INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Global Framework for Motion Capture

Stéphane Ménardais — Franck Multon — Bruno Arnaldi

N° 4360

Janvier 2001

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4360--FR+ENG

A Global Framework for Motion Capture

Stéphane Ménardais , Franck Multon* , Bruno Arnaldi

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projets Siames

Rapport de recherche n° 4360 — Janvier 2001 — 21 pages

Abstract: Human motion is so complex that model-based animation techniques still do not produce enough realistic trajectories. Consequently techniques based on real trajectories are commonly used in several application fields in order to animate human-like figures. Nevertheless the use of motion capture systems still remains difficult when the movements become complex (such as gymnastic figures) or are adapted to fit geometric constraints. We propose a software system to overcome several of these drawbacks and make it possible to directly apply captured trajectories to virtual actors. This system deals with the loss of data encountered in all optical systems, the various skeletons morphologies and the blending of several motions together in real time.

Key-words: optical motion capture, computer animation, motion adaptation, knowledge-based interpolation, motion blending

* Laboratoire de Physiologie et Biomecanique de l'Exercice Musculaire, Universite de Rennes 2

Une chaîne de traitement pour la capture de mouvement

Résumé : L'utilisation de systèmes d'acquisition optique de mouvements est de plus en plus répandue pour animer des personnages synthétiques. Contrairement aux modèles d'animation, ils offrent l'avantage de produire des séquences très personnalisées mais posent encore à l'heure actuelle un certain nombre de problèmes. Ces problèmes entraînent des temps de post-production assez longs et coûteux. Nous proposons une chaîne de traitement afin de raccourcir ces temps de production en gérant les problèmes d'occlusion, les différences de morphologies entre squelettes et l'animation interactive.

Mots-clés : capture de mouvements optique, animation, adaptation de mouvements, reconstruction de trajectoires acquises, mélange de mouvements

1 Introduction

Nowadays a lot of applications require to visualize realistic synthetic 3D motions. Video games and special effects are the most popular applications that require these high quality motions. For a few years motion capture systems have been widely used to create realistic synthetic motions. The most common systems are generally of two types. Magnetic-based systems are composed of sensors that are associated with their cartesian coordinates and orientations in space. This is widely used in real-time computer animation and especially virtual reality. The main advantage of such a technique is that the computation of the 6 position and orientation parameters can be computed rapidly to animate virtual puppets in real-time. Nevertheless the field of capture is not large enough to deal with motions such as trampoline figures, long-jumps and sporting movements that require large environments. Moreover the subjects are covered with wires and equipments that may change their natural gestures. The most commonly used system in video games and special effects is the optic-based motion capture system composed of cameras and markers. Several commercial and prototypes systems have been proposed but the most common technique is based on infrared sensors. They are composed of stromboscopes that light the scenery at a given frequency. Small reflective markers are placed on anatomical landmarks of the body. The main problem of such systems is linked to the use of cameras. Indeed if an object is placed between the camera and a marker the latter is not viewed. If a marker is not viewed by at least two cameras its 3D-coordinates cannot be retrieved by the system. In commercial systems several interpolation techniques are proposed to guess the position of the missing markers. All these techniques are based on the interpolation of the 3D trajectory without paying attention to other known details such as the shape of the skeleton and the angle limits at each articulation. We propose a technique that deals with the anatomical details and makes it possible to recover most of the missing points with convenient accuracy.

Once the 3D position of the markers are reconstructed several commercial plug-ins and tools are generally used in order to animate a synthetic actor whose inputs are quaternions or Euler angle articulation. The computation of these angular trajectories from 3D Cartesian data requires us to deal with a synthetic skeleton. The use of two skeletons (the real and the synthetic one) involve studing the links between these two structures. Several works focused on the use of such data to animate synthetic creatures.

Molet et al. [12, 13] underlined the problem of anatomical correction which makes it possible to compensate the measurement errors that are cumulated during the capture while using a magnetic-based motion capture system. With optical-based systems the problem is different because the errors occur at various levels of measurement: definition of the camera, 3D reconstruction, position of markers compared to the location of the bone... Some work has been carried-out in order to compute the motion of the bones according to the markers [2]. This work consists of searching for the articulation center and the corrected placement of the body segment according to the position of the markers.

Other authors proposed algorithms in order to adapt a motion to a skeleton that has a different morphology [1]. This technique is necessary in order to ensure that the animation is collision-free and that the motion is reproduced with the same geometric constraints (such

as ensuring foot contact with the ground). Nevertheless one can wonder if a real subject with the synthetic actor's anatomy would have produced these corrected trajectories. The algorithm is never fully automatic and users have to give constraints to the adaptation process to ensure that the main caracteristics of the motion are respected.

For most computer animation applications the captured motion is modified in order to create a variety of specific animations that may take the synthetic environment into account. For instance when interaction between two synthetic actors is required movements have to be tuned in order to ensure contact between them. To this end, two main families of techniques have been introduced in the last few years: motion blending and motion warping. Motion blending requires a database of motions (described either in the frequency or temporal domain) and consists of interpolating between their parameters in order to produce new trajectories. These parameters can be either linked to a temporal [9, 15, 17, 17, 14] or a frequential [16] representation of the motion. The main drawback of this kind of technique is that it generally requires us to deal with a large database of pre-recorded motions. Indeed the size of this database generally increases proportionally to the number of parameters and the complexity of the motion. Moreover motion blending doesn't eliminate sliding effects or unwanted collisions.

We call motion warping all techniques that take well-known trajectories (described by key-frames or captured data) and modify them in order to obtain new motions. To this end, two main groups of approach are considered: signal analysis in the temporal or in the frequency domain. Witkin and Popovic [18] modified a reference trajectory $\theta_i(t)$ (where i represents the i^{th} parameter of the system) by interactively tuning the position of selected key-frames and by scaling and shifting $\theta_i(t)$. Ko and Badler [10] also introduced a method to modify a reference motion recorded as sequences of key-frames. By changing the morphology of the synthetic figure, new parameters are automatically computed to make the original and the new characters move the same way. Instead of studying trajectories in the temporal domain Bruderlin and Williams [3] applied image and signal processing in the frequency domain to re-use, modify and adapt animated human motion. The goal is to make libraries of animated motion with high level motion editing at interactive speeds. To this end multi-resolution filtering of angular trajectories have been used to define a reduced set of motion parameters. To produce a new motion the signal is reconstructed differently by tuning weights associated with each parameter. This method is not really suitable for tuning a motion to verify a point-to-point constraint. Consequently it involves a large database of prerecorded motions that make it possible to tune a motion that is not so far from the solution. It often remains difficult to obtain the desired motion and requires a long trial and evaluation process.

Another promising technique consists of using spacetime constraints in order to adapt the motion to another skeleton [7] or to change the captured motion [8]. This technique has been successfully applied to several kinds of motions but requires lots of computations that are not compatible with real-time animation for complex figures (large number of degrees of freedom). The main drawback of these techniques is that they do not take dynamics into account and may produce unrealistic sequences. Some new approaches propose modi-

fying motions with dynamic simulation [19]. This makes it possible to take changes in the environment or in the actor's anatomy itself into account. Nevertheless it requires lots of computations and implies to proposing a mechanical model that must be adapted to each creature we wish to deal with.

We propose a global framework to simplify the use of motion capture for computer animation. We first focus on the process dealing with captured trajectories to compute final angular trajectories required to animate a H-ANIM compliant VRML model[1]. The motion capture files exhibit occlusions, noise and do not have any angular information. Our system automatically computes the angles for each degree of freedom without missing data and noise. It also adapts the motion to respect new skeleton morphologies and compress the final motion in order to minimize data space. The motions can be blended and synchronized together in an interactive simulation.

This paper is organized as follows. Section 2 presents an overview of the framework. Section 3 describes the algorithms developed in order to fill in the missing points, adapt and compress the motion. Section 4 deals with the realtime animation system based on these motions and gives somes results.

2 Overview

2.1 Outputs of motion capture systems

First of all, let us consider the main problems of an optical-based motion capture system. We use a Vicon370[2] system made-up of seven 60Hz infrared cameras. Such a system is composed of markers that are placed on the anatomical landmarks. As for all optical-based systems,the position of these markers in space is measured by placing a set of cameras in the scenery. Infrared cameras are equiped with an infrared stromboscop that lights the scenery. The goal is to capture the motion of a subject by filming the displacement of the markers. At least two cameras are required in order to retrieve the 3D position of such markers by using stereoscopic techniques such as the DLT algorithm requiring a calibration. Figure 1 depicts such a system and illustrates the principle of 3D reconstruction. In this picture one can see the images provided by the two cameras. Highlighted pixels correspond to a marker that is viewed by the camera. All the other pixels are black. As the sun and many lights contain infrared rays the camera currently capture so-called "ghosts" (points that do not correspond to markers).

The nature of this kind of system implies dealing with the occlusions. Hidden markers are not reconstructed and the resulting 3D trajectory exhibits holes, as depicted in figure 1. Of course, before animating a human-like figure one has to retrieve these data.

Once the 3D trajectories are reconstructed for each marker the resulting data is a set of unlabeled trajectories with holes and noise. The task of the user is then to label all the trajectories by associating a name with each marker. This task makes it possible to recover

[1]for more information on H-ANIM please consult *http://h-anim.org/*.
[2]Vicon370 is a product of Oxford Metrics *http://www.vicon.com*.

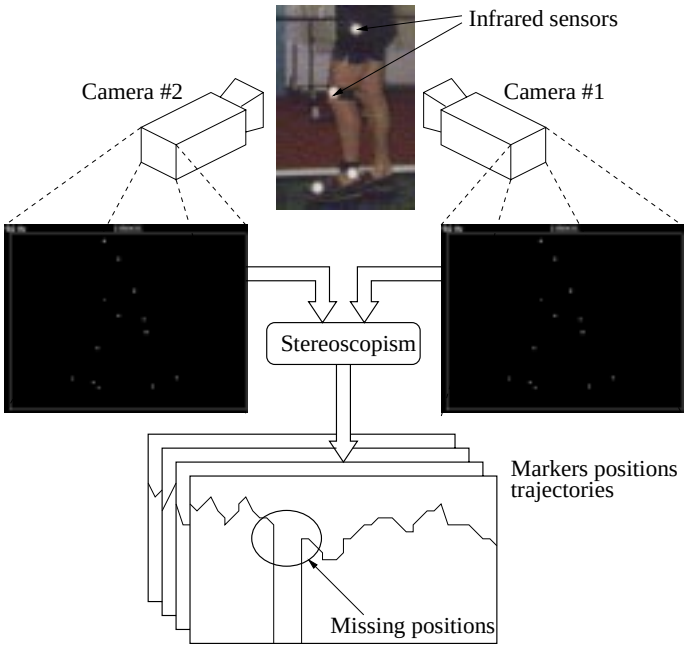

Figure 1: Description of an optical motion capture system.

the skeleton of the subjects by measuring the average distance between the markers (such as the distance between the knee and the ankle to obtain the length of the leg). This last value can be used as an estimation of the error computed during the capture. Indeed as the markers are placed on the skin the distance between those placed on the same rigid body generally vary. The larger the variations the worse the markers placement quality.

2.2 Dealing with human bodies

The goal of this framework is to animate a human-like figure according to motion capture. Several problems occur and especially the difference of morphology between real and synthetic skeletons. The first step is consequently to define a structure for each of them:

- The *general structure* that describes the hierarchy of frames (each frame is associated with a body segment) and the sets of markers that are supposed to be placed on it. This description is available for every kind of human subject and defines exactly the available degrees of freedom.
- The *origin structure* that contains the average distances between markers of a same rigid body and other information helpful for 3D rescontruction. This description is specific to the subject that performed the motion.

- The *animation structure* composed of the distance between the articulation and the orientation of the synthetic actor's frames used for computer animation.

Once these structures are defined, one has to design algorithms to make the origin and the animation structures correspond to a given motion.

2.3 From reconstruction to animation

Our framework proposes a tool to export fully-processed and adapted motions from captured trajectories. To this end we provide the user with an interface that gathers all the needed functions (see figure 2). First it allows him to load a motion capture file (*c3d* files in the Vicon370 system) and a *general structure*. The user at this level can only see the markers' positions, body segments and *c3d* internal data. Once an *origin structure* is loaded, the user can run the reconstruction process in order to recover missing markers positions and avoid marker inversion. The next step is available once an *animation structure* is loaded. The final morphology is known at this step and one can adapt the motion to the synthetic figure. The user can specify constraints (like which markers positions to maintain) at this step to help the adaptation process. The resulting motion can be filtered, cyclified and processed in order to make the main displacement follow a given direction. Before saving the motion, the user can add specific constraints such as foot contact with the ground. The saved trajectories are encoded in order to minimize the number of keyframes while tuning a precision threshold that controls the compression quality.

Once several motions are encoded, the system allow the user to replay them together while changing the position and orientation of the synthetic figure. It also ensures the synchronization of motions thanks to constraints (such as foot prints).

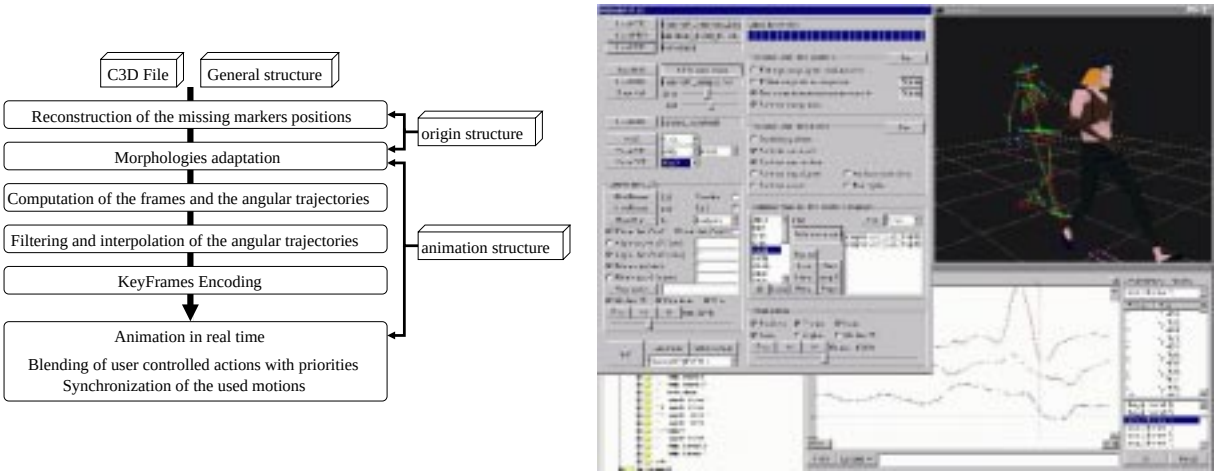

Figure 2: The framework main steps and the corresponding user interface

3 Dealing with the data

This section deals with the methods used in order to recover missing points and apply constraints to the skeleton.

3.1 Reconstruction of the missing points

Recovering the missing points is performed by a 3-step process:

First step: naïve interpolation and distance estimations This step deals with a first approximation of the missing points without taking the morphology into account. In addition, it evaluates how the distances change depending on time for all couples of neighbors.

Second step: association of uncertainty values and reconstruction ordering It consists of preparing the data in order to compute the missing points. For example, it computes the reconstruction order that optimizes the interpolation quality.

Third step: computation of the missing points Depending on the number of neighbors and the distances evaluated in the first step, it recovers the positions of the missing points.

3.1.1 First step: naïve interpolation and distances estimation

First a naïve interpolation using cubic splines is performed in order to eliminate holes in the trajectories. This interpolation does not take the skeleton structure into account but ensures continuity in the motion. Nevertheless using cubic splines implies constant accelerations during periods of time while actual accelerations should be different. The resulting trajectories do not exhibit holes but may not respect the structure of the skeleton.

In order to reconstruct the missing points, we need to estimate the supposed constant distances between neighbors. These distances are not really constant due to noise factors like clothes, skin or muscle movements. So, if we want to have an accurate reconstruction we need to take these variations into account. It's the reason why for each rigid link we compute the segments length for each frame and consider it as a continuous function.

Depending on the missing data interval length, the naïve interpolation may be far from the average length given by the *original structure*. So we make the curve interpolation tend towards the average length if the missing data interval is too long. Furthermore we associate with each estimated distance an uncertainty coefficient depending on the current range between the interpolation curve value and the average length. This coefficient is used in the next step that consist of finding the best order to reconstruct the missing points for each frame.

3.1.2 Second step: association of uncertainty values and reconstruction ordering

Let A be the set of markers (considered as nodes) that compose the skeleton. Let $B \subset A$ be the set of nodes that have to be reconstructed (corresponding to missing points). The first step of the algorithm consists of associating an uncertainty value (used as a weight) I_i to each element $a_i \in A$. If $a_i \notin B$ then the marker is present and I_i is set to its residual factor included in the primary motion capture file. This residual factor is computed at each frame by the motion capture system and stands for the accuracy of the measurement of a_i. The weight I_i is computed by scaling and borning the residual factor in the interval $[0, 1]$. We divide it by a maximum acceptable value. The weight is normalized o that 1 stands for the least accurate measurement. For the other nodes (missing points) I_i is obtained by a weighted function of a_i's neighbors and neighborhood proprieties. This structure is described in a non-oriented graph.

This global graph G describes the *general structure* and is computed as follows. A file containing all the rigid links between the markers placed on the skeleton is loaded and examined. Each rigid link is also represented by a link in G. The nodes of G are markers placed on the subject. The links of G correspond to constraints that stand for a constant distance between two nodes. During the capture some of the markers may be lost. Hence the graph contains two kinds of markers: the present and the missing ones. Let represent graphically the missing markers by triangles and present markers by squares. In addition each node is associated with its weight I_i.

Let I_i be the weight of the missing marker a_i computed thanks to the nodes included in set M_i with $M_i \subset A$. Each element of M_i is either present or missing. M_i stands for the set of a_i's neighbors). Hence if $a_j \in M_i$ is missing its weight is computed thanks to another set of nodes M_j (containing neighbors) and so on recursively until there is no new missing node added into the graph. For a given node a_i their is a specific graph G_i that is built thanks to a_i and its recursively associated nodes $\{a_j\},\ j \neq i$ as described above. For example let consider five nodes $\{a_1, a_2, a_3, a_4, a_5\}$ where a_1 and a_4 are missing. The graph G_1 is depicted in figure 3: a_1 is linked to each other node and a_4 is also linked to a_3 and a_5. In this example, a_1 is associated to a weight I_1:

$$I_1 = f(I_2, I_3, I_4)$$

where f is a function that computes a weight and $I_4 = f(I_3, I_5)$. Hence

$$I_1 = f(I_2, I_3, f(I_3, I_5))$$

f is a $[0-1]^n$ to $[0-1]$ function that computes a weight thanks to the set of the neighbor's associated weights. The resulting value accounts for the weight of all the neighbors and all the estimated distances that can be used to reconstruct the missing point. The more confident the neighbors (two or three if possible) and distances (little missing data) the more confident the results. This means that the weight associated with a reconstructed marker a_i depends on the maximum weight of all the used neighbors. If a marker is missing

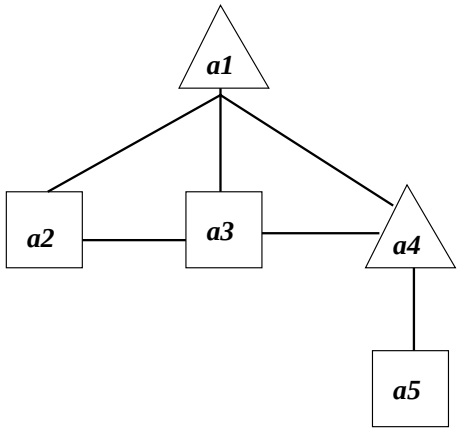

Figure 3: Example of graph with five nodes.

and has no neighbor its weight is set to 1, the maximum possible value. Hence a possible function for f is the max function. In the example of figure 3 the weight I_1 of a_1 is equal to $max(I_2, I_3, max(I_3, I_5)) = max(I_2, I_3, I_5)$. Actually we use a weighted max function that depends on the number of neighbors used for the reconstruction. If two neighbors are considered, the weight is set to a higher value than for three neighbors.

As a first step all the weights are computed for each node and the nodes are sorted in the increasing order of their weight. This makes it possible to reconstruct the missing points from the most confident one to the least confident one. If a_i is more confident than a_j it is reconstructed before a_j. Hence a_j that may use a_i to be reconstructed is obtained with neighbors that are the most accurate possible.

3.1.3 Third step: computation of the missing points

Three neighbors at least are required to reconstruct with accurately a missing point. The set of neighbors $\{a_1, \ldots, a_n\}$ $n \geq 3$ defines a frame in which the missing point a_m has constant coordinates in time (x, y, z). Nevertheless because of inaccuracies during the measurements these supposed constant coordinates change slightly. Consequently we have introduced a cost function g that has to be minimized. For a given missing point a_m, we compute the actual distance to each neighbor a_i. g returns the sum of the differences between these last values and the average distances l_i. Consequently, the minimum of this function is the position a_m that best verifies the estimated distance with each of its neighbors and therefore the structure of the skeleton.

$$g(a_m) = \sum_{i=1}^{n} (dist(a_i, a_m) - l_i)^2 \tag{1}$$

where a_m is the missing point, l_i is the supposed-constant distance between a_m and a_i (the length of the body segment) and n the number of neighbors used for the reconstruction of a_m. The initial value of m at time t is given by the naive interpolated trajectory $\Psi_m(t)$. In order to minimize this function we use the steepest descent algorithm with a variable step.

If only two neighbors are known let us consider two spheres S_1 and S_2. S_i is the sphere whose center is a_i and whose radius is l_i. For example, let us consider that the marker a_m placed on the elbow is missing. Let a_1 be the shoulder and a_2 be the wrist. Due to the structure of the skeleton, the distance between a_m and a_1 (resp. a_2) is equal to l_1 (resp. l_2). So a_m is placed on the intersection of these two spheres. This intersection is a circle in space. The position of a_m is then supposed to be placed on this circle, the closest to the naive

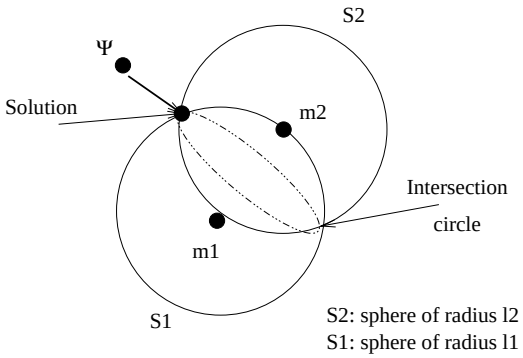

Figure 4: Reconstruction thanks to two neighbors.

interpolated trajectory $\Psi_m(t)$ as possible. Due to imprecision during the measurements this intersection may be empty: the two markers a_1 and a_2 are too far from each other. In that case we consider that the articulation is extended to its maximum value. Consequently the two neighbors and the missing point are supposed to be placed on a straight line. The position of a_m on this line depends on l_1 and l_2.

If only one neighbor is known, then a_m is not reconstructed at this step. Indeed there is not enough data to reconstruct correctly this point and we suppose that such an interpolation would create wide errors that would depreciate the quality of the reconstruction. Hence it is with the next step (reconstruction of the angular trajectories thanks to the position of the nodes) that we deal with these last missing points. It is more convenient to interpolate the angular trajectories in order to produce good-looking movements instead of interpolating the points in space that may cause discontinuities and bad-looking sequences.

3.2 Morphologies and adding constraints

Once the positions of the markers are available, we can compute the angular trajectories. However, we need to examine the differences between the morphologies of the *original structure* and the *animation structure*. Moreover the movement generally contains constraints

such as ensuring contact at foot prints and when handing something. Our system provides two methods for the skeletal adaptation. The user can scale the human-like figure to fit the real's actor morphology. So the *original structure* and *animation structure* become equal. This solution keeps the original dynamics of the motion which is important for precise applications such as biomechanics. The second solution doesn't change the final *animation structure* and let the user choose what kind of constraints are needed. First of all, the system can scale all the original marker positions to obtain a human height equal to that of the *animation structure*. This method makes it possible to avoid asking a child model to walk in giant's footprints. Once the *original structure* (walking in giant's footprints) is adapted to the global size of the *animation structure* (a child) by scaling the Cartesian trajectories a second step can be carried-out. This last step consists of adapting the resulting trajectories in order to respect constraints such as ensuring foot contact with the ground.

Hence the user can choose which extremities need to be constrained. For the child, it consists of dividing the skeleton into six independent sub-skeletons (two arms, two legs, the body and the head). As not only the size, but also the proportions of the limbs may be different between the child and the giants, inverse kinematics are used to verify the constraints. This method is applied to each sub-skeleton independently.

In order to divide the skeleton into six independent parts the following algorithm is used. Let us consider *mshoulder* the name of the joint node between the neck and the two shoulders (as depicted in figure 5). We first evaluate for each frame the new positions of the root and *mshoulder*. Let us consider the example of the giant and the child. For the giant (resp. child), the distance between the root and the feet at rest is l_o (resp. l_a). As depicted in figure 5, we compute a ratio $\frac{l_a}{l_o}$ used in order to estimate for each frame t $l_a(t)$ (desired distance between the root and the feet for the child at frame t) according to $l_o(t)$ (same distance for the giant at frame t):

$$l_a(t) = l_o(t) * \frac{l_a}{l_o}$$

As a result the feet position and the ideal distance to the root are known for each frame. This process is repeated for each sub-skeleton. The result may produce contradictory root positions. Consequently we have to compute the root position that best fits the length constraints for all the sub-skeletons by using optimization. The optimization process takes the positions of the original movement into account. Next the leg kinematics (and the same for all the other sub-skeletons) are obtained by applying inverse kinematics.

3.3 Synchronization of captured motions for realtime animation

The end of the framework deals with the realtime animation of human-like figures using motion capture. The user can gather and blend several actions such as locomotion, vision targeting, object handing, motion replay... Each action is linked to an independent module with its own variables. In addition, for each node, the user can define priorities that can be tuned during the animation. This priority can be specified for each node and is not general

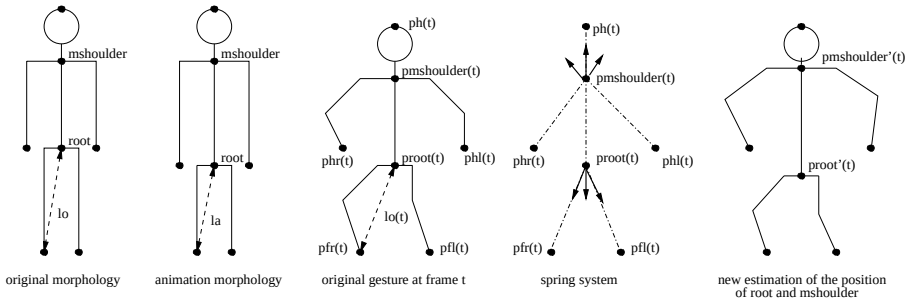

Figure 5: The spring-like system

to the action. Hence during locomotion one can assign a high priority to the knee and a lower priority to the hip. In this example, the high priority assumes that the knee trajectory is more important to preserve than the hip one. Consequently the locomotion has a greater probability than other actions (especially with higher priorities on the hip) to take control of this node. On the contrary, for the knee, the locomotion should be the main action and should be better preserved. The actions can be interactively started and ended by the user. For example, human locomotion gives naturally high priority to the legs and low priority to the remainder of the body. While walking or running the human-like figure can also take objects that engender higher priorities to the arms. Therefore taking an object will progressively take control of the arms by automatically increasing the corresponding action priority. A special action is always performed even if no action is specified by the user. This action consists of animating the human-like figure at rest with a set of predefined motions.

Another value is used to control each action. This value is a normalized coefficient ranging from 0 to 1 and stands for the state of the action. It differs from the priority because it globally acts on the action without having a specific control over the nodes. 1 stands for an action that is fully activated while 0 stands for an unactivated one. The use of such a coefficient makes it possible to smooth the transition between one motion and another by tuning their corresponding coefficient. If the coefficient increases (resp. decreases) the action starts (resp. stops). When the user launches an action, he only specifies the beginning of the action, and the system automatically increases the coefficient.

When an action is started, we first estimate if the action can be carried out with the other actions. In the current version, our system only deals with feet sequencing constraints. For example a single-support phase followed by a double support phase while walking or a single-support phase followed by a non-contact phase while running. As each action is linked to a captured motion the feet sequence is intrinsically defined in the motion capture file. When using only one action, we are sure that the action can be launched. When blending several actions this is not true. Consequently before launching a new action the system verifies if it can be synchronized to the current ones. For example, the system will

not launch immediately a run beginning with a left foot-strike if a right foot-strike has just been detected. It will wait for the next left foot-strike.

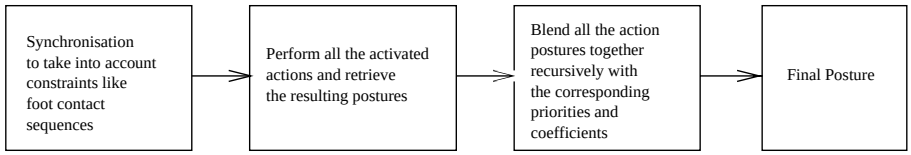

Figure 6: The animation steps

Consequently the priorities and the coefficients are used together to blend postures given by the activated actions. For each node the blending algorithm is a recursive process that computes the quaternion $\mathcal{Q}$ applied to each articulation. Let us consider that $n+1$ actions are activated. Let $h_0 \dots h_n$ be the $n+1$ priorities of these actions for a specific node sorted in the decreasing order. The blending algorithm consists of stating that $\mathcal{Q}$ mainly depends on the quaternion $\mathcal{Q}_{h_0}$ (corresponding to the action with the highest priority) and the quaternion resulting from the remainder actions $\mathcal{Q}_{h_1 \dots h_n}$.

$$\mathcal{Q}_{h_i \dots h_n} = blend(\mathcal{Q}_{h_i}, \mathcal{Q}_{h_{i+1} \dots h_n}) \tag{2}$$

where *blend* is a function that performs a weighted sum of its parameters. The weight for each quaternion is a function of the actions' priority and coefficient. Once the quaternions for each node are computed in this way the algorithm takes the external constraints (such as feet contact) into account. To this end we use the method described in section 3.2. At the end of the process, the final posture is sent to the 3d engine to animate the virtual actor.

4 Results

Our results mainly demonstrate that such a system makes it possible to decrease the cost of using motion capture to produce realistic interactive animations. To this end we first focus on the new capabilities brought by our main contributions: reconstruction of missing points, adaptation to every kind of human-like skeleton and blending/merging of captured motions with dynamics priorities that ensure synchronization.

To better appreciate the quality of the reconstruction we suppressed artificially some points and compared the real trajectories to the reconstructed ones. The following table describes the mean error and the maximum error reached for several motions: running, vertical jumps and somersaults. These motions represent three levels of difficulty for the reconstruction algorithm. Indeed, with an optical system, many factors like the camera position or the accelerations of the body segments strongly depend on the kind of motion. The three kinds of motion chosen represent three distinct levels of difficulties for the reconstruction algorithm. A run on an endless belt requires a shorter space than a trampoline to perform motion capture. So the camera don't need to be very distant for a simple run. Moreover a

run and vertical jumps produce lower accelerations than a somersault and therefore less high frequencies and occlusions. For each motion we have deleted consecutively 6, 30, 60, 120 and 180 points (at 60Hz it corresponds to 0.1s, 0.5s, 1.0s, 2.0s and 3.0s) and estimated the error introduced by our technique. The deleted point was placed on the pelvis and produces more than three neighbors.

Gesture (duration of deletion)	Mean error	Maximum error
Trampoline somersault (0.1s)	2.7 mm	3.3 mm
Trampoline somersault (0.5s)	4.7 mm	7.6 mm
Trampoline somersault (1.0s)	4.9 mm	10.61 mm
Trampoline somersault (2.0s)	5.3 mm	14.1 mm
Trampoline somersault (3.0s)	6.1 mm	23.9 mm
Trampoline vertical jumps (0.1s)	1.7 mm	2.8 mm
Trampoline vertical jumps (0.5s)	5.4 mm	10.3 mm
Trampoline vertical jumps (1.0s)	6.4 mm	13.1 mm
Trampoline vertical jumps (2.0s)	6.0 mm	13.1 mm
Trampoline vertical jumps (3.0s)	5.58 mm	12.8 mm
Run (0.1s)	1.5 mm	2.1 mm
Run (0.5s)	3.1 mm	4.6 mm
Run (1.0s)	3.8 mm	5.9 mm
Run (2.0s)	4.74 mm	7.2 mm
Run (3.0s)	4.9 mm	7.8 mm

Table 1: Average and maximum error for several tests.

In this table one can see that the mean error sometimes decreases when the number of missing points increases. This phenomenon is due to the fact that at a tiny period of time the motion is very complex and makes it difficult to interpolate the missing point correctly. As the number of points increases the influence of this tiny critical period of time is lowered compared to better approximations calculated in its neighborhood.

We have also tested the motion adaptation to several morphologies. If the two morphologies are too distinct, some problems can occur. Let us consider a very little human-like figure. For some motions, he would be unable to reach the extremities such as the foot prints of tall people even if the adaptation algorithm is efficient. Consequently it is up to the user to decide to scale the *original structure* to fit as best as possible the *animation structure*.

The animation system offers useful functionalities for the animator such as automatic motions synchronization in real-time. However this system is intended to be used for a high number of avatars together. Consequently we evaluated the time-cost of our system during a 60Hz animation. First let us consider the theoretical complexity of our algorithm. This complexity is somehow linear depending on the number of blended motions. Practically we measured the amount of time spent by the system to synchronize the motions involved in figure 7. On a PentiumIII 450 this required time was evaluated to approximately 1% of the

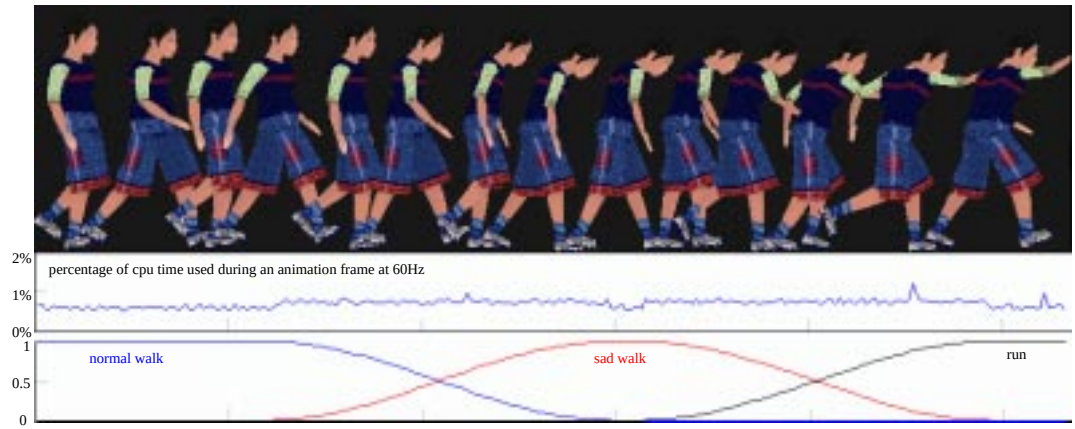

Figure 7: Example of motion blending and synchronization.

animation rate $\frac{1}{60}$ second. In the bottom of figure 7 one can see this percentage of time depending on time.

Our tools have been used for several applications (see figure 8 for an illustration of two applications). The reconstruction algorithm has been widely used in biomechanics for the processing of complex motions such as handball throws, trampoline figures, athletics, gymnastics... This tool is currently used in the Laboratory of Biomechanics placed at Rennes. As a result, several studies can be carried out whithout which they should not be considered (too many occlusions).

The motion adaptation has been efficiently used in order to animate several characters with various morphologies without carrying-out lot of motion capture experiments. For example, the characters have to perform common tasks that are linked to only one motion capture file (such as locomotion, sitting down...). Moreover this technique enables the animation of virtual characters with motion captured from real actors with a different morphology. In other systems this task is also possible but the final motions have to be changed manually in order to ensure foot contact and avoid collisions. In our system we take the feet contact into account and consequently ensure that the virtual actor moves over the ground. This tool has been used in several applications such as:

- crowed simulation with autonomous actors guided by customized behavior [5],
- the visit of a museum in collaboration with the French City of Science in Paris where 15 different characters were animated this way,
- the control of human-like figures according to visual constraints [4],
- the development of prototypes for behavioral animation in collaboration with PhD students [11, 5],

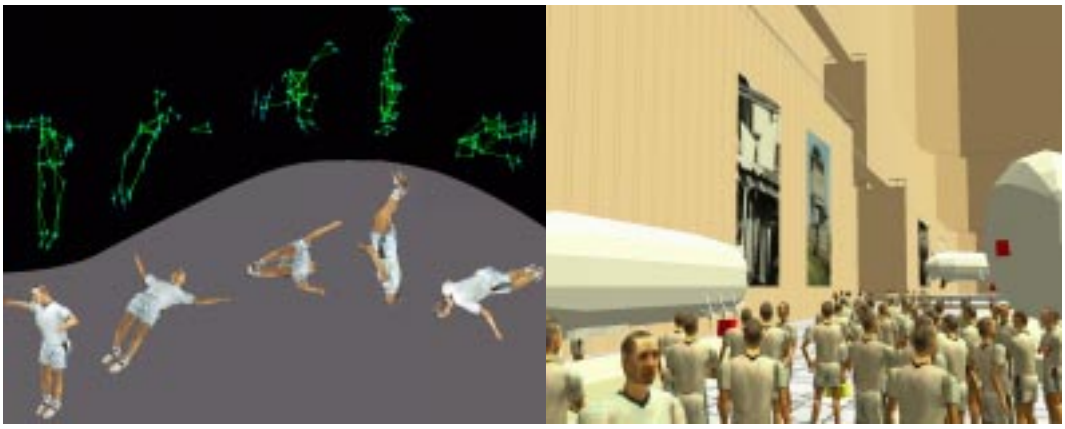

Figure 8: Two applications: use of motion reconstruction (here complex trampoline motions) in biomechanics and virtual visit of a museum by autonomous agents guided by behavioral models.

- and other demonstrations and prototypes on a virtual reality platform in the SIAMES project.

For all of these applications involving several different characters and motions such a technique strongly decreases the time spent by animators. For an animator the work was organized as follows:

1. get one captured motion for the main gestures involved in the scenario (walking, running, grasping...),
2. extract the *original structure* of the real subject and get a first virtual character to load an *animation structure*,
3. specify the desired constraints such as foot contact,
4. isolate a part of the captured motion that best responds to the scenario wishes,
5. eventually modify the motion if required,
6. and so on for the other virtual actors.

In order to perform this process we provide the user with a specific interface depicted in figure 2. For each new *animation structure* with the same motion, one just has to load the virtual actor. All the other parameters are stored to simplify the task of the user. Consequently for the other virtual actors the motion is automatically adapted without any manual help. Let us consider the example of the visit of a museum. 15 characters were used and have to move in the museum and each of therm can perform specific tasks. Thanks to

our technique only 35 motions have been captured including: walking forwards, backwards, walking sideways in both directions, running in the same conditions.... If no adaptation is possible we would have to consider 15 times the number of motions. As each motion capture file has to be processed (point labelling and interpolation) such a tool enables a shorter production time.

The animation part has been tested on several animations like a virtual museum (see figure 8) with interactive actors. This application which use a lots of human actors together has shown the advantage in modeling and adapting motion capture. Only one cycle of different walk was necessary to obtain various kind of walk and transitions in real-time. The figure 7 describes a transition beetween a simple walk, a sad one and a run with the hand forward. Moreover, with a cycle of sideways walk and one of backwards walk the user can make the virtual actor walk in all directions (not only the four cardinal directions). This kind of possibility is very useful for real-time avatar collision avoidance. Furthermore, some low complexity behavior algorithms have shown that independent actions with customizable priorities are well suited for human motion control.

5 Conclusion

We have presented a new tool to directly and automatically use motion capture in computer animation. This tool enables the user to interpolate missing points with convenient accuracy.

Compared to commercial tools such as Maya and Kaydara Filmbox, our system is capable of interpolating trajectories while ensuring anatomical constraints. It also enables to adapt the motion to a different skeleton while respecting the position or orientation constraints over some particular points. Whereas Molet et al. [12, 13] used anatomical corrections for magnetic-based motion capture we propose a new method for optic-based systems that are widely used in entertainment and biomechanics. This kind of system implies new constraints such as dealing with occlusions. The main contribution of our system is to gather new methods for motion capture and adaptation in a unique tool. This tool was conceived for animators in order to improve motion design efficiency.

It is also designed for biomechanics where motion capture is one of the major measurements. This tool has been used by a biomechanics laboratory in Rennes in order to process very complex motions with a large number of occlusions. For example, we experimented this tool with trampoline figures that are performed in wide environments, with complex postures where parts of the body are totally hidden and that exhibit very high accelerations (up to 5 times the acceleration of gravity). The results are very promising and enable the use of optical-based motion capture for such complex motions. Without this tool it would not have been possible to analyze these motions. The tool has also be used successfully in handball throwing, running, gymnastics, soccer.

As stated in the results section this tool has also been successfully used in the production of computer animations in the GASP environment [6]. This environment embeds some of our tool functionalities such as real-time motion warping while preserving geometric constraints

and motion blending. This computer animation engine is now used for behavioral animations in our laboratory. Hence an animator can:

- either tune a motion off-line until it corresponds to his wishes,
- or produce rapidly an approximate motion that can be adapted in real-time during computer animation.

These results have been obtained with a human-like figure but the same software could be used in the same way in order to animate other animals and vehicles. To this end the user just has to specify the new markers hierarchy and the links between them. Of course, our system has some limitations. First, even an efficient interpolation technique can only produce interpolated trajectories that may differ from the real ones. Some occlusions may be due to very complex motions with rapid rotations and change of gesture. In that case, our system cannot recover trajectories if no other information is provided. Moreover constraints applied to the skeleton may act in the opposite way one from another. Let us consider two distant goals that have to be reached by two hands. In that case one has to choose if the goals have to be exactly reached even if the original motion is modified or the contrary. The choice of one or the other solution could engender unwanted postures that are very difficult to control with such methods.

Future works will tend to add knowledge to help the users to design animations from motion capture. In that case, the process of computer animation from motion capture should be totally reviewed. We also may develop new techniques for motion editing that enable animators to be more efficient when dealing with motion capture.

Acknowledgments

The authors thank the French Minister of Sport and Youth for its financial support for this project. Thanks also to the French Federations that have cooperated with the design of such a tool. The geometric H-ANIM compliant models were designed by Dominique Favotti and the Cité des Sciences in Paris.

References

[1] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture: Dealing with the data. In *Eurographics Workshop on Computer Animation and Simulation*, pages 3–18, September 1997.

[2] R. Boulic, P. Fua, L. Herda, M. Silaghi, JS. Monzani, L. Nedel, and D. Thalmann. An anatomix human body for motion capture. In *Proc. of EMMSEC'98*, Bordeaux, France, September 1998.

[3] A. Bruderlin and L. Williams. Motion signal-processing. *Proceedings of ACM SIGGRAPH'95*, pages 97–108, August 1995. Los Angeles, California.

[4] N. Courty and E. Marchand. Computer animation: a new application for image-based visual servoing. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 2001. IEEE; Piscataway, NJ, USA.

[5] F. Devillers. *Langage de scenario pour des acteurs semi-autonomes.* PhD thesis, Université de Rennes I, September 2001.

[6] S. Donikian, A. Chauffaut, T. Duval, and R. Kulpa. Gasp: from modular programming to distributed execution. In *Proceedings of Computer Animation 1998*, Philadelphie, USA, June 1998.

[7] M. Gleicher. Retargetting motion to new characters. In *Proc. of ACM SIGGRAPH*, pages 33–42, July 1998.

[8] M. Gleicher and P. Litwinowicz. Constraint-based motion adaptation. *Journal of Visualization and Computer Animation*, 9(2):65–94, 1998.

[9] S. Guo and J. Roberg. A high-level mechanism for human locomotion based on parametric frame space interpolation. *Computer Animation and Simulation'96*, pages 95–107, 1996.

[10] H. Ko and N.I. Badler. Straight line walking animation based on kinematic generalization that preserves the original characteristics. In *Graphics Interface*, pages 9–16, Toronto, Ontario, Canada, May 1993.

[11] F. Lamarche and S. Donikian. The orchestration of behaviours using resources and priority level. In N. Magnenat-Thalmann and D. Thalmann, editors, *Proceedings of the Eurographics Workshop on Computer Animation and Simulation (EGCAS-01)*, pages 171–182, Wien, Austria, September 2001. Spinger-Verlag.

[12] T. Molet, R. Boulic, and D. Thalmann. A real time anatomical converter for human motion capture. In *Eurographics Workshop on Computer Animation and Simulation*, pages 79–94, September 1996.

[13] T. Molet, R. Boulic, and D. Thalmann. Human motion capture driven by orientation measurements. *Presence*, 8(2):187–203, 1999.

[14] C. Rose, M. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[15] C. Rose, B. Guenter, B. Bodenheimer, and MF. Cohen. Efficient generation of motion transitions using spacetime constraints. *Proceedings of ACM SIGGRAPH'96*, pages 147–154, August 1996.

[16] M. Unuma, K. Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. *Proceedings of ACM SIGGRAPH'95*, pages 91–96, August 1995. Los Angeles, California.

[17] D.J. Wiley and J.K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Application*, 17(6), November 1997.

[18] A. Witkin and Z. Popovi'c. Motion warping. *Proceedings of ACM SIGGRAPH'95*, pages 105–107, August 1995.

[19] V.B. Zordan and J.K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *Proc. of EGCAS'99*, pages 13–22, September 1999.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399