Morphology-independent representation of motions for interactive human-like animation — Source link

Richard Kulpa, Franck Multon, Bruno Arnaldi

Institutions: University of Rennes

Published on: 01 Sep 2005 - Eurographics

Topics: Skeletal animation, Joint constraints, Inverse kinematics and Animation

Morphology-independent representation of motions for interactive human-like animation

Richard Kulpa, Franck Multon, Bruno Arnaldi

EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (2005), Number 3

Morphology-independent representation of motions for interactive human-like animation

R. Kulpa[1,2], F. Multon[2,1] and B. Arnaldi[1]

[1] SIAMES Project - IRISA, Rennes, France
[2] LPBEM - University of Rennes 2, France

Abstract

This paper addresses the problem of human motion encoding for real-time animation in interactive environments. Classically, a motion is stored as a sequence of body postures encoded as a set of joint rotations (quaternions, Euler-like angles or rotation matrices). As a consequence, Cartesian constraints must be solved using inverse kinematics and/or optimization. Those processes involve computation costs that do not allow real-time animation of several characters in interactive environments. To solve such a problem with a minimum computation time, we designed a motion representation independent from the morphology and containing the constraints intrinsically linked to the motion such as feet contacts with the ground. With such a description, a unique motion can be shared by several characters with different morphologies and in different environments. We also adapted a Cyclic Coordinate Descent algorithm that takes advantages of this representation in order to rapidly deal with complex tunable spacetime constraints. For example, this method enables to interactively control at least eight characters with different morphologies that interact each other during a fight training. Hence, each character has to deal with geometric constraints that can change at every time, depending on the opponents' morphology and gestures.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation

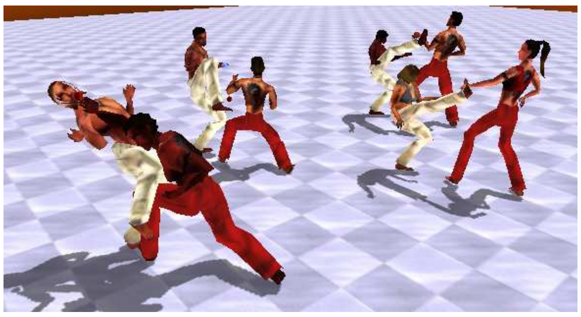

Figure 1: *Eight characters dealing with several constraints in real-time.*

1. Introduction

In interactive environments, designing a set of behaviors for an avatar is difficult primarily due to the real-time constraint, especially if we wish to use a relatively poor motion capture database. In such interactive environments, selecting behaviors in a database requires to pre-calculate all the possible transitions between them by using motion graphs [KGP02] [LL04]. An alternative consists in merging several elementary motions thanks to frame-space interpolation [GR96]. Nevertheless those techniques generally require large motion capture databases. An alternative is to use motion adaptation that enables to create new behaviors with a small motion capture database. The main point is then to ensure that adaptations preserve the style and realism of the original motion. The motion must be adapted to the skeleton of the new character. The environment should also be considered to adjust the motion to new constraints.

To solve this problem, two main approaches could be identified: kinematic or dynamic based techniques. While the former allows low computation time, the latter enables to take into account balance, force magnitude control, etc. Dynamic methods are not yet suitable for interactive environments where a user can interfere with several synthetic characters concurrently. Moreover, techniques involving spacetime constraints and optimization cannot be considered because they require a complete knowledge of the constraints whereas a user can interact at any time. Previous works are generally based on the modification of angular trajectories whereas constraints are generally given in the Cartesian frame. The resulting process consequently involves using inverse kinematics.

We propose a specific description of motion in order to accelerate and optimize the process of motion adaptation. To this end, instead of using angular trajectories we propose to deal with data that are not linked to the character's anthropometric properties. This method enables us to animate in real-time several different characters who have to deal with constraints that can change at every frame (see figure 1).

2. Related works

Adapting motions to new constraints has been investigated by adding displacement maps to the original trajectories [Gle98] [LCR*02] [LS99]. Although those techniques are very efficient to edit motions, a complete knowledge on spacetime constraints is required for all the sequence. As a consequence, modifying those constraints in real-time is not possible. Let us consider a kick that must drive the foot to the opponent's head while the other character is also moving. If no assumption of the opponent's head trajectory is possible, displacement maps calculated at the beginning of the sequence may not verify the constraints at the end of the sequence given that the opponent's head is moving unpredictably.

An alternative is to use statistical analysis [BH00] [LWS02] [SHP04] in order to decrease the number of degrees of freedom that are controlled and consequently to simplify the control problem. Learning a search-space is also possible to perform so-called style-based inverse kinematics [GMHP04]. Whatever the technique, a large database of motions is required to obtain accurate controllers whereas we aim at using a small motion capture database.

As constraints are generally expressed in the Cartesian frame whereas joints are controlled using rotations, another solution consists in using inverse kinematics at each frame. This technique allows interactive motion deformation with prioritized constraints [CB04] including the control of the center of mass trajectory [BB04]. In order to decrease computation time, the method can be applied on only subsets of the entire kinematic chain [BCHB03]. A dynamic filter can also be used to ensure that the resulting motion verifies dynamic constraints [TySK02]. This approach is an alternative to the search of a controller that drives a dynamic system by computing the forces and torques that minimize a given objective function [PW99] [FP03] [SP05].

Although those methods provide a powerful tool to adapt motions to new constraints rapidly, the computation time does not allow the control of several characters in real-time. Improvements have been proposed to apply inverse kinematics with less computation time, such as using specific formulations for anthropometric limbs [TGB00]. However, controlling sub-parts of the skeleton independently is not always adequate to solve constraints that require the displacement of all the body. [SLSG01] extent this approach by using an iterative process in order to deal with the complete body. Their goal was to map the movements of a performer to an animated character only considering constraints on the end-effectors. Another improvement consists in using an intermediate skeleton [MBBT00] that has less degrees of freedom and then in recalculating the remaining degrees of freedom analytically. This improvement was also used to control a simplified mechanical system [PW99] while inverse kinematics was used to recover the remaining degrees of freedom. Those works demonstrate the interest of using a specific representation of motion (through simplified skeletons) and we propose to go further those steps. Hence, we propose to define a data structure especially dedicated to motion adaptation: it is supposed to decrease the computation time required to solve complex Cartesian constraints that could change at each frame.

To accelerate the constraints solver based on this data structure, we propose to use the Cyclic Coordinate Descent technique (denoted CCD in the remainder of the paper) [WC91] [Lan98]. Although this method offers rapid solving of inverse kinematics problems, it is not possible to add secondary tasks and consequently to use specific laws. As a consequence, some artifacts may occur in the resulting sequence, such as an inhomogeneous repartition of the deformation along the kinematic chain [Wel93]. Several heuristics were proposed to make the system converge to an acceptable solution. For example, a threshold can be used at each time step in order to limit the quantity of rotation taken by the first joints. This technique, called damping, makes the solution be more homogeneous but it engenders more iterations to reach it. Joint limits can also be considered [Lan98]. However, as CCD does not consider movement continuity, it must be improved to provide acceptable solutions for motion adaptation. Moreover, it does not take comfort information such as using preferably the arm and not the trunk during grasping.

In this paper, we propose to use the advantages of several of the above techniques by introducing a specific data structure to encode the motion. We also propose an enhancement of the CCD that enables to have a more realistic adaptation and to respect the center of mass (called COM in the remainder of the paper) as well as to control every points of the character's body.

3. Overview

As stated above, classical approaches dealing with joint angles require optimization and inverse kinematics to solve

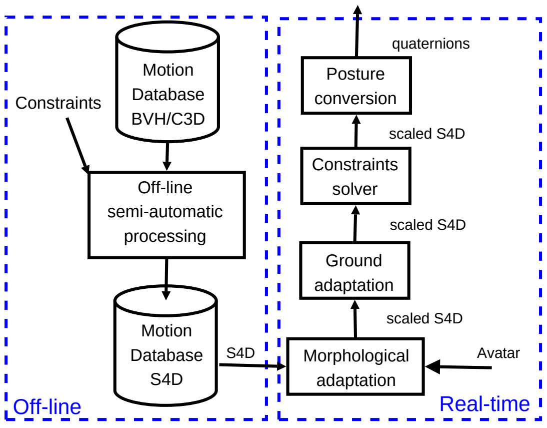

Figure 2: *Overview of the motion adaptation process.*

Cartesian spacetime constraints, such as moving on an uneven terrain, grasping objects at various positions in space… Even for flat grounds, scaling data to new skeletons requires inverse kinematics or optimization to ensure feet contacts with the environment. To take those requirements into account without using complex inverse kinematics methods, we propose to subdivide the skeleton into subparts. Human body is considered as a hierarchy of subparts (such as an arm, the trunk…) expressed relatively to their parent, coupling Cartesian and angular data. For example, instead of storing all the joint angles, we select only the angles that are not directly linked to anthropometric data, such as the angle of the plane containing the shoulder, the elbow and the wrist. The Cartesian data are normalized so that only adimensional data are stored in order to simplify the scaling process.

However, motion is not limited to a sequence of postures but also intrinsically contains constraints, such as contacts between parts of the skeleton and the environment. Foot contact can be semi-automatically identified by selecting the times where the foot height and velocity are below a given threshold [MKMA04]. Other constraints depend on the nature of the motion and must be edited by a user. Those tasks are performed off-line for each captured motion in a so-called *Off-line semi-automatic process* (see figure 2). The inputs are captured files in C3D or BVH format and the outputs are files stored in our format called S4D.

Given a file stored in this format and an avatar (described with its geometry and anthropometric data), a three-steps process is performed in our real-time animation engine (called MKM) in order to adapt the motion to a new character and to the environment. First, the adimensional data stored in the file are scaled to take the new character's anthropometric data into account (called *Morphological adaptation* module). The output is called scaled S4D given that it is still based on our motion representation. Second, the feet that must be in contact with the ground are displaced to a convenient position (called *Ground adaptation* module). As a consequence, the root is recalculated to be compatible with the new feet positions. The data are still in the scaled S4D format. Third, the other constraints are solved in the *Constraints solver* module. Those constraints are the ones edited by the user in the *Off-line semi-automatic process* and others that can be added interactively in the real-time environment. At this step, the scaled S4D format is still used but other points directly linked to the geometric constraints are calculated when needed. All the constraints require geometrical data provided by the real-time environment (such as actual position of a target that have to be reached or actual size of an object that must be held). Finally, joint angles are calculated and sent to visualization through the *Posture conversion* module. Only at this step, our specific format is abandoned to provide visualization with a classical posture representation.

Section 4 describes the data structure used to encode motion. Then section 5 explains the constraints associated to this representation. Section 6 shows that this adimensional representation allows simple and rapid morphological adaptation. Section 7 describes how the motion is adapted to the ground. The way all the joints are calculated according to our data structure is presented in section 8. Next, section 9 deals with the adapted CCD algorithm that makes it possible to control several characters in real-time while respecting the COM. Some results are presented in section 10.

4. Normalized skeleton

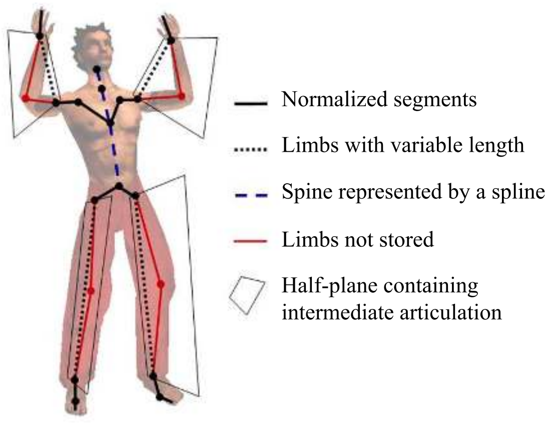
Normalized segments
Limbs with variable length
Spine represented by a spline
Limbs not stored
Half-plane containing intermediate articulation

Figure 3: *Normalized representation of the skeleton.*

The motion is stored using a normalized representation of the skeleton and a set of associated constraints. In this section, we describe the representation of a posture. Classically, the human body is considered as a set of kinematic chains starting from the root to extremities. In our representation, the human body is also subdivided into kinematic subchains that describe parts of the skeleton (see figure 3). Those kinematic chains are divided into three main parts:

- *the normalized segments* are composed of only one body segment (such as the hands, the feet, the clavicle and the scapula),
- *the limbs with variable length* that encode upper and lower limbs; in this representation, the intermediate joints (elbows and knees) are not encoded because their position is directly linked to the character's anthropometric properties,
- and *the spine* represented with a spline that could be subdivided into as segments as wishes in the real-time animation module.

Whatever the kind of kinematic chain KC composed of segments S_j, only normalized values are stored. Let E and R be respectively the extremity and the root of the kinematic chain KC. In order to obtain a normalized representation of KC we can use the following equation:

$$normalizedKC = \frac{E-R}{\sum_j length(S_j)} \tag{1}$$

The *limbs* are stored using only two data: a reference frame attached to the root of the corresponding kinematic chain (shoulder or hip) and a scalar. The axes of the frame are depicted in figure 4: they are defined in order to represent the half-plane in which the intermediate articulation (elbow or knee) takes place. The scalar represents the normalized length of this limb, that is to say a percentage of the maximum limb length.

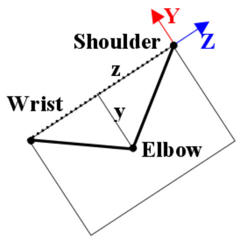

Figure 4: *Specification of the half-plane represented by the frame (X,Y,Z) which contains the elbow. The local coordinates (y, z) of the elbow are computed analytically.*

In this last representation, each point of the skeleton can be retrieved relatively to the root position. Contrary to classical representation, the instantaneous orientation of the root is divided into two main components. The global orientation deals with the global direction of the motion. The local orientation is the additional rotation applied to the global orientation in order to obtain the actual root orientation. During a walk, it represents the pelvis oscillations around the global direction. In order to encode movements without taking anthropometric properties into account, the position of the root is normalized by the leg length.

5. Constraints

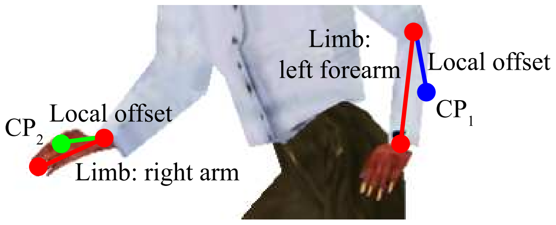

Figure 5: *Specification of the point that is constrained.*

As introduced in section 3, constraints are also stored with the motion. The feet contacts with the ground are necessary to avoid sliding effects or unwanted penetration of the feet into the ground. Additional constraints can also be added by the user. For example, considering a clapping motion, at least one constraint must be added to the motion: the contact between the two hands that occurs at repetitive periods of time. To model a constraint C_i several parameters are necessary:

$$C_i = \{CP_i, T_i, KC_i, P_i, A_i\}$$

The first parameter CP_i is the constrained point (see figure 5). It is linked to a body segment and its position is defined using a 3D local offset from the root of this segment. Next parameter T_i is the type of the constraint among the following ones:

- Position: the user can impose a desired position for a given point of the skeleton. Since all the computations are performed in real-time, the imposed position can change every time. For example, it is then easy to use the position of an avatar's articulation as the constraint for another character (see section 10);
- Orientation: the user can enforce a segment's orientation. This orientation can be directly applied to the corresponding reference frame;
- Distance: the user can impose a distance between two points of the skeleton. This constraint is also used to ensure contact between two points by imposing a null distance. For example, this constraint makes it possible to hold several different objects with different sizes according to a unique two-hands holding motion provided by the motion database;
- Restricted area: the user can constraint points into a spatial area (see *wristsAboveTable.avi*). Figure 6 shows the use of such constraints applied to the wrists. The avatar in the background talks while placing the wrists on his thighs, as in the original captured motion. In the front, the character plays the same motion while adapting it in order to keep the wrists above the table.

Depending on the constraint, specific parameters must obviously be added such as the desired position of the constraint or the dimensions of the restricted area.

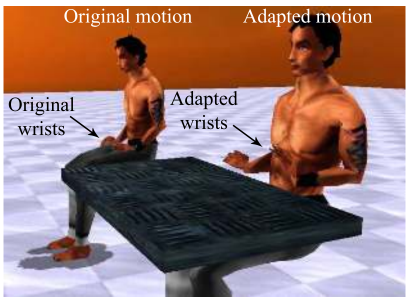

Figure 6: *The wrists are constrained in the area above the table.*

The next parameter KC_i defines the kinematic chain associated to the constraint. It offers the user the possibility to specify the set of usable body segments in order to solve the constraint. For example, in figure 7, three constraints are specified. C_1 is applied on the right hand and acts on all the segments ranging from the hand to the abdomen. The constraint C_2 only involves the arm and the clavicle.

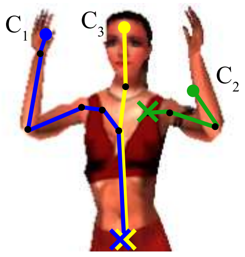

Figure 7: *Constraints are associated to a subset of the complete kinematic chain. The three different constraints are represented with big dots. The lines are their kinematic chains and their root are represented with the big crosses.*

The priority of the constraint, called P_i, indicates the importance of a constraint compared to others. The constraints with low priorities are only verified after those associated to higher priorities. Finally, the user can start and stop constraints using the parameter A_i which is the activation of the constraint (see figure 8). A_i includes the start and stop times of a constraint as well as the duration of activation and deactivation phases.

6. Morphological adaptation

With the representation of posture presented in this paper, morphological adaptation can be easily performed. Indeed, it consists in inverting the segments normalization process (described in equation 1) using the dimensions of the new body segments:

$$newKC = \sum_j newLength(S_j) * \frac{E-R}{\sum_j originalLength(S_j)} \quad (2)$$

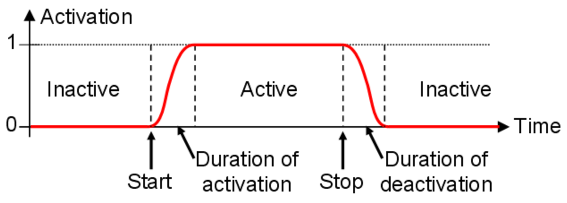

Figure 8: *Activation of a constraint.*

This equation demonstrates that no loss of information occurs by using our representation if *newLength* is equal to *originalLength*. This morphological adaptation can be performed rapidly in the real-time animation module. Consequently, a motion stored with this representation can be shared by all the avatars and adapted to new anthropometric properties. In the video (see *morphoAdaptation.avi*), several characters share the same dancing motion. In this video, the displacement of the young girl in black is smaller than the others. The step length is automatically scaled to her small legs. As morphological adaptation is performed in real-time the system is able to deal with body segments that could increase and decrease at each time step.

At this step of the whole process, intermediate articulations such as the elbows, the knees and the vertebrae are not yet reconstructed. Moreover, since the position of the root is normalized using the leg length, it is also adapted in order to be at the correct height from the floor. But if the ground is different or changes in real-time, ground adaptation is needed.

7. Ground adaptation

The ground adaptation consists in adapting the scaled relative posture according to the footprints placed on the ground. These footprints are calculated according to the foot-contact constraints specified in the motion file. If a function is able to provide the height of the ground (axis Z) for a given 2D position (X,Y), the new footprints can then be placed automatically. This function allows to handle any kind of ground even those that are interactively modified.

Ground adaptation ensures that the new position of the pelvis (and consequently the root) is compatible with the leg lengths while being as near as possible from the position stored in the original motion file. First, the desired positions of the ankles are retrieved according to the shape of the ground. Actually, it corresponds to a translation of the ankles along the Z-axis from their original positions (denoted *desired ankle* in figure 9, part (b)).

Second, the width of the pelvis is suppressed from the ankles positions in order to express them relatively to the root (see (c) in figure 9). The next step consists in calculating the new height of the root (see (d) in figure 9). The root's position must verify two conditions. It should be close to the original height $h1$ in order to preserve the initial style. It should also ensure that the root is reachable according to the two desired ankles positions (providing two maximum heights $h2$ and $h3$ for the two legs). The height of the root is then the minimum of these three heights: $hRoot = min(h1, h2, h3)$.

Next, the hips are retrieved by adding the pelvis width to the root position resulting from the previous steps (see (e) in figure 9). Our representation then enables to easily adapt the legs in order to respect the resulting hips and ankles. Indeed, the blue dot-lines in the figure represents the limbs with variable length. Adapting the leg is then simply achieved by aligning the Z axis of the limb frame with the new vector $\overrightarrow{ankle - hip}$. The normalized length of the leg is set to $\|\overrightarrow{ankle - hip}\|$.

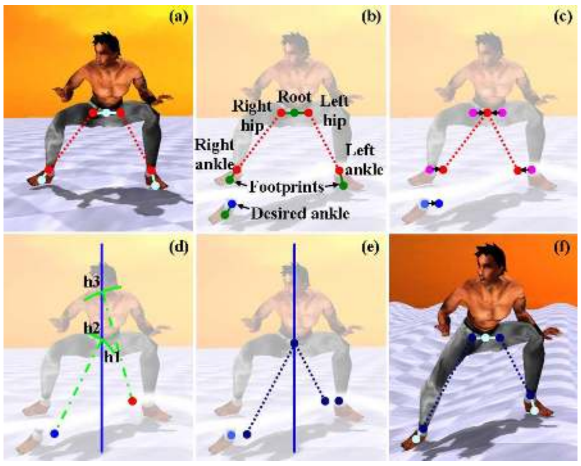

Figure 9: *Our normalized representation is suitable for ground adaptation of posture. (a) Original posture. (b) The blue dot is the desired position for the right ankle. (c) Pelvis width is suppressed to all the positions. (d) Three heights are considered for the root: its actual height and the maximum height the hips can reach keeping the feet on the ground. (e) The lowest height is chosen. (f) Final posture: with the normalized skeleton, legs are reconstructed automatically.*

8. Posture conversion

None of the previous adaptations use intermediate articulations (the elbows, the knees and the vertebrae). However, when the posture is rendered, those articulations are needed. Consequently, a conversion is needed to transform our representation to a more classical one. The calculation of some intermediate articulations can also be useful for complex constraints solving where knees or elbows are involved.

The conversion is performed according to the type of the kinematic chain associated to the constraint: normalized segments, limbs or the spine. To retrieve the position of a vertebrae, the spline representing the spine is simply discretized according to the distances between the vertebrae. For the limbs of variable length, the position of the intermediate articulations can be obtained analytically. To explain this computation, let us consider how the position of the elbow is retrieved, given that the knee is retrieved the same way. The position of the elbow is placed on the intersection of two spheres (resulting in a circle). The first sphere is centered at the shoulder and its radius is equal to the arm's length. The second sphere is centered at the wrist and its radius is equal to the forearm's length. The position of the elbow is then placed at the intersection of the circle and the half-plane which orientation is stored in the motion file. Moreover, since this half-plane is defined in the shoulder reference frame, the local position of the elbow in this frame (see figure 4) is easily computed:

$$z = \frac{armLgth^2 - forearmLgth^2 + limbLgth^2}{2 * limbLgth}$$

$$y = \sqrt{armLgth^2 - z^2}$$

where z and y are the local coordinates of the elbow in the shoulder frame, $armLgth$ (resp. $forearmLgth$) is the length of the arm (resp. of the forearm) and $limbLgth$ is the current distance between the wrist and the shoulder.

9. Specific constraints solver

To deal with all the constraints, we adapted a Cyclic Coordinate Descent algorithm that takes advantages of our representation. By nature, CCD method converges to a unique solution given that no secondary task can be specified contrary to classical approaches [CB04]. With CCD those secondary tasks are replaced by heuristics in the iterative search algorithm. Nevertheless, the choice of those heuristics is a difficult task. Selecting bad heuristics may engender an inhomogeneous repartition of the deformation along the kinematic chain. A commonly used heuristic, called damping, is to threshold the computed rotations applied to the body segments at each iteration. This method enables to improve the homogeneity of the solution but requires more iterations.

Instead of using a recursive search of the solution such as Badler et al. [BMB87] have done, we propose to subdivide the skeleton into six main groups (see figure 10): the two legs, the head, the two arms and the trunk. Each group G_j (where j is the number of the group) has a set of constraints $CS_j = \{C_k\}$ which is a subset of the complete set of constraints CS imposed by the user: $CS_j \subset CS$. All the constraints contained in CS_j are linked to at least one body segment belonging to the group G_j. For example, if a constraint C_i is acting on the right forearm, it is obviously linked to the right arm group G_a and consequently added to CS_a. If the kinematic chain KC_i associated to C_i also involves the trunk (belonging to the trunk group G_t), the constraint is then added to the set of constraints CS_t associated to G_t.

As the trunk group uses constraints shared with other

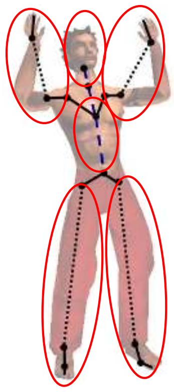

Figure 10: *Six groups used to order the CCD algorithm: the head, the two arms, the two legs and the trunk.*

groups (such as the arms or the head), our constraints solver makes the motion adaptation starts from the legs to the trunk in the following order: the legs, the head, the arms and finally the trunk. The main iteration loop is then:

```
try = 0;
Do
  adaptGroup(RIGHT_LEG)
  adaptGroup(LEFT_LEG)
  adaptGroup(HEAD)
  adaptGroup(RIGHT_ARM)
  adaptGroup(LEFT_ARM)
  adaptGroup(TRUNK)
  adaptRoot()
While ( (try++ < maxTries)
        && (adaptation not completed) )
```

The iteration loop ends when no more adaptation is necessary or when the number of iterations is too high.

For each group, adaptation is performed from the extremity to the root of the kinematic chain. In this adaptation process, the first adapted body segments get larger transformations than the last ones. For the arms example, the algorithm starts from the wrist to the clavicle. For each articulation, either an analytical computation or a rotation is performed. In the following algorithm, CP_i is the current position of the constraint C_i and *Target* is its desired position:

```
For all the articulations A in the group
  If A in limb with variable length
    Analytical computation
  Else
    Apply on A the angle to align
      A->CPi with A->Target
  End of if
End of for
```

For the limbs with variable length, analytical computations are performed, as described in section 7 for the legs. In the analytical computation, the constraint with the highest priority is verified first. The other constraints are then solved while ensuring that the previous constraints with higher priorities are not broken. Part (a) of figure 11 shows the adaptation of a constraint C_1 with a high priority. Since the target is reachable, the constrained point is directly placed to the convenient configuration using an analytical computation. Part (b) shows that the constraint with lower priority is computed using a rotation around the axis that goes through the point constrained with C_1 and the root. It provides a circle of solution in which the point that is the closer to the desired target for constraint C_2 is selected.

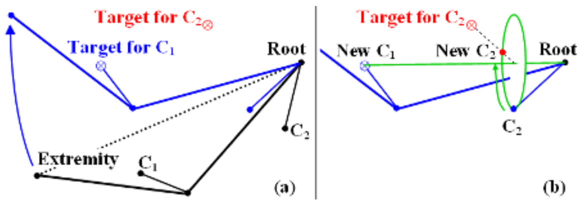

Figure 11: *Analytical solution for two constraints. In part (a), the constraint C_1 with a higher priority is verified. Part (b) is the adaptation of the constraint C_2 of lower priority while preserving C_1 and Root.*

With such a method, only the body segments that are necessary to solve the constraints are used. On the opposite, in the classical CCD methods all the kinematic chain is used. For example, during a grasp, the CCD iterates from the hand to the abdomen in order to verify the constraint. Generally, the constraint is not verified with only one step of the algorithm, the trunk is consequently modified. The resulting motion could be unrealistic because no trunk adaptation is actually required in order to reach targets close to the body. With our method, the arm is moved to the desired constraint in only one step without involving the trunk if only the arm is required. Otherwise, as the arm is displaced to minimize the distance between the current constraint position and the target, the trunk moves with only the minimum required rotation.

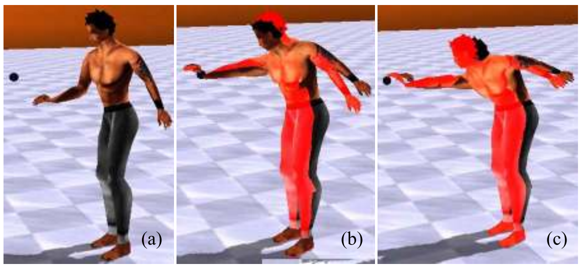

Figure 12: *(a) Original grasping motion. (b) and (c) The character's wrist is constrained while preserving the original COM position. The red character does not preserve the COM. (c) The constraint applied to the wrist cannot be verified.*

When a character is grasping a target that is really far, the arm and the trunk are simply bended in the direction of the target. Since no other limb is used to counterbalance those bending motions, the COM is not placed at its original position. This problem is solved by including the control of the COM in the iteration loop, just after the adaptation of the trunk, inthe function $adaptRoot()$. Hence, at the end of each iteration, the solver translates the root in order to replace the COM to its desired position. This position can simply be the original position of the COM in the original motion or can be obtained using inverse kinetics. Part (a) of figure 12 shows the original grasping motion. In part (b), this motion is adapted in order to verify three constraints while preserving the original COM position: the feet contacts on the ground and a constraint associated to the wrist. The red character, used as a reference, does not perform the control of the COM position while the other one moves its pelvis to counterbalance the upper-body motion. In part (c), the same constraints are used by the target is unreachable.

10. Results

To evaluate our method, we have implemented these algorithms in a more general real-time animation framework [MMKA04]. This framework synchronizes and blends motions in real-time. It also allows to extract the avatar morphology from H-ANIM compliant characters designed in VRML standard format (thanks to Avatar Studio, a product of Canal NuMedia, France). In figure 13 we depict a character that has to deal with four main constraints: two constraints to ensure feet contacts and two constraints to drive the wrist and the elbow of the same arm. The kinematic chain involved in the foot-contact constraints includes only the lower limbs and the pelvis. The constraints on the wrist and the elbow have the same kinematic chain: the corresponding upper-limb and the trunk. The highest priority is given to the wrist constraint whereas a lower one is associated to the elbow. With such priorities, we wish to ensure wrist-contact with the target and, if possible, to verify the constraint on the elbow.

Part (a) of figure 13 represents the original posture without constraint. In part (b), one can see that both the wrist and elbow constraints are verified. In this figure, the targets are represented with red and blue spheres, the yellow sphere locates the position of the COM. In part (c), the wrist reaches its target but the constraint on the elbow is not verified. This result demonstrates task-priority capabilities of our method with very few computation time.

Next, we used eight characters with different morphologies. Those characters are divided into two teams: the red team with red trousers and the white team with white ones (see *fightTraining.avi*). All the characters share the same motion database composed of only three motions: one is for the rest, one is a kick and the last motion allows the avatar to ward off the opponent's actions and then counterattack with a punch. In the sequence of fight training depicted in figure 14, the characters are working in pair. In part (a), white characters kick the red opponents trying to reach the torso (a constraint on the ankle to an absolute position). Concurrently, red characters ward off the kick thanks to the middle of their forearm (a constraint aiming at the opponent's ankle) before performing a punch to their opponents' head (a constraint aiming at the opponent's head or chin). Finally, the white characters bend their trunk and head back in order to avoid the punch, illustrated in part (b) of the figure. This last adaptation is simply done by using a constraint on the head with a target defined as an absolute position. The main complexity of this sequence is that the targets of the constraints change at each time depending on the opponents' gestures and anthropometric data. In order to avoid unwanted gestures at the beginning of the sequence, the constraints are activated after the gesture is initiated. Indeed, as the white character's calf is close to the ground at the beginning of the sequence, activating the constraint sooner would make the red character's forearm follow an unrealistic trajectory, beginning with a down motion before touching the calf at a high position. In the resulting sequence, the COM is constrained to follow the trajectory directly calculated from the captured file. This sequence is calculated in real-time for eight characters at 30Hz (on a P4 2.8GHz with GeForceFX5600) and additional characters could be added before decreasing this frame rate. For example, when the two arms and the trunk are adapted in order to respect two unreachable constraints on the wrists, up to 22 characters can be animated without visualization.

11. Discussion

In this paper, we proposed a representation of motion that simplifies the process of motion retargeting and adaptation in interactive environments where constraints can change at any time. In this representation, the data do not depend on body segments length. Consequently, scaling those data to

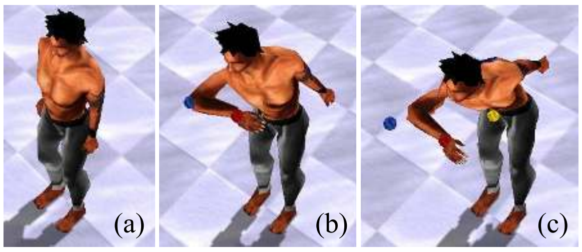

Figure 13: *Control of the wrist (with high priority) and the elbow (with a low priority) positions in order to reach two targets. The yellow sphere is the position of the COM. (a) Original posture without constraint. (b) Posture adaptation if the two targets are reachable. (c) Posture adaptation when only the constraint on the wrist can be verified.*

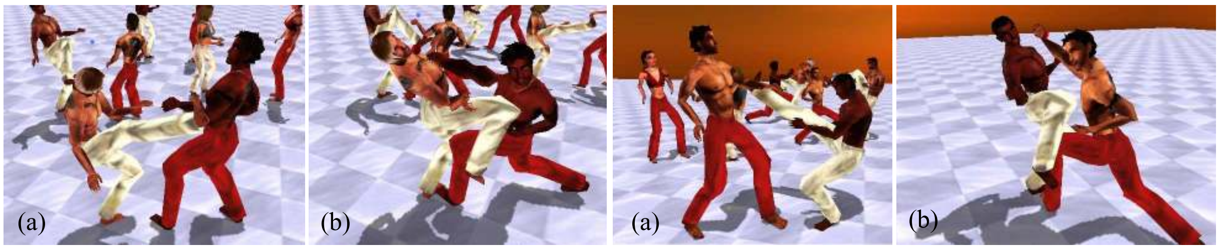

Figure 14: *(a) Red fighters ward off the opponent's kick with the forearm. (b) White fighters avoid the opponent's punch.*

new characters is performed very simply and rapidly. Moreover, the constraints that are intrinsically linked to the motion are also stored ensuring that the main properties of the original motion are preserved (such as foot-contacts). Previous works, such as [MBBT00], proposed to use an intermediate simplified skeleton to perform motion retargeting. The work presented in this paper is based on such an approach but also proposes a global framework that allows to take a better advantage of such a representation by proposing an adaptation of the CCD method.

In our representation of the motion, the root translation is normalized by the leg length which handles most of the motions. It would be interesting to extend this in order to take some specific motions into account. For example, for a handstand, the root translation should be normalized by the arm and trunk lengths.

Although an iterative process is used to adapt the motion at each time step, only few computation time is required to solve many constraints on a complete skeleton. Indeed, instead of modifying all the body segments in an iteration step, we only change groups of body segments. For example, the adaptation of the arm and the forearm orientation becomes the modification of the relative position of the wrist in the shoulder reference frame. Hence, for example, a constraint which is applied on one point of the forearm and which goal is a target in the Cartesian frame, is solved in one step with our method. If the target is not reachable then the arm is placed at the configuration that minimizes the error between the constraint position and the target. Moreover, with this hierarchical process, if the solution is reached by only moving the arm, there is no need to continue the adaptation to other groups of joints (such as moving the clavicle and the trunk). Consequently, this process naturally moves the heavier and proximal segments after the distal ones. This segmental sequence is classically identified in biomechanics for many natural gestures [Put93].

As prioritized inverse kinematics with Jacobian [CB04] does, we can deal with priorities. Consequently, high-priority constraints are verified first (if possible) before trying to verify constraints with lower ones. The user can modify these priorities interactively in order to obtain the desired animation. Finally, it would be interesting to offer the user another level of control of the constraints, making him access to different priorities for each articulation of a unique kinematic chain.

A control of the COM is proposed but is limited to the kinematic chain linked to the activated constraints. In our system, the COM is implicitly associated to the highest priority. Consequently, some other constraints may not be verified if they are not reachable without moving the COM. Future developments will allow to use the COM as the other groups, with its own priority. In the current version of this work, the control of the COM is ensured if a constraint changes the initial posture. Hence, the body segments involved in the control of the COM are limited to those included in the kinematic chain associated to the activated constraint. Consequently, if the constraint only acts on an arm and the trunk, it is not possible to recruit the legs or the other arm to control the COM position. As a perspective, we are working on a new algorithm that would be able to use other body parts in order to completely control the COM, if necessary.

Despite those limitations, our method provides animators with powerful tools for real-time animation of groups of humans in interactive environments. The characters can deal with many geometric constraints and only a small set of captured or edited motions is required. The method presented in this paper was implemented in industrial software environments for applications in multimedia and video games.

Acknowledgments

We would like to gratefully thank Stéphane Donikian for its helpful review of this paper and its suggestions. This work was partially funded by the French research program RIAM within the Semocap project.

References

[BB04] Baerlocher P., Boulic R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *Visual Computer 20*, 6 (2004), 402–417.

[BCHB03] Boulic R., Callennec B. L., Herren M., Bay H.: Motion editing with prioritized constraints. In *Proceedings of RichMedia* (Lausanne, October 2003).

[BH00] BRAND M., HERTZMANN A.: Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192.

[BMB87] BADLER N. I., MANOOCHEHRI K. H., BARAFF D.: Multi-dimensional input techniques and articulated figure positioning by multiple constraints. In *Proceedings of the 1986 workshop on Interactive 3D graphics* (1987), ACM Press, pp. 151–169.

[CB04] CALLENNEC B. L., BOULIC R.: Interactive motion deformation with prioritized constraints. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), ACM Press, pp. 163–171.

[FP03] FANG A. C., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Trans. Graph. 22*, 3 (2003), 417–426.

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM Press, pp. 33–42.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Trans. Graph. 23*, 3 (2004), 522–531.

[GR96] GUO S., ROBERGÉ J.: A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (1996), Springer-Verlag New York, Inc., pp. 95–107.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 473–482.

[Lan98] LANDER J.: Making kine more flexible. *Game Developer Magazine 1* (November 1998), 15–22.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 491–500.

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), ACM Press, pp. 79–87.

[LS99] LEE J., SHIN S. Y.: *A hierarchical approach to interactive motion editing for human-like figures*. Acm press/addison-wesley publishing co., Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999.

[LWS02] LI Y., WANG T., SHUM H.-Y.: Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 465–472.

[MBBT00] MONZANI J.-S., BAERLOCHER P., BOULIC R., THALMANN D.: Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum 19*, 3 (August 2000). ISSN 1067-7055.

[MKMA04] MÉNARDAIS S., KULPA R., MULTON F., ARNALDI B.: Synchronization for dynamic blending of motions. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (August 2004), pp. 325–336.

[MMKA04] MÉNARDAIS S., MULTON F., KULPA R., ARNALDI B.: Motion blending for real-time animation while accounting for the environment. In *Computer Graphics International* (June 2004).

[Put93] PUTNAM C.: Sequential motion of body segments in striking and throwing skills: description and explanation. *Journal of biomechanics 26* (1993), 125–135.

[PW99] POPOVIĆ Z., WITKIN A.: Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 11–20.

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph. 23*, 3 (2004), 514–521.

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Trans. Graph. 20*, 2 (2001), 67–94.

[SP05] SULEJMANPAŠIĆ A., POPOVIĆ J.: Adaptation of performed ballistic motion. *ACM Trans. Graph. 24*, 1 (2005), 165–179.

[TGB00] TOLANI D., GOSWAMI A., BADLER N. I.: Real-time inverse kinematics techniques for anthropomorphic limbs. *Graph. Models Image Process. 62*, 5 (2000), 353–388.

[TySK02] TAK S., YOUNG SONG O., KO H.-S.: Spacetime sweeping: An interactive dynamic constraints solver. In *Proceedings of the Computer Animation* (2002), IEEE Computer Society, p. 261.

[WC91] WANG L.-C. T., CHEN C. C.: A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. On Robotics and Applications 7*, 4 (August 1991), 489–499.

[Wel93] WELMAN C.: *Inverse Kinematics and Geometric Constraints For Articulated Figure Manipulation*. Master's thesis, Simon Fraser University, 1993.