Morphology independent motion retrieval and control

Nicolas Pronost[1], Franck Multon[2,3], Qilei Li[1], Weidong Geng[1], Richard Kulpa[3] and Georges Dumont[4]

[1]*Zhejiang University, State Key Laboratory of CAD&CG*
[2]*University of Rennes 1, IRISA - Bunraku*[*]
[3]*University of Rennes 2, M2S Laboratory*
[4]*ENS Cachan, IRISA - Bunraku*[*]

***Abstract*— This paper addresses the problem of selecting and adapting a motion in order to make a virtual human interact with a user in real-time. We describe a method that is able to retrieve the most appropriate motion in interactive environments where both the constraints and the character's morphology are not known in advance. The method described here is based on a morphology-independent representation of motion to perform fast motion retargetting and retrieval. Because the size of the database is not infinite, even the best candidate motion may not satisfy precisely all the imposed constraints. As a second step of the algorithm, the selected motion is locally adapted in order to accurately satisfy these constraints. This adaptation allows dealing with a large set of possible movements even if no candidate in the database exactly fits the constraints.**

***Index Terms*—Virtual human, Morphology independent representation, Motion retargetting and retrieval, Interactive animation.**

I. INTRODUCTION

Virtual reality has become a popular tool for studying human interaction with his environment and with other characters. It enables scientists to carry-out experiments where the virtual world is totally under control which is impossible in real life. However, this kind of application requires dealing with a wide variety of constraints and subjects, leading to difficult challenges for animating avatars and autonomous virtual humans. Animating virtual humans in such highly interactive applications implies to deal with two main issues: controlling the motion of the character and obtaining natural-looking movements. In computer animation, controlling motions is generally solved by associating space-time constraints that generally requires applying inverse kinematics. This process strongly depends on the morphology of the character that has to perform the task. Naturalness can be addressed by using motion capture data for which the movements are perceived as realistic. However, several sub-problems occur when using motion capture data, such as finding the most appropriate motion clip according to the situation and adapting the resulting trajectories to the morphology of the virtual human and to its environment.

[*] Bunraku is an INRIA jointed team
Manuscript Received on 18 December 2008.
E-Mail : nicolas.pronost@gmail.com

In this paper, we propose a method to select an appropriate motion in a small database in order to reduce precomputation, search and manual editing time. To us, an appropriate motion should deal with the morphology of the character and unpredictable constraints imposed in real-time by the user.

Indeed, the size of an avatar is not always known in advance in VR applications and the environment induces many adaptations of its original captured motion. In this work, we illustrate our method on a kung-fu fighter that has to kick and punch targets that are interactively displaced. 75 kung-fu motions were performed by 4 actors and will be used by virtual fighters with different sizes and proportions. We so allow the morphology of the virtual fighter to change during the simulation. The method will adapt the animation according to this new character. The target is driven by a real-time motion capture system which tracks the position of the head of the user in an immersive environment. This example demonstrates the ability of our method to deal with average-size databases in an interactive application without requiring long preprocessing.

II. RELATED WORK

Several solutions have been proposed to deal independently with the morphology of the character and with space-time constraints. For selecting most convenient motions, two main approaches have been proposed. The first one leads to organize a database of motions as a graph, called motion graph [1, 2, 3] in which nodes are poses and links are possible transitions. After precomputation of the links using a cost function, the resulting graph can be used to find a path between the current state of the character and a desired one [4]. However, the large size of the graph leads to numerous computations in order to find a correct path. On one hand, the larger the number of motions, the higher the quality of the result. On the other hand, the larger the number of motions, the higher the computation time for searching into the graph. Then, methods have been proposed to create constrained motions from motion graphs. Hence, well-connected motion graph on rather small graph size [5] and interpolated motion graph [6] both use interpolations to satisfy accurate by predefined end-effector constraints.

A second approach to retrieve a motion which fits the current situation consists of defining motion templates or patterns [7]. It enables the retrieval of a set of candidate motions that best correspond to a given set of properties. This kind of method must associate a dual representation of the motion with each motion clip. Liu *et al.* [8] thus partitioned a motion library and constructed a motion index tree based on a hierarchical motion description. The motion index tree serves as a classifier to determine the sub-library that contains the promising similar motions to the query sample.

All the above methods generally assume that the anthropometric dimensions of the virtual character are compatible with those of the actor. However, in many VR applications, these dimensions are generally different. As a consequence, the motion selected by these methods will not be compatible with the morphology of the virtual human. Motion retargetting [9] has been introduced to solve this problem. It assumes that the problem is mainly solved by satisfying kinematic constraints. Displacement maps [10] can then be used to solve the constraints for the whole sequence as they add a continuous trajectory to the current one. However, it means that all the constraints are known in advance for the whole sequence in order to calculate the trajectory. In interactive applications, the user can act at any time which may change the constraints in real-time. Let us consider our virtual fighter that has to punch and kick a target interactively displaced by a user. The target is continuously moving in an unpredictable way and the motion of the fighter has to be adapted at each frame. In that case, displacement maps are inappropriate. Inverse kinematics can be adapted and used in a per-frame constraint solver. Using inverse kinematics is necessary because constraints are generally expressed in the Cartesian frame whereas the pose of the skeleton is modeled as a set of joint angles. One alternative consists in replacing this model by a set of mixed Cartesian and angular values that are normalized to become independent from the morphology of the character. These morphology-independent representations [11, 12] lead to very efficient algorithms for motion retargetting and adaptation.

The above approaches can locally adapt the motion to a given set of constraints whereas the original motion may be badly chosen from the morphological point of view. Let us consider again a kung-fu fighter that has to kick an opponent. The kick selected for a two-meter high fighter to hit a target 2-meter away is certainly not the most convenient motion for a small fighter. Another motion of the database may be more adapted to this character in this particular situation, as shown in Fig.1.

Let us consider now a database of motions performed by various actors with different morphologies. Imagine that the synthetic character is manually tuned by the user, as commonly observed in video games or virtual reality applications. Retargetting motions of the database to this virtual human cannot be performed off-line. Hence, the database cannot be preprocessed with a classical retargetting algorithm as we do not know in advance the morphology of the character. But the database could be stored in a morphology independent representation. However, if the constraint is applied to a point that is different from the extremity (such as touching a target with the middle of the forearm), it would require to transform this representation to a classical one to evaluate the distance between the poses and the constraints. Hence, when designing the database, we do not know in advance which type of constraint will have to be satisfied in the real-time application. Therefore, deleting the information about the morphology of the actor seems to be inappropriate for such kind of problem. In this paper, we propose a new search algorithm (section III-C) that embeds a very fast retargetting process (section III-B) that enables us to solve this problem (section III-D).

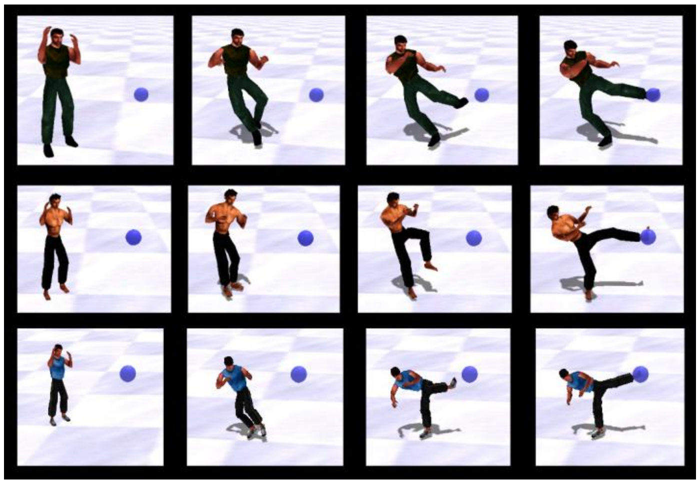


Fig. 1. Three characters with different morphologies satisfy the same constraint (reaching the blue sphere) by performing different natural looking motions.

III. METHODS

3.1 Overview

The first part of the method is a retargetting process, the second part is a motion retrieval algorithm and the last part is a motion adaptation module. The role of the retargetting process is to adapt the current posture of the animated virtual fighter to the morphologies of the actors who performed the motions. The role of the motion retrieval algorithm is to search the database to find the motion most suited to the virtual character's posture and to the real-time constraints. The role of the last module is to adapt the selected motion in order to satisfy the constraints. In our example of a kung-fu fighter, the inputs are the current pose and morphology of the virtual fighter and the position of the target. The overall process is depicted in Fig.2

In our experiment, the motion of the subject's head is captured in real-time with an AR-Tracking system composed of 5 infrared cameras cadenced at 60Hz. With these data, the stereoscopic vision and the point of view of the camera are corrected in real-time. The target that the fighter should punch or kick is assumed to be the middle of the 5 reflective markers placed over the head of the subject. A supervisor interacts at any moment thanks to a keyboard in order to command the fighter to punch or kick the head of the subject using its left or right limbs. This example demonstrates the ability of our algorithm to deal with average-size databases in an interactive application without requiring long preprocessing.

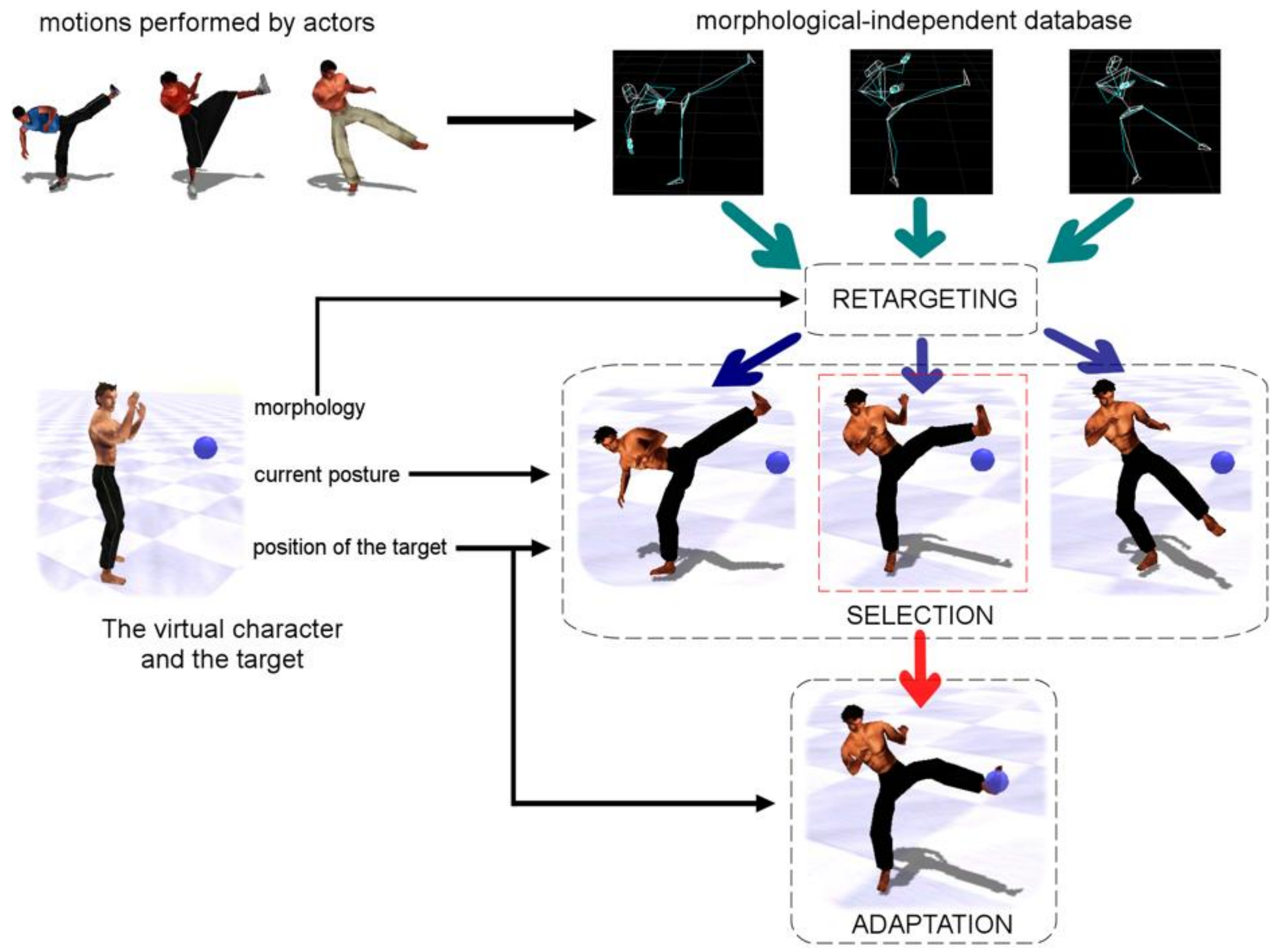

motions performed by actors
morphological-independent database
RETARGETING
morphology
current posture
position of the target
The virtual character and the target
SELECTION
ADAPTATION

Fig. 2. Synopsis of the morphology independent retrieval and control using an average-size database of motions.

Once a corresponding motion is retrieved from the database, it is adapted to the skeleton of the virtual character. At any time, the supervisor can change this character from a library and immediately visualize the result. The kinematic constraints (such as avoiding foot-skating and reaching the target) are solved on the fly. The resulting motion of the virtual fighter is displayed on a 9-meter wide screen for immediate feedback. The motion database is composed of files stored with a standard format (Biovision's BVH files) including the motion and the definition of the skeleton. The database gathers clips performed by 4 various actors. 3 males (1.20m, 1.62m and 1.83m tall) and 1 female (1.55m tall) have performed multiples kung-fu motions leading to 75 sampled motions (130s with a 30Hz sample frequency). Each motion clip is linked with one semantic information (to state if it is a left/right punch/kick) and necessary quantitative data (such as the hit time). We now present how to deal with the morphology independent representation and its use by the modules involved in the process.

3.2 Dealing with morphology

Retargetting is a major issue of this paper because the morphology of the character is a key point in the selection of the motion. Retargetting is used at two different phases. Firstly, it adapts the current pose of the kung-fu fighter to the skeletons of the actors that participated in the motion capture session. Secondly, the motion selected by the motion retrieval algorithm is adapted to the morphology of the kung-fu fighter in real-time. To solve these problems, people usually apply joint angles from character #1 to character #2 while dealing with geometric constraints [9] in order to avoid foot-skating and other unwanted artifacts. However, this conversion is generally time-consuming and implies a complete knowledge of the constraints in advance which is not possible in interactive environments.

An alternative consists in using a morphology independent representation of a character, as presented in [11]. As motion retargetting generally involves solving geometric constraints, this representation is based on Cartesian data instead of joint angles. Hence, the arm is modeled as a line between the shoulder and the wrist and an angle to locate the elbow. Finally, only the relative position of the wrist in the shoulder reference frame and the angle are stored. The relative position is divided by the upper-limb total length (arm and forearm) leading to nondimensional data. The same way, all the intermediate body parts are modeled with relative position of the distal point in the proximal reference frame, divided by the total length. We use this approach to model the legs, the arms, the feet, the hands and the head. The torso is modeled with a spline that can be sampled to as many segments as one wishes. Our model is thus based on the lengths of the limbs, and not on their shapes (the circumferences are not used), because these parameters are the most influential on the retargetting of a motion. The conversion into the classical representation, based on joint angles, is a real-time process. To illustrate this, let us play two motions (a kick and a punch) on four various characters. The average value of the conversion is about 21 µs on a common laptop: Pentium M 2.4GHz (configuration used during all the experiments described in this paper). This value is almost the same for the two motions (differs from 4 µs), and only depends on the configuration of the motion; *i.e.* if some adaptations are needed. This value is also almost the same for the four characters (differs from up to 3 µs) because the multiplications due to the scaling are equivalent but the resulting position of the extremities can lead to different types of correction.

3.3 Motion retrieval

As described above, our morphology independent representation of motions is suitable when we have to deal with several virtual characters. Thus, for the same reason, we will use this representation to retarget the motions of the database which were performed by several actors. Our motion retrieval process, using the advantages of such a representation, is depicted in Fig.3.

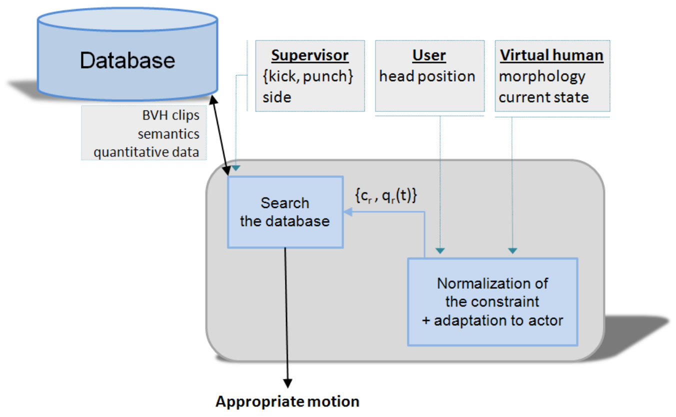


Fig. 3. Overview of the motion retrieval process.

As the simulated character is different from the actors which performed the motions, some adjustments are required. Firstly, the position of the constraint has to be scaled to fit the size of the character. For example, a constraint placed at 1-meter in front of the virtual character does not have the same influence if this latter is 2m or 1.5m tall. To provide a size-relative constraint c_r, we define the following normalization:

$$c_r = c \times \frac{size(Limb_{actor})}{size(Limb_{character})} \quad (1)$$

where c is the relative (to the root reference frame) three-dimensional position of the constraint, *size(Limb*$_{actor}$*)* and *size(Limb*$_{character}$*)* are the cumulative sizes of the limbs from the root to the constrained joint (for example the right foot if the request is a right kick). We divide the relative position of the constraint by the cumulative sizes of the character's limbs. The result is a nondimensional constraint which is multiplied by the cumulative sizes of the actor's limbs.

Secondly, the current pose of the virtual character *q(t)* is retargetted to the actors belonging to the database, providing a set of q_r. Consequently, the distance calculated between a pose in each clip and the associated q_r is based on the same skeleton. Motion retargetting is performed by the method described in section III-B. Hence, the inputs of the motion retrieval algorithm (denoted *Search* in Fig.3) are c_r and the set of q_r. Searching the database for the motion that corresponds to both the constraint and the current pose could lead to lot of computation time. To speed up the process, we propose to organize the database in an efficient manner.

1) *Design of the database*

Since the quality of the output motion mainly depends on the content of the database, we have carefully selected a set of fighting motions which are suitable for the boxing scenario (punch and kick). This selection is preprocessed among a large motion database composed of motion capture data stored in the BVH format. With toolkits developed in our lab, the user can extract the most important subsequences from selected motions (such as times during which the actor is punching or kicking). For each resulting subsequence, the user specifies the name of the hit segment (such as the right hand or the left foot) as well as the semantics (such as punching or kicking). The system automatically calculates the trajectory of the hit point. For each motion, the system also automatically calculates the bounding box of this trajectory, as shown in Fig.4. The *Search* algorithm will then eliminate motions within which the bounding box does not contain the constraint c_r.

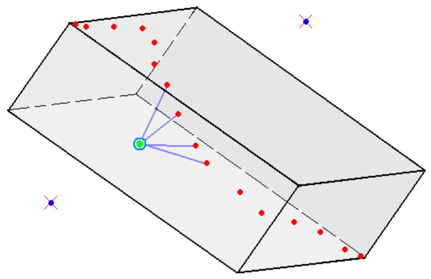


Fig.4. Bounding box of the hit point trajectory modeled with red spots (such as the trajectory of the wrist). The constraint c_r is modeled by the green spot. If the constraint is out of the bounding box of the trajectory (blue dots), the motion which has this trajectory will be discarded, otherwise the distance between the trajectory and the constraint will be calculated.

2) *Precomputation*

In order to perform the motion retrieval process in real time, we adopt a precomputation technique based on clustering. The main idea of this algorithm is to divide the motion database into groups where all the motions inside one group have similar initial postures. Furthermore, the bounding box of the hit trajectory in each motion is used in order to make the hit point filter process faster. The classic K-mean clustering algorithm is chosen to group the motions in our database. The goal of this algorithm is to gather poses of various motions into clusters in order to minimize the total intra-cluster variance. The algorithm starts by randomly partitioning the input motion poses into k initial sets. It then calculates the mean value of the constraint for each cluster (the mean value is also named centroid $\overline{Pose^{G}}$). It builds a new partition by associating each pose with the closest centroid. Then, the centroids are recalculated for the resulting clusters. The method (building new partitions and computing the centroids) is repeated until convergence. Convergence is assumed when the motion frames no longer switch from one cluster to another. We also stop the algorithm when centroids are not modified after an iteration [13].

We introduce a distance value d_{q_r} to compare the similarity between two postures. Since the frames of the motion clips have variety of orientations and absolute positions on the horizontal plane, to define this distance is not obvious. One solution consists of moving the root node of the two corresponding motions to a unique position and, then, rotating them along the vertical axis in order to align them along the same orientation. A better solution proposed by Kovar *et al.* [14] consists of minimizing the distance between the two poses by tuning the alignment parameters of the two motions:

$$d_{q_r} = \min_{\theta, x_0, z_0} \sum_i w_i \left\| p_i - T_{\theta, x_0, z_0} p'_i \right\|^2 \qquad (2)$$

where

$$\theta = \arctan\left(\frac{\sum_i w_i (x_i z'_i - x'_i z_i) - \frac{1}{\sum_i w_i} (\bar{x}\bar{z}' - \bar{x}'\bar{z})}{\sum_i w_i (x_i x'_i - z'_i z_i) - \frac{1}{\sum_i w_i} (\bar{x}\bar{x}' - \bar{z}'\bar{z})} \right) \qquad (3)$$

and

$$x_0 = \frac{1}{\sum_i w_i} (\bar{x} - \bar{x}' \cos(\theta) - \bar{z}' \cos(\theta)) \qquad (4)$$

$$z_0 = \frac{1}{\sum_i w_i} (\bar{z} - \bar{x}' \sin(\theta) - \bar{z}' \sin(\theta)) \qquad (5)$$

In these equations, w_i is a weight associated to point p_i of a trajectory and $T_{\theta, x_0, z_0} p'_i$ is another trajectory transformed with a translation to (x_0, y_0) and a rotation along the vertical axis with angle θ.

3) *Search algorithm*

After clustering, the database is organized as a set of groups *G* associated with a typical pose (or centroid) $\overline{Pose^{G}}$. The main principle of the *Search* algorithm is depicted in Fig.5. It consists of selecting a cluster which best corresponds to the initial pose q_r and in associating the distance d_{q_r} to each pose of this cluster. To this end, all the centroids $\overline{Pose^{G \in \{cluster\}}}$ of cluster *G* are compared with the initial pose q_r. Let G_i be the cluster which typical pose $\overline{Pose^{G_i}}$ is the one that best corresponds to q_r. G_i is then selected for carrying-out a comparison between all the corresponding poses $Pose_{j \in G_i}^{G_i}$ and q_r. The poses are stored as leaves of a hierarchical tree data structure. All the pose nodes are associated with a set of most similar poses $similar(Pose_j^{G_i})$. These ones are neighbors in the data structure that allows direct addressing of all the similar poses.

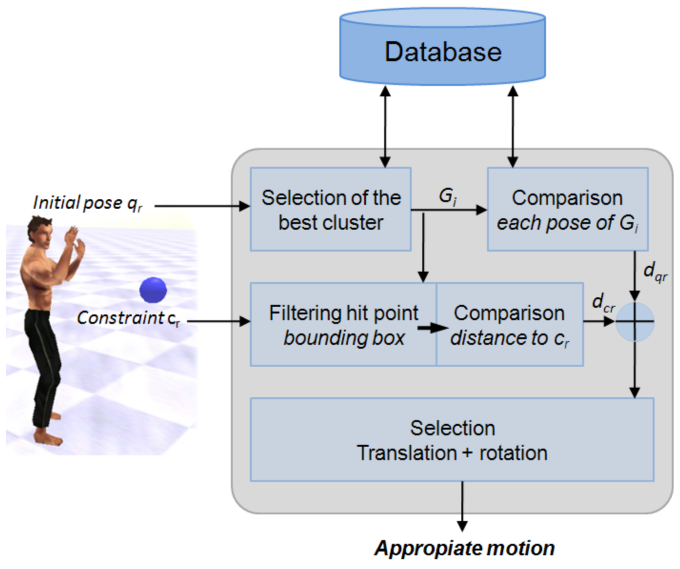

Database
Initial pose q_r
Selection of the best cluster
G_i
Comparison each pose of G_i
d_{qr}
Constraint c_r
Filtering hit point bounding box
Comparison distance to c_r
d_{cr}
Selection Translation + rotation
Appropiate motion

Fig. 5. Overview of the *Search* algorithm.

Even if $Pose_i^{G_1}$ seems to be the best candidate according to the distance d_{q_r} to q_r, it may not be the optimal choice for other criteria. For example, this pose may correspond to the beginning of a punch or a kick but not to the time when the strike occurs. In that case, the similar poses are referenced by $similar(Pose_i^{G_1})$ and can be rapidly analyzed in order to decide if they better correspond to a strike. Let us suppose that one of these similar poses better correspond to a strike than $Pose_j^{G_i}$. This latter is finally selected even if its distance to the constraint is larger than the one of $Pose_j^{G_i}$ in order to avoid unrealistic punches and kicks. This selection only uses the initial pose but solving constraint c_r is also an important problem. The resulting candidate poses need to be filtered in order to eliminate those which the corresponding trajectory of the hit point is not compatible with c_r. This task is performed by eliminating the motions whose the bounding box does not contain q_r. For each remaining pose, we calculate the minimum Cartesian distance d_{c_r} between the trajectory of the hit point and c_r. As the number of candidate motions after this heuristic is low (due to the size of the database), we choose to adapt only the best motion. With larger databases, building an interpolation space between several resulting motions could be done [15]. The pose (and the corresponding motion) that is finally here selected is the one that minimizes the weighted sum of the two distances:

$$d = w_{q_r} \times d_{q_r} + w_{c_r} \times d_{c_r} \qquad (6)$$

where w_{c_r} and w_{q_r} are set at the initialization process in the application. In our experiments, the distances on the posture are almost 30 times larger than the distances on the constraint in an average scale. The motion associated with the lowest distance *d* is then rotated and translated to fit the initial pose q_r.

As the *Search* algorithm is performed for every input posture according to the morphologies of the *n* actors, then a sequence of local minimal distances is calculated, which is denoted as

$D=\{d_1,\ldots,d_n\}$ and finally the global minimal distance $d_{global}=min(D)$ is selected. The corresponding motion and actor are found accordingly and are provided to the *motion adaptation* module.

3.4 Motion adaptation

The goal of this module is to adapt the selected motion M_s to the current situation. As shown in Fig.6, it involves: i) synchronizing and blending M_s with the current active motion, ii) retargetting the motion to the animated character, and iii) solving all the constraints (such as guiding the wrist to the target and correcting feet artifacts).

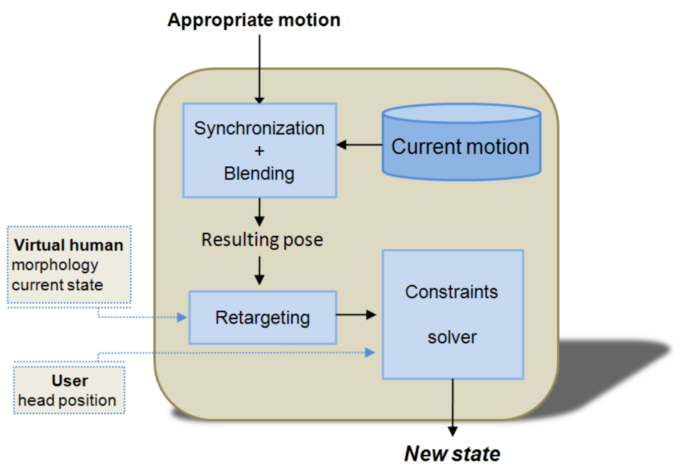


Fig. 6. Overview of the motion adaptation process.

Retargetting mainly consists of multiplying the corresponding nondimensional data by the dimensions of the animated character and correcting the feet artifacts. The resulting pose is compatible with the morphology of the character and respects the ground surface but does not satisfy the other constraints. Let us recall here that the database is small compared to motion graphs or other classical approaches. Consequently, the selected motion is certainly not accurately adapted to the constraints and has to be modified. In our virtual kung-fu application, we have to satisfy one constraint: the position of a limb reaching the target at the hit time. But our method is generic enough to involve several kinematic constraints on any part of the body simultaneously.

Solving kinematic constraints is generally performed with displacement maps. The main advantage of such a method is to ensure continuity in the resulting motion while satisfying the constraints. However, displacement maps are supposed to know all the constraints in advance in order to ensure continuity. In our case, when the supervisor provides an order, the system should react rapidly, as a real human does. We consequently use a frame-by-frame solver driven by continuous constraints. In the selected motion M, the duration between the current frame t_0 and the one corresponding to the contact with the target t_c is known. At t_c, the constraint corresponds to the vector $d(t_c)$ between $M(t_c)$ and the actual target X. At t_0, this distance is $d(t_0)=0$. In between, $d(t)$ is an interpolation between $d(t_0)$ and $d(t_c)$. In our example, we choose to keep the target to the original position X at t_0 even if the head of the user moves. As a consequence, the hand/foot of the virtual fighter will reach the position of the head of the user when the request was made. It enables the user to dodge the punch/kick if its head moves in the meantime. Updating the position of the target during the adaptation phase can be done in real-time as it requires only few computation time. Indeed, t_0 and t_c would not change, only $d(t_c)$ will be updated. The result of the interpolation will take the same amount of time in both cases.

At any time t, we thus have to adapt the joint angles of the character in order to make the wrist or the foot reaches the new position $p(t)+d(t)$, where $p(t)$ is the initial position without adaptation. To solve this problem, inverse kinematics with Jacobian and prioritized constraints is generally used. However, it leads again to computations that may be incompatible with interactive animation. In our system, the pose is not described with joint angles but with a mixed representation (using both angular and Cartesian values, as shown in section III-B). Thus we can use the position of the constraints instead of the joint angles in order to solve this problem. Hence, modifying the position of an extremity is immediate if the constraint is directly reachable.

However, the constraint is often not reachable by only using the corresponding limb. In that case, we use an iterative algorithm derived from the Cyclic Coordinate Descent method [16] (denoted CCD). The method tries to reach the target sequentially for all the body parts involved in the motion. The classical CCD generally leads to unrealistic poses and requires numerous iterations to converge. Lee and Shin [10] have proposed a hybrid inverse kinematics method in order to overcome this limitation. It has been again improved [17] in order to speed-up the process by gathering body-segments into groups allowing a local analytical solution. Hence, the distal groups are tried first (using a direct analytical solution) and then the proximal groups until the root. If the constraint is still not satisfied, a new iteration is processed. Each iteration is very fast because the alignment leads to either solving one or two-link kinematic sub-chains. Finally, kinetic adaptations are applied in order to ensure overall naturalness through a balance strategy. During the adaptation, if only one segment (arm or leg) is involved, the method only requires 34 µs, including time for retargetting. For a constraint that need to use three body parts (such as the arm, the clavicle and the torso), computation time is approximately 40 µs. For unreachable constraints, at worst, computation time is about 115 µs which is still compatible with the 30Hz frame-rate commonly used in computer animation.

IV. Experiments and Results

A subject was asked to participate in this study. He was equipped with stereo glasses (Crystal Eyes glasses for active stereovision). 5 reflective markers were attached to the glasses on a rigid body. The subject took position in the immersive environment in order to interact with virtual fighters.

When the subject moves into the environment, the opponent also moves to stay at a reachable distance for punching and kicking him. Its behavior is implemented with a simple finite state machine driven by the distance between the virtual fighter and the user. In Fig.7, the head of the user is modeled by a blue sphere and the motions (selected into the database) are adapted to reach the position of the sphere.

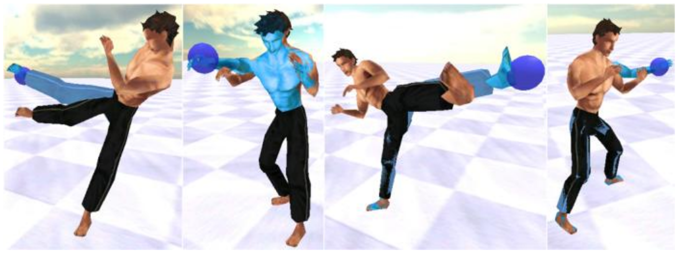


Fig. 7. Adaptation of various punches and kicks selected into the database. The modified posture is represented in light blue and the target is the blue sphere.

As we can observe from the table 1, the global computation time linearly depends on the size of the database. The required time from the beginning of the search algorithm to the return of the selected motion is about *0.54*s with our full database. But as we use two separated threads to search and to display, the system continues animating the virtual human during the process, leading to a continuous animation.

TABLE 1 : COMPUTATION TIME ANALYSIS AGAINST THE NUMBER OF MOTIONS IN THE DATABASE

Number of motions	75	60	45	30	15
Computation time (s)	0.54	0.52	0.46	0.38	0.34

Table 2 shows computation times according to the clustering parametrization. In our experiments, the algorithm gets best performance when the number of clusters is set to approximately 100. This value, related to the number of poses (*i.e.* number of frames), may change according to the size and diversity of the database, or to the required constraints.

TABLE 2 : COMPUTATION TIME ANALYSIS AGAINST THE NUMBER OF CLUSTERS USING THE 75 MOTIONS

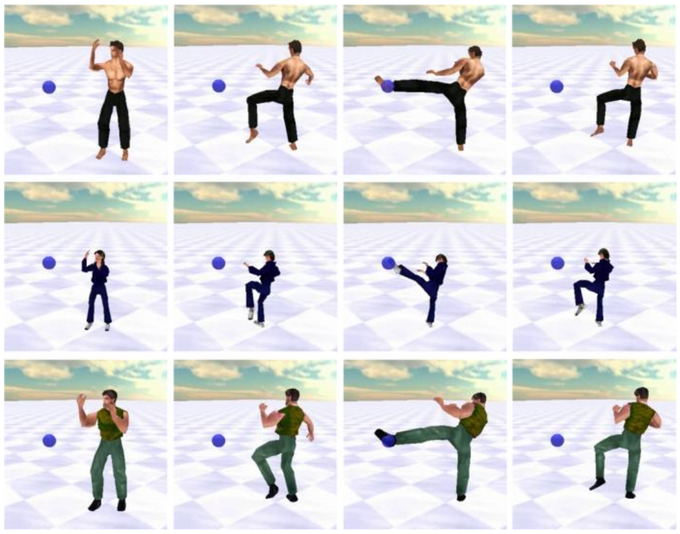
Number of clusters	10	50	100	200	1000
Computation time (s)	5.68	1.24	0.54	1.56	4.87

Computation time is not the only criterion to consider. Let us consider now some examples where the morphology of the subject significantly influences the choice of the motion in the database. If we consider a unique constraint expressed relatively to the root position, then different morphologies of the virtual human lead to different results of the motion retrieval module (as shown in Fig. 8). These results illustrate the fact that constraints are normalized using the morphology which enables the system to produce more appropriate motions.

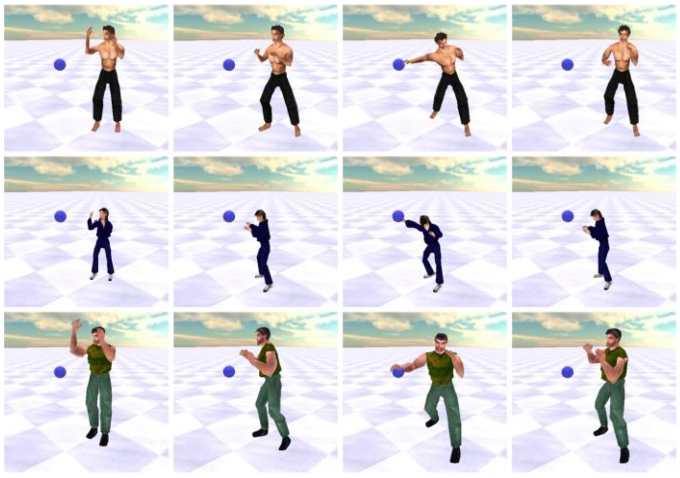
V. Conclusion

The method proposed in the paper aims at searching efficiently a motion into a heterogeneous database obtained with actors with various morphologies and according to a set of constraints imposed in real-time by a user. Motion graphs would require large databases in order to deal with a large variety of constraints. Moreover, motion graphs are generally based on a homogeneous database obtained on a unique actor or preprocessed with retargetting algorithms. A classical alternative is to adapt some selected motions using displacement maps. In the example of the kung-fu fighter, in the off-line process, the constraints imposed to the character are unknown. They will be known in the real-time application only. Moreover the position of the constraint changes in an unpredictable way, depending on the displacement of the user. Hence, it is impossible to solve space-time constraints in this highly interactive application.

Because of the wide variety of constraints that may be imposed by the user in real-time, most of previous methods fail in dealing with a heterogeneous database. One of the main problems is to design a convenient distance metrics that overcomes the problem of using different morphologies. Some methods have proposed normalizations by the body size but they cannot handle actual changes in morphology, such as big



(a) A kick with the left foot



(b) A punch with the right hand

Fig. 8. The morphological difference of the characters can lead to different choices.

differences in proportions. Instead of searching a distance metrics, we have inserted a fast conversion module that ensures compatibility between the current pose of the character and those stored into the database. This approach avoids performing fastidious preprocessing of the data and offers a lot of flexibility for real-time adaptation to a wide range of unpredictable constraints. However, the per-frame constraints solver may lead to some discontinuities if the activation of the constraints is also discontinuous. For example, if we only solve the constraint when the hit occurs, the resulting animation would certainly be unrealistic. We consequently apply a smooth activation of the constraint by defining a transition time during which the constraint is continuously added to the current pose of the character. For very fast motions as kicks and punches, fast acceleration still occur compared to displacement maps. However, displacement maps would fail in satisfying the constraints that change at each time step because the user continues moving after the hit is initiated. Some discontinuities may also appear during the connection between the current motion and the new one selected by the search algorithm. The search algorithm is supposed to retrieve a motion which first frame is not so far from the current pose of the character. However, as the database is small, the most convenient motion may be not close enough to avoid strange acceleration. We used classical real-time blending methods to partially solve this problem. In this paper, we wished to demonstrate that it was possible to handle a lot of different cases with only few motions as the method is based on real-time motion adaptation. Our method could be applied on wider databases, improving the quality of the transitions between motion clips.

In the current version, computation time for searching an appropriate motion is over *1/30*s. For the demos, we decided to use concurrent threads that enabled us to continue animating the virtual human while calculating the appropriate motion. The consequence is a small latency before the character actually performs the motion, which is acceptable for a large variety of interactions. However, it should be improved in future work.

REFERENCES

[1] L. Kovar, M. Gleicher, and F. Pighin, “Motion graphs,” ACM Transactions on Graphics, vol. 21, no. 3, pp. 473–482, 2002.

[2] M. Choi, J. Lee, and S. Shin, “Planning biped locomotion using motion capture data and probabilistic roadmaps,” ACM Transactions on Graphics, vol. 22, no. 2, pp. 182–203, 2003.

[3] H. Shin and H. Ko, “Fat graphs: Constructing an interactive character with continuous control,” in Proc of Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2006.

[4] J. McCann and N. Pollard, “Responsive characters from motion fragments,” ACM Trans. on Graphics (ACM SIGGRAPH 2007), vol. 26, no. 3, 2007.

[5] L. Zhao and A. Safonova, “Achieving good connectivity in motion graphs,” *Graph. Models*, vol. 71, no. 4, pp. 139–152, 2009.

[6] A. Safonova and J. Hodgins, “Construction and optimal search of interpolated motion graphs,” ACM Trans. Graph., vol. 26, no. 3, pp. 106–116, 2007.

[7] [M. Muller and T. Roder, “Motion templates for automatic classification and retrieval of motion capture data,” in Proc. of Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2006, pp. 137–146.

[8] F. Liu, Y. Zhuang, F. Wu, and Y. Pan, “3d motion retrieval with motion index tree,” Comput. Vis. Image Underst., vol. 92, no. 2-3, pp. 265–284, 2003.

[9] M. Gleicher, “Retargetting motion to new characters,” in Proc. of ACM SIGGRAPH, Jul. 1998, pp. 33–42.

[10] J. Lee and S. Shin, “A hierarchical approach to interactive motion editing for human-like figures,” Proceedings of ACM SIGGRAPH 99, pp. 39–48, Aug. 1999.

[11] R. Kulpa, F. Multon, and B. Arnaldi, “Morphology-independent representation of motions for interactive human-like animation,” Computer Graphics Forum, Eurographics 2005 special issue, vol. 24, no. 3, pp. 343–352, 2005.

[12] C. Hecker, B. Raabe, R. Enslow, J. D. Weese, J. Maynard, and K. van Prooijen, “Real-time motion retargeting to highly varied user-created morphologies,” ACM Trans. on Graphics (Siggraph proceedings), 2008.

[13] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” ACM Computing Surveys, vol. 31, no. 3, pp. 264–323, 1999.

[14] L. Kovar, M. Gleicher, and J. Schreiner, “Footstake cleanup for motion capture,” ACM Siggraph Symposium on Computer Animation 2002, 2002.

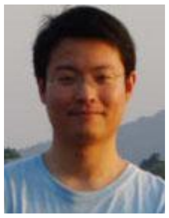
[15] L. Kovar and M. Gleicher, “Automated extraction and parameterization of motions in large data sets,” ACM Trans. Graph., vol. 23, no. 3, pp. 559–568, 2004.

[16] L.-C. T. Wang and C. C. Chen, “A combined optimization method for solving the inverse kinematics problem of mechanical manipulators,” IEEE Trans. On Robotics and Applications, vol. 7, no. 4, pp. 489–499, August 1991.

[17] R. Kulpa and F. Multon, “fast inverse kinematics and kinetics solver for human-like figures,” in Proceedings of IEEE Humanoids, Tsukuba, Japan, december 2005, pp. 38–43.

Nicolas Pronost is a postdoctoral researcher in the VRLab at EPFL, Switzerland. He studied Computer Science at the University of Rennes1 and worked within the SIAMES Project of the INRIA/IRISA Laboratory (Rennes, France). He received his Ph.D. in Computer Science in December 2006, and then he was a postdoctoral researcher at the State Key Lab of CAD & CG at Zhejiang University (China). The main topics of his work are the analysis and the synthesis of human motions using normalized approaches and musculoskeletal simulations.

Franck Multon is Professor in University Rennes2 in France. He is performing his research in biomechanics in M2S Lab and in character simulation in Bunraku/INRIA Rennes. His research interests are biomechanics, character simulation, and interaction between real and virtual humans. He defended his PhD in 1998 in INRIA Rennes on motion control of virtual humans. Since 1999 he was Assistant Professor in University Rennes2, has defended his "authorization to supervise research" in 2006 and has been hired as full Professor in 2008. He published 20 journal papers and 23 conference papers in several domains including computer animation, robotics, virtual reality, biomechanics, neurosciences and anthropology. He is member of ACM SIGGRAPH, IEEE, and the European Society of Biomechanics. He has reviewed papers in several conferences and journals in the above domains, and was member of the international program committee of ACM SIGGRAPH SCA, CASA, IEEE-VR, GRAPP and ACM SIGGRAPH VRST.



Qilei Li is currently a PhD student at the Department of Computer Science, Zhejiang University, China, supervised by Professor Weidong Geng. His research interests include human motion synthesis and human computer interaction.

Weidong Geng is currently a professor of College of Computer Science, Zhejiang University, China. He received the BSc degree from the Computer Science Department in Nanjing University, China, in 1989, and a M.Sc. degree from the Computer Science Department of National University of Defense Technology in 1992. In 1995, he received his PhD from the Computer Science and Engineering Department of Zhejiang University, China. From 1995 to 2000, he was in Zhejiang University, where he took charge of a number of projects about CAD/CG, and intelligent systems. He joined Fraunhofer Institute for Media Communication (former GMD.IMK), Germany, as a research scientist in 2000. In 2002, he worked in Multimedia Innovation Center, The Hong Kong Polytechnic University, Hong Kong. Since 2003, he works in State Key Laboratory of CAD&CG, and the research interests are computer aided design, computer animation, multi-modal interface, interactive media and digital entertainment.

Richard Kulpa is a Lecturer in the M2S laboratory (University Rennes2) where his works deal with biomechanics, perception in sport and interaction between real and virtual humans. He also works on the real-time animation of virtual humanoids and the link between control and behavior in simulations of such characters in the Bunraku team (IRISA-INRIA Rennes).

Georges Dumont has been an assistant professor at Ecole Normale Supérieure de Cachan since 1994 and is currently the scientific leader of Bunraku team at IRISA (Bunraku is an INRIA jointed team). He gratuated in mechanical engineering from the National School of Ponts et Chaussées (Paris, France) and received his Ph.D. in 1990 at IRISA (UMR 6074) and his Higher Doctorate (Habilitation à Diriger des Recherches) in 2005. He has worked at Michelin and at EDF as a research engineer on the finite element method. His interests are in scientific computations for VR, CAD/CAM, and virtual prototyping and virtual human beings.