Virtual Human Locomotion Synthesis: A Survey

Xochitl Hernández V., Abraham Sánchez L. and David Nuñez R.

Facultad de Ciencias de la Computación, BUAP
14 Sur esq. San Claudio, CP 72570
Puebla, Pue., México
hevexo3@hotmail.com, asanchez@cs.buap.mx, dnunezr@yahoo.com.mx

Abstract. The ideal animation engine is the one that exactly matches the human motions, with its subtle and unpredictable changes and variations. Virtual characters need to navigate in order to interact with their environment. The autonomy is ensured by locomotion controllers to accomplish navigation tasks. This paper surveys the set of techniques developed in Computer Graphics for virtual human locomotion synthesis. We review the recent advances in this field by presenting some of them in our multipurpose tool developed for research and teaching.

1 Introduction

The synthesis of realistic human motion is a challenging research problem with broad applications in movies, special effects, cartoons, virtual environments and games. Due to the quality and realism of the result, the use of motion captured data has become a popular and effective means of animating human figures. Since it is an inherently off-line process, there has been great interest in developing algorithms that are suitable for interactive applications [1].

Traditionally, human animation has been separated into facial animation and body animation, mainly because of lower level considerations. Facial animation results primarily from deformations of the face. Controlling body motion generally involves animating a *skeleton*, a connected set of segments corresponding to limbs and joints. Using geometric techniques, animators control the skeleton locally and define it in terms of coordinates, angles, velocities, or accelerations. The simplest approach is motion capture.

We introduce a multipurpose tool for the animation of virtual characters (MAVIC) as an available simulation platform for research and teaching. MAVIC is an extensible simulation framework and rapid prototyping environment for computer animation.

MAVIC offers solutions to the problems of flexible reuse of controllers, actuators and interactive participation in 3D animation. In addition, MAVIC provides basic modeling capabilities and support for kinematic animation such as motion capture and keyframing. In this first version, the tool includes some elements for the support of physical simulation of virtual humans.

The paper is organized as follows. Section II presents the state of the art in human virtual animation. To create realistic animation, Section III describes the

Received 14/03/09
Accepted 27/04/09
Final version 05/05/09

two-stages locomotion planning as an effective new approach in this direction. Section IV presents the various modules integrated in our system from locomotion planning to the dynamic control. Finally, we give some concluding remarks and future work in Section V.

2 State of the Art in Character Animation

Existing motion synthesis techniques can be divided into three research directions: hand-driven, model-driven and data-driven methods [2], [3], [4], [5], [6], [7], [8].

The hand-driven methods are the oldest and simplest ways for animation creation. An animator determines manually postures ("keyframes") of an articulated character by defining positions and orientations on its joints at specific animation times ("key-times"). The final motion results in a smooth interpolation between the corresponding key-frames.

Model-driven methods explicitly describe how movements are generated, and use computation to propagate changes in high-level parameters to changes in low-level parameters. Therefore, these methods typically have a small number of high-level parameters which can be modified to change the generated motion. We can distinguish two classes of model-driven techniques, either based on kinematics or physics.

Motion capture technique offers an alternative source of highly realistic movement sequences used for a variety of data-driven motion synthesis methods. The motion acquisition systems allow the estimation of the joint positions and/or orientations of a real performer. These parameters are then adapted to a virtual character in order to result in an animation which reproduces the original motion accurately.

All presented methods, classified by hand-driven, model-driven and data-driven techniques have advantages and disadvantages. Illustrated in Figure 1, the different approaches are summarized according to two criteria: the motion realism and the motion control freedom provided to an animator. Hand-driven techniques allow the creation of realistic animations with a very important freedom for the animator to control the motion, constrained neither by visual realism nor physical accuracy. Actually, most of the 3D movies and games use keyframe animation sequences. However, this very flexible method has a price. It requires incredible investment of time and only skilled and talented designers produce realistic human motions. Model-driven approaches generate motions with less motion control freedom and realism. However, kinematic methods concentrate the control on a small set of high-level parameters, allowing an easier and more intuitive motion parameterization. In addition, the motion is generally created on-the-fly, but suffers from too robotic and jerky movements. The other part of model-driven methods, namely those based on physics, produce more realistic results but need lot of computational time (over several minutes). Moreover, for a valid resulting motion, physical accuracy does not imply visual realism. For example, a grasping movement can be simulated with various physically correct alternatives some of which may appear oddly and unnatural. Apart from these drawbacks, the motion control is driven by many parameters, difficult to directly interpret. Hence, data-driven methods have been developed.

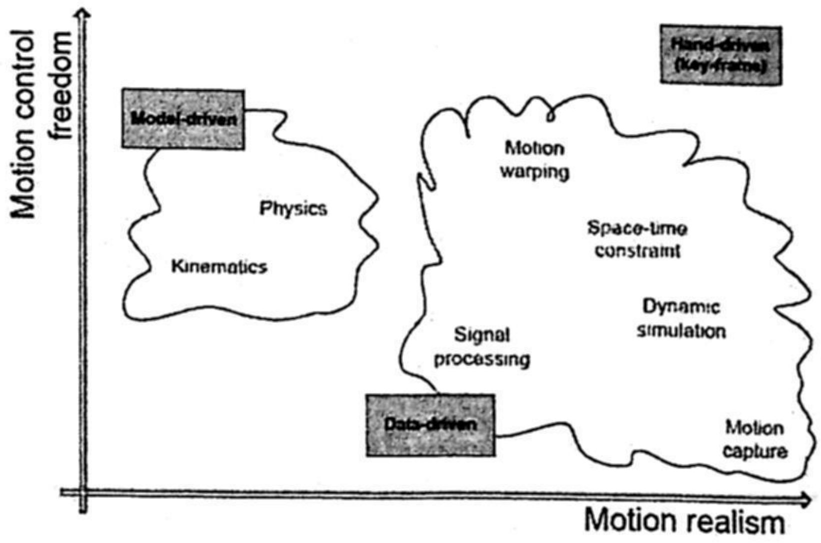

Fig. 1. Summary of motion synthesis approaches according to the motion realism versus motion control freedom for animators.

2.1 Discussion

Since the advent of motion capture systems which return high realistic motion without any control, the number of data-driven methods has exploded. Generally, these approaches produce motions with high realism thanks to the captured input data. The methods based on signal processing offer a weak control on the animation. In fact, the correspondences between motion parameters and frequencies are difficult to establish and the induced filtering can lead to less impressive results as the original motions. Motion warping techniques give lot of control freedom to the animators as key-frames are modified by hand. However, the motion realism is not ensured on the entire final sequences. Results can be improved with space-time constraint methods which consider a motion as a whole continuous sequence. In contrast, the constraint definitions have to be as intuitive as possible, reducing also the animator control. Finally, dynamic simulation allows to modify and enhance an original motion capture data by adding physical accuracy. The main advantage is that simple input data like rough keyframes are sufficient in order to produce convincing results. However, besides their expensive computational cost, these methods are limited to high dynamic motions. Nevertheless, the data acquisition is one of the stumbling blocks for data-driven methods. In fact, motion capture systems are expensive and need lots of pre-processing work to clean the recorded motions before using them.

Moreover, while data-driven synthesis could be applied to any creature whose movements can be captured, in practice it is primarily applicable to humans. In contrast, most of the hand- and model-driven methods can be applied to a wider range of body, with varying topology and joint structure.

3 The Two Stages Locomotion Planner

Motion planning increases the autonomy of virtual characters. Generally, given initial conditions and a goal, a path planner method provides a path to follow. Concerning the special case of locomotion tasks, the planner provides the path taking into account the obstacle avoidance problem. Additionally, this path is transformed into a trajectory which gives the adequate locomotion parameters at each time step. We have been inspired by an interesting proposal that raises the solution to the locomotion problem in two stages [9]. Authors apply PRM (Probabilistic Roadmap Methods) to get a collision-free 2D path in a 3D environment. This path represented by Bézier curves generates a trajectory encapsulating linear and angular speed variations. Finally a walking animation is generated according to those variations.

The two steps of the approach previously mentioned are the following: 1) Bounding the character's geometry by a cylinder allows motion planning for navigation to be reduced to planning collision-free trajectories for this cylinder in 3D. 2) Simply computing a collision-free path in the environment is not enough to produce realistic animation. A motion controller is used to animate the actor along the planned trajectory. We detail the two stages of the approach.

3.1 Path Planner

The RRT approach, introduced in [11], has become the most popular single-query motion planner in the last years. RRT-based algorithms were first developed for non-holonomic and kinodynamic planning problems. RRT-based algorithms combine a construction phase with a connection phase. Until now we do not have knowledge if some author uses this method for the planning process. In fact, in our implementation we can use the unidirectional planners or the bidirectional ones.

3.2 Motion Controller

While motion capture provides eye-believable motion, recorded sequences are fixed. The objectives of a motion controller are to readapt captured sequences to fit a given scenario while preserving their believability. A motion controller computes automatically time-parameterized trajectories driving all the degrees of freedom of a character (or any mechanical system), given the input state that defines a goal to reach.

We address the problem via a geometrical formulation in the two dimensional control space, inspired by Pettré et al [9]. 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 allowing efficient queries for point location and nearest neighbor computations. The control scheme is based on blending operator working from the motion capture library.

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 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 animation (see Figure 2).

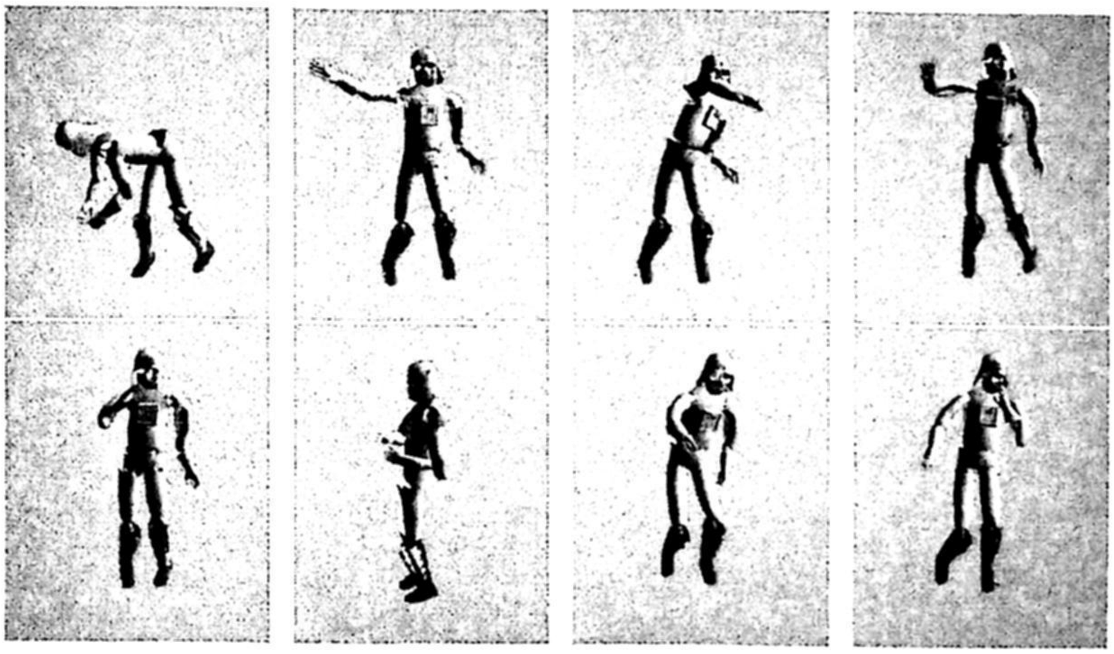

Fig. 2. Examples of different postures obtained with the proposed system. These motions can be characterized by emotional expressiveness or control behaviors.

4 System Design

MAVIC's design focus is on the creation of a modular and open animation system. In some cases, compromises had to be made to find a balance between competing requirements, like speed and data abstraction. Other times, seemingly contrasting goals like tight integration and modularity had elegant solutions that allowed both to be fulfilled. MAVIC was implemented on an Intel © Pentium IV processor-based PC running at 2.6 GHz with 2 GB RAM, using C# and OpenGL. An important feature of our system is the use of ODE as a physics engine, and in addition as collision checker. ODE is an open source, high performance library for simulating rigid body dynamics. It has advanced joint types and integrated collision detection with friction.

Animating very complex model such as virtual humans is usually done by extracting a simpler representation of the model, a skeleton, that is an articulated figure made of rigid links connected by hinges. The number of joints of the model and the degrees of freedom (dofs) depend largely on the desired reality or quality. In this work, we use a relatively simple model with 52 dofs. Motion capture data depend on the model [1], the structure of the virtual human is modeled in two levels. Pelvis and legs are used for the locomotion, all the 18 dofs are said to be active dofs. The 34 other ones are said to be reactive dofs, they deal with the control of the arms and the spine (see Figure 3).

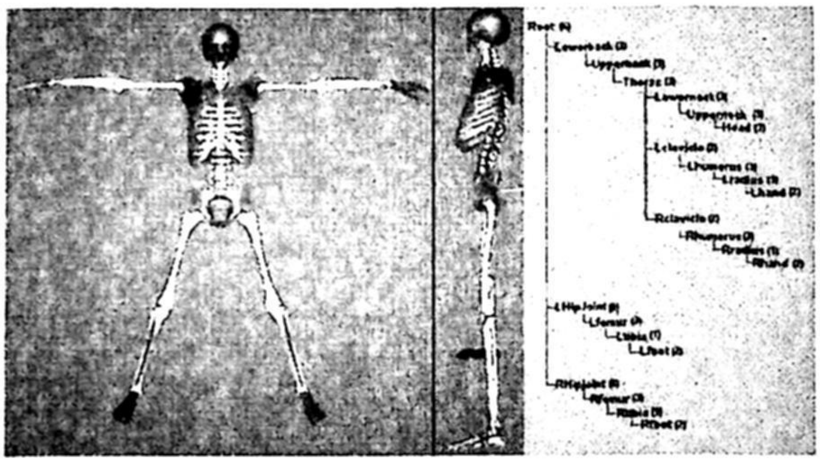

Fig. 3. Functional model for human locomotion.

Human figure walking or running is essentially a quasi-nonholonomic system, since the typical turning radius is usually subject to some minimum value depending upon the velocity [2]. For ease of control, we abstract a human-like character as a particle with an orientation. The particle is constrained to move on a floor. The orientation of the particle is aligned with its velocity. Such a particle is often called as a vehicle. Humans tend to only walk forward, not backward or sideways (no direction reversals during the path following) [10].

The path following phase is modeled as one involving an oriented disc smoothly tracking a geometric path in the plane. The disc center corresponds to the projected point at the base of the character's geometry, and the orientation of the disc corresponds to the character's forward-facing direction (see Figure 4). Since the linear velocity of the disc is constrained to always lie along the forward-facing

[1] The motion library is provided by the CMU Graphics Lab Motion Capture Database.

[2] Of course, a person can turn in place, this will only happen at the beginning or end of a path, not in the middle of a path.

direction, the character can walk or run forward. While turning is modeled by considering the disc's rotational velocity about its center.

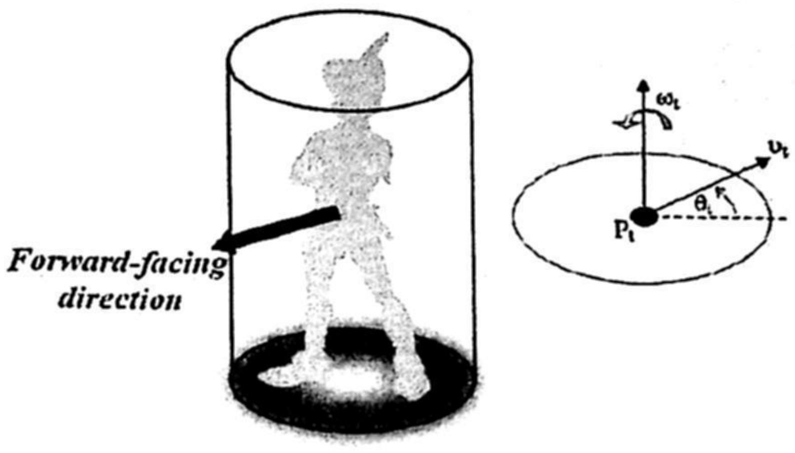

Fig. 4. The character's geometry is bounded by an appropriate cylinder. The center of the disc is the projection of the origin of the root joint of the character onto the walking surface.

A discrete time simulation of the following state variables is used: P_t position (x_t, y_t) of the disc center, θ_t orientation, v_t linear speed along the direction of θ_t, ω_t angular speed about P_t.

The tuple $(P_t, \theta_t, v_t, \omega_t)$ represents the simulated state of the character at time t. At each time step, any combination of the following two controls may be applied: a_t linear acceleration along the direction of θ_t, α_t angular acceleration about P_t. These controls model the four basic fundamental actions for the character (speed up, slow down, turn left, turn right). Speeding up and slowing down are represented by positive and negative values of a_t respectively. Positive values of α_t correspond to left turns, while negative values correspond to right turns.

Once the controls a_t and α_t have been specified, the state variables are integrated forward discretely by the time step Δt. In these experiments, simple fixed-step Euler integration can be used. For this kind of integration, the state propagation equations are approximated by:

$$\begin{aligned}
x_{t+\Delta t} &= x_t + (v_t \cos\theta_t)\Delta t \\
y_{t+\Delta t} &= y_t + (v_t \sin\theta_t)\Delta t \\
\theta_{t+\Delta t} &= \theta_t + \omega_t \Delta t \\
v_{t+\Delta t} &= v_t + a_t \Delta t \\
\omega_{t+\Delta t} &= \omega_t + \alpha_t \Delta t
\end{aligned}$$

The simulation proceeds in this fashion iteratively. As long as the values of the controls are reasonable relative to the size of the time step Δt, the motion will be smooth and continuous. The method to compute the two controls is based on proportional derivative control. Given the current state of the system, a desired state is calculated. The controls are then computed to move the system towards the desired state.

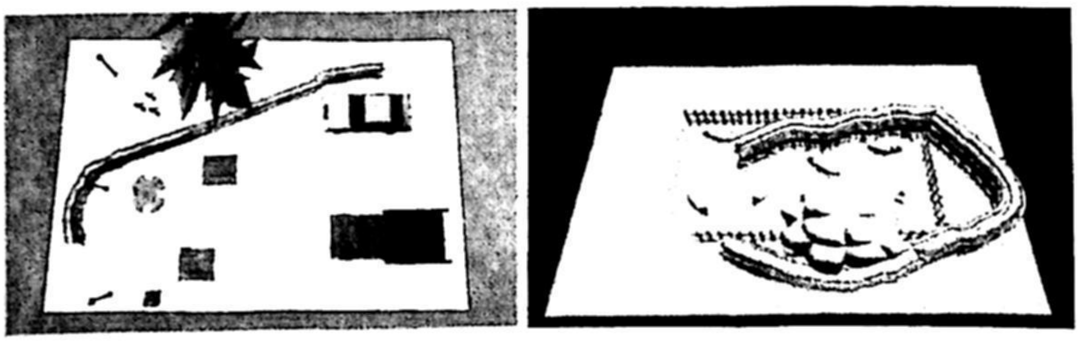
Fig. 5. A complete locomotion planning in two complex environments.

To deform a motion with precise goals, the first solution consists in modifying the body joint orientation in order to get a new posture. In [7], the authors introduce a variant of displacement mapping called motion warping. The animator interactively defines a set of keyframes inducing a set of constraints. These are used to derive a smooth deformation preserving the fine structure of the original motion. However, it is difficult to ensure the geometric constraint enforcement between keyframes. In addition, motion warping methods are purely geometric techniques and operate on each dof independently, without understanding the motion structure. They are not well suited for adjustments requiring coordinate movements, such as grasping actions with modification of object location. In such cases, not only the joint hand is affected, but also other joints like the elbow or shoulder.

In order to alter coherently multiple dofs of an original motion and over a continuous time period, constraint-based techniques can be applied. Discussed and classified in [12], these methods provide effective tools to interactively manipulate a motion clip by changing some important movement properties. Space-time constraints were first introduced to the graphics community by Witkin and Kass [5]. In our system, we have implemented both strategies to avoid collisions of the upper bodies of the virtual character, see Figure 6.

Given an appropriate model of the environment and a simulator, physics-based animation enforces realistic motion. The cost of this realism is the loss of fine-grained control over objects in the scene. An animator can no longer simply specify an object's trajectory or velocities over time, since those values may not be physically achievable. For example, the center of mass of an object in flight, in the absence of any external forces except gravity, is constrained to follow a parabolic path.

While fine-grained control is necessary to achieve certain effects, an animator might prefer to let specific objects behave autonomously. This is particularly

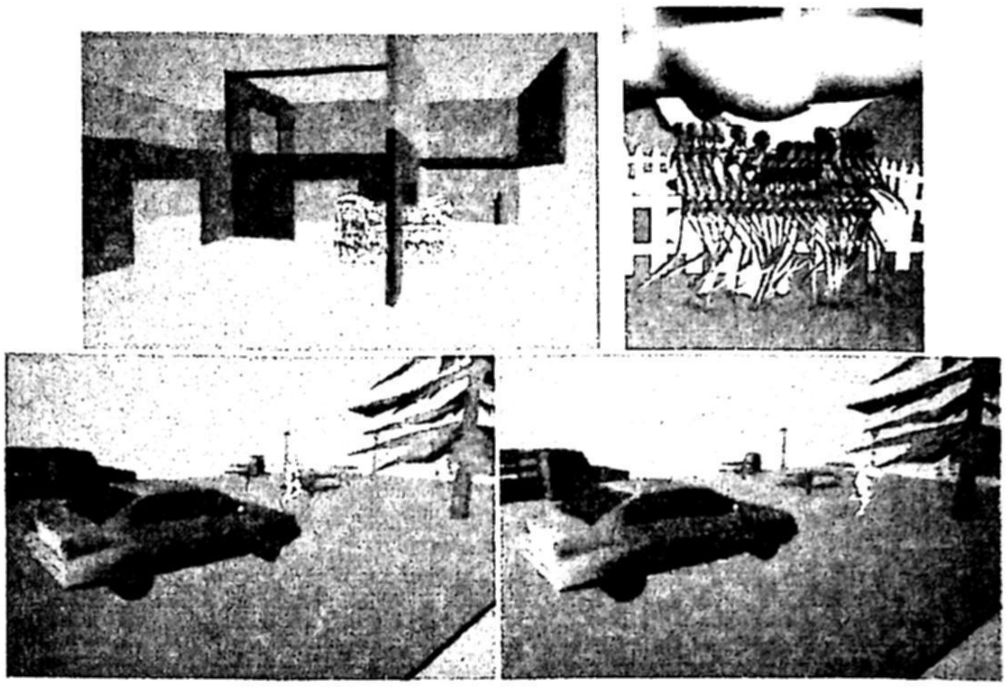

Fig. 6. Motion warping vs. constraint-based techniques.

useful in a scene where many objects are interacting in a complex manner. Rather than tediously specifying the trajectories of all the objects, the animator would often prefer to focus on control of a few foreground characters, allowing the background characters to evolve in a natural manner.

Control of animated objects in a physics-based environment ultimately involves the specification over time of actuations consisting of forces and torques. Given these, the equations of motion determine resulting accelerations, which the simulator then integrates to produce velocities and positions to update the state vector.

Synthesizing the control required to produce a desired motion is a difficult problem. Consider the control that humans use to produce a walking motion. While much progress has been made in a general understanding of the principles of biped locomotion, there has been little progress made toward a controller for physics-based human locomotion that can be applied to general gaits and terrains. Yet, walking is second nature to us; conscious control is rarely required for navigating smooth terrains. In fact, the human body senses and interprets many cues from the environment, using these to orchestrate muscle activity in a manner that puts most sophisticated robotic control systems to shame.

A general solution of the control problem produces a mapping from the state of a system to actions that will produce an appropriate path to a desired goal state. In physics-based animation, the state is described in terms of the dofs of the system and their derivatives. The space spanned by the dofs and their derivatives is called the state-space. The problem can be thought of as a motion

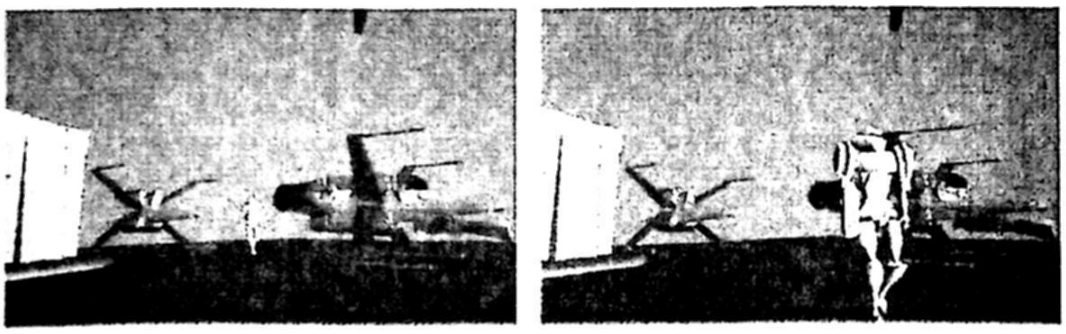

Fig. 7. With the motion controls implemented it is possible in real-time to change of the walking task to the running task.

planning problem through state-space as seen in Figure 8, which shows a two-dimensional space. A major obstacle to solving this problem is dealing with the size of the space, which grows exponentially with the number of dimensions. This is often referred to as the curse of dimensionality. For high-dimensional spaces, it can be difficult or altogether impossible to compute a path between points in the state-space, even when they are close together. These difficulties arise because of the inherent constraint of a physical environment such as inertia or gravity. A path between two points in state-space does not exist when no actuations exist that can control the system to go from one state to the other.

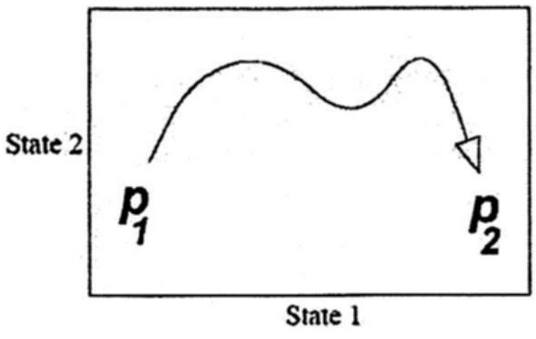

Fig. 8. A path through state-space, p_1 and p_2 are initial and goal states.

In order to make the approach practical, LaValle and Kuffner [13] integrated it within the framework of probabilistic algorithms, with the RRT approach. Here, the author's idea is to propagate the state of the robot into the future by choosing random bang-bang controls. From the state p_1, an exploration tree is thus built by applying random control inputs u over a constant time interval δt. The search is finished when p_2 is approximated with the predefined accuracy ϵ. A tradeoff has to be found between the accuracy with which the goal is reached and the time it takes to find a solution. Although the RRT approach does not produce a time-optimal solution it provides, in reasonable time, a solution for real-sized problems.

Our approach for the dynamic motion controller is the following. The method controls a character based on input motion specified by a user, and environmental physical input in a physically based character animation system. In the system, the angular acceleration of character's joints are controlled so as to track the user-input motion. Dynamic simulation then generates the resulting animation. When environmental physical inputs is applied to the character, the dynamic motion control computes the angular joint accelerations in order to produce dynamically changing motion in response to the physical input.

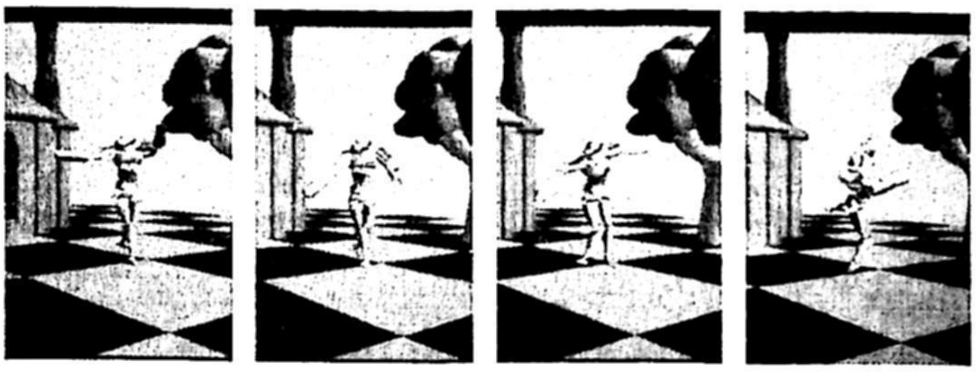

Fig. 9. Dancing animation with environmental physical input.

5 Conclusions and Future Work

Synthesizing realistic animation motions remains one of the great challenges in computer graphics. It seems that obeying physical laws is an important criterion of plausibility of motion. An interesting method to produce such physically motions is to animate characters from captured motion data that are inherently valid. These motions are adapted to different representations of the character to various environments or to additional kinematical constraints. The kinematic and kinetic adaptations (by interpolation, edition, retargeting or blending) may introduce physical inaccuracies in virtual human animation. It is thus necessary to be careful when such methods are used. Whenever the modifications introduce visually apparent errors in the physics of a motion, dynamics improvements may be added as a post-process or may correct the adaptation algorithm. We can for example, use the approach proposed by Safonova and Hodgins [14] to analyze the correctness of some physical properties.

With MAVIC, we have built an open, extensible animation system that engages the animator to interactively direct a 3D animation. The plug-in architecture allows a large library of diverse actuators and controllers to be implemented and integrated using a standard object-oriented interface We use MAVIC to build sophisticated, interactive 3D environments. We see the MAVIC platform as the basis for research into computer graphics in the areas of character animation, physically-based motion and control. An interesting aspect in the locomotion planning is the coordination of multiple virtual humans. Our work group have developed an interesting approach for the coordinated motion of virtual characters by combining centralized and decoupled planning methods. In the future, we will integrate to MAVIC this interesting solution.

References

1. N. M. Thalmann and D. Thalmann (Editors), "Handbook of virtual humans", *John Wiley & Sons, Ltd*, (2004)
2. F. Multon, L. France, M. P. Cani and G. Debunne, "Computer animation of human walking: A survey", *Journal of Visualization and Computer Animation*, Vol. 10, (1999) 39 -54
3. N. I. Badler, J. D. Korein et al., "Positioning and animating figures in a task-oriented environment", *The Visual Computer*, Vol. 1, No. 4, (1985) 212-220
4. A. Watt and M. Watt, "Advanced animation and rendering techniques: Theory and practice", *ACM Press*, (1992)
5. A. Witkin and K. Kass, "Space-time constraints", *Proc of the ACM SIGGRAPH*, (1988)
6. A. Bruderlin and L. Williams, "Motion signal processing", *Proc of the ACM SIGGRAPH*, (1995)
7. A. Witkin and Z. Popovic, "Motion warping", *Proc of the ACM SIGGRAPH*, (1995)
8. Z. Popovic and A. Witkin, "Physically-based motion transformation", *Proc of the ACM SIGGRAPH*, (1999)
9. J. Pettré, J. P Laumond and T. Siméon, "A 2-stages locomotion planner for digital actors", *Eurographics, Symposium on Computer Animation*, (2003)
10. J. J. Kuffner, "Goal-directed navigation for animated characters using real-time path planning and control", *Proc. of the CAPTECH*, (1998) 171-186
11. S. LaValle and J. J. Kuffner, " Rapidly-exploring random trees: Progress and prospects", *Algorithmic and Computational Robotics: New Directions, A K Peters*, (2001) 293-308
12. M. Gleicher, "Comparing constraint-based motion editing methods", *Graphical Models*, Vol. 63, No. 2, (2001) 107-134
13. S. LaValle and J. J. Kuffner, "Randomized kinodynamic planning", *International Journal of Robotics Research*, Vol. 20, No. 5, (2001) 378-400
14. A. Safonova and J. K. Hodgins, "Analyzing the physical correctness of interpolated human motion", *Eurographics Symposium on Computer Animation*, (2005) 171-180