up the constraint-solving process [30]. This method is also based on the morphology-independent representation introduced in the previous section. Continuity is ensured by applying continuous constraints on the feet. Indeed, if a contact (or a release) is expressed as a discontinuous constraint, the per-frame solver will lead to discontinuous poses. Hence, each constraint is associated with a state that evolves continuously depending on time: from 0 (totally inactive) to 1 (fully active).
Taking dynamics into account

The kinematic constraint solver may lead to unbalanced poses that should be adjusted. For dynamic motions, it consists in adapting the joint angles if the corresponding whole-body motion doesn’t satisfy the mechanical laws. For example, a linear walking that is adapted in order to follow a curved path should lead to leaning the whole body in order to take centripetal accelerations into account. The character can also have to carry or push more or less heavy objects. For all of these cases, we can consider external forces $F_e$ added to those originally embedded in the current motion clip. Thus, the system should be able to adapt the sequence of poses to react to these perturbations in a per-frame solver. To do so, the system can act with two complementary methods:

- displacing the center of pressure $COP$ into the base of support in order to create a new torque $\Delta COP \times R$ where $R$ is the ground reaction force. If the required $COP$ remains in the base of support to counteract all the perturbation, no other change is needed; nothing is changed in the character’s pose.

- if the first solution is not enough to counterbalance $F_e$, the next step consists in changing the ground reaction force $R$. Theoretically $R$ becomes $R - F_e$. All the components of $R$ change but the vector should stay within the cone of friction forces. Each change of the ground reaction force leads to a change of the global orientation and position of the center of mass. As the character is attached through kinematic constraints to the ground, it leads to a change of the pose of the character, as depicted in figure 4.
For the latter, the main problem is to express the Newton’s laws as a function of the global orientation of the character $q_1$ at time $t$:

$$M_W(q_1) + M_R(q_1) + M_{F_e} - \dot{H}(q_1) + M(m\ddot{x}(q_1)) = 0$$  \hspace{1cm} (1)

$M_W(q_1)$ is the torque due to the body weight, $M_R(q_1)$ is the one due to the ground reaction force (linked to the position of the center of pressure), $M_{F_e}$ is the one due to the additional external forces and $\dot{H}(q_1)$ is the derivative of the angular momentum. The center of pressure is assumed to be the Zero Moment Point, usually utilized for stabilizing biped robots [16]. Without additional external forces, when a motion is retargetted to a new character, this sum is certainly different from 0. This is due to a set of approximations: the model is different from a real human body, the body segments length and mass properties are also different. So equation 1 becomes:

$$\min_{q_1} \left( M_W(q_1) + M_R(q_1) + M_{F_e} - \dot{H}(q_1) + M(m\ddot{x}(q_1)) \right)^2 \hspace{1cm} (2)$$

This expression has to be minimized to calculate the global orientation of the character that best satisfies equation 1. When the global orientation is known, the kinematic constraints may be unsatisfied and a new inverse kinematics step is again performed to locally correct the pose of the character, as depicted in figure 1. As a consequence, the global orientation is modified not only to compensate the change of morphology but also the additional external forces. $q_1$ should also remain in the neighborhood of the current pose in order to avoid discontinuities. Hence, the search space is quite small and a simple iterative method can find an optimal solution with only few steps.
Left part of figure 4 depicts a character that has to push a furniture. Depending on the weight of the furniture and the friction forces on the ground, the virtual human has to compensate an external force applied to his hands. In this figure, the light-gray character stands for the original pose, without additional external force. The other character is obtained according to our method for furniture which weight is 50Kg. The system automatically rotates the whole body to compensate the external forces exerted by the furniture. The kinematic constraints (connecting the feet without sliding on the ground and putting the hand on the furniture) are also satisfied. This correction is performed in real-time.

Right part of figure 4 depicts a character that counteracts an external force $F_e$ applied perpendicularly to his displacement. The system leans the whole-body in the opposite direction of the exerted force to compensate this perturbation. As a consequence, the system rotates the whole body which corresponds to turning with a radius corresponding to a centripetal acceleration equal to $F_e/m$ (where $m$ is the mass). He consequently turns in the direction of the origin of the applied force $F_e$. Hence, to drive the displacements of the character, it is possible to either tune these external forces or apply a radius (that leads to an external force equal to $m \times \ddot{x}$ where $x$ is his position).

**Layered parametric controllers**

Thanks to the automatic features described in the previous sections, such as automatic morphological adaptation, constraints solving and blending of motions [29], the process of de-
signing interactive controllers is simplified. Let us consider the example of a controller that is supposed to make a virtual human displace in a complex environment with various possible styles, such as walking, running, dancing, jumping, etc. In classical approaches, each new controller generally requires a complete redesign; a controller for displacements cannot be reused, even partially, for grasping. In the previous part of the paper, we have introduced basic functionalities that can be organized as a layered controller (ranging from morphology-independent representation of motion to real-time dynamic correction of motions), as depicted in figure 5. This layered motion control is compatible with models published in psychology [33]. In this architecture, designing a specific controller consists in building a new layer that will benefit from the existing layers and functionalities. Compared to the popular displacement maps which constraints (for retargetting and geometric constraints solving) are all solved concurrently, this layered architecture simplifies the design of new real-time controllers. In that case, a controller is viewed as an object that can make some requests to the solvers embedded in the lower layer. Hence, a controller for grasping or displacements is simply designed by specifying a sample motion and by accessing to the below constraints solvers according to high-level parameters (such as direction and velocity, or the location of the target that has to be grasped).

Let us consider now two examples of a controller that manages the displacements of a character simply by using one motion that contains the animation (conveying the desired style) and a destination that the humanoid has to reach. One of the tasks of the controller is to compute the per-frame correct orientation of the character in order to walk in the direction
of the goal. This new orientation is calculated by adding the change of angle $\Delta \theta$ between the current direction of the displacement and the one of the target. To have a realistic animation, a maximum limit is set on the allowed angular speed to avoid too fast rotations. These simple calculations are sufficient to create this first controller. The framework computes the new position of the footprints according to the original one. It applies a rotation of $\Delta \theta$ which center is placed on the contact point between the foot and the ground (where the ground reaction force takes place).

Once the new footprints are computed at each frame, the controller provides the IK solver with the corresponding constraints. As the constraints are solved at each time step, continuity is ensured by applying a continuous change of the constraints (provided by the controller). Thus, the simple use of a control of the orientation of the character leads to a generic model of displacement whatever the kind of motion is chosen by the user. The remaining problems are automatically solved by the layered framework. For example, figure 6 shows the use of this model on a complex motion with an odd style such as a drunk man who is walking. To be used by the controller, a motion should be composed of a sequence of stances that provides the corresponding sequence of footprints.

This very simple model can be enhanced in order to adapt the footprints to complex environments. The key idea is the same: providing the constraint solver layer with continuous constraints that are adapted to the situation. To this end, the proposed controller has two main types of parameters. The first type is based on the position and orientation of the footprints during the stances, as in the previous controller. The layered framework returns
the current state of the character, including the footprints stored in the selected motion. The second type consists in topological data of the ground that may lead to changes in the displacement of the character. For uneven grounds or stairs (see figure 7a for an example), the controller computes the corresponding footprints and provide it to the constraints solver layer. The computation of the footprints according to the environment may be complex. It should take complex steering behaviors into account, such as anticipating a climbing of stairs or avoiding obstacles (see figure 7 for such kind of examples). In some cases, from the behavioral’s point of view there is no need to control exactly the position and orientation of the footprints but just the global displacement. Designing realistic steering methods is beyond the scope of this paper but the layered controller offers a simple way to such kind of model. For example, this layered framework can be used with real-time path planning [34], as depicted in figure 7.

If we now consider a controller for grasping objects at various locations and with various styles, we can proceed the same way. The grasping controller is designed as an object composed of sample motions (conveying the desired style) and a knowledge of the location of the target. This latter parameter is associated with a constraint provided to the constraints solver layer. As for displacements, this constraint should be continuously tuned to avoid discontinuities. It leads to designing a continuous trajectory $\Delta X(t)$ that has to be added to the original trajectory of the hand in order to reach the target. The user can parameterized this trajectory in order to convey a given style (by changing the speed profile, for example). As grasping an object in various places may lead to very different motions (such as flexing
the knee to reach low targets), the model is using several sample motions according to the area in which the target is placed. Thus, the controller selects the most convenient sample according to the location of the target. The resulting model can be used alone (as depicted in figure 8) or in association with the above displacement controller thanks to the blending functionality of the framework.

**Conclusion**

In this paper, we have described a layered framework to control the movement of virtual characters thanks to motion capture data. This framework ensures that the motion is correctly adapted to the size of the character. Moreover, it adapts the motion to kinematic constraints with very few computation time. For example, it allows the character to move on any ground even if this latter evolves in real-time. Classical approaches that modify an entire motion are not suitable to manage such kind of challenge. The system is also able to correct poses that does not satisfy general mechanical laws by considering only general mechanical quantities such as angular and linear momentums.

With this layered framework, creating a real-time controller simply implies defining the constraints applied to the motion, such as the footprints during while walking and the location of the target for grasping. All the adaptations are performed automatically in real-time so that the system can adapt to unpredictable changes of the environment.

This framework has been associated with high-level models (including steering and be-
havioural models). In this paper, we focused on a displacement and a grasping controller but it could be easily illustrated on other types of motions, such as fighting, working in a virtual plant... In the current version the key idea was to preserve the style conveyed in the original motion clip. As a future work, we wish to also change this style by introducing a style-translation layer. However, one of the main challenges of this layered architecture is to ensure convergence. Hence, solving kinematic constraints after having correcting mechanical unfeasible poses may lead to new unfeasible poses. We will now continue working on this problem to improve the quality of the resulting motions.

References


Figure 1: Overview of the motion adaptation framework.

Figure 2: Morphology-independent representation of motion.
Figure 3: Dozens of characters interactively controlled by a user who set a set of constrains on the wrists and the ankles. The center of mass is also constrained to keep the position of its projection on the ground at the same place.

Figure 4: Left: the motion of the character is adapted in real-time in order to push a heavy cupboard while the original motion was simply walking with the two hands placed in front of him (light-gray character); Right: the character reacts in real-time to an external force exerted on his shoulder while walking.
Figure 5: Organization of the features in layers. It ensures that all the features can be used together and provides an interesting support for the easy creation of controllers.

Figure 6: Resulting animation when the controller ensures that the drunk man reaches the vertical arrow. The captured motion is a walk of a drunk man following a straight line. In this sequence of poses, the figure shows the new position of the footprints computed to ensure that the character will reach the target.
Figure 7: a) Example of pose obtained during a displacement of the character on stairs. The controller ensures that the footprints are correctly placed on the ground. b) Example of the use of a controller of displacement in complex environment.

Figure 8: a) Grasping controller applied to an object placed in front of the character b) same controller applied to an object placed on the ground.