Planning human walk in virtual environments

J. Pettré, T. Siméon, J.P. Laumond

LAAS/CNRS,
7, avenue du Colonel Roche, 31077 Toulouse Cedex 04 - France
{jpettre,nic,jpl}@laas.fr

Abstract

This paper presents a method for animating human characters, especially dedicated to walk planning problems. The method is integrated in a randomized motion planning scheme, including a steering method dedicated to human walk. This steering method integrates a character motion controller assuming realistic animations. The navigation of the character through the virtual environment is modeled as a composition of Bezier curves. The controller is based on motion capture data editing techniques. This approach satisfies some essential computer graphics criteria : realistic result, low response time, collision-free motion in possibly constrained 3D environments. The approach has been implemented and successfully demonstrated on several examples.

1 Introduction

Animating human characters has become a very active research field since cinematographic and entertainment industries have demonstrated the greatness of the application field. Many animation techniques recently developed allow to reduce production time and also improve the realism of the animations[4]. Animating automatically virtual actors raises problems due to models and control complexity. Indeed, automation suppose motion synthesizing capabilities. On one hand, realistic human motion controllers exist in computer animation literature[4][14]. On the other hand, robotics has developed algorithms efficient over complex and constrained motion planning problems[12][13]. Our contribution is to take advantage from both sides. We propose a solution for planning human walk through virtual environments, taking advantage of randomized motion planning algorithms, combined to a character motion controller based on motion capture editing.

Walking virtual actors Through the character animation problem [4], walking is crucial for navigating in a virtual world. Synthesizing walk animations is traditionally approached by three techniques: kinematic (forward and inverse), dynamic and motion data based.
Kinematic approach is commonly dedicated to hand made animations, eventually assisted by specific interpolation techniques to generate transition positions. Inverse kinematics reduces the body positioning time, as animator acts on the end effector only. It also can be used as an animation filter in order to solve kinematic constraints violation [2]. A dynamic approach is described in [5]. Respecting physical laws intrinsically increase realism but high computation power is needed. Motion capture data based techniques are presented in [20], [3] and [16]. The data are fixed recorded motion sequences warranting realism[1]. Moreover, re-targeting is possible to animate different trajectories thanks to motion blending and warping. Motion blending [19] consists in interpolating parameters of several motions in order to produce new motions, whereas motion warping [20] consists in modifying a single motion in order to fit to a new trajectory. See [16], [14] or [7] for a more detailed state of art.

Path planning Motion planning algorithms are well-developed in robotics literature [12]. Robot motion planning is concerned by collision free motion in constrained environments, system's control[13] and complex tasks achievements. Randomized motion planning is an efficient solution taking into account these constraints: description is given in[9], [8].

Human motion planners Closing together motion planning algorithms issued from robotics and human character animation has been approached in [6] and [15]. Applications to humanoid robots control are also presented in [11] and [10] with stability constraints. In [6], Kuffner divides path planning and path following problems. A dynamic environment is considered. A path is planned for the character's bounding cylinder, a controller is used to produce an animation along this path. A collision invalidates the path and process is repeated. In

[15], Raulo also considers dynamic environments. During planning phase, a simplified representation of environment is used. The "Virtual Robot" tracks the planned trajectory, while informations are collected. The informations allow to choose a strategy for modifying the resulting animation and solving collisions.
Our strategy differs by integrating the human motion control during the path planning. Consequently, our contribution consists in synthesizing a steering method integrated in a randomized planner scheme. The potential of randomized motion planner algorithms over highly constrained problems, respecting low computing times, benefits to our solution.

In section 2, the model of the human character is introduced. Section 3 described first level of the steering method, while the core is detailed in section 4. Section 5 describes integration and implementation of the method in a randomized motion planner architecture.

2 Model and Motion Data

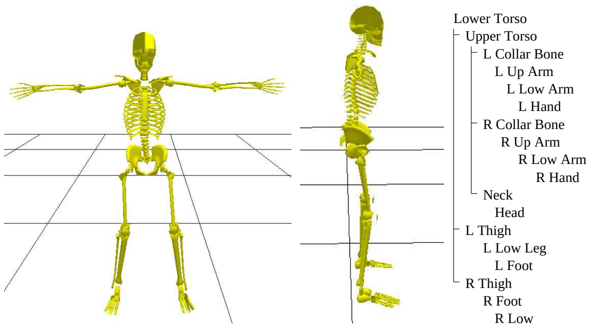

Figure 1: *Character's model*

Our the virtual actor is modeled as a classic opened kinematic structure with 57 degrees of freedom (dof) detailed on figure 1. "Lower Torso" is the root of the kinematic structure. Position and orientation of the root gives us the situation of the character, noted $[x, y, z, \theta, \phi, \psi]$. The whole configuration of the character is then given by it's situation attached to the angular values of each dof, and is noted $[x, y, z, \theta, \phi, \psi, q_1, ..., q_n]$. Motion capture data depend on the model and represent the evolution of the configuration through the time, in several cases : walks, turns, runs, etc.

3 Root's trajectory

Root's trajectory problem concerns only $[x, y, \theta]$ parameter's evolution. Inputs of this functionality are

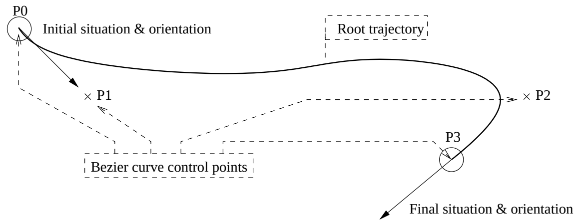

Figure 2: *Example of a root trajectory*

two configurations of the virtual actor's free configuration space ($\mathcal{C}_{free}$). Human's local navigation is modeled as a third degree Bezier curves. As a result, global path resulting from the whole planner algorithm is a composition of Bezier curves respecting C^1 continuity constraint, i.e. a B-Spline. Control points are computed taking in account initial and final θ values and in-between configurations distance. Parameter's evolution are given by cartesian parametrics equations :

$$\begin{cases} x(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)} t^i (1-t)^{n-i} P_{xi} \\ y(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)} t^i (1-t)^{n-i} P_{yi} \\ \theta(t) = arctan(y'(t)/x'(t)) \end{cases} \quad (1)$$

with $n = 3$, $0 \leq t \leq 1$, and P_{xi} the x coordinate for the control point number i, etc. Other values are needed : arc length, linear and rotation speeds evolution. Parameters can be analytically derived from equation 1.

4 Animation Module

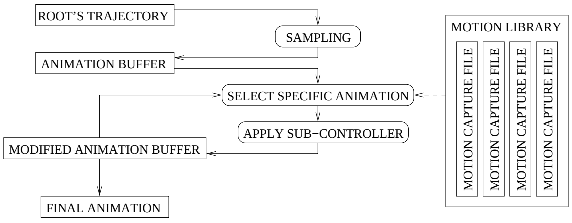

Figure 3: *Animation Module Structure*

The animation module is composed by two elements : a motion library and a set of specific sub-controllers. Successive application of sub-controllers using some motion data of the library compose our human motion controller. The principle of the module is to sample the previously computed root's trajectory. The animation buffer stores the samples. The content of the motion library is scanned, and specific sub-controllers are applied. The process is described on figure 3. The animation module is designed to be easily modified by adding new sub-controllers.

4.1 Motion Library

The motion library is a container for motion capture files. Each motion capture file is pre-processed: filtered (in the case of walk cycles) and characterized as explained in next sections. The motion library eventually contains different walk cycles, and other actions descriptions : movements, postures, etc.

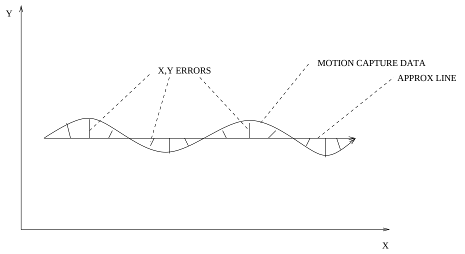

Figure 4: *Computing (x,y) motion capture data situation's errors around an approximative trajectory*

Preparing Walk Cycles. The walk sub-controller is based on motion capture blending and warping ([3],[20] and [16]). This method needs a preparation of motion data as supposed to be interpolated (harmonization need), repeated along the trajectory (cyclic data need).
The user is in charge to select in a given motion capture file, a walk sequence approximatively cyclic, and to add descriptors (see section 4.1). This operation is only made once to format data.
Next phases of preparation are automatically done during the planner's initialization phase. In the case of straight walks, parameters of the position $x(t), y(t), z(t)$ given by the motion capture data are approximated by a line whose characteristics are computed with a linear less squares fitting technique, defining $\tilde{x}(t), \tilde{y}(t), \tilde{z}(t)$ and average orientations. In the case of a turning walks, the approximation is a circle portion. Some parameters derived from the approximation are kept: average linear and rotation speeds.
Situation data $[x, y, z, \theta, \phi, \psi]$ are computed as errors around the approximation previously done. Errors are computed by $\epsilon_x = \tilde{x}(t) - x(t)$, etc. This computation is illustrated on figure 4 over both $(x(t), y(t))$ parameters. The set of parameters defining motion is now noted $[\epsilon_x, \epsilon_y, \epsilon_z, \epsilon_\theta, \epsilon_\phi, \epsilon_\psi, q_1, ..., q_n]$ and is almost cyclic. Fourier expansions are computed over these parameters, and a low-band filter applied [19] in order to make sequences cyclic. Consequently, a set of parameters (α_k, β_k) characterize each motion style. The evolution of a parameter p can be computed using equation 2.

$$p(t) = \alpha_0 + \sum_{i=1}^{n} \alpha_i cos(2\pi ti) + \beta_i sin(2\pi ti) \qquad (2)$$

where $0 \leq t \leq 1$.

Characterising Files. Adding descriptors to the motion data is crucial. As filters differ from a file type to another, the user has to mention characteristics. First, motion data are not limited to walk sequences; they possibly correspond to specific postures or movements. Also, some data do not concern all character's dof (eg. an arm-only movement description). The two following characteristics must be given :

- Animation type: for example walk (with sub-characteristic : straight or not), movement, posture, blocking task (character must stop to execute the movement) etc.
- Animation chronological validity: the motion controller needs to know when the motion data must be taken into account. Walk cycles are generally always valid. But we could imagine that run cycles are valid only when the character is far from obstacles. Also, movement and postures are generally valid at a given moment : at starts or ends of paths, etc.

4.2 Motion Controller

Sub-controllers. A sub-controller is an application rule dedicated to a data file type. The inputs are motion data, validity moments and actual state of the animation buffer. The output is a modification to the animation buffer. The set of all sub-controllers present in the animation module constitute our character's motion controller.

Walk controller This controller solves navigation problems. We explain it through an example with three motion capture files: straight, left turn and right turn motion cycles.
The principle of the walk controller is to scan each frame of the animation buffer. Given the frame's situation parameters, local linear and rotation speeds (v, ω), the walk controller computes a locally valid set of parameter (α_b, β_b) and generates a matching configuration from these parameters, projected on the animation frame's situation. Fitting the motion data to the linear speed value is realized using warping techniques. Motion data blends are used to fit to the rotation speed value.
The controller's algorithm is schematically illustrated on figure 6. For readability reasons, we illustrate the process of $[x, y]$ parameters only. On figure 5, parameters issued from root's trajectory are given (see section 3 for equations). Figure 6 shows the initial motion data at the top (errors around approximated trajectory), and the evolution of these errors through warp and blend operations.

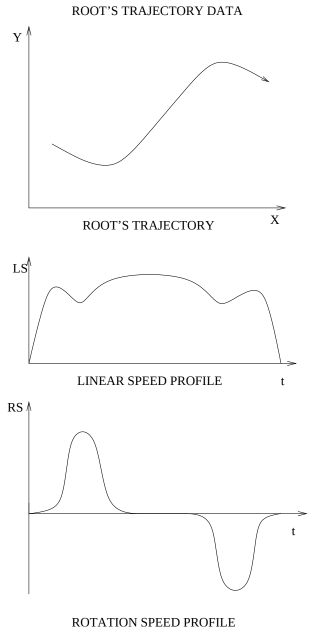

Figure 5: *Root's trajectory available data*

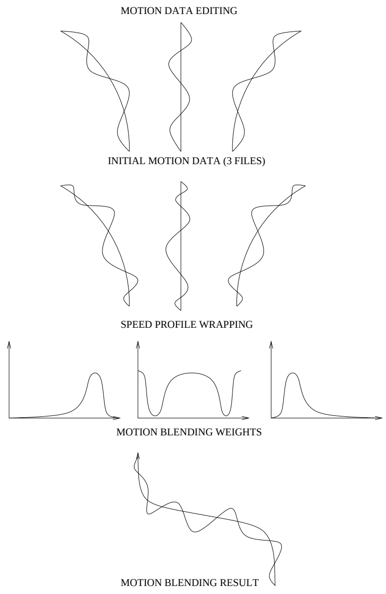

Figure 6: *Motion warp and blend steps*

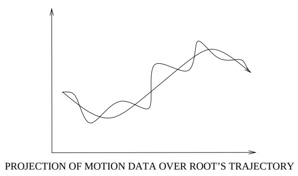

Figure 7: *Synthesized motion projection*

The projection over root's trajectory of the motion editing result is illustrated on figure 7.
Warping a motion cycle to fit to the linear speed is realized by applying a coefficient to the motion specific set of parameters (α_k, β_k). The same coefficients are applied to the average speeds (described in section 4.1), in order to be able to reconstruct correctly the motion.
Blending different motion data consists in constructing a motion with a desired average rotation speed. This is made by interpolating the motion data with different known rotation speeds. For example a slow left turn is obtained by interpolating a rapid left turn with a straight walk. Average rotation speeds of motion cycles (left, straight, right) are noted $(\omega_1, \omega_2, \omega_3)$. The weights $a(t), b(t), c(t)$ of each motion respectively defined with (α_1, β_1), (α_2, β_2) and (α_3, β_3) can be computed as follows.

$$\omega(t) \leq 0 \Rightarrow \begin{cases} a(t) = abs(\frac{\omega(t)}{\omega_1}) \\ b(t) = 1 - a(t) \\ c(t) = 0 \end{cases}$$

$$\omega(t) > 0 \Rightarrow \begin{cases} a(t) = 0 \\ b(t) = 1 - c(t) \\ c(t) = abs(\frac{\omega(t)}{\omega_3}) \end{cases}$$

Finally, the evolution of blended motion parameters $(\alpha_b, \beta_b))$ is computed :

$$\begin{cases} \alpha_b(t) = a(t)\alpha_1 + b(t)\alpha_2 + c(t)\alpha_3 \\ \beta_b(t) = a(t)\beta_1 + b(t)\beta_2 + c(t)\beta_3. \end{cases} \tag{3}$$

A final step consists in finding the moment in the synthesized motion to be used for computing the configuration's parameters. The problem is solved by comparing the distance obtained by executing root's trajectory and the distance obtained by playing the computed motion. When equivalents, an inverse Fourier transform is applied to determine all the configuration parameters values. Finally, a projection of errors on the root's trajectory is done and angular values are copied. The projection's result is illustrated on figure 7.

Other Sub-controllers The animation module architecture is designed to integrate other sub-controllers. For example :

- A partial configuration evolution can be described in a motion capture data: arm-only movement for example. When the chronological validity (defined in subsection 4.1) is verified, contained data replace the animation buffer data on the concerned dof.
- Another sub-controller allows to call the motion planner (with another steering method such as a

linear one) in order to generate a transition between two configurations. This is how postures in the motion library are taken into account.

The evolution of the animation buffer due to successive applications of the sub-controllers is described in the next paragraph.

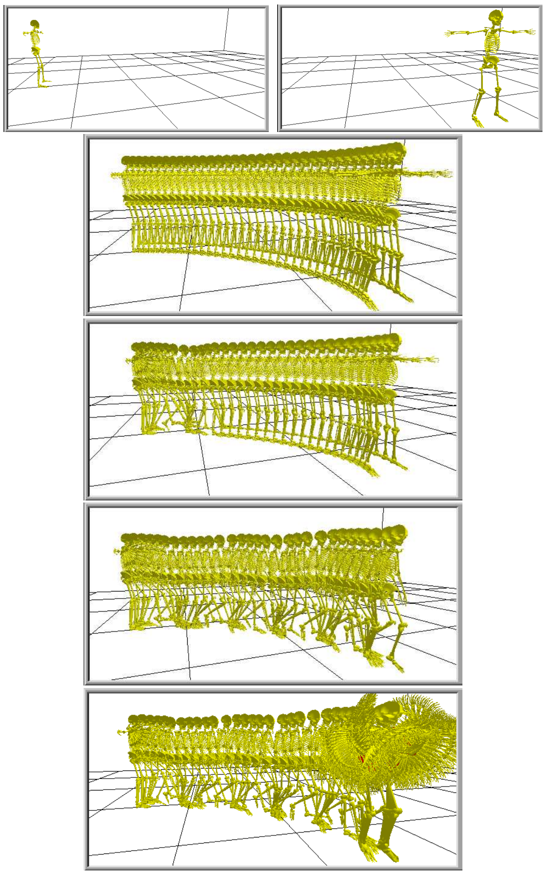

Figure 8: *An example of the construction of an animation : a/ b/ Inputs, c/ Root's Trajectory, d/ Start sequence introduction e/ Walk Cycle introduction f/ Macarena postures introduction*

Evolution of the Animation. The evolution of the animation buffer, between each sub-controller application, is illustrated on figure 8. In order to link different movements, linear interpolations between animation buffer, active motion data, and eventually other motion data are applied.

In this example, a sequence is first inserted to the start the walk . A straight walk cycle is then applied on the remaining part of the trajectory. Transition is made by computing the closest configuration between the data the issued from the walk cycle and the last data computed with the start sequence. A "roll-back" is done, and an interpolation is computed between the starting sequence and the walk cycle in order to smooth the transition.

At last, a set of postures is imposed, which correspond to the macarena's dance steps. This is why at the bottom of the figure 8 the skeleton is shaking arms.

5 Walk planning

The steering method described in the two previous sections has been implemented within Move3D [17]. Move3D is a motion planning platform integrating several randomized algorithms. Main components of such algorithms are:

- the steering method to compute admissible paths,
- a collision checker which is used both to select the nodes of the roadmap and to check whether an admissible path is collision-free or not.
- a roadmap builder in order to generate or extend the roadmap,
- a roadmap explorer in order to solve specific problems (which eventually calls the roadmap builder).

Move3D's global planner uses visibility PRM technique [18] that captures the configuration space connectivity into a small roadmap. Figure 9 shows an example of collision free walk automatically computed by our planner.

Considering the whole kinematic structure and the description of the environment in three dimensions allows us to find solutions in constrained environments. Figure 9 illustrates this potential, with a resulting trajectory close to the furniture (note that the skeleton's arm pass above the piano's stool). This result could not be obtained with a bounding cylinder around the human model.

6 Conclusion

We have presented a human walk planning method associating randomized motion planning and motion capture editing techniques. First results obtained are promising.

Several developments could improve realism, computing time, and application field. We are interested in investigating reactive planning methods to deal with dynamic environments. Character's controller is also in permanent evolution. Fourier expansions are not completely satisfying as applicable only to cyclic motion data. We are investigating a wavelet based solution. Also, we want to develop new sub controllers, especially for environment, objects or characters interaction problems.

Acknoledgment

Thanks to F. Forges from Ex-Machina who gave us the motion capture data used in our implementation.

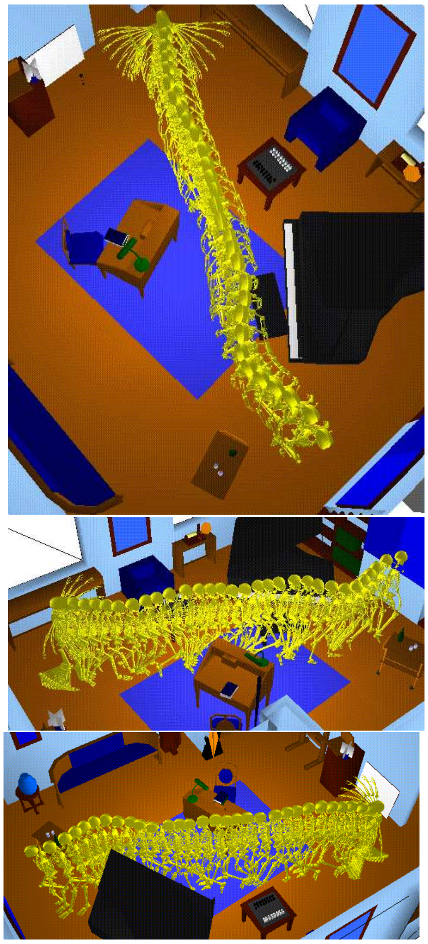

Figure 9: *A complete walk planning through a living room (3 different points of view : a/ b/ c/)*

References

[1] Bobby Bodenheimer, Chuck Rose, Seth Rosenthal, and John Pella. The process of motion capture: Dealing with the data. *Computer Animation and Simulation '97, Eurographics*, pages 3–18, 1997.

[2] R. Boulic, M. Thalmann, and D. Thalmann. A global human walking model with real-time kinematic personification. In *The visual computer*, pages 344–358, 1990.

[3] A. Bruderlin and L. Williams. Motion signal processing. In *Proc. SIGGRAPH'95*, 1995.

[4] Rae Earnshaw, Nadia Magnetat-Thalmann, Demetri Terzopoulos, and Daniel Thalmann. Computer animation for virtual humans. *IEEE Computer Graphics and Applications*, pages 20–23, September/October 1998.

[5] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proc. ACM SIGGRAPH 2001 Conference, Los Angeles, CA*, 2001.

[6] James J. Kuffner Jr. Goal-directed navigation for animated characters using real-time path planning and control. In *CAPTECH*, pages 171–186, 1998.

[7] Shih kai Chung. *Interactively Responsive Animation of Human Walking in Virtual Environments.* PhD thesis, George Washington University, May 2000.

[8] L. Kavraki and J. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA), pp. 2138-2145, 1994.*, 1994.

[9] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical Report CS-TR-94-1519, 1994.

[10] James Kuffner, Masayuki Inaba, and Hirochika Inoue. Automating object manipulation tasks for humanoid robots. *Proc. of 1st Int. Conf on Robotics and Mechatronics (ROBOMEC'00)*, pages 2P2–79–103, 2000.

[11] James Kuffner, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Graphical simulation and high-level control of humanoid robots. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'00)*, 2000.

[12] Jean-Claude Latombe. *Robot Motion Planning.* Boston: Kluwer Academic Publishers, Boston, 1991.

[13] Jean Paul Laumond. *Robot motion planning and control.* Springer-Verlag, 1993.

[14] Franck Multon, Laure France, Marie-Paule Cani, and Gilles Debunne. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation*, 10:39–54, 1999. Published under the name Marie-Paule Cani-Gascuel.

[15] D. Raulo, J.M. Ahuactzin, and C. Laugier. Controlling virtual autonomous entities in dynamic environments using an appropriate sense-plan-control paradigm. In *Proc. of the 2000 IEEE/RS. International Conference on Intelligent Robots and Systems*, 2000.

[16] Charles F. Rose. *Verbs and Adverbs : Multidimensionnal motion interpolation using radial basis functions.* PhD thesis, Princeton University, June 1999.

[17] T. Siméon, JP. Laumond, and F. Lamiraux. Move3d: a generic platform for motion planning. In *4th Internation Symposium on Assembly and Task Planning, Japan.*, 2001.

[18] T. Siméon, J.P. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. In *Advanced Robotics Journal 14(6)*, 2000.

[19] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *Proc. od SIGGRAPH'95*, 1995.

[20] A. Witkin and Z. Popovic. Motion warping. In *Proc. SIGGRAPH'95*, 1995.