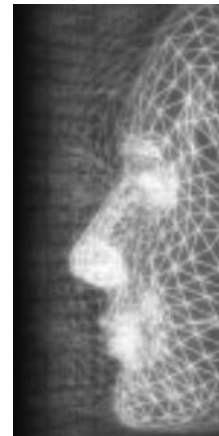


A motion capture-based control-space approach for walking mannequins

By Julien Pettre and Jean-Paul Laumond*



Virtual mannequins need to navigate in order to interact with their environment. Their autonomy to accomplish navigation tasks is ensured by locomotion controllers. Control inputs can be user-defined or automatically computed to achieve high-level operations (e.g. obstacle avoidance). This paper presents a locomotion controller based on a motion capture edition technique. Controller inputs are the instantaneous linear and angular velocities of the walk. Our solution works in real time and supports at any time continuous changes of inputs. The controller combines three main components to synthesize locomotion animations in a four-stage process. First, the Motion Library stores motion capture samples. Motion captures are analysed to compute quantitative characteristics. Second, these characteristics are represented in a linear control space. This geometric representation is appropriate for selecting and weighting three motion samples with respect to the input state. Third, locomotion cycles are synthesized by blending the selected motion samples. Blending is done in the frequency domain. Lastly, successive postures are extracted from the synthesized cycles in order to complete the animation of the moving mannequin. The method is demonstrated in this paper in a locomotion-planning context. Copyright © 2006 John Wiley & Sons, Ltd.

Received: December 2004; Revised February 2005; Accepted: April 2005

KEY WORDS: digital mannequins; locomotion control; motion capture; motion blending; motion planning

Introduction

Computer-aided motion is an active research area, stimulated by a large scope of applications ranging from the simulation of industrial processes (virtual reality, virtual prototyping) to video games and graphics. The animation of mechanical systems is part of a long 25-year story in robotics, merging algorithmic planning and automated control.^{1,2} In this story, the animation of human figures appeared as a challenging special case at the beginning of the 1990's.³ Indeed, virtual humans appear as complex mechanical systems, composed with numerous degrees of freedom. But also (and above all), believability and realism-related constraints are critical issues, which are not required by robot applications. In this perspective, the main pro-

gress has been supported by the development of motion capture technologies, giving rise to a specific and active research area focusing on the animation of virtual humans.⁴

While motion capture provides satisfactory eye-believable motion, such recorded sequences are fixed. Motion control objectives are to readapt captured sequences to fit a given scenario while preserving their believability. This is the question addressed by this paper.

The role of a motion controller is to compute automatically time-parameterized trajectories driving all the degrees of freedom of a mechanical system, given the input state that defines a goal to reach. As an example, the motion controller of a robot manipulator defines the whole motion that the robot has to perform in its joint space, in order to place its end-effector at some location expressed in the workspace, driving the velocities at the same time. In our context, the inputs of a walking control are the desired linear and angular velocities of the mannequin's root (e.g. its pelvis). Outputs are the

*Correspondence to: J-P. Laumond, LAAS-CNRS, 7, avenue du Colonel Roche, 31077 Toulouse, France. E-mail: jpl@laas.fr
Contract/grant sponsor: European Project FP5 IST 2001-39250 MOVIE.

time-parameterized trajectories defining the evolution of all the mannequin's degrees of freedom compatible with desired goals.

Motion control for virtual humans is an active research area in computer graphics (see references 3, 5, 6 and 7 for various overviews). Robotics approaches^{8,9} are sufficiently powerful to design task controllers. However, their applications perform better on humanoid robots than on digital actors. Indeed, eye-believability motion is not guaranteed. Key-framed animations and kinematics-based engines first appeared in Zelter.¹⁰ Physics-based solutions benefited from computers' power increase;¹¹⁻¹⁴ however, calibrating such solutions to ensure motion believability remains a difficult task for the user. Controller design based on motion capture editing aims at providing such convincing motion. The core of these approaches relies on signal-processing techniques introduced by Bruderlin and Williams.¹⁵ New animated sequences are computed by filtering, blending and warping captured motion.¹⁶ Motion blending is achieved using various approaches;¹⁷⁻²⁰ the walking controller proposed in this paper falls into this class.

We address the problem via a geometrical formulation in the two-dimensional control space, allowing:

- fast and automated selection of the motion capture data to be blended;
- automated computation of the respective blending weights.

The key idea is to transform each motion capture of a given database into a single point lying in a two-dimensional velocity space (linear and angular velocities). These points are then structured into a Delaunay triangulation, a well-known data structure in computational geometry²¹ allowing efficient queries for point location and nearest neighbour computations. Our control scheme is based on a blending operator working from three motion captures, inspired by Unuma *et al.*¹⁷ Respective weights are automatically computed by solving a simple linear system with three unknown variables.

After stating our contribution with respect to related work, we present a general overview of the controller. Modelling of a motion capture database as a set of points in the velocity space as well as transformation of the motion capture data into the frequency domain are then presented. We then address the selection and weighting of three motion-captured samples to be blended from a given control. Next we present the blending phase, followed by an analysis of the controller and results. Before the conclusion we illustrate the application in a collision-free motion-planning context.

Related Work and Contribution

Locomotion is certainly the most complex motion among all human tasks. Indeed locomotion involves coordination of numerous degrees of freedom to reach a simple goal, usually expressed by two position variables and one orientation angle. Understanding the locomotion control laws then appears as a challenging research issue in various disciplines, including neurosciences,²² biomechanics,^{23,24} or robotics.²⁵⁻²⁷ For computer graphics²⁸ the objective is to synthesize controllers that provide believable motion (without imposing a priori any bio-inspired models).

Wiley and Hahn⁸ provide a solution for synthesizing new motions from a mixture of pre-recorded data. The proposed formulation is a reference for any motion library-based solution. The interpolation relies first on a hybrid position and orientation representation of the articulated figure; second, on a linear interpolation of position vectors and a spherical linear interpolation of orientation quaternions; and third on a pre-computed motion data resampling. Our method is inspired by this approach but extends it in many ways. We consider continuously changing inputs. The interpolation technique is based on motion data expressions using Fourier expansion and allows the solution of motion periodicity defaults interactively (see section 'Motion Data Periodicity Issues' for details). The resulting interpolation remains valid (no recomputation) while controller inputs are unchanged (making it efficient for animating simultaneously several characters). Also, the control space approach is efficient for storing and accessing motion data (with respect to their characteristics and a given control).

An original interpolation technique is presented in Unuma *et al.*¹⁷ Fourier expansions are used to interpolate motion data in the frequency domain. Examples of synthesized motion transitions or motion variations by superposition illustrate the method. Although this inspired our interpolation technique, we exploited other properties of Fourier expansions (such as motion data filtering-related ones) and limited their use for interpolating cyclic motions of a similar style.

Sun and Metaxas²⁹ introduce a method for automating gait generation. Input motion captures are expressed using a sagittal plane-based representation (with respect to the terrain geometry). The method is integrated in a three-level architecture. The first level considers user-defined paths, transforms them into parameter values for the second level, which determines length and height steps, and adequately modifies the initial motion

dataset. Finally, the last level computes animations (using the new dataset) by combining motion data and procedural animation techniques. As mentioned by the authors, the method does not consider interrelationships between curvature of the path and part of the locomotion parameters, which may lead to incorrect gait modelling. Our approach is mainly based on the analysis of these interrelationships, especially between locomotion velocities (and consequently path curvature) and motion data.

Park *et al.*¹⁹ present a motion blending-based method for controlling locomotion, allowing continuous changes of inputs and including a solution to the motion-retargeting problem. Compared to this approach, we focus on the initial control problem. Although one of our approach objectives is to decrease computational costs (especially sensitivity to the number and the size of considered motion captures), the control space representation also allows the characterization of controllable velocities. Determining automatically the range of feasible velocities with respect to any set of motion captures is crucial to answer the path-tracking problem (among others). Indeed, computing a feasible velocity profile to execute the locomotion is required.

Kovar *et al.*³⁰ introduce *motion graphs*. The technique combines motion captures to provide new motion. It consists of capturing feasible transitions between a set of labelled motion captures. Transitions (graph nodes) can occur at the ending parts of each clip (graph edges), as well as at intermediate frames (new nodes are then inserted in the graph). Motion graphs are built in a three-stage process where possible transitions are first detected, second selected and finally created. A sequence of connected edges produces a believable animation, composed of the corresponding motion capture parts and transitions. The user can control the mannequin by sketching paths to follow. The path that minimizes the error between the animated path and the user-specified path is automatically computed.

Another similar approach is presented by Lee *et al.*,³¹ who consider a database of captured motion where a subject performs several behaviours: walking around, interacting with surrounding objects, jumping, etc. A two-layer structure is used to represent human motion data. The lower layer retains details of the motion, whereas the higher one captures similarities between motion frames and possible interconnections. The *cluster forest* thus built is the core of the architecture, providing three interactive control modes to the user. First, the user can choose a performable action at any time among a set of possible ones according to the

context. Second, the user can mime desired actions in front of a camera. Computations then consist of minimizing the difference between the features present in the rendered version of that action and those in the video sequence. Finally, the user can sketch paths to follow. The controller determines sequences of actions minimizing the distance between the sketched path and the avatar's centre of mass path. Choi *et al.*³² puts into practice the method in the context of planning (i.e. including obstacle avoidance constraints).

Both motion graph and cluster forest approaches exploit motion capture databases to the full and perform very well in practice. Compared to these approaches, our work focuses only on the walking task. Our original approach emphasizes controllability issues (i.e. path tracking or real-time control accuracy). Discrete data structures (graphs or forests) cannot guarantee that any planned path can be followed in an accurate manner. This comes from the fact that the motion library appears as a *discrete* control space. Our contribution is to work in the *continuous* two-dimensional velocity control space. Our approach accepts *any* desired inputs expressed in the velocity space.

Kovar and Gleicher³³ also introduced *registration curves*, consisting of an automatic analysis of a set of motion captures. The technique correlates timing, local frame coordinates and constraints of the input motions. Detected relationships improve classical blending operations and extend the range of candidates for these operations. This technique also addresses the motion control problem. The user controls the locomotion speed, curvature or style by providing a continuous weight function driving the blending operations. This approach focuses on improvement of the motion-blending process itself. Comparatively, our approach focuses on automatic computation of the weight function (user-defined in Kovar's approach), given the high-level goals assigned to the character.

Ménardais *et al.*³⁴ combines motion blending and inverse kinematic techniques to control the motion of virtual actors. While trajectories of skeleton extremities (hands, feet and head) result from motion captures, positions of intermediate articulations (elbow, knees, etc.) are computed using inverse kinematics. This extends the user directives driving the actor. As an example, the distance between the hips and the ground can be controlled. The method is applicable to locomotion-related problems.

Boulic *et al.*³⁵ present the '*versatile walk engine*', based on a direct kinematics technique. The real-time controller relies on a parametric expression of the human gait.

Some of parameters allow adaptation of the gait to any bipedal kinematic structure. The rest of the parameters define the mannequin's behaviour: desired style, target position and speed. One original aspect of the engine is that it supports a continuous change of its inputs, i.e. the angular and linear velocities of walk. In that sense our work follows the same objective. The difference comes from the approach: our approach is fundamentally based on motion imitation that aims to preserve motion realism, while Boulic's approach is based on fine kinematic analysis. It is difficult to compare the approaches on a formal analysis (e.g. controllability properties are the same and both approaches allow real-time control). Only the realism of the resulting animations can make a difference.

An online reactive method for generating locomotion cycles with desired characteristics is presented in Glardon *et al.*³⁶ Principal component analysis (PCA), a statistical method, is used to compress and emphasize the similarities between the input motion-captured data (a set of locomotion cycles). A hierarchical data structure allowing the generation of new motions is obtained at a pre-processing stage, which consists of a successive application of the PCA method over a set of input motions. Each level of the structure identifies relationships between a motion property (e.g. speed, style or personification) with a main PCA component. The method synthesizes a full locomotion cycle for a given vector of parameters. Therefore, the locomotion synthesis method has to be invoked only when necessary, i.e. when the desired locomotion characteristics are modified. As a result, the technique particularly fits the animation of crowds, or more generally the animation of any secondary character in a given scene for which the control parameters are not renewed at each frame.

To summarize, our contribution is to address the walking control by an explicit continuous representation of the two-dimensional velocity control space, which is structured from a motion capture library. Then any desired angular and linear locomotion velocities are considered. Such an approach allows path following, trajectory tracking, goal reaching and real-time control as well. The key issue is the introduction of a 2D geometric approach to select and weight the motion captures to be blended. The motion-blending technique is an extension of the Fourier expansion-based interpolation introduced by Unuma *et al.*¹⁷ Here we consider simultaneously three input motions.

Compared to previous approaches we focus on locomotion controllability-related problems (can the char-

acter reach a desired state and how?). The control space approach allows such considerations. As the controller is aimed at being plugged into motion-planning platforms, controllability is a crucial issue. Our method for analysing and representing the motion dataset allows us to determine easily constraints on paths to be followed. As a result, a software platform invoking the controller is able to provide varieties of feasible trajectories (i.e. with respect to the motion library content). Credibility of results could be improved in several important ways (see sections 'Animation Results' and 'Conclusion') on which other approaches focus. Our main concern is to exploit to the full results of the proposed method for analysing motion data, in order to provide feedback to the platform handling our controller, and enable fully automatic use without any user intervention.

Overview of the Walking Controller

The global architecture of the walk controller is summarized in Figure 1. Main components are the Motion Library, the Control Space and the Blending Space.

The Motion Library stores motion capture samples. All of them correspond to walking motions at different velocities. The principle is to decline a same locomotion according to the parameter that is aimed at being controlled, as introduced by Wiley and Hahn.¹⁸ As we want to control locomotion velocities, the samples should respect the following rules:

- motion is recorded on a flat terrain;
- motion captures use the same kinematics model (that of the animated actor);
- followed paths are straight lines and arcs of circles (see below for explanation);
- paths are followed at constant velocities;
- a motion sample describes a complete locomotion cycle;
- samples are in phase: the cycle always starts at the same specific posture (e.g. left foot strike).

The library automatically analyses the average velocities at which the mannequin moves in each sample. Thus, each capture is represented by a point in a 2D Control Space (v, ω) , where v is the linear velocity and ω the angular one. All the points in the Control Space are structured into a Delaunay triangulation.

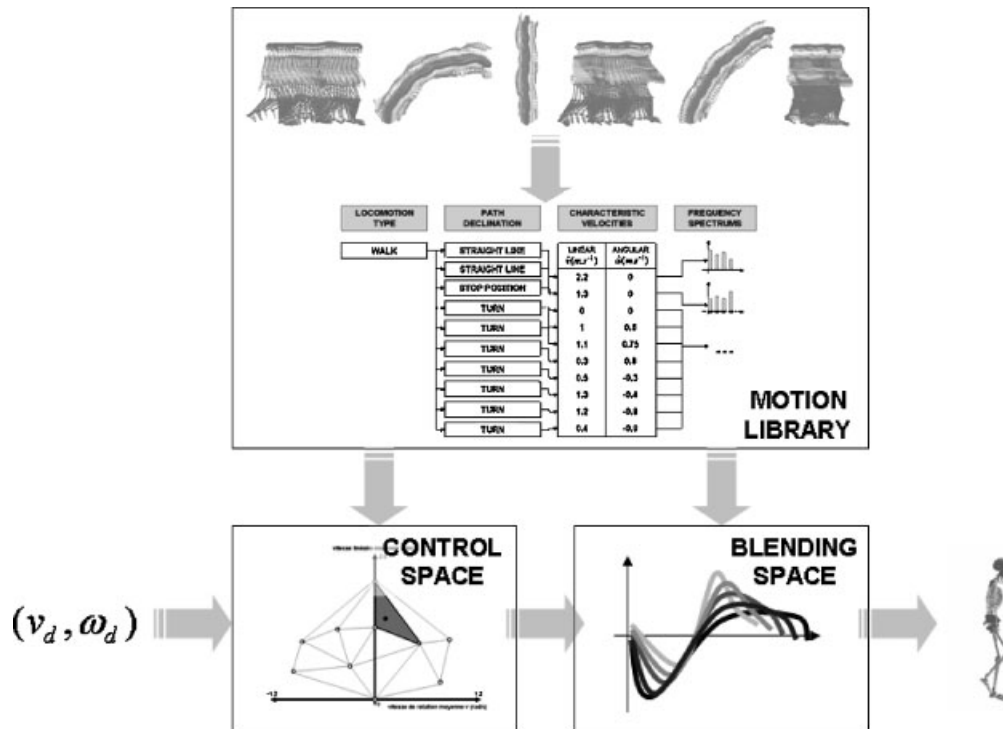


Figure 1. Overview of the controller architecture.

A desired control (v_d, ω_d) being given, the vertices of the triangle containing (v_d, ω_d) determine the three nearest motion samples selected to be blended. The blending is done in the Blending Space according to the weights corresponding to the position of (v_d, ω_d) in the Control Space triangle. The Blending Space expresses the motion data into the frequency domain with Fourier expansions. A linear interpolation of the frequency spectra allows the synthesis of a new locomotion cycle with the desired characteristics.

The following sections develop the underlying techniques of the three main components.

Motion Library

Figure 2 illustrates the structure of the Motion Library. Motion samples are first organized according to user-provided labels describing the captured *locomotion type* and the *path type*. The controller always blends captures from the same type. The structure of the Motion Library is general. Locomotion types can be 'walk', 'side steps', 'run', or more subtle: 'tired walk', 'angry walk'. However our current implementation restricts the locomo-

tion type only to walk.* The path type allows us to adapt the analysis as described below.

Analysis of Motion Captures

The average velocities $(\hat{v}, \hat{\omega})$ at which the mannequin walks characterize each motion sample. These values are automatically derived from the captured trajectory $(x_t, y_t, z_t, \theta_t)$ of the mannequin's root (the pelvis). The position and orientation of the pelvis are expressed in the absolute coordinate system. Unlike other internal degrees of freedom (expressed in relative coordinate systems) these data are not periodic. We then devise a least-square filter allowing to decompose the root trajectory into a linear and angular component (due to the displacement of the mannequin at constant velocities) and into a periodic component (due to the oscillations of the pelvis around followed trajectories).

*This restriction is due to the absence of a direct access to motion capture technologies that would have allowed the authors to build a rich database by themselves. The current database is restricted to simple walking behaviours. It has been provided by the former company Exmachina and by F. Multon (LPBEM/Université de Rennes 2).

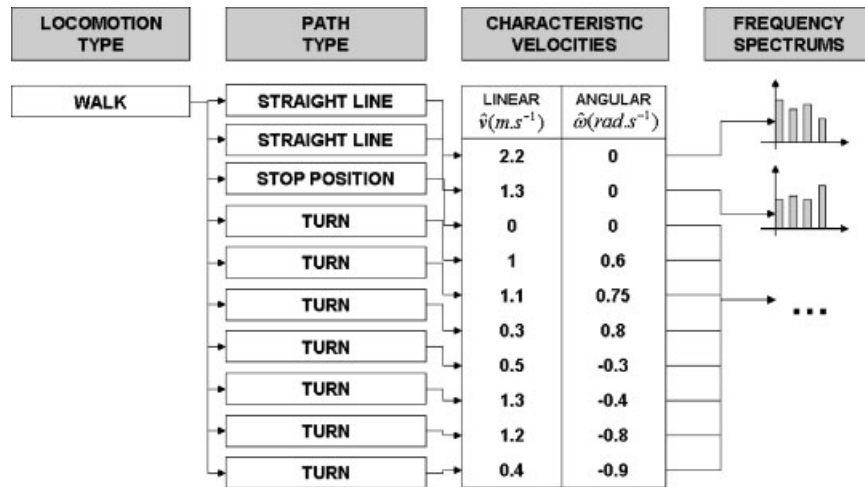


Figure 2. Architecture of the Motion Library.

Figure 3 illustrates the successive stages of the process along a straight line. On the left, the walking mannequin is displayed from above and the successive pelvis positions are shown as black circles.

The first stage consists in computing analytically a simple geometrical pattern corresponding to the path approximately followed by the mannequin in the considered motion sample. Where the mannequin walks along a straight line, the searched pattern is a line segment. Where he is turning, the searched pattern

is the arc of a circle. The time-parameterized equation of the *identified trajectory* is computed using a least-square fitting technique (Figure 3, second image).

We derive the average velocities ($\hat{v}, \hat{\omega}$) of the walk cycle from the equation of the pattern. For that, we consider the curvature and length of the computed pattern and the cycle duration.

Also, we extract the periodic component of the root trajectory as mentioned before. The *positioning error*

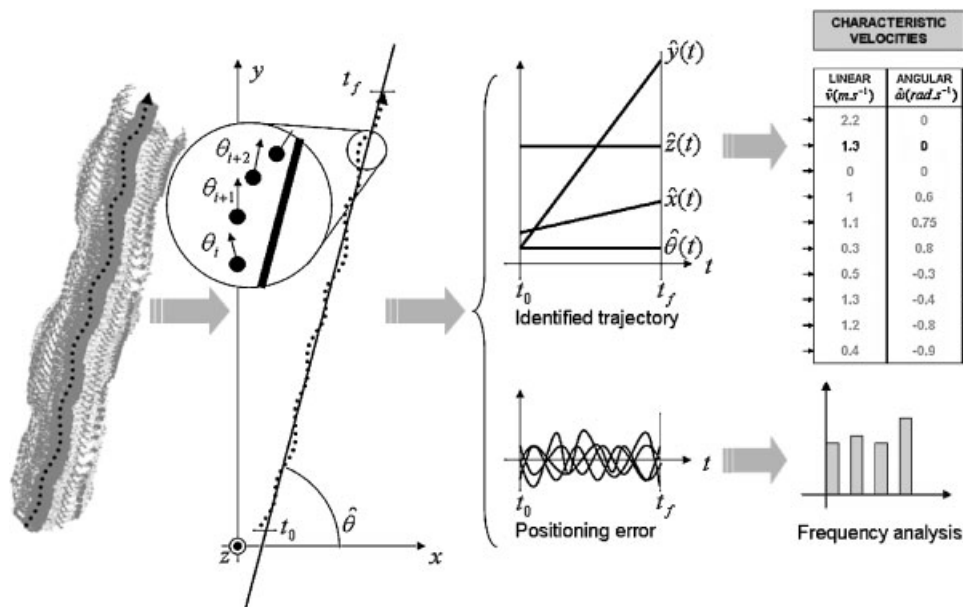


Figure 3. Analysis of a motion capture along a straight-line motion.

(Figure 3, third image, bottom) at a given instant t is computed by subtracting the position estimated by the previously computed pattern $\hat{P}_t = [\hat{x}(t), \hat{y}(t), \hat{z}(t), \hat{\theta}(t)]^T$ from the position of the root $P_t = [x_t, y_t, z_t, \theta_t]^T$.

The root motion data, decomposed as previously explained, are much easier to manipulate than the initial ones. Classically, 2D transformations are performed on motion data before their blending in order to align captured positions on the same axis (e.g. the *coordinate frame alignment* process³³). In our case, we correlate walk velocities and pelvis positions in a relative coordinate system. Later, the relative positions of the pelvis and those of the internal degrees of freedom will be blended in the same manner. As a result, a 2D transformation of the motion data is not required.

Expression of the Data in the Frequency Domain

Locomotion is a periodic event for the internal degrees of freedom of the mannequin. The motion of the arms, legs and head is periodic. As the trajectory of the degrees of freedom (angular trajectories) is described in motion captures using relative coordinate systems (i.e. the limbs of the mannequin are positioned relatively to their ascendant in the kinematic structure), this periodicity also appears directly in the data.

We now compute the discrete Fourier transform of each angular trajectory. There are, however, a few subtleties in the interpretation of discrete Fourier transforms. This operation is well known in the domain of signal processing. It is an invertible linear transformation of N samples composing a discrete signal into a set of N complex numbers. The modulus of the i th complex number reveals the strength of the signal over the frequency $f_i = iT^{-1}$ (where T is the duration of the capture). A suitably scaled plot of the complex modulus of a discrete Fourier transform is commonly known as a power spectrum.

Assuming that the considered signals are (almost) periodic, relationships allow us to transform the real and imaginary parts of the complex numbers into the coefficients (α_i, β_i) of Fourier series expansions. The set of pairs (α_i, β_i) constitute the frequency spectrum of the input motion signal.

Note that with such a method we obtain a continuous expression of the input angular trajectories. Let us consider the discrete motion signal m_t . Its continuous expression is

$$m_t \equiv m(t) = \frac{1}{2}\alpha_0 + \sum_{k=1}^N \alpha_k \cos\left(\frac{k\pi t}{T}\right) + \beta_k \sin\left(\frac{k\pi t}{T}\right)$$

It is crucial to represent all the angular trajectories of the motion-captured samples with a set of pairs (α_i, β_i) of the same dimension. For that, each angular trajectory must be composed of the same number of sampled values. As the motion captures do not have the same duration, in the general case this condition is therefore not satisfied. The user is in charge of arbitrarily choosing a number N of samples to represent every angular trajectory. Before being expressed in the frequency domain, angular trajectories are interpolated linearly to respect this number of samples. The operation is equivalent to a frame rate change to represent the motion.

Now the Motion Library initialization is complete. The two next sections present how the library is exploited to achieve animation synthesis from the user's directives.

Control Space: Selection and Weighting of Motion Captures

A desired control (v_d, ω_d) being given, this section describes how three motion captures to be blended are selected in the Motion Library. The captures are then weighted to balance their respective influence in the blending stage. The principle is illustrated in Figure 4.

Selecting the Motion Samples

The first stage consists of selecting the three captures in the library whose characteristic velocities $(\hat{v}, \hat{\omega})$ are the nearest to the directives (v_d, ω_d) . Each motion capture contained in the library is represented as a 2D point in this space whose coordinates are $(\hat{v}, \hat{\omega})$. The points are structured into a Delaunay triangulation.^{21,38} Figure 4 (top) displays the representation of the content of the Motion Library in such a way.

The user's directives are then taken into account: (v_d, ω_d) are the coordinates of the directives point. The three vertices of the Delaunay triangle containing (v_d, ω_d) correspond to the three motion captures with the nearest characteristics. This selection method is presented in Figure 4 (bottom). Note that usually the point is included in a triangle. The case where (v_d, ω_d) lies outside the envelope of the motion sample points

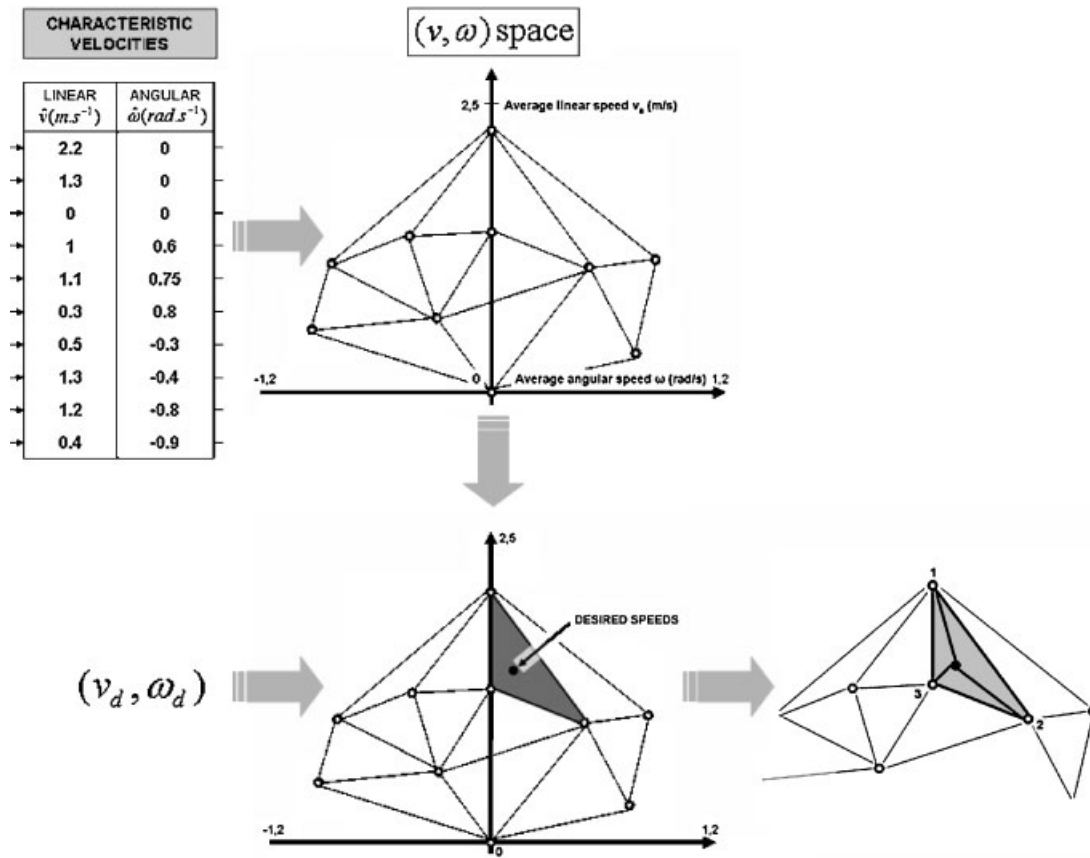


Figure 4. Motion capture selection and weighting process.

may be considered as well. This technical issue is sketched in the next section and more deeply addressed in the section ‘Analysis and Performance’.

Weighting the Selected Samples

The next stage consists of weighting the three selected motion captures to be blended. The principle is as follows: the more the characteristic velocities of a selected sample are close to (v_d, ω_d) , the more this sample influences the result. The sum of the influences of each sample must be normalized in order to obtain a consistent result. The solution is given by solving the following simple linear system:

$$\begin{cases} a\hat{v}_1 + b\hat{v}_2 + c\hat{v}_3 = v_d \\ a\hat{\omega}_1 + b\hat{\omega}_2 + c\hat{\omega}_3 = \omega_d \\ a + b + c = 1 \end{cases}$$

where v_d, ω_d are the controller inputs; $\hat{v}_1, \hat{v}_2, \hat{v}_3$ are the average linear speeds of each selected motion sample;

$\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3$ are the average angular speeds of each selected motion sample; and a, b, c , unknown variables, are the respective influence of each selected motion sample.

As (v_d, ω_d) is included in the triangle $((\hat{v}_1, \hat{\omega}_1), (\hat{v}_2, \hat{\omega}_2), (\hat{v}_3, \hat{\omega}_3))$, we ensure that $a, b, c \in [0, 1]$. The blending operation is an interpolation. In the case where the directives point is located outside the coverage of the Delaunay triangles, it is still possible to select the nearest neighbours. However, because a, b or $c \notin [0, 1]$, the blending operation is an extrapolation. Extrapolations are feasible with our blending method but could lower the believability of the resulting animations (especially for high-amplitude extrapolations).

Motion Capture Blending

Motion capture blending and extraction of postures from the synthesized locomotion cycles is the two last stage of the controller process. The blending involves interpolating the angular trajectories of the three

selected motion captures according to their respective weight. The interpolator manipulates the motion frequency spectra previously computed for each motion sample. A complete locomotion cycle is synthesized with characteristics corresponding to the user's directives. Finally, postures are extracted from the new cycle to produce the animation.

Interpolating Motion Data

The interpolator is presented in Figure 5. From the parameters computed during the previous stage and the frequency spectra contained in the library, we interpolate linearly rank by rank the coefficients of the Fourier expansions weighted by a , b and c . Interpolated coefficients are the following:

For each degree of freedom and for $i : 0 \rightarrow N$,

$$\text{then } \begin{cases} \alpha_i^d \leftarrow a\alpha_i^1 + b\alpha_i^2 + c\alpha_i^3 \\ \beta_i^d \leftarrow a\beta_i^1 + b\beta_i^2 + c\beta_i^3 \end{cases}$$

The resulting set of coefficients (α_i^d, β_i^d) is the frequency spectrum of a new angular trajectory driving one degree of freedom of the mannequin.

A pair (α_j^i, β_j^i) issued from the j th motion capture refers to the frequency $f_i = iT_j^{-1}$ depending on T_j , the duration of the j th capture. As duration varies for each selected capture, we have to compute the duration of the synthesized locomotion cycle with respect to each selected motion capture duration:

$$T_d = aT_1 + bT_2 + cT_3$$

Each angular trajectory of the synthesized locomotion cycle can be then expressed analytically, by computing the Fourier series with respect to (α_i^d, β_i^d) and T_d :

$$m_d(t) = \frac{1}{2}\alpha_0^d + \sum_{k=1}^N \alpha_k^d \cos\left(\frac{k\pi t}{T_d}\right) + \beta_k^d \sin\left(\frac{k\pi t}{T_d}\right)$$

Extracting Postures

A controller iteration produces one single posture to animate the mannequin. Yet, we have computed a complete locomotion cycle whose characteristic velocities are (v_d, ω_d) . While the user's directives (v_d, ω_d) remain unchanged, given the animation timing and the analytical expressions of the synthesized locomotion, a posture can be immediately deduced. Nevertheless, the controller supports continuous changes of the directives. As a result, the synthesized locomotion cycles evolve as well as their duration T_d . We now present the mechanism allowing extraction of postures from the synthesized locomotion cycle while preserving the animation smoothness (see Boulic *et al.*³⁷ for a similar method).

The solution is presented through an example (Figure 6) where a linear acceleration is imposed. Δt corresponds to the user-defined frame rate of the output animation. At $t = t_i$, a 'slow' walk cycle is synthesized. A single arbitrary posture is extracted from this cycle and copied towards the animation. We normalize the duration of the cycle (see the progression bar under the walk cycle in Figure 6). $p_i \in [0, 1]$ indicates where the posture used

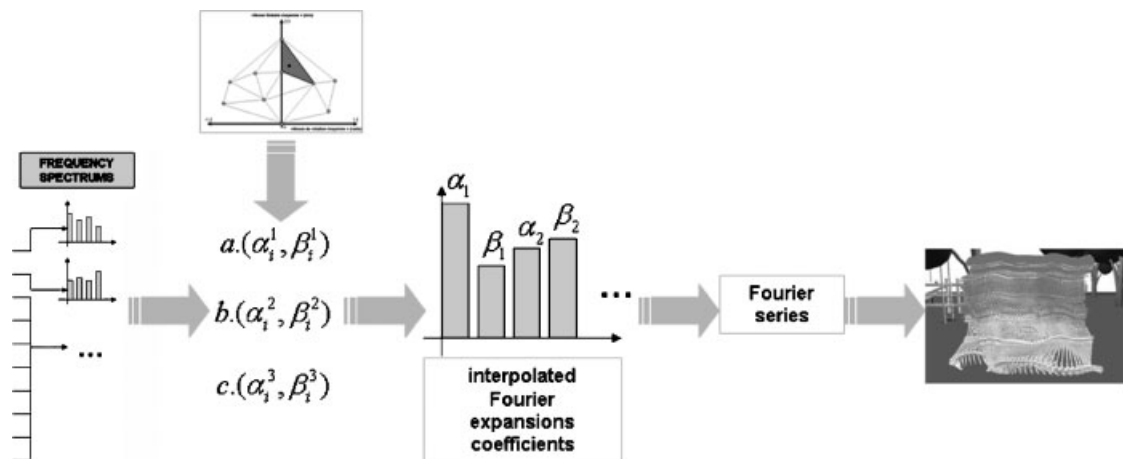


Figure 5. Motion blending.

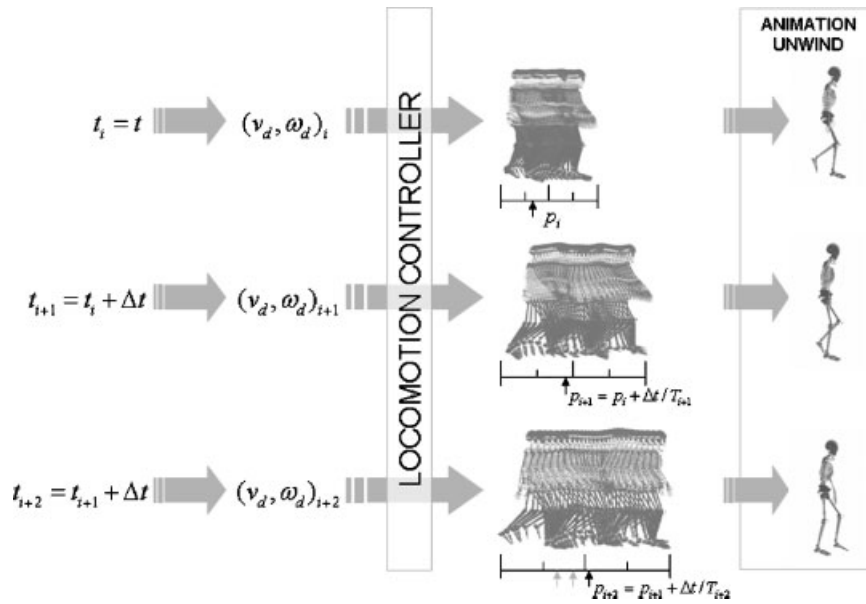


Figure 6. Continuous control of locomotion.

to animate the mannequin is positioned in the synthesized walk cycle (in the given example $p_i \approx 0.3$).

In the second stage of the example, the user's directives (v_d, ω_d) evolve, and a 'normal' walk cycle is produced. The duration of this cycle (T_{i+1}) differs from the previous one. Considering the normalized duration of the cycles, we can report the relative position of the last posture used in the new cycle (see the greyed arrow in the progression bar under the second cycle). To ensure a smooth and continuous animation of the mannequin, we compute the correct position of the next posture to extract: $p_{i+1} = p_i + \Delta t / T_{i+1}$. Indeed, we start from the same normalized position as the previous one, and then we progress to the new cycle with respect to its duration: the longer the new cycle, the less we progress into this cycle. In our case, $p_{i+1} \approx 0.45$.

This process is repeated in the third stage of the example, where $p_{i+2} \approx 0.55$.

Analysis and Performance

Comments on Motion Library Quality

The Motion Library content defines the possible behaviours of the mannequin. Direct relationships exist between the motion sample characteristic velocities and the reachable velocities for the mannequin. Figure 7 (top left) displays the previously defined (v, ω) space. The envelope containing all the motion sample points is

underlined. Each point inside this envelope leads to an interpolation. When the point is outside the envelope the blending extrapolates the motion data (Figure 7, bottom). Even if extrapolations are feasible they should be avoided. Indeed, the larger the envelope, the larger the reachable speeds are. Coverage is not the only important characteristic. The envelope should also be equally spread over the angular velocities axis: the mannequin should be able to reach similar positive or negative angular velocities.

The density of the motion samples inside the envelope is also critical (Figure 7, top right). On one hand, a high density of the motion sample prevents high amplitude warps of the initial samples. On the other hand, a high density affects the controller computational performance since the complexity of both the point location problem and the weighting computation are respectively logarithmic (w.r.t. the number of motion captures) and constant.

Figure 7 illustrates a medium-quality Motion Library. The envelope correctly covers the linear velocities axis. The mannequin can go from a stop position to a high-speed walk (almost a run at 2.3 m s^{-1}), but he can neither turn without a linear velocity nor turn while walking fast.

Motion Data Periodicity Issues

Motion captures' quality conditions the believability of output animations. One classical default observed in the

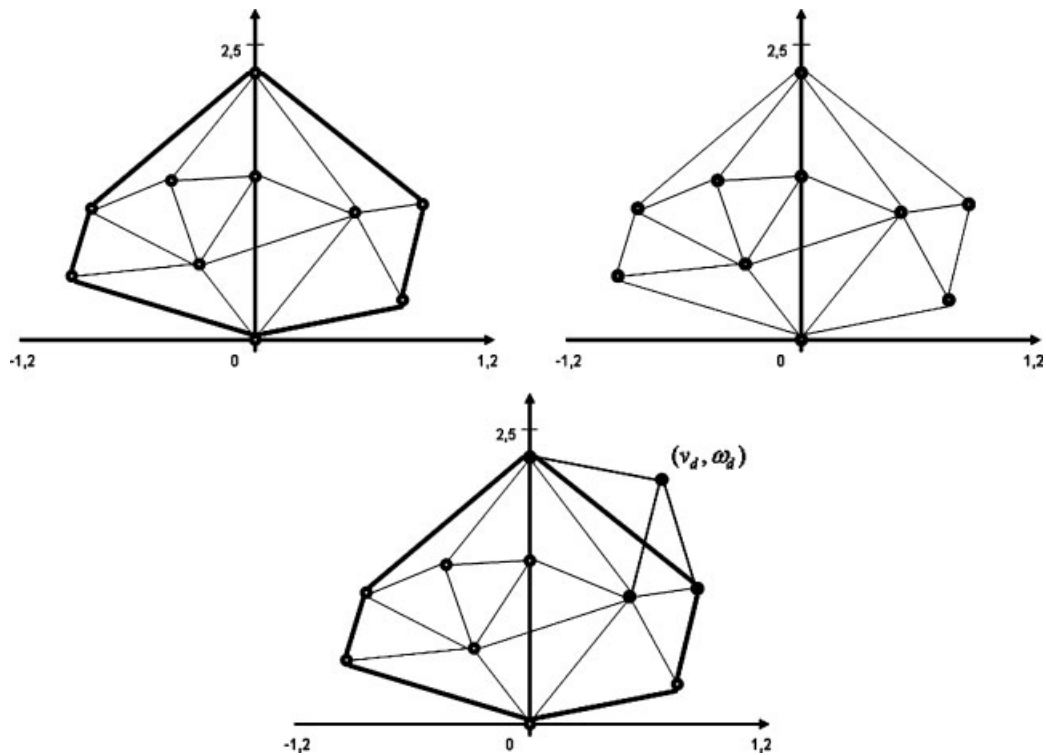


Figure 7. Covering and density of the Motion Library (top); extrapolation case (bottom).

captures concerns their periodicity. The captured human performs *almost* periodic motions. The captures should be carefully pre-processed to avoid such problems: interpolating between captures' first and last frames is a well-known solution.²⁰

However, here, motion blending works in the frequency domain and can solve these problems on the fly. Whereas human motion signals appear in the lowest frequency bands, periodicity defaults provoke high-frequency signals in the data and are represented in the highest frequency components (α_i, β_i) of the Fourier expansions (i.e. for the highest values of i). Neglecting the last rank pairs (α_i, β_i) comes down to applying a low-pass band filter to the data.

The controller lets the user decide the rank from which the pairs (α_i, β_i) are to be neglected. Experimentally, neglecting (α_i, β_i) for frequencies over 15 Hz corrects the periodicity defaults without negative effects on the animation believability.

In special cases, sudden changes in the gain of the low-pass band filter may provoke oscillations in the synthesized motions. Lowering the gain progressively solves this problem ((α_i, β_i) pairs are linearly lowered to zero between two given values of i).

Inputs

Real-Time Control. The inputs of the controller are the angular and linear velocities at which the actor is required to move. Any hardware interface with at least 2 degrees of freedom can be used to drive the character, e.g. mouse, joystick or keyboard. The interface state is evaluated at each time step, and transformed into numerical values v_d, ω_d . A filter should be introduced between the user interface and the controller. The role of this filter is the avoidance of unrealistic directive variations, provoking unbelievable accelerations of the mannequin.

Higher-level directives could be immediately provided to the user by adding plug-ins between the interface and the controller inputs. For example, playing automatically on the ω_d value can satisfy the 'walk in that direction' directive. Other useful directives such as 'follow this path' or 'go to that position' raise more difficult problems, addressed in the two next sections.

Path Following. The main difficulty in path following is to transform a given path $S(u)$ into a trajectory $S(t)$. $S(u)$ is the parametric expression of the geometrical

curve defining the path to follow. $S(t)$ adds to the previous geometrical definition a time law for the execution of the path. The difficulty of the parameter substitution is to respect given criteria, which in our case are bounded linear and angular accelerations and velocities. These criteria ensure that the reachable velocities are not overtaken (see previous section) and that unbelievable accelerations are not provided. The problem is well known in robotics, and solutions exist.³⁹

Given $S(t)$ and a frame rate for the output animation, we can immediately deduce a set of successive locations spread on the time axis, as well as discrete velocity profiles which constitute the inputs of the walk controller.

Steering Method. A steering method addresses 'go to that location' directives. It synthesizes a path from two given initial and a final mannequin position. The path is then followed in the same way as described in the previous section. Steering methods are also known as local methods in the motion-planning context (see below).

Figure 8 presents a Bezier-based steering method. The initial and goal positions and orientations are displayed. The resulting path shown is a third-degree Bezier curve, driven by four control points. The control points are displayed as P_0, P_1, P_2 and P_3 . P_0 and P_3 are located at the initial and goal positions of the mannequin. P_1 and P_2 are located given the initial and goal orientations of the mannequin and a distance D (experimentally we choose D equal to the height of the mannequin). The curve results from the control point barycentre computation with weights evolving according to Bernstein polynomials.⁴⁰ Such curves allow C_1 continuity by composition.

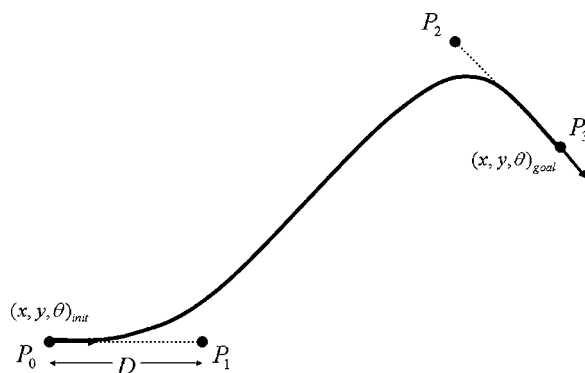


Figure 8. A Bezier-based steering method.

Performance

The walk controller works in real time. The motion capture analysis stage is negligible since it runs once during the controller offline initialization. The dominant computing times are those related to the three main stages of the controller:

- the selection process is logarithmically dependent on the number of motion captures;
- the weighting process runs in constant negligible time thanks to an analytical solution;
- the blending and extraction stages are both linearly dependent on the mannequin's number of degrees of freedom, and on the considered number of interpolated pairs (α_i, β_i) .

In the example presented in Figure 11, the controller computes a posture in less than 1 ms (0.92 ms exactly for the motion capture selection, weights computation, interpolation, Fourier series computation and extraction of postures from the synthesized cycles), on a Sun Blade 100 (UltraSparc processor IIe 700 MHz, 768 Mo RAM, OS Solaris 8). This leaves enough time for the rendering in interactive use. In our case, the mannequin has 62 degrees of freedom, five motion captures are present in the motion library, 32 couples of coefficients (α_i, β_i) characterize each angular trajectory, and these terms are neglected after the eighth rank.

Results

This section focuses on the motion-blending technique results. We present interpolations of three angular trajectories (sinusoids with different characteristics). These results are then followed by examples of animations produced by the controller.

Interpolation of Sinusoidal Signals

Figure 9 illustrates the interpolation of three sinusoidal signals, with different periods, phases and amplitudes. Sinusoidal signals are representative of the kind of angular trajectories contained in walk motion captures, and also ensure the results' readability.

In the same manner as mentioned in the section introducing the selection and weighting process, we associate each blending source with a point in the control space (top). In this space, the location of the 'directive point' relative to the 'input points' determines the weight of each sinusoid for their blending.

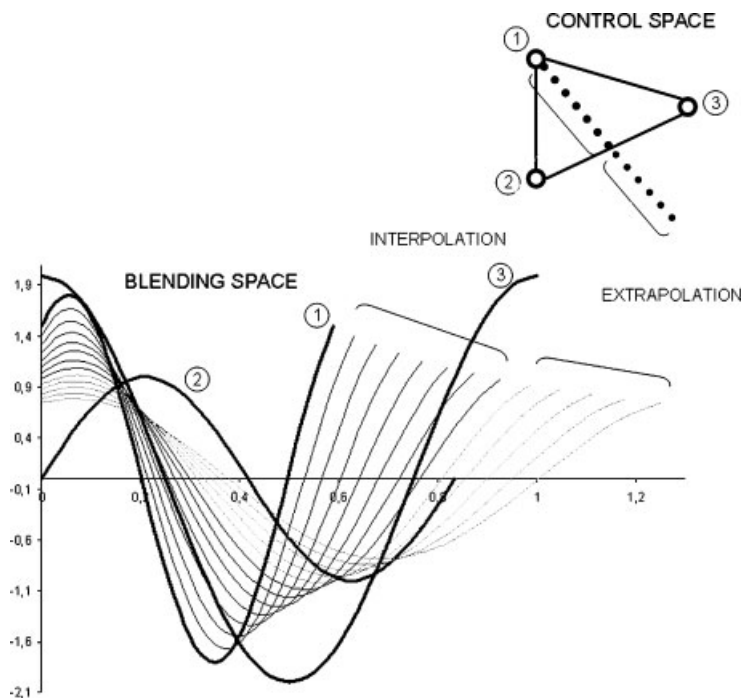


Figure 9. Interpolation of three sinusoidal signals.

The successive locations of the directive point used in the example of Figure 9 are displayed. The black positions of the directive point correspond to interpolations with different balances of the three input signals. The grey ones correspond to extrapolations, given that the directive point is outside the triangle formed by the input points.

The graph (Figure 9, bottom) displays the three input sinusoids (thick black lines), as well as the resulting interpolations and extrapolations (thin dark and light grey lines). Correspondences between the directive point locations and the curves are indicated in the figure.

At the beginning, the control point is first located near point 1 and, as a result, the corresponding curve in the graph is very similar to sinusoid 1.

The location of the control point then evolves along a line. Input signals 2 and 3 become progressively the only interpolation sources with equivalent influences. Given that those signals have higher periods compared to the first one, the interpolated signal period is progressively increased. In the same way, the amplitude of the interpolated signal decreases owing to the influence of sinusoid 3. The sinusoidal structure of the interpolated signal is preserved during this transition.

Finally, the control point evolves outside the triangle, and the input signals are extrapolated. The weight of sinusoid 1 becomes negative, while the other weights

still increase. The sinusoidal structure of the result is still preserved, while its period amplitude and phase evolve according to the respective weight of each signal.

Animation Results

Figure 10 displays four screenshots of the controlled mannequin. The first image illustrates a walk along a straight line while the mannequin accelerates (from 0 to 2 m s^{-1}). This example is similar to the one detailed in Figure 6.

The second image is a screenshot of the mannequin's walk along a path produced by the Bezier-based steering method. Initial and goal positions are joined in a realistic way. The mannequin starts to walk, turns to the right and accelerates to get closer to the goal. Finally the mannequin turns to the left in order to adjust its orientation with respect to the goal. The third image proposes another point of view to display the same motion. The fourth image focuses on one of the left foot support stages occurring in this motion. Thanks to the quality of the motion samples, no foot stake is observed. Nonetheless, unbelievable motion details (foot stakes, limb interpenetrations, etc.) can be observed in the resulting animations under certain conditions. Either captured motion quality or blending

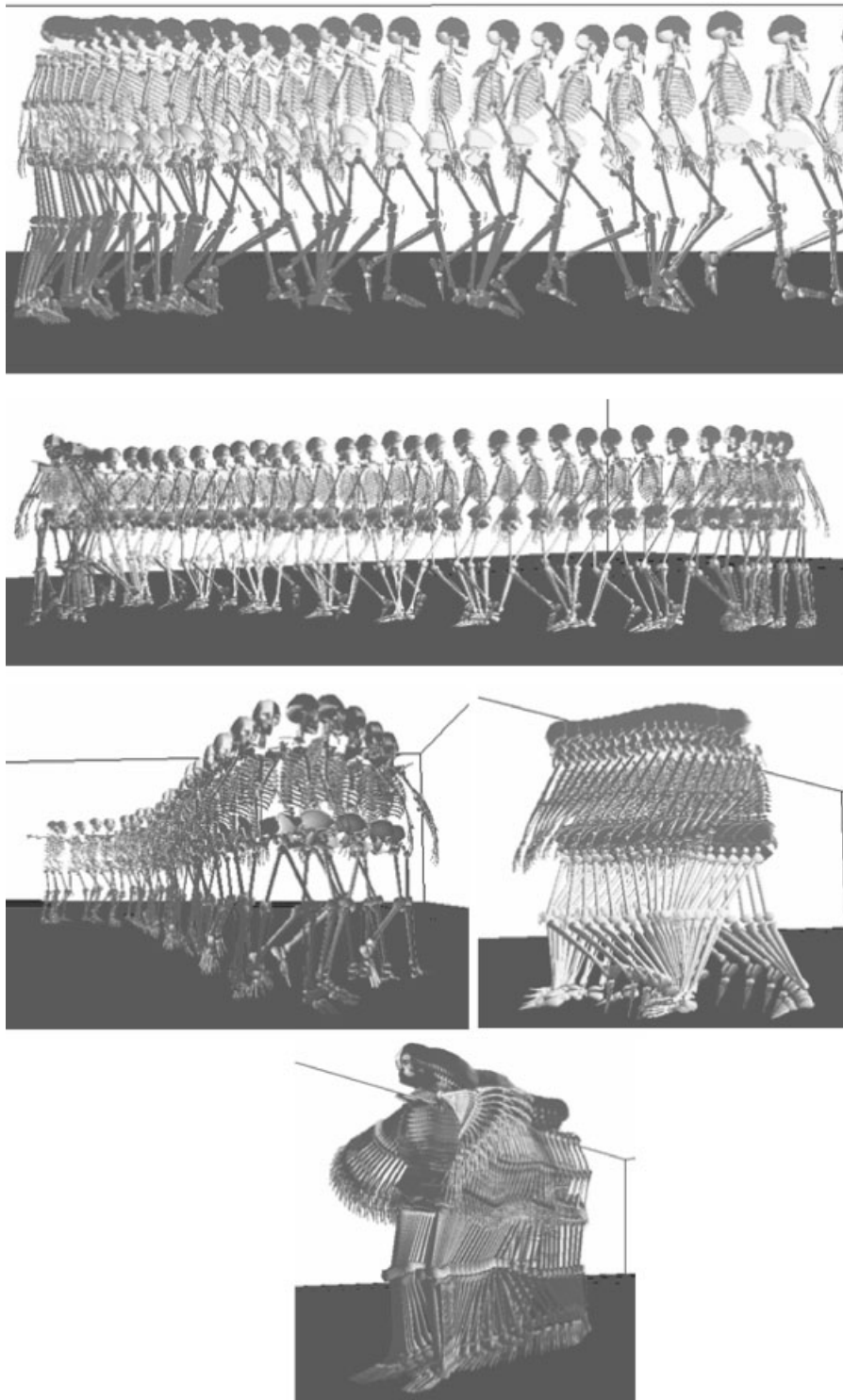


Figure 10. Animation results.

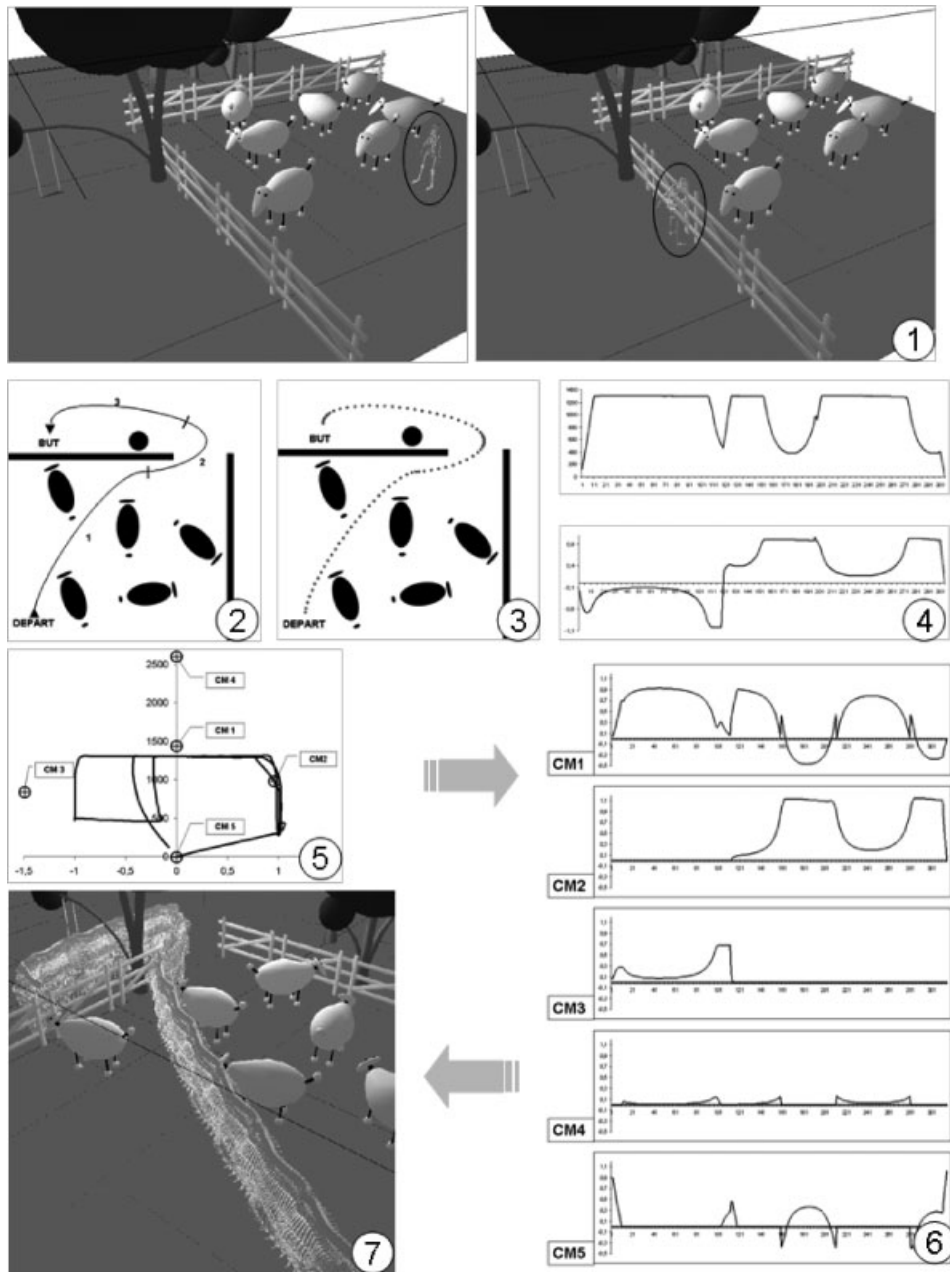


Figure 11. A complete motion-planning problem.

operations can be the origin of these defaults. The literature proposes improvements to motion-blending operators³³ or adequate preparation of motion data⁴¹ in order to solve (or at least lower) these faults. Here, our main concern is the controllability of the locomotion, whereas necessary refinements of the produced animations should be addressed in future work, as detailed in the Conclusion.

The fifth image illustrates an extended feature of the controller: any stop position can be used. When the mannequin decelerates and stops, the chosen stop position is interpolated progressively within the locomotion captures. As a result, the motion progressively derives towards this position. In the motion library, a position can be represented in the same manner as a locomotion cycle. Its frequency spectrum is defined with only α_0

values whereas the other pairs (α_i, β_i) are equal to zero. In the displayed example, a stop position where the actor opens the arms is used.

A Worked-Out Example of Use in a Motion-Planning Context

We now illustrate our locomotion controller on a motion-planning problem. Figure 11 provides an example that takes place in a virtual outside environment. The input data are the geometric modelling of the environment (a soup of 3D polygons), the articulated mannequin, a start configuration and a goal to be reached. A collision-free motion is then computed in an automatic way without any operator tuning. Here are the main steps of the process (see references 42–44 for details).

- *Step 1.* The first images in Figure 11 illustrate the environment in which the problem takes place and the user directives (the initial and goal positions of the mannequin). The mannequin stands in a sheep pen. He is ordered to get out of this pen and to stand in front of the wooden barrier.
- *Step 2.* The motion planner provides a solution to this problem. The motion planner is Move3D.⁴⁵ The planner relies on a probabilistic motion-planning method.⁴⁶ The human locomotion path is a sequence of third-degree Bezier curves.

Briefly, such a motion planner relies on a roadmap (an oriented graph), which captures the connectivity of the collision-free configuration space. The nodes of the roadmap are collision-free positions of the mannequin, whereas the arcs are collision-free local paths (computed by the steering method). The algorithm tends to connect new nodes in the roadmap in a random manner. The process stops as soon as the initial and the goal position are connected in the roadmap or it stops after some fixed computation time if no solution is found.

In the case of our example, the result is composed of four elementary Bezier curves, joining in a natural way the user-defined initial and final configurations.

- *Step 3.* The previously computed path is transformed into a trajectory. Given the chosen frame rate for the animation, the trajectory is sampled.
- *Step 4.* From the previous step, we can immediately deduce the evolution of the inputs $(v_d, \omega_d)_t$, as displayed on the graph.
- *Step 5.* The fifth image of Figure 11 represents the content of the used motion library in the (v, ω) space.

We have considered a reduced set of example motions: a stop position (CM5 on image 5), a normal straight walk (CM1), two turns (CM2 and 3) and a run (CM4) compose the library. The graph represents the projection of the evolution of the controls in the (v, ω) space. Note that the speed limits previously mentioned have been chosen in correlation with the content of the library (and especially the aspect of the envelope of the reachable speeds, introduced in Figure 7).

- *Step 6.* At this stage, the evolution of the influence of each motion capture is computed. The ‘normal straight walk’ (CM1) is the most influential of the motion examples, whereas the ‘run’ (CM4) has a low interference. Indeed, the linear speed is limited by too small a value (1.3 m s^{-1}). The influence of the right and left turns alternate with respect to the trajectory curvature. One can finally observe that some negative weights are sometimes attributed to the motion samples. Indeed, the directives overtake the envelope of the reachable speeds.
- *Step 7.* Thanks to the previous computations, the interpolation formulas can be parameterized for each posture to compute, and the animation is synthesized.

Conclusion

We have presented a motion capture editing-based locomotion controller for virtual mannequins. The solution supports a continuous change of the desired linear and the angular velocities of the mannequin’s displacement. It allows real-time use. The believability of the results is appreciable (video can be found at <http://www.laas.fr/RIA/RIA-research-motion-character.html>). One original aspect of this approach is the control space-based selection and weighting of the blended captures. We have revisited and extended motion-blending techniques working in the frequency domain.

The controller can be improved in several ways. One limitation of the method is the requirements on the captures constituting the motion library. Our technique for analysing automatically the content of the captures is limited to restricted cases. Identifiable trajectories should be extended to more general cases. Our directives are limited to locomotion speeds. The control space dimension may be extended to extend the controller ability to decline the locomotion styles on other criteria such as tiredness and anger. We are also limited to a flat

terrain. At first guess, we could envisage a (v, ω, \dot{z}) exactly on the same principle as those exposed in this article for a (v, ω) control. Finally, a motion blending-based technique can produce inconsistent motion (foot stakes) due to capture quality. The use of filters can be envisaged to solve such problems (see Kovar *et al.*⁴¹ for an example of a foot stake clean-up filter).

Our solution has been mostly evaluated in a motion-planning context. We hope to test it in other contexts, especially a more interactive one such as a video game.

ACKNOWLEDGEMENTS

We would like to thank F. Multon, R. Boulic, C. Esteves and F. Forge for their valuable help. This work is supported by the European Project FP5 IST 2001-39250 Movie.

References

1. Latombe JC. *Robot Motion Planning*. Kluwer: Boston, MA, 1991.
2. Laumond JP (ed.). *Robot motion planning and control*. In *Lectures Notes in Control and Information Science*, Vol. 229. Springer: Berlin, 1998.
3. Badler NI, Phillips CB, Webber BL. *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press: Oxford, 1992.
4. Earnshaw R, Magnenat-Thalmann N, Terzopoulos D, Thalmann D. Computer animation for virtual humans. *IEEE Computer Graphics and Applications* 1998; **18**(5): 20–23.
5. Parent R. *Computer Animation: Algorithms and Techniques*. Morgan-Kaufmann: San Francisco, 2001.
6. Magnenat-Thalmann N, Thalmann D (eds). *Interactive Computer Animation*. Prentice-Hall: Englewood Cliffs, NJ, 1996.
7. Watt A, Watt M. *Advanced Animation and Rendering Techniques: Theory and Practice*. ACM Press: New York, 1992.
8. Khatib O, Brock O, Chang K, Conti F, Ruspini D, Sentis L. Robotics and interactive simulation. *Communications of the ACM* 2002; **45**(3): 46–51.
9. Yamane K. Simulating and generating motions of human figures. In *Springer Tracts in Advanced Robotics*, Vol. 9. Springer: Berlin, 2004.
10. Zelter D. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications* 1982; **2**(9): 53–59.
11. Witkin A, Kass K. Spacetime constraints. In *Proceedings of ACM SIGGRAPH*, 1988.
12. Hodgins JK, Wooten WL, Brogan DC, O'Brien JF. Animating human athletics. In *Proceedings of ACM SIGGRAPH*. Addison-Wesley, Reading, MA, 1995.
13. Shiller Z, Yamane K, Nakamura Y. Planning motion patterns of human figures using a multi-layered grid and the dynamic filters. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'01)*, 2001.
14. Faloutsos P, van de Panne M, Terzopoulos D. Composable controllers for physic-based character animation. In *Proceedings of ACM SIGGRAPH 2001 Conference*, 2001.
15. Bruderlin A, Williams L. Motion signal processing. In *Proceedings of SIGGRAPH'95*, 1995.
16. Witkin A, Popovic Z. Motion warping. In *Proceedings of SIGGRAPH'95*, 1995.
17. Unuma M, Anjyo K, Takeuchi R. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH'95*, 1995.
18. Wiley D, Hahn J. Interpolation synthesis for articulated figure motion. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS'97)*, 1997.
19. Park S, Shin H, Shin S. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA'02)*, 2002.
20. Rose CF, Cohen MF, Bodenheimer B. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications* 1998; **18**(5): 32–41.
21. Sack J, Urrutia G (eds). *Handbook of Computational Geometry*. Elsevier Science: Amsterdam, 2000.
22. Bernstein N. *The Coordination and Regulation of Movements*. Pergamon Press: Oxford, 1967.
23. Beckett R, Chang K. An evaluation of the kinematics of gait by minimum energy. *Journal of Biomechanics* 1968; **1**: 147–159.
24. Townsend MA, Seireg A. The synthesis of bipedal locomotion. *Journal of Biomechanics* 1972; **5**(1): 71–83.
25. McMahon TA. Mechanics of locomotion. *International Journal of Robotics Research* 1984; **3**: 4–28.
26. Girard M, Maciejewski AA. Computational modelling for the computer animation of legged figures. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991.
27. Alexander RMcN. The gaits of bipedal and quadrupedal animals. *International Journal of Robotics Research* 1984; **3**(2): 49–59.
28. Multon F, France L, Cani M-P, Debunne G. Computer animation of human walking: a survey. *Journal of Visualization and Computer Animation* 1999; **10**: 39–54.
29. Sun HC, Metaxas DN. Automating gait generation. In *Proceedings of SIGGRAPH'01*, 2001.
30. Kovar L, Gleicher M, Pighin F. Motion graphs. In *Proceedings of SIGGRAPH'02*, 2002.
31. Lee J, Chai J, Reitsma PSA. Interactive control of avatars animated with human motion data. In *Proceedings of SIGGRAPH'02*, 2002.
32. Choi MG, Lee J, Shin SY. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM transactions on Graphics* 2003; **22**(2): 192–203.
33. Kovar L, Gleicher M. Flexible automatic motion blending with registration curves. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation (SCA'03)*, 2003.

34. Ménardais S, Kulpa R, Multon F. Synchronization of inter-actively adapted motions. In *ACM, SIGGRAPH/EUROGRAPHICS Symposium of Computer Animation*. ACM Press: New York, 2004.
35. Boulic R, Ulicny B, Thalmann D. Versatile walk engine. *Journal of Game Development* 2004; 1(1): 29–50.
36. Glardon P, Boulic R, Thalmann D. A coherent locomotion engine extrapolating beyond experimental data. In *Proceedings of Computer Animation and Social Agent (CASA)*, 2004.
37. Boulic R, Magnenat-Thalmann N, Thalmann D. A global human walking model with real-time kinematics personification. *The Visual Computer* 1990; 6(6): 344–358.
38. Delaunay BN. Sur la sphère vide. In *Proceedings of the International Mathematics Congress*, Toronto, 1924.
39. Lamiroux F, Laumond JP. From paths to trajectories for multi body mobile robots. In *Fifth International Symposium on Experimental Robotics (ISER'97)*, 1997.
40. Bézier P. *Courbes et Surfaces* (2nd edn). Hermès: Paris, 1986.
41. Kovar L, Gleicher M, Schreiner J. Footstake cleanup for motion capture. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation (SCA'02)*, 2002.
42. Pettré J, Laumond JP, Siméon T. A 2-stage locomotion planner for digital actors. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'03)*, 2003.
43. Pettré J. Planification de mouvements de marche pour acteurs digitaux. PhD thesis, University of Toulouse III, 2003.
44. Pettré J, Siméon T, Laumond JP. Planning human walk in virtual environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2002)*, 2002.
45. Siméon T, Laumond JP, Lamiroux F. Move3D: a generic platform for motion planning. In *Fourth International Symposium on Assembly and Task Planning (ISATP'01)*, 2001.
46. Kavraki L, Svetska P, Latombe JC, Overmars M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *Proceedings of IEEE Transactions on Robotics and Automation*, 1996.
47. Boulic R, Mas R, Thalmann D. A robust approach for the center of mass position control with inverse kinematics. *Journal of Computers and Graphics* 1996; 20(5): 693–701.

Authors' biographies:



Julien Pettré is about to start postdoctoral research at the VRLAB of the EPFL. He qualified as an engineer in 1998 and received a PhD in Robotics from Toulouse III University in 2003, after a 4-year stay at LAAS-CNRS. His research interests include digital actor animation, motion capture-based control and collision-free path planning.



Jean-Paul Laumond is Directeur de Recherche at LAAS-CNRS in Toulouse, France. In Fall 1990 he was invited to be a senior scientist at Stanford University. He has been a member of the French Comité National de la Recherche Scientifique from 1991 to 1995. He is currently a member of the board of the ACI Neurosciences Intégratives et Computationnelles. He has been coordinator of two European Esprit projects—PROMotion (1992–1995) and MOLOG (1999–2002)—both dedicated to robot motion planning technology. In 2001 and 2002 he created and managed Kineo CAM, a spin-off company from LAAS-CNRS devoted to developing and marketing motion planning technology. Kineo CAM was awarded the French Research Ministry prize for innovation and enterprise in 2000. He teaches robotics at ENSTA and Ecole Normale Supérieure in Paris. He has edited three books. He has published more than 100 papers in international journals and conferences in computer science, automatic control and robotics.