Computer Animation of Human Walking: a Survey

Franck Multon[1], Laure France[2], Marie-Paule Cani-Gascuel[3], Gilles Debunne[3]

[1] IRISA,
Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
[2] BIP/INRIA,
ZIRST- 655 ave. de l'Europe, 38330 Montbonnot Saint-Martin, France
[3] iMAGIS-GRAVIR/IMAG
BP 53, F-38041 Grenoble cedex 09, France

Abstract: This paper surveys the set of techniques developed in Computer Graphics for animating human walking. First, we focus on the evolution from purely kinematic "knowledge-based" methods to approaches that incorporate dynamics constraints, or use dynamics simulations to generate motion. Then, we review the recent advances in motion editing, that enable the control of complex animations by interactively blending and tuning synthetic or captured motions.

Keywords: computer animation, human walking, motion synthesis and control, kinematics, dynamics, motion capture.

1 Introduction

Animation of human walking is a crucial problem in Computer Graphics: Many synthetic scenes involve virtual humans, from special effects in the film industry to virtual reality and video games. Synthesizing realistic human motion is a challenge: all viewers of an animation are also all experienced observers of human locomotion and will notice any artifact in the motion. However, we may be unable to give an objective reason why a motion is unrealistic. These points increase the difficulty of the task.

The study of walking motions has generated much interest in other fields such as biomechanics and robotics. It appeared about fifteen years ago in Computer Graphics, with the first work on "knowledge-based" animation of human figures. The interest for this area has never decreased in the Computer Animation community, even though the task is not easy, since techniques based on kinematics, dynamics, biomechanics, robotics and signal processing may be required. Indeed, animation or simulation of human walking interest several fields of application including robotics (such as the biped robot developed by Honda), art, behavioral simulation, entertainment, education and biomedical researches. According to the kind of application, the constraints that the model has to solve are different. On the one hand, for video games and virtual reality, one of the major constraints is to minimize the ratio between the computation cost and the capabilities of the model (adapting the gait to the environment, taking external forces into account, etc.). On the other hand, biomedical researches require to develop accurate models that obey the physical laws. Other applications of Computer Graphics such as video-clips or special effects in cinema also require to design accurate models that deal with artistic or cartoon's laws [7] without caring of the computation cost. For these reasons, models currently proposed in computer animation are designed in order to be applied in a specific application area. Hence, we can group these models in two main families: interactive models that involve low computation costs and off-line models that can use heavy computation time in order to obtain the required motion. Nowadays, several commercial packages provide the user with tools that automatically synthesize walks and deal with motion-capture data. However, the capabilities of such packages can be improved in order to be applied in other application areas than entertainment: interactive animation, biomedical simulation, etc. Thus, a survey on human-like figure animation is interesting in order to underline what are the solutions available at present time (included in commercial packages) and what should be the future developments (current topics of research).

To this end, this survey provides a comparative study of previous work in the area of motion control in Computer Graphics, focusing on the solution each method brings to the animation of human walking according to the kind of application. Whereas robotics proposes several kinds of walking models, this survey only focuses on the solutions that have been adopted in computer animation (coming from robotics or other research areas such as biomechanics or simulation). Rather than presenting a purely historical review, we classify previous work into three main groups, and emphasize the intrinsic advantages and limitations of each approach:

- Procedural methods based on knowledge-based kinematic animation are reviewed in Section 3.
- The attempts to incorporate dynamics constraints in the generation of motion or to use dynamics simulation are presented in Section 4.
- Lastly, the approaches enabling the interactive edition of either captured or synthetic walking motions are detailed in Section 5.

This classification clearly shows the evolution stream of the field, since the first group of approaches were mainly developed from 1982 to 1990, the second group from 1988 to 1996, and the last group has just arisen within the last two years. Moreover, this decomposition can also be viewed as a way to identify three ways of introducing knowledge in the walking models. Procedural approaches embed biomechanical and empirical knowledge on human locomotion, such as walking cycles studied in biomechanics. Dynamics is a way to incorporate mechanical knowledge as differential equations that enforce the respect of dynamic laws. Finally, by modifying captured motions, the system directly works on basic knowledge on human motion: real trajectories. In this last type of approach, trajectories are the knowledge that is used to compute human motion.

Section 6 concludes by listing the problems that are still open and that will undoubtedly inspire future research in the next few years.

2 Background

2.1 Animating virtual humans

Animating very complex models such as virtual humans is usually done by extracting a simpler representation of the model, a "skeleton", namely an articulated figure made of rigid links connected by hinges. Motion is first computed for this "skeleton", that can be displayed interactively. Once the animator is satisfied with the global motion sequence, he may compute higher quality representations of the moving character, by coating the skeleton with deformable surfaces modeling skin [15, 47, 44, 46] or clothes [28, 13, 53, 52].

This paper focuses on methods for generating the motion of skeletons of articulated figures that will represent virtual humans. An example of such an articulated figure is shown on Figure 1. Mathematically speaking, the skeleton is a hierarchy of local frames, each of which is characterized by its position and orientation with respect to its parent frame. The set $\theta = (\theta_1, .., \theta_N)$ of parameters corresponding to each degree of freedom of the figure, together with its derivative according to the time, is called the "state vector" or the "generalized coordinates" of the articulated figure. Synthesizing motion of the skeleton thus consists in defining how the state vector changes over time.

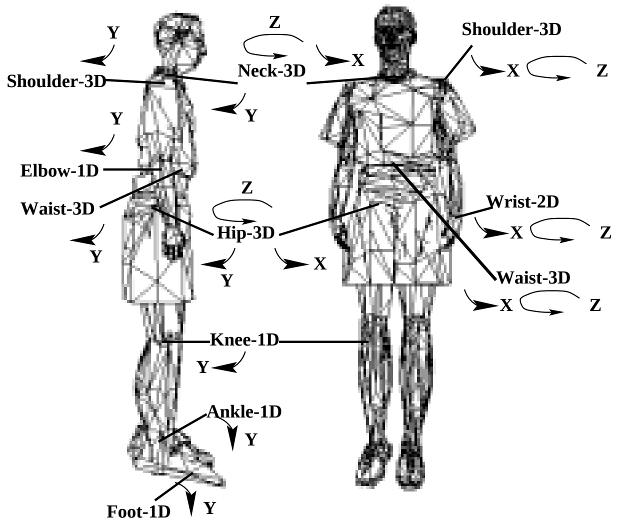

Figure 1: Articulated figure with thirty degrees of freedom.

2.2 Human walking

Researchers in biomechanics [40, 17] characterize human walking as the succession of phases separated by footstrikes FS (the foot is in contact with the ground) and takeoffs TO (the foot leaves the ground). In gait terminology, a stride is defined as a complete cycle from a left foot takeoff to another left foot

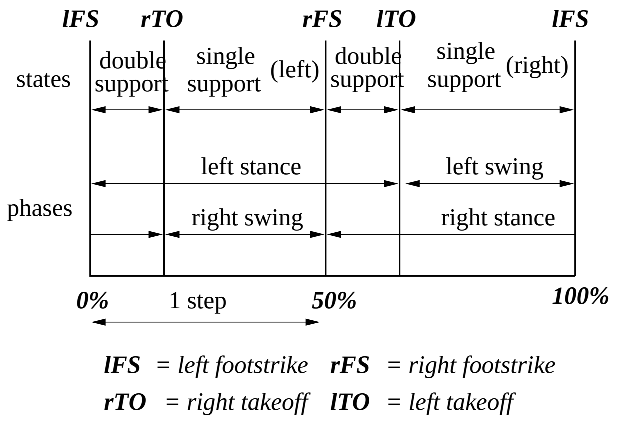

Figure 2: Characteristic phases of a walking motion.

takeoff, while the part of the cycle between the takeoff of the two feet is called a step. Four footstrike and takeoff events occur during a stride: left takeoff (lTO), left footstrike (lFS), right takeoff (rTO), and right footstrike (rFS). This leads to the characterization of two motion phases for each leg (see Figure 2):

1. the period of support which is referred to as the *stance phase*,
2. the period of non-support which is known as the *swing phase*,

A more accurate description of the walking phases is provided by Nilsson et al. [40]. They use extension and flexion steps for defining locomotion sub-cycles associated with each articulation. Each extension/flexion step is characterized by its place in the walking cycle, its duration, and the associated lower and upper angle bound values. Nilsson et al. study how these values change according to the velocity of walking. When the velocity increases, the duration of the double support state decreases to zero, which means that the walker has switched from a walk to a run.

The main problems encountered while designing a walking model depend on the kind of application, but include:

- ensuring that the motion of the body parts looks realistic,
- to verify that the contact between the human-like figure and the environment (especially the terrain) is realistic,
- accommodating variable grounds such as slope terrains or stairs,
- adapting the motion to the synthetic actor's anatomy,
- personifying the gait such as making the human-like figure walk as a woman, be less or more tires, etc.,
- accounting for changes in the mechanical structure of the walker which makes it possible to modify the motion when it carries heavy objects or is submitted to the win,
- making the walker react to external events or forces such as pushes or collisions,
- making sure that the forces and torques required to execute the computed motion are realistic.

Of course, depending on the kind of application, these problems are more or less important. In virtual reality, accounting for the environment is more essential than computing realistic forces and torques whereas, in biomedical applications (generally carried out in predefined environments) the relevance of the torques is essential. The difficulty of designing a walking model is proportional to the number of the above problems it solves. For example, designing a model that only focuses on computing realistic motions of the legs without taking the environment into account is easier than making a synthetic actor walk on complex terrains while maintaining it balance and being subject to external forces. All along this survey, we recall for each kind of model the problem that can be solved and what are its limits.

One problem in synthesizing realistic walking motion, common to several types of application, is the penetration of the feet into the ground. Under the hypothesis that the feet do not slide, the inequality constraints specifying that the feet should stay above the ground may be converted into equality constraints holding during the support phases. Modeling double support is a problem since computer animation algorithms, that easily cope with open chains, have difficulty handling closed loops. This is one of the reasons why walking motions are harder to model than running, for which there is no double support.

The question of handling interactions with the ground leads us to a strategic question: should we use physically-based simulation to help compute realistic motions? The answer is unclear. First, the role played by dynamics is less important in a walking motion than in other human motions such as running, jumping or diving [27]. Second, the way somebody walks tells us a lot on his/her personality and mood, which will be by essence very difficult to evaluate and to model. Half of the models presented in this paper use dynamics, while the other half do not. In the first case, a good simulator providing accurate modeling of contacts/friction with the ground will be required. The main problem will be to find parameterized controllers that generate muscular actions over time. These actions should make the human model walk while maintaining its balance. Here, techniques developed in the robotics community for the control of biped robots may help (we will address this issue in Section 4). If a kinematic animation method is chosen, the model will mostly rely on descriptions of walking motions such as those provided in biomechanics.

3 Kinematic animation

The first set of tools developed for motion specification in Computer Animation is based on forward and inverse kinematics. We first present these tools, and then look at the way they were applied to the generation of human walking. These techniques use empirical and biomechanical knowledge on human motion in order to compute realistic motions.

3.1 Forward and inverse kinematics

Forward kinematics consists in specifying the state vector of an articulated figure over time. This specification is usually done for a small set of "key-frames", while interpolation techniques are used to generate in-between positions. The main problems are the design of convenient key-frames, and the choice of adequate interpolation techniques. The latter problem, and in particular the way orientations can be represented and interpolated has been widely studied [54]. Designing key positions is usually left onto the animator's hand, and the quality of resulting motions deeply depends on his skills. In many cases, available physical and biomechanical knowledge such as the characterization of motion phases described in Section 2.2 for human walking, can help the animator to create relevant key-frames.

The exclusive use of forward kinematics makes it difficult to add constraints to the motion, such as those specifying that the feet should not penetrate into the ground during the support phases. These constraints may be solved using inverse kinematic algorithms. Here, motion ΔX of the end link of a chain (ie. a foot) is specified by the animator in world coordinates. The system computes the variation $\Delta\theta$ of the state vector (ie. the orientations between intermediate links) that will meet the constraint. The relation between the "main task" ΔX and the angular displacements $\Delta\theta$ takes the form:

$$\Delta X = J\Delta\theta \tag{1}$$

where J is the Jacobian matrix of the system [54]. J is often not directly invertible, due to the different dimensions of X and θ (ie. there is an infinity of angular positions at joints that lead to the same Cartesian position of a foot). So the most frequently used solution is [4]:

$$\Delta\theta = J^+\Delta X + \alpha(I - J^+J)\Delta z \tag{2}$$

where J^+ is the pseudo-inverse of the Jacobian matrix J, α is a penalty constant, I is the identity matrix, and Δz is a constraint to minimize, called the secondary task. This secondary task is enforced on the null space of the main task. Thus, the second term does not affect the achievement of the main task, whatever the secondary task Δz is. Generally, Δz is used to account for joint angular limits or to minimize some energetic criteria.

3.2 Kinematic animation of human walking

Most of the kinematic approaches used for generating synthetic human locomotion rely on biomechanical knowledge, and combine forward and inverse kinematics for computing motions.

In the eighties, methods were developed to automatically generate families of key-frames from biomechanical informations, thus providing easy tuning of the resulting motion. This is done through finite state machines controlled by high level parameters such as step length and step frequency. The first approach of that class is defined by Zeltzer [58, 59]. It embeds biomechanical knowledge of locomotion into hierarchical concurrent state machines which control the gait of a synthetic skeleton. A key-posture is associated to each state. These postures are linearly interpolated to produce in-between angular values.

A first method for ensuring non-penetration with the ground while using forward kinematics consists in changing the root limb of the skeleton when the support foot changes: at each foot-strike, the new support foot becomes the root, its position being fixed in world coordinates. This method is used by Bruderlin and Calvert [8], who simulate an inverted pendulum for computing realistic motion for the stance-leg. In that paper, only one joint of the support leg moves at a given time, which may lead to slightly artificial motions. Bruderlin and Calvert [9] improve the technique by simultaneously simulating all the degrees of freedom, leading to smooth and parameterizable walking gaits. Note that the same kind of approach has more recently been used for the animation of human running [10].

Another way to maintain extra constraints on the foot position is to use inverse kinematic algorithms [22, 21, 4, 6]. Boulic et al. [4] first use a standard forward kinematics approach, generating key-positions that are interpolated. A "leg-correction" process is then used to modify invalid in-between postures:

if a foot penetrates the ground, an inverse kinematic algorithm is applied to modify its position, thus modeling contact with the ground (see Figure 3). As shown in [5], the secondary task provided by inverse kinematic approaches may be used to maintain the character's balance.

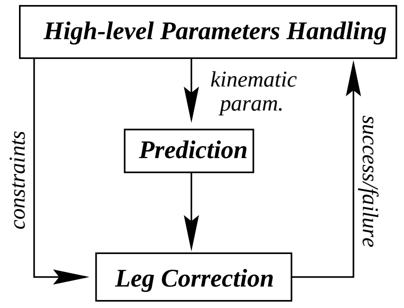

Figure 3: Synopsis of Boulic's locomotion system.

3.3 Discussion

Compared to the direct design of key-frames, the kinematic techniques presented above rely on a certain understanding of the basic walking motion mechanisms, at least from a purely descriptive point of view. The quality of the motion thus relies on the quality of the model (ie. the finite state machine that synthesizes the motion), rather than on an artist's skills. One of the main advantages of these models is the high level parameters they provide (such as velocity, step length, etc), leading to the generation of families of different gaits. Another advantage is the low cost of computations. The complexity of inverse kinematic algorithms is $\mathcal{O}(n^3)$ with respect to the number of joints (due to the inversion of the Jacobian matrix), but since the algorithm is usually used for one leg, this does not have an important effect on the efficiency of computations.

Most shortcomings come from the geometric interpolation process which is used for generating intermediate frames: angular trajectories are independently computed for each joint, although there should be a strong coupling between joint motions. Moreover, the interpolation process may filter the intrinsic dynamics of the locomotion, leading to a loss of realism. Lastly, although most terrains are not flat, there is no easy way to generate walking motion on a rough terrain from these approaches. The adaptation of the walk to such terrains would have to be explicitly described while designing the model.

4 Dynamics

In many cases, accounting for dynamics is essential to ensure that the motion is realistic:

- if the synthetic actor has to carry loads,
- if the human-like figure has to react to external forces such as pushes or the wind,
- if the ground is complex such as stairs or slope terrains,
- if the energy requires to produce the motion belongs to a realistic interval.

Dynamics approaches may be used either for adding constraints that guarantee a certain realism to a predefined motion, or for directly synthesizing the walk. Thus, in this kind of approach, the main knowledge that is used to ensure realism is dynamics. As in the previous section, we first give a short global review of the dynamics simulation techniques before describing the applications to human walking.

4.1 Forward and inverse dynamics

Newton's laws are known as the fundamental principle of dynamics. They link the forces (resp. torques for rotations) to the resulting motion (resp. rotation) from the equation:

$$f = m.\ddot{x} \tag{3}$$

where f represents the force applied to an object, m its mass and $\ddot{x}$ the second derivative with respect to time of the position vector x. The equation looks similar for rotations, where torques and angular positions are linked by:

$$t = i.\ddot{\theta} + \dot{\theta} \times i\dot{\theta} \tag{4}$$

where t is the torque, i the inertia matrix, $\dot{\theta}$ is the angular velocity and $\ddot{\theta}$ the angular acceleration. **Forward dynamics** is the application of these laws which calculate the motion generated by a given force. Conversely, **inverse dynamics** methods calculate the forces that would generate a given motion.

The equations above hold for a single solid. Specific methods [29, 2, 20, 1] have been used to apply these equations to an articulated skeleton, modeled as a hierarchy of rigid solids (namely the limbs) connected by joints. A full description of these methods is beyond the scope of this paper. We briefly describe the two basic approaches.

The first method was introduced by Isaacs and Cohen [29] and relies on a Lagrangian formalism. Using generalized coordinates (each limb angular position is expressed in its father's local coordinate system), a "generalized mass matrix" M can be computed. Computing motion then means solving an equation of the form: $F = M.\ddot{X}$, where F now represents a generalized force vector (consisting of all the different forces applied on each solid of the system) and X is the generalized coordinate vector. Since the matrix M depends on the relative positions of the solids, it changes over time and has to be inverted at each time step. Inverse dynamics can easily be combined with forward dynamics using this method: a value is given to some of the unknowns $\ddot{X}$ while tagging as unknown some components of F. This leads to animations where the motion of certain degrees of freedom is specified by the user while other joint motions are automatically computed from applied forces.

Another approach that avoids this expensive matrix computation was first introduced by Barzel and Barr [2]. Each solid is independently simulated at each time step. Extra forces are then computed to guarantee constraints on joints. These forces restore the joint constraints, and the next time step can then be simulated. Adding inverse dynamics is also straightforward with this method: a motion is specified for parts of solids. The others for which motion is computed by simulation, will then be "pulled" by the user-controlled solids.

4.2 Dynamics as an a posteriori constraint

One approach for benefiting from the realism offered by dynamics laws without having to specify the forces that create the motion is to use dynamics as a constraint. Motion is first computed a standard kinematic model. A post-processing stage then checks the physical relevance of motion.

The system of Ko and Badler [30, 31] (see figure 4) uses inverse dynamics to compute torques needed to perform a given motion. They then verify their validity (do they maintain balance and comfort? do they keep joint stress under a threshold?). Motion is corrected if needed, with hand-tuned coefficients that link kinematic positions of the synthetic actor to inverse dynamics results. This correction consists in adding $\Delta\theta(t)$ (computed by the inverse dynamics balance control module in order to retain balance) to the kinematic initial trajectory $\theta_0(t)$. For the next computation step, the inverse dynamics balance control accounts for the corrected joint angles:

$$\theta(t + \Delta t) = \theta_0(t) + \Delta\theta(t)$$

If the update should be done on the movement of one foot that is in contact with the ground, it also affects the kinematic module for the generation of the next step.

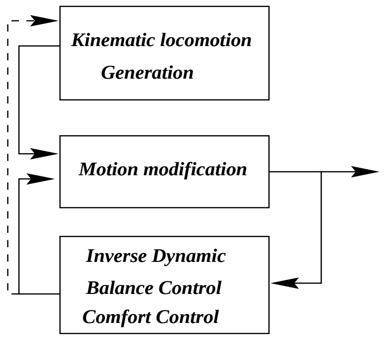

Figure 4: Synopsis of the locomotion system from Ko and Badler.

Thanks to this system, the synthetic actor automatically adjusts his posture by bending his pelvis if he is holding a heavy weight. During a walking cycle, the impacts with the ground make it difficult to compute the equivalent forces and torques at contact time. Ko and Badler simplify the problem by assuming a linear displacement of the contact point on the foot during the support phase. A more precise solution, based on an interaction model [18] which includes collision detection and Coulomb friction with the ground is used by Faure et al. [19], which provide the torque curves that correspond to a family of parameterized kinematic motions.

Adding dynamics constraints to kinematically computed motion results in a good compromise between realism and control, since it helps the animator to design more realistic motions without changing his way of interacting with the system. However, the two-stages process produces extra computations and does not guarantee the convergence of the process: a number of steps could be needed before the corrections of the kinematic model lead to a physically valid motion (imagine for instance a virtual actor trying to control balance at each time step while walking down stairs). In this kind of system, nothing ensures that that their is no conflict between the specified kinematics and the fixes required by the inverse dynamics. Moreover, one can wonder if the inverse dynamics module that is used to account for new physical phenomena (such as maintaining the balance in getting up and down stairs) makes it possible to compute the accurate required $\Delta\theta(t)$. Even if possible, would it be more efficient to directly simulate the model in that case ? So, we now present techniques that directly produce realistic (dynamically sound) motions.

4.3 Spacetime constraints

The combination of forward and inverse kinematics allows parts of articulated figures to follow predefined key-framed trajectories, but does not provide any help for defining these trajectories. Witkin and Kass [56] propose a new formulation of the problem, called "spacetime constraints": the basic idea is to compute the figure motion and the time varying muscular forces on the whole animation sequence instead of doing it sequentially in time. The discrete values of forces, velocities and positions over time are set in a very large vector of unknowns. A set of constraints between these unknowns is specified. They include:

- constraints on initial, final and some intermediary positions and velocities, through which the user controls the motion;
- constraints that limit muscular forces, or that model contact with the ground;
- the Newtonian physical laws that provide a constraint between forces and positions at two consecutive time steps. This constraint is added for each instant in time.

The vector of unknowns is computed during an iterative constrained optimization process. This is done by specifying a cost function, which is then minimized. This function is often set to the sum of squared muscular forces over time, which means that the system is trying to find the motion that spends as little energy as possible given the user-defined goals. A typical approach for solving a constrained optimization problem is the Sequential Quadratic Programming method: at each iteration, the cost function is minimized first, then the error over all constraints is minimized. This process is applied iteratively until the animator is satisfied.

However, this method suffers from several limitations. Firstly, the high computational complexity of the problem (and the large number of unknown) limits the length of animation sequences. The specification and the numerical integration steps are non-interactive, which makes it difficult to embed these techniques in an animation design system. Because constraints and goals may be non-linear, the user is not guaranteed that the numerical process will converge to an acceptable solution. Second, since dynamics is treated as a constraint between unknowns, and since the system compromises between all constraints to be able to find a solution, dynamics laws may not be entirely respected when the user stops the iterations (this may not be a problem if the animator is just looking for visual realism). Lastly, since every constraint in the animation must be specified "a priori", collisions and contacts between objects are modeled by constraints on positions that hold during specific time intervals. This prevents the use of an automatic collision detection and response module during the animation.

Cohen [16] presented an improved method for solving space-time constraints. It relies on "spacetime windows" which are designed interactively, and enable the solution of a given part, in space and time, of the animation. A window is defined on a sub-set of degrees of freedom and on a sub-interval of time. An iterative optimization process searches for functions describing the motion of each degree of freedom in the window, that minimizes a goal while maintaining a set of weighted constraints. The solution for each spacetime window represents a partial solution of the entire animation. The global solution, continuous in space and time, is obtained by combining the partial windows solutions. Due to visual and numerical feedback on the progression of animation, the animator can add new constraints, increase or decrease previous constraint weights, or modify intermediary solutions. The animator thus interacts and guides the optimization in order to make it converge to an acceptable solution. Several other improvements have been proposed such as the reformulation of the evolution of the degrees of freedom in hierarchical wavelet representation [36]. This representation allows the significant reduction of the computational cost. Nevertheless, a major drawback of space-time constraints still remains: collisions with the environment are not detected and treated automatically. They are simply considered as contact constraints, that are specified from their beginning to their end by the animator.

This kind of technique has been more specifically applied to the animation of biped walks by van de Panne [49]. His approach consists in using planification algorithm in order to compute footprints. These footprints are considered as constraints that drive the motion of the biped's center of mass submitted to the gravity and external forces. An optimization technique is used to minimize a two-term function in order to prevent non-physical and unnatural motions (comfort). Finally, the motion of the legs is computed by using inverse kinematics. Gleisher [23, 24] go further this approach by defining a set of spacetime constraints for all the trajectories and not only for the center of mass. Indeed, as inverse kinematics generally considers only one pose in its computation, he prefers to control all the trajectories along the whole sequence. To this end, he constrains the variations $d(t, x)$ (where t is the time and x is a vector that represents the parameters of the motion) around an initial motion $m_0(t, x_0)$. On the one hand, these constraints consist in equations and inequations that represent specific aspects of the motion that should be maintained. On the other hand, they drive the changes $d(t, x)$ in order to reach specific points at specific times. This method does not directly account for dynamics but is another way to use spacetime constraints in order to compute several realistic walks. Moreover, the kinematics constraints used in these approaches are quite simpler to define than dynamics constraints. Indeed, the desired motion is easier to specify this way than tuning energetical and mechanical criteria.

All the above approaches tend to offer the animator a tool to interactively modify the process of motion computation. In contrast, the next approaches that we will describe propose *on-the-shelves* models of human locomotion that deal with the intrinsic dynamics of human motion. Controllers make it possible to drive these dynamics while accounting for empirical and biomechanical knowledge on human gait.

4.4 The use of Controllers

The aim of the use of controllers is to animate a synthetic human figure with forward dynamics, which allows us to automatically take into account the effects of the interactions of the figure with the virtual environment. In this framework, the main problem is to find the actuator forces (modeling the action of the "muscles" of the virtual human), that will make it perform the desired motion.

Computing the actuator actions is usually done in a hierarchical process: The animator uses classical key-frame or kinematic techniques to define a desired motion. A "dynamics controller" then computes forces and torques required to achieve the goal. Finally, the standard forward dynamics process computes the motion that results from the set of applied forces and torques. In this framework, the specification of the goal motion offers high level control to the animator, while physical realism is obtained through the use of the dynamics controller, and through the integration of forces (that includes collision and friction forces with the environment). An intermediate controller is often introduced to allow gait refinements and coordination of the movements.

Before the animation starts, the animator specifies the behavior he wishes to simulate by providing the high level controller introduced above. The dynamics controller used to compute forces during the simulation is responsible for stabilizing posture, maintaining the locomotion cycle, controlling the speed and the direction of motion, and regulating the behavior of the joints. Based on the current state of the model and on the desired state, the controller computes forces and torques to apply to each actuator located on the joints. These forces can be specified by proportional-derivative controllers (PD controllers), that specify the dynamics behavior of the joints:

$$f = -k_p(q - q_d) - k_v(\dot{q} - \dot{q}_d)$$

where k_p and k_v are proportional and derivative gains respectively, $(q, \dot{q})$ is the current state of the system (position and velocity), $(q_d, \dot{q}_d)$ is the desired state of the system.

Van de Panne et al. [51] introduce a two-layered architecture to control the gait of such a dynamics system. The high level controller consists of a finite state machine, called *pose control graph*, in which states are associated to a pose of the figure (ie. a shape, but not a position). The transition between two adjacent states, which determines the motion between the corresponding poses, is realized by PD controllers. These PD controllers, placed on each joint, compute the required forces and torques to lead the articulated figure to the specified pose (the next state in the graph). Figure 5 depicts such a graph for the human walk. This concept is particularly well suited to cyclic motions, such as walking: the movement

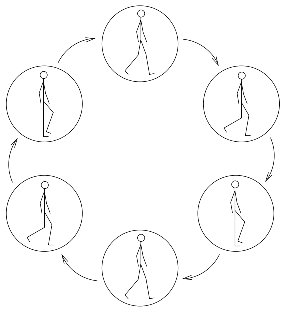

Figure 5: Finite state machine for the walk of an articulated figure

is generated by a cyclic graph that only describes a cycle of the considered motion [42, 27, 33, 51]. Cyclic pose control graphs provide a mean to represent and automatically control open-loop periodic motions. This technique can be extended to closed-loop control, that uses information provided by sensors. For instance, transitions between poses may be parameterized by contact information.

4.4.1 Hand-tuning of controllers

The approach introduced above has been widely studied for various kind of motions. Specific controllers have to be designed for a given behavior. For instance, Raibert and Hodgins [42] defined a controller to animate dynamics running of quadruped and biped creatures, by regulating the running speed, organizing the use of legs and maintaining the balance. Several other approaches [27, 26] make it possible to animate dynamics athletic behaviors such as running, bicycling, vaulting and diving, by controlling appropriate features of the considered movements.

These techniques seem to necessitate a long period of interactive tuning, since parameters are first found by trial and error, and then tuned. Indeed, empirical laws determine angles in the state machines that specify a high level description of the behavior to be simulated. Moreover, an open-loop control strategy with fine-tuning of control parameters is needed, which makes it difficult to reuse these parameters for another figure with a different morphology [41].

4.4.2 Automatic generation of controllers

An alternative to hand-tuned controllers is their automatic generation. Some approaches of this kind attempt to start from scratch: The user defines an articulated figure and a goal function to optimize (for instance, going as far as possible in a given time). The system automatically generates and optimizes a set of controllers, that may result into very different and surprising locomotion modes. Van de Panne and Fiume [50] model controllers as "Sensor-Actuator Networks" ie. non-linear directed graphs with weighted connections, that compute actuator torques from internal nodes linked to binary sensors data (giving information about the environment such as contact). The internal parameters of the system are first generated randomly, and then optimized according to a given criterion. Ngo and Marks [39] use a similar approach, where parameter values for the controller are searched with genetic algorithms. Sims [45] goes further in the use of genetic algorithms by simultaneously computing the evolution of the creature morphology that best suits a given mode of locomotion. As the only interface to control the motion is a cost function, these kinds of methods require lots of trials, and consequently need huge computational resources. In addition, most of them were only used in 2D and there is very little chance that the resulting motion resembles well-known motion such as human walking.

Applying dynamics control methods to 3D figures gives rise to the challenging problem of maintaining the balance. Although walking motions should be cyclic, the open-loop control of a 3D figure most often yields a motion that takes several steps, and then the figure falls over. Laszlo et al. [35] propose an automatic method for generating walking motions that maintains the balance from given open-loop controllers (such as a pose-control graph). The method is based on "limit cycle control". This extra control module tailors the open-loop controller by regulating some of its variables according to observed perturbations on motion. This automatic addition of closed-loop control yields stable and robust walking motions, that however do not have the visual quality of real human motion.

4.5 Discussion

Adding biomechanical knowledge concerning human motion to the inverse dynamics module allows us to guarantee that the computed motion is anatomically feasible. If the forces are properly and robustly computed, the character may be able to handle interactions with his environment. This may be essential for animating autonomous actors in virtual reality applications. Collisions with other objects, walking on a variable terrain, walking in water or against the wind could be simulated using these methods, offering a powerful tool to animators.

However, techniques based on controllers and dynamics simulation suffer from a high computational cost compared to kinematic techniques. This makes them difficult to use in interactive animation tools. Moreover, the mechanical parameters are very difficult to calibrate with respect to human anatomy (damping effects between bones, etc.). In addition to the calibration of the mechanical model, the designer of such a system has to deal with the calibration of the controller gains in order to obtain realistic motions. Consequently, resulting motions are usually not as realistic as those generated by hand using kinematics, which offer direct control on the effective motion rather than trying to model its causes.

5 Animation Based on Motion Data

We have just seen that many dynamics models were designed in the nineties in order to synthesize human figure motion, but it is unlikely that dynamics simulation will solve all animation problem. Researchers have thus turned to other kinds of approaches. The recent progress in motion capture techniques makes it possible to directly use human motion data. In this last type of approach, the knowledge that is used to ensure realism consists in the captured trajectories themselves. We now describe some of these approaches.

5.1 Motion Capture

All the previous approaches deal with biomechanical, mechanical or empirical knowledge on human motion. The goal of this kind of approaches is to work directly on data which are provided by captured motions. Thus, in last few years, motion capture techniques have been widely used to animate 3D rigid-body skeleton. By using magnetic or optical technologies, it is possible to store the positions and orientations of points located on the human body. A further computation provides the link between the synthetic skeleton and the real skeleton, in order to adapt data to the new morphology. Several approaches [37, 3] have introduced techniques to adapt captured trajectories to a different synthetic skeleton. Such systems also deal with errors introduced during the capture process, such as numerical approximations, calibration error, electronic noise, etc. The method consists in recovering angular trajectories which are applied to a synthetic articulated body. Given sensor positions and orientations, a modified inverse kinematic optimization algorithm is used to produce the desired joint trajectories. The synthetic skeleton thus plays exactly the same motion as the real actor.

For most applications, the captured motion needs to be modified in order to create a variety of specific animations, that may take the synthetic environment into account. For instance, when interaction between two synthetic actors is required, their movement has to be modified to model this interaction (by dealing with problems of contacts, trajectory tracking, etc.). To this end, two main families of techniques have been introduced in the last few years: motion blending and motion warping.

5.2 Motion blending

Motion blending needs a database of characteristic motions (described either in the frequency or temporal domain) and consists in interpolating between their parameters in order to produce new motions.

Unuma et al. [48] used Fourier expansions of experimental human motions to interpolate or extrapolate the human locomotion. First, angular trajectories are expressed using Fourier series in the frequency domain:

$$\theta_n = \alpha_0 + \sum \alpha_k sin(2\pi k \frac{n}{N} + \phi_k) \tag{5}$$

A low-band filter is applied to keep only the first parameters ($\tilde{\theta_p}, p \in [0...3]$). Once these parameters are obtained for several different locomotion styles, it is possible to interpolate from one set of parameters (α_k^1, ϕ_k^1) to another (α_k^2, ϕ_k^2):

$$\tilde{\alpha_k} = s\alpha_k^2 + (1-s)\alpha_k^1 \tag{6}$$

where s is a real value ranging from 0 to 1. New motions can be obtained by interpolating between two pre-computed motions. Resulting motions are completely defined by the value of s and by the two sets of parameters of original motions. For example, to make a synthetic actor walk in a more or less tired fashion, it is possible to interpolate between a normal and a tired gait.

Whereas Unuma et al. worked with trajectories in the frequency domain, Guo and Roberge [25] used parametric frame space interpolation of key-framed motions. Every motion with m articulations is expressed in a m-dimensional space $P^m = (a_1, a_2, ..., a_m)$. Thus, for n key-frames $K_1^m, K_2^m, ..., K_n^m$, it is possible to compute in-between angular trajectories by a m-dimensional interpolation function $F(s)$, where s is the arc-length parameter. The resulting angular trajectories are mapped to a horizontal line segment through a parameter conversion to obtain a 1-D frame space. Once these operations have been carried out on k referenced motions, it is possible to create new motions as linear combinations (or weighted sums) of the reference interpolation functions $F_i(x)$, $(i = 1, 2, ..., k)$, where x is the coordinate of the 1-D frame space. For human locomotion, four different locomotion styles are considered: short-step (**ssw**), long-step (**lsw**), short-step running (**ssr**) and long-step running (**lsr**). Figure 6 depicted an example of frame space interpolation for the human locomotion cycle.

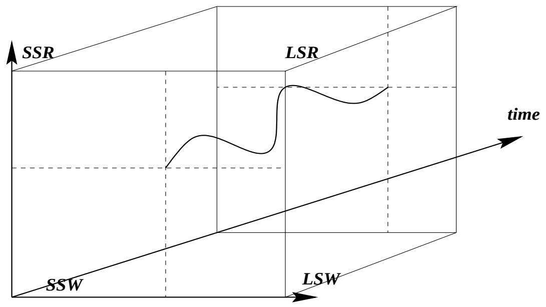

Figure 6: A position curve defined in the frame space.

Wiley and Hahn [55] have also introduced a technique to interpolate pre-recorded captured motions. Numerous captured motions define a parameter space (such as the position of an end-effector, styles of walk, etc.), and new motions are obtained by interpolating parameters of some captured motions with cubic splines. The problem of this technique is the size of the database that depends on the required precision of the resulting motion. Moreover, for human motion, several different postures correspond, in general, to a unique parameter (such as the position of the hand). These customized postures may be taken into account in this kind of model if new parameters are added (according to these families of motions). Consequently, the volume of the database of pre-recorded motions increases proportionally to this number of parameters. To decrease the volume of this database, Ko and Cremer [34] introduced a system called VRLOCO, which automatically blends motions computed by kinematic models of human locomotion (described in [30]) instead of captured motions. Depending on direction and velocity, the system automatically generates footprints and selects the appropriate style of gait.

Whereas the previous approaches use linear interpolation to blend two pre-recorded motions, Rose et al. [43] use spacetime constraints to transit from one movement to another. Basic operators such as parallelism, addition and subtraction are defined for a basis of elementary motions. The transition between two motions is then obtained while dealing with dynamics constraints such as minimizing the required energy (as described in subsection 4.2).

The main advantages of the motion blending techniques are the low computation cost and the use of motion capture trajectories as primary data (to ensure the maintainance of intrinsic dynamics of the movement). The main drawback of these techniques is that the number of possible effects such as weariness, nervousness, etc. relies on the number of pre-recorded motions. Making a human figure walk in a more or less tired manner is possible if and only if a normal walk and a tired walk are recorded. Moreover, for these techniques, the authors generally assume that transitions between two parameter sets are linear (or cubic) and continuous in time. In frequency and temporal domains however, nothing ensures that an interpolated (or extrapolated) parameter set produces realistic motions. Finally, in some cases, interpolation between two parameter sets is dangerous. For instance, when we move our hand, we may decide to completely change the body configuration: for example, by squatting if the altitude of the hand is lower than a given threshold. Interpolated postures, in this particular case, will not be realistic.

5.3 Motion warping

We call motion warping all techniques that take well-known trajectories (described by key-frames or motion captured trajectories) and modify them in order to change the motion. To this end, two main groups of approaches are considered: signal study in the temporal or in the frequency domain.

Witkin and Popovic [57] modified a reference trajectory $\theta_i(t)$ (where i represents the i^{th} parameter of the system) by interactively tuning the position of selected key-frames and by scaling and shifting $\theta_i(t)$:

$$\begin{array}{c} \forall i,\ \theta'_i(t) = a(t)\theta_i(t) + b(t) \\ t = g(t') \end{array} \tag{7}$$

Function $a(t)$ is used to scale the signal and $b(t)$ is used to change the center of scaling of a. The deformation from time t to t' is a constrained interpolation based on Cardinal splines. Thus, the resulting sequence satisfies the constraints of new key-frames with respect to the pattern of the initial motion. Moreover, blending of several motions can be obtained by weighted sums. Ko and Badler [32] also introduced a method to modify a reference motion recorded as sequences of key-frames:

$$\mathcal{Q} = \{(t_i, v_i) \| i = 1, \ldots, n\}$$

where t_i is a real number with $t_i < t_{i+1}$, and v_i is any dimensional vector. By changing the morphology of the synthetic figure, new parameters (t_i, v_i) are automatically computed to make new character walks. Whereas these parameters are changed, the characteristics of the original gait are preserved.

Instead of studying trajectories in the temporal domain, Bruderlin and Williams [11] applied image and signal processing techniques in the frequency domain to reuse, modify and adapt animated human motion. The goal is to make libraries of animated motion with high level motion editing at interactive speeds. To this end, multi-resolution filtering (figure 7 depicts such a multi-resolution filtering process) of angular trajectories have been used to define a reduced set of motion parameters:

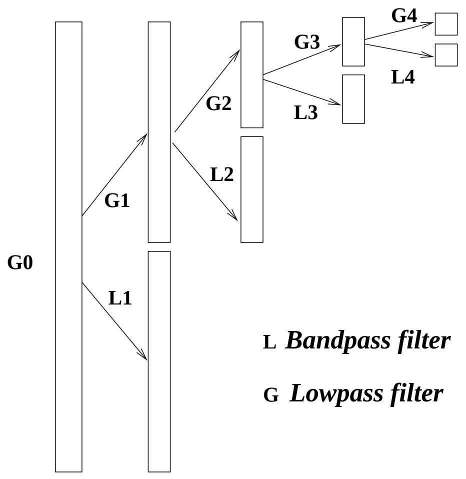

Figure 7: Principle of the multi-resolution filtering process.

$$G_0 = L_0 + L_1 + ... + G_n \tag{8}$$

where G_0 represents the initial signal and $L_{n-1} = G_{n-1} - G_n$. To produce new motions, the signal is reconstructed differently by tuning weights g_k associated to each parameter:

$$G_0 = G_{f_b} + \sum_{k=0}^{f_b - 1} (g_k . L_k) \tag{9}$$

where $f_b = log_2(m)$, if m is the length of G_0. Depending on which frequency band g_k is modified, the resulting motion is tuned (more or less tired, nervous, exaggerating, etc.). Moreover, this technique makes it possible to interpolate from one set of frequency parameters to another, as in motion blending techniques. Compared to Fourier analysis, multi-resolution analysis is interesting for non-periodic perturbed signals. But, in the special case of locomotion, that is a quasi-periodic motion, the most important part of the signal is contained in the few first harmonics. It is possible to achieve the same kind of approach by using Fourier principles.

For the two approaches mentioned above, the same problem as kinematic or procedural animation techniques is encountered. Because these modifications do not handle dynamics effects, it is impossible to ensure that resulting motions are realistic. Moreover, as for kinematic or procedural approaches, the trajectories are decoupled so that the strong coupling of the articulation motion may be lost in many cases. Even if all frequencies are changed in the same manner, no attention is paid to the shift that naturally occurs between the articulations in a real locomotion cycle.

5.4 Discussion

Compared to procedural and kinematic techniques, motion modification techniques offer the advantage of using intrinsically realistic motion (ie. captured motion) to animate human-like figures. Moreover this technique provides animators with tools that are compatible with their usual work on key-framed or captured trajectories.

Table 1: Comparison between the models.

Model	Cost	Advantages	Disadvantages
Procedural approaches	$\mathcal{O}(n)$	low computation cost	no automatic dynamics effects
Kinematic correction	$\mathcal{O}(n^3)$	ensure realistic poses	no automatic dynamics effects coherence between modules
Dynamic constraints	$\mathcal{O}(n)$	avoid unrealistic dynamics effects	Desired dynamics effects implies ad'hoc constraints coherence between modules
Space-time constraints	$\mathcal{O}((nm)^2)$	dynamically-sound interpolation	Huge computation cost
Simulation	$\mathcal{O}(n^3)$	accounts for dynamics effects automatic simulation code gen.	Computation cost calibration of parameters physical realism vs. natural looking
Editing Motion Data	$\mathcal{O}(n)$	Computation cost Realistic trajectories	hard interactive control volume of the motion database

On the other hand, the intrinsic realism included in captured trajectories might be lost for extreme modifications. Indeed, the modifications applied to these trajectories do not ensure the preservation of intrinsic dynamics of human motion. Even though the quality of the animations produced by procedural techniques depends on the model, the quality of the resulting sequences depends on the users skills. For motion warping techniques, there is no control that ensures that the resulting sequence is the one desired. Consequently, the user of such a system has to modify by trial and error parameters of trajectories, as well as interpolation weights, in order to produce a specific desired motion.

Moreover, in several of such approaches, using filters or blending motions makes it difficult to ensure that holonomic constraints are verified. For example, a filtered trajectory does not exactly go through the constraints points that might be relevant in order to make the motion be realistic. In the same manner, the result of two blended trajectories is really hard to control in order to maintain several simultaneous point-to-point constraints.

6 Conclusion

This paper classifies Computer Graphics techniques for the simulation of human locomotion into three main categories : kinematic animation, dynamically-based animation and motion-data edition. These categories are identified by the type of knowledge that they use in order to compute realistic motions: empirical/biomechanical knowledge, dynamics and captured trajectories. Table 1 summarizes the main advantages and shortcomings of all these techniques. Their development over time shows the evolution described above. For each kind of approach, table 2 recalls the main problems that the method deals with and its best suitable type of application.

First, Researchers attempted to reproduce some specific behaviors by providing a way to describe these effects. This kind of approach requires a low computational cost which depends on the selected interpolation function. Degrees of freedom are computed separately, as if they were decoupled, which involves a theoretical computation cost of $\mathcal{O}(n)$ if n is the number of degrees of freedom. When taking constraints on foot/ground contact into account, an additional cost of $\mathcal{O}(n^3)$ is generated by the required inverse kinematic algorithm. Nevertheless, the quality of the motion mainly depends on the quality and the quantity of knowledge necessary to reproduce the desired effect. For human locomotion, the way the angular trajectories change depending on high level parameters (such as velocity, step length, frequency and height) has to be described “a priori”. Thus, the designer generally has to create a new model from scratch to account for a new effect.

Then, several approaches attempted to improve these models under the hypothesis that dynamics laws were responsible for much of these effects. Methods producing “dynamically-sound” motions were developed. Some of them just modify the results of kinematic models. Other methods go further by directly embedding dynamics in the computation of angular trajectories. Thus, controllers applied to more or less complex mechanical models have been presented. Nevertheless, for complex models such as the human body, the computational cost increases in $\mathcal{O}(n^3)$ due to the inversion of large matrices. Moreover, some of the dynamically-sound motions do not look realistic.

Motion capture systems have been widely used during the last few years for creating new animations of virtual humans. These approaches assume that realistic motions can be obtained by editing and tuning a library of motion data. This data is based on captured motion, but may also include artificial motions computed with previous approaches. The computational cost is low ($\mathcal{O}(n)$) because each articulation is computed separately. Although several convincing animations have been designed using these methods, the parameter control is not simple: a number of trial and error iterations is often needed before obtaining the desired result. Moreover, the modifications applied to predefined trajectories have to be small in order to ensure that the resulting sequence remains realistic. Problems such as avoiding inter-penetration of the feet into the ground for edited motions, and ensuring non-sliding contact for the support leg, are not handled by most of these approaches. We believe that these techniques would lead to very promising developments if they are coupled with knowledge on human motion and with dynamically-based constraints.

Another important point is to be able to choose the method that suits the best a specific application. For instance, kinematic models can be successfully applied to virtual reality applications where all possible basic motions are designed in advance and no dynamics effect is required. These methods are suited to real-time animation of several characters [38]. Dynamic models have to be used in other virtual reality animations that deeply rely on physically-based interactions. However, precise simulation techniques that have a heavy cost make the control of several characters difficult in shared virtual reality applications such as the VLNET system [12]. Dynamic techniques that require off-line computations have rarely been used in audiovisual applications (such as special effects in cinema) because of their lack of control. But improved models could find a good field of applications there in the future. Finally, motion-data-based animation techniques are suitable for video-games and applications in movie production. Reference motions are first captured and then tuned either by the user or to solve the environment constraints (adapting a captured motion to another synthetic world). These techniques offer interesting capabilities. For instance, they express emotions (which may be already contained in the captured data) more easily than physically-based techniques. Future works enhance motion editing techniques by accounting for dynamics effects and

Table 2: Comparison of the methods according to the kind of application.

Model	problems taken into account	suitable for (applications)
Procedural approaches	realistic motion of the articulations	virtual reality (without collision), video games, behavioral simulation
Kinematic correction	realistic motion of the articulations verifies the contact with the ground accommodate variable grounds	virtual reality, video games, behavioral simulation, art
Dynamic constraints	realistic motion of the articulations accounts for mechanical changes reacts to external forces computes the required torques	art, virtual reality
Space-time constraints	accounts for mechanical changes reacts to external forces computes the required torques	art, biomechanical simulation
Simulation	accommodates variable grounds accounts for mechanical changes reacts to external forces computes the required torques	biomedical simulation
Editing motion data	realistic motion of the articulations personification of the gait	art, virtual reality, video games, behavioral simulation

other global knowledge on motion. On the other hand, physically-based models may also be improved, and coupled with kinematic techniques in order to be applied to interactive avatars or to behavioral simulations.

In conclusion, we believe that the various methods we have presented should not be used alone, but rather combined according to the specific application which is developed. An interesting way to do this is to adapt the level of detail in motion generation when a character comes closer [14], which may be done by switching from a model to another one. Several challenging problems such as the design of smooth transitions are still to be solved in this area.

References

[1] David Baraff. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH 96 Conference Proceedings*, Computer Graphics Proceedings, Annual Conference Series, pages 137–146. ACM SIGGRAPH, Addison Wesley, August 1996. ISBN 0-201-94800-1.

[2] R. Barzel and A.H. Barr. A modeling system based on dynamics. In *Proceedings of ACM SIGGRAPH*, pages 179–188. Addison Wesley, July 1988.

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

[4] R. Boulic, N. Magnenat-Thalmann, and D. Thalmann. A global human walking model with real-time kinematic personification. *Visual Computer*, 6(6):344–358, December 1990.

[5] R. Boulic, R. Mas, and D. Thalmann. A robust approach for the center of mass position control with inverse kinetics. *Journal of Computers and Graphics*, 20(5), 1996.

[6] R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figures motion editing. *Computer Graphics Forum*, 11(4):189–202, 1992.

[7] A. Bruderlin. Hierarchical virtual characters. In *Siggraph course note 17, Virtual Humans: Behaviors and Physics, Acting and Reacting*, May 1997.

[8] A. Bruderlin and T. Calvert. Goal-directed, dynamic animation of human walking. In *Proceedings of ACM SIGGRAPH*, pages 233–242. Addison Wesley, July 1989.

[9] A. Bruderlin and T. Calvert. Interactive animation of personalized human locomotion. In *Graphics Interface*, pages 17–23, 1993.

[10] A. Bruderlin and T. Calvert. Knowledge-driven, interactive animation of human running. In *Graphics Interface'96*, pages 213–221, May 1996.

[11] A. Bruderlin and L. Williams. Motion signal processing. In *Proceedings of ACM SIGGRAPH*, pages 97–104, Los-Angeles, California, August 1995. Addison Wesley.

[12] TK. Capin, LS. Pandzic, H. Noser, N. Magnenat-Thalmann, and D. Thalmann. Virtual human representation and communication in vlnet. *IEEE Computer Graphics and Applications*, 17(2):42–53, 1997.

[13] Michel Carignan, Ying Yang, Nadia Magnenat-Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. *Computer Graphics*, 26(2):99–104, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[14] D.A. Carlson and JL. Hodgins. Simulation level of detail for real-time animation. In *Proceedings of Graphics Interface'97*, pages 1–8, 1997.

[15] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):243–252, July 1989.

[16] M.F. Cohen. Interactive spacetime control for animation. In *Proceedings of ACM SIGGRAPH*, pages 293–302. Addison Wesley, July 1992.

[17] R. M. Enoka. *Neuromechanical Basis of Kinesiology (2nd Edition).* Human Kinetics, 1994.

[18] F. Faure. An energy-based method for contact force computation. In *Proceedings of Eurographics'96*, pages 357–366, August 1996. Computer Graphics Forum, Volume 15, Number 3.

[19] F. Faure, G. Debunne, M.P. Cani-Gascuel, and F. Multon. Dynamic analysis of human walking. In *Eurographics Workshop on Computer Animation and Simulation*, pages 53–66, September 1997.

[20] Jean-Dominique Gascuel and Marie-Paule Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, 10(4):191–204, March 1994.

[21] M. Girard. Interactive design of 3d computer-animated legged animal motion. *IEEE Computer Graphics and Applications*, 7(6):39–51, June 1987.

[22] M. Girard and A.A. Maciejewski. Computational modeling for the computer animation of legged figures. In *Proceedings of ACM SIGGRAPH*, pages 263–270. Addison Wesley, July 1985.

[23] M. Gleisher. Motion editing with spacetime constraints. In *Proc. of Symposium on Interactive 3D Graphics*, April 1997.

[24] M. Gleisher. Retargeting motion to new characters. In *Proc. of ACM SIGGRAPH'98*, Orlando, FL, July 1998. Addison Wesley.

[25] S. Guo and J. Roberge. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Eurographics Workshop on Computer Animation and Simulation*, pages 95–107, Poitiers, France, September 1996. Springer Verlag.

[26] J. Hodgins. Three-dimensional human running. In *Proceedings of the IEEE Conference on Robotics and Automation*, April 1996. Minneapolis, Minnesota.

[27] J.K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F O'Brien. Animating human athletics. In *Proceedings of ACM SIGGRAPH*, pages 71–78, Los Angeles, California, August 1995. Addison Wesley.

[28] Donald House, David Breen, and Philipp Getto. On the dynamic simulation of physically-based particle-system models. In *Third Eurographics Workshop on Animation and Simulation*, Cambridge, England, September 1992.

[29] P.M. Isaacs and M.F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In *Proceedings of ACM SIGGRAPH*, pages 215–224. Addison Wesley, July 1987.

[30] H. Ko. *Kinematic and Dynamic techniques for Analyzing, Predicting, and Animating Human Locomotion*. PhD thesis, University of Pennsylvania, 1994.

[31] H. Ko and N. I. Badler. Animating human locomotion in real-time using inverse dynamics. *IEEE Computer Graphics & Applications*, 1996.

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

[33] H. Ko and N.I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2):50–59, March 1996.

[34] H. Ko and J. Cremer. Vrloco: real-time human locomotion from positional input streams. *Presence*, 5(4):367–380, 1996.

[35] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of ACM SIGGRAPH*, pages 155–162, New Orleans, Louisiana, August 1996. Addison Wesley.

[36] Z. Liu, S.J. Gortler, and M.F. Cohen. Hierarchical spacetime control. In *Proceedings of ACM SIGGRAPH*, pages 35–42, Orlando, Florida, July 1994. Addison Wesley.

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

[38] S. R. Musse and D. Thalmann. A model for human crowd behavior: Group inter-relationship and collision detection analysis. *Eurographics Animation and Simulation Workshop*, September 1997.

[39] J.T. Ngo and J. Marks. Spacetime constraints revisited. In *Proceedings of ACM SIGGRAPH*, pages 343–350, Anaheim, California, August 1993. Addison Wesley.

[40] J. Nilsson, A. Thorstensson, and J. Halbertsam. Changes in leg movements and muscle activity with speed of locomotion and mode of progression in humans. *Acta Physiol Scand*, pages 457–475, 1985.

[41] J.K. Hodgins N.S. Pollard. Adapting simulated behaviors for new characters. In *Proceedings of ACM SIGGRAPH*, Los Angeles, California, August 1997. Addison Wesley. Computer Graphics Forum, Volume 15, Number 3.

[42] M.H. Raibert and J.K. Hodgins. Animation of dynamic legged locomotion. In *Proceedings of ACM SIGGRAPH*, pages 349–358. Addison Wesley, July 1991.

[43] C. Rose, B. Guenter, B. Bodenheimer, and M.F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of ACM SIGGRAPH*, pages 147–154, New Orelans, Louisiana, August 1996. Addison Wesley.

[44] Jianhua Shen and Daniel Thalmann. Interactive shape design using metaballs and splines. In *Implicit Surfaces'95—the First Eurographics Workshop on Implic it Surfaces*, pages 187–195, Grenoble, France, April 1995.

[45] K. Sims. Evolving virtual creatures. In *Proceedings of ACM SIGGRAPH*, pages 15–22, Orlando, Florida, July 1994. Addison Wesley.

[46] D. Thalmann, J. Shen, and E. Chauvineau. Fast human body deformations for animation and vr applications. In *Proceedings of Computer Graphics International '96*, pages 166–174. IEEE Computer Society Press, June 1996.

[47] Russel Turner. Leman: A system for construsting and animating layered elastic chara cters. In *Computer Graphics- Developments in Virtual Environments*, pages 185–203, Academic Press, San Diego, CA, June 1995.

[48] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of ACM SIGGRAPH*, pages 91–96, Los Angeles, California, August 1995. Addison Wesley.

[49] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, October 1997.

[50] M. van de Panne and E. Fiume. Sensor-actuator networks. In *Proceedings of ACM SIGGRAPH*, pages 335–342, Anaheim, California, August 1993. Addison Wesley.

[51] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. In *Graphics Interface*, pages 208–215, Banff, Alberta, Canada, May 1994.

[52] Volino, N. Magnenat-Thalmann, S. Jianhua, and D. Thalmann. Am evolving system for simulating clothes on virtual actors. *IEEE Computer Graphics and Applications*, 16(5):42–51, 1996.

[53] Pascal Volino, Martin Courchesne, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *Computer Graphics*, pages 137–144, August 1995.

[54] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. ACM Press, 1992.

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

[56] A. Witkin and M. Kass. Spacetime constraints. In *Proceedings of ACM SIGGRAPH*, pages 159–168, Atlanta, Georgia, August 1988. Addison Wesley.

[57] A. Witkin and Z. Popovic. Motion warping. In *Proceedings of ACM SIGGRAPH*, pages 105–108, Los Angeles, California, August 1995. Addison Wesley.

[58] D. Zeltzer. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, 2(9):53–59, November 1982.

[59] D. Zeltzer. Knowledge-based animation. In *ACM SIGGRAPH/SIGART, Workshop on Motion*, pages 187–192, April 1983.