Real Time Animation of Virtual Humans: A Trade-off Between Naturalness and Control

H. van Welbergen[1] B. J. H. van Basten[2] A. Egges[2] Zs. Ruttkay[1] M. H. Overmars[2]

[1]Human Media Interaction, University of Twente, Enschede, The Netherlands
[2]Center for Advanced Gaming and Simulation, Utrecht University, The Netherlands

Abstract

Virtual humans are employed in many interactive applications using 3D virtual environments, including (serious) games. The motion of such virtual humans should look *realistic (or 'natural') and allow interaction with the surroundings and other (virtual) humans. Current animation techniques differ in the trade-off they offer between their motion naturalness and the amount of control that can be exerted over the motion. We give an overview of these techniques, focusing on the exact trade-offs made. We show how to parameterize, combine (on different body parts) and concatenate motions to gain control. We discuss several aspects of motion naturalness and show how it can be evaluated. We conclude by showing the promise of combinations of different animation paradigms to enhance both naturalness and control.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation

1. Introduction

Virtual environments inhabited by virtual humans (VHs) are now commonplace in many applications, particularly in (serious) games. Animation of such VHs should operate in real-time to allow interaction with the surroundings and other (virtual) humans. For such interactions, detailed *control* over motion is crucial. Furthermore, the motion of VHs should *look* realistic. We use the term *naturalness* for such observed realism.

Many techniques exist that achieve real-time animation. These techniques differ in the trade-off they offer between the amount of control over the motion of the body, the naturalness of the resulting motion and required calculation time. Choosing the right technique depends on the needs of the application. This STAR aims to help the reader in this choice, by providing an overview of real-time animation techniques. We give a short summary of each technique, and focus on the trade-offs made.

First we discus models of the VHs body that are steered by animation. Then we give a short overview of technologies from robotics and film/cartoon making that are used in computer animation. In section 4 we classify animation techniques that are used to generate short animation segments with a consistent function and discus their strengths and weaknesses. In section 5 we show how to parameterize, combine (on different body parts) and concatenate motion generate by these techniques to gain control. We discus several aspects of naturalness and show how it can be evaluated. We conclude by discussing the power of combinations of animation paradigms to enhance both naturalness and control.

2. Modeling the VH

Animation steers the body of a VH. Here we show how the body is modeled as a skeleton, articulated set of rigid bodies and biological system.

2.1. Skeletal Model of the VH

VHs are mostly represented by polyhedral models or *meshes*. Animating all these polygons individually can be very tedious, therefore it is very common to work with the underlying *skeleton* instead of the mesh itself. A skeleton is an *articulated structure*: its segments are connected by means of joints in a hierarchical structure. The joints and segments (also called *bones*) define the structure of the skeleton. A *pose* of a VH is set by rotating the joints. The

world location of a joint is determined by the joints above it in the hierarchy. For example, a rotation in the shoulder will not only affect the position of the elbow joint, but also that of the wrist joint. How the skeleton deforms the mesh is beyond the scope of this article, we refer the interested reader to [MTSC04].

Every joint has several *degrees of freedom* or *DoFs*. The DoFs are the parameters that define a configuration of a joint. For example, the elbow joint has only one DoF, while a shoulder joint has three. The global translation of the skeleton is represented by a translation of the root joint (hence, a root joint has 6 DoFs). The pose of a skeleton with n rotational DoFs can be described by an $n+3$ dimensional vector $\mathbf{q} = [r_x, r_y, r_z, \theta_1, \ldots, \theta_n]^T$ where $\mathbf{r} = [r_x, r_y, r_z]^T$ is the root translation.

Standardizing the skeleton topology improves reusability of motions. Motions created for one VH can be transfered to another VH more easily. The H-anim [Hum05] standard provides a complete set of standardized joint names and their topology, that specifies their resting position and how they are connected.

2.1.1. Representations of Joint Rotation

One of the most common representation of rotations is a 3×3 *rotation matrix*. A rotation matrix can only represent a rotation if its determinant is 1 and its columns are of unit length and mutually orthogonal. The set of all such matrices form the group $\mathbf{SO(3)}$ under the operation of matrix multiplication.

Euler angles represent rotation by three rotations around the unit axes. Sometimes these angles are called *yaw, pitch and roll*, but mostly these three terms are used to define three rotations around the principal axis of the body itself. A disadvantage of Euler angles is that the order in which the three rotations are applied is crucial. A different order results in a different rotation since rotations do not commute. So, the angles are not independent. Another problem with Euler Angles are Gimbal locks. In case of Gimbal locks, one DoF is lost. A Gimbal lock occurs when a series of rotations at 90 degrees are performed. Due to the alignment of the axes, these rotations might cancel each other out.

A *quaternion* $\mathbf{p}$ consists of a scalar value w, and a vector in the imaginary ijk space.

$$\mathbf{p} = w + xi + yj + zk \tag{1}$$

So, a quaternion can be interpreted as the sum of a scalar part and a vector part. The quaternion space is denoted as $\mathbb{H}$. We define *unit quaternions* as quaternions with norm 1.

$$\|\mathbf{p}\| = \sqrt{w^2 + x^2 + y^2 + z^2} = 1 \tag{2}$$

The unit quaternion space, denoted as $\mathbf{S^3}$, is a 4D unit hypersphere. When considering a rotation θ around an axis $\mathbf{n}$, the corresponding unit quaternion will be

$$\mathbf{p} = (\cos\frac{\theta}{2}, \sin(\frac{\theta}{2})\mathbf{n}) \tag{3}$$

In the *exponential map* representation [Gra98], rotations are represented in a linear domain. Using the exponential map representation, we are also able to employ techniques that only operate in linear domains, such as principal component analysis [EMMT04]. In this representation, a rotation θ around a unit axis $\mathbf{n}$ is represented by a vector $\mathbf{r} \in \mathbb{R}^3$ where $|\mathbf{r}| = \theta$ and $\frac{\mathbf{r}}{|\mathbf{r}|} = \mathbf{n}$. There are some limitations of the exponential map. First, a rotation in $\mathbf{SO(3)}$ maps to an infinite number of vectors in $\mathbb{R}^3$. Second, there is no simple operation to combine rotations as there is when using quaternions.

2.2. Physical Model of the VH

In physical simulation, the body of the VH is typically modelled as a system of rigid bodies, connected by joints. Each of these rigid bodies has its own mass and inertia tensor. Movement is generated by manipulating joints torques.

Most physical animation systems [HWBO95, WH95, Woo98, ZH99, YLS04, ZvdP05] assume a uniform density for each rigid body. The density of the rigid bodies can be measured directly from cadavers, or using scanning systems that produce the cross-sectional image at many intervals across the segments [Win04]. The mass, center of mass and inertia tensor can then be calculated via the volume of the mesh that corresponds to the rigid body [Mir96].

To allow for collision detection and collision response, a geometric representation of the rigid bodies is needed. The mesh of the VH can be used for this representation. However, collision detection between arbitrary polygonal shapes is time consuming. Computational efficiency can be gained at the cost of some accuracy by approximating the collision shape of rigid bodies by basic shapes such as capsules, boxes or cylinders.

2.3. Biomechanical/Neurophysical Models of the VH

The central nervous system (CNS) is used to control our muscles, on the basis of sensor input. Here we describe some sensors used in biomechanical movement controllers, the employed muscle model and some models and invariants for motor control.

2.3.1. Sensors

Motor control needs information on the state of the VH. This information can be calculated from the DoFs, their velocities and the physical representation of the VH's body. It is convienent to compute intuitive higher level *sensors* that can be shared among different motion controllers [FvdPT01b]. Examples of such sensors are the center of mass (CoM) of the VH, the velocity of the CoM, contact information (are the

feet or other body parts in contact with the ground?), the location of the support polygon (the convex hull of the feet), and the zero moment point (ZMP). The ZMP is the point on the ground plane where the moment of the ground reaction forces is zero. If the ZMP is outside the support polygon, the VH is unbalanced and should fall over.

2.3.2. Modeling Muscles

Over 600 muscles can apply forces to our bones by contracting. One muscle can cover multiple joints (e.g. in the hamstring and muscles in the fingers). In real-time physical simulation methods, muscles are typically modeled as torque-motors at joints. Such a model provides control in real-time and has a biomechanical basis: it is hypotized that the CNS excerts control over joints at a joint or similar higher level [Win04]. To determine the torque applied by these motors, muscles are often modeled as a system of springs (representing elastic tendons) and dampers that cause viscous friction [Win04]. Joint rotation limits and maximum joint strength can be obtained from the human factors literature [WTT92, BPW93].

2.3.3. Motor Control

Motor control deals with steering the muscles in such a way that desired movement results. Robotic systems rely mostly on feedback control using very short feedback delays. In biological movement, feedback delays are large (150-250 ms for visual feedback on arm movement), so precise control of fast movement (as exhibited by humans) cannot be achieved using solely feedback control [Kaw99]. According to Schmidt [Sch75] people construct parameterized General Motor Programs (GMPs) that govern specific classes of movement. Different movements within each class are produced by varying the parameter values. The relation between parameter values and movement 'outcome' is learned by practicing a task in a great variety of situations. According to the equilibrium point hypothesis, control is not explicitly programmed, but emerges from the dynamic properties of the biomechanical system. In this model, the spring-like properties of muscles in, for example the arm, are used to automaticly guide the hand to an equilibrium point. Movement is achieved by a succession of equilibrium points along a trajectory. Feedback control (see 4.2.1.1), GMPs (explicitly in [Zel82, KW02], implicitly in 4.2.2, 4.1.3) and equilibrium point control (see 4.2.1.1.2) have been used in computer animation.

The GMP theory is supported by invariant features that are observed in motion. Gibet et al. [GKP04] give an overview of some of such invariant features, including Fitts' law, the two-third power law and the general smoothness of arm movement. Fitts' law states that the movement time for rapid aimed movement is a logarithmic function of the target size and movement distance [PM54]. The two-third power law [VT82] models the relation between the angular velocity and the curvature of a hand trajectory. Movement smoothness has been modeled as a minimization of the mean square of hand jerk (derivative of acceleration) [FH85] or the minimization of the change of torque on the joints executing the motion [UKS89]. Harris and Wolpert [HW98] provide a generalized principle that explains these invariants by considering noise in neural control. The motor neurons that control muscles are noisy. This noise is signal dependent: the variability in muscle output increases with the strength of the command. For maximum accuracy it is therefore desirable to keep the control signals low during the whole movement trajectory, thus producing smooth movement. Faster movement requires higher control signals, thus higher variability which leads to reduced precision. In computer animation, movement invariants have been used both in animation techniques [GLM01, KW02] and as evaluation criteria for the naturalness of animation (see 6.5.2). The notion of signal dependent noise has been exploited in the generation of motion variability (see 6.4.3).

3. Technologies From Related Fields

Several technologies from the fields of biomechanics, cartoon/film making and robotics are currently used in computer animation. Here we give a brief overview on such technologies, and show how they can be used in animation kinematics and physics. Most of these technologies are currently available in software libraries or toolkits. It is advisable to have some knowledge on how these technologies work, both to select the right software for your application and to apply them in a robust and efficient manner.

3.1. Kinematics

Kinematic technologies can be used to control or analyze information of a kinematic nature, such as joint angles, joint angle velocity or joint angle acceleration.

3.1.1. Keyframe animation

Keyframe animation is a technique borrowed from traditional cartoon animation, where a senior artist draws the key animation frames and his assistants draw the 'inbetweens'. Burtnyk and Wein [BW76] first proposed using keyframing for skeletal animation. In keyframe skeletal animation, an animator specifies the rotation of joints at certain moments, producing the so-called keyframes. The rotation of the joints in inbetweens is obtained by interpolating between those keyframes.

3.1.2. Motion Capture

Using motion capture, very detailed motions can be created. Motion capture tracks the movement of markers on a human performer at a high frequency. The recorded marker movement is used to reconstruct the rotations of the joints on a skeleton with similar proportions as the actor. This provides

animation keyframes. Since the VH to be animated typically has different body properties as the performer, the animation has to be *retargetted* [Gle98] to the body of the VH.

3.1.3. Interpolation

In order to determine the inbetweens one needs to interpolate the translational and rotational DoFs of the VH. Translational DoFs can be linearly interpolated, which results in C^0-continuity, or one can use piecewise, higher-order polynomials (*splines*) to enforce a higher order of continuity. For a more thorough explanation on splines, we refer the reader to a graphics textbook [WP00].

Unlike rotation matrices and Euler angles, quaternions are very well suited for rotation interpolation and some well-defined interpolation methods exist [Gra98]. The most common is the spherical linear interpolation (slerp) [Sho85]. A slerp will result in the shortest possible path on the surface of the 4D hypersphere. Shoemake [Sho85] explains how to construct spherical Bézier curves over the hyper sphere so one can blend a series of rotations smoothly. Applying this gives a higher order of continuity of the path on the Quaternion sphere. One can also use Spherical Cubic Interpolation (squad) [Boe82] that will also result into C^1 continuous paths on the hypersphere.

A straight line in between exponential maps in $\mathbb{R}^3$ is not the same as a slerp in $\mathbf{S^3}$. But, when the rotation axis of the two rotations do not differ too much and the right log map is used, the interpolations are visually indistinguishable. A log map maps an orientation in $\mathbf{SO(3)}$ to an infinite number of points in $\mathbb{R}^3$, corresponding to rotations of $2n\pi+\theta$ about axis $\mathbf{n}$ and $2n\pi-\theta$ about axis $-\mathbf{n}$. The log map should select the rotation in $\mathbb{R}^3$ that minimizes the Euclidean distance to the mapping of the previous rotation. Note that due to the linearity of the exponential map, one can also use other interpolation techniques, such as splines, for a higher order continuity.

3.1.4. Forward Kinematics

The generalized location of an *end effector* (the joint at the end of a chain of joints) $\mathbf{s}$ is a function of the rotations and translations $\mathbf{q}$ of all joints in the chain.

$$\mathbf{s} = \mathbf{f}(\mathbf{q}) \tag{4}$$

$\mathbf{f}$ is defined by the topology of the skeleton. Besides the desired world position of the end effector, $\mathbf{s}$ can also contain its world rotation. Forward Kinematics finds $\mathbf{s}$, given $\mathbf{f}$ and $\mathbf{q}$.

3.1.5. Inverse Kinematics

Inverse kinematics (IK) specifies the inverse problem: finding $\mathbf{q}$, given $\mathbf{s}$.

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{s}) \tag{5}$$

Often this problem of finding joint rotations is overspecified, that is, there are multiple combinations of joint DoF values that put the end effector in the right location. Several techniques exist to solve this problem. The IKAN toolkit [TGB00] solves anthropomorphic limbs analytically. It finds all joint configurations that solve the IK problem for an arm or leg. For larger chains numerical solutions are necessary. If these numerical techniques start out in a natural starting pose (denoted by $\mathbf{q}$) in which the end effector is already close to the goal, a natural pose will often be achieved. Several numerical techniques are outlined below.

3.1.5.1. Jacobian inverse method The Jacobian inverse method linearizes the problem about the current joint configuration [Wel93]. The relation between the joint velocities and the velocity of the end effector is

$$\dot{\mathbf{s}} = \mathbf{J}\dot{\mathbf{q}} \quad \text{with} \quad \mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \tag{6}$$

$\mathbf{J}$ is an $m \times n$ matrix, with m the dimension of the end effector (3 for just position, 6 for position+rotation) and n the number of joint variables. Inverting equation 6 this gives the joint velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{s}} \tag{7}$$

The following iterative approach is then used to find $\mathbf{q}$:

1. Find the derivative of $\mathbf{s}$: $\dot{\mathbf{s}} = \mathbf{s} - \mathbf{f}(\mathbf{q})$
2. Calculate $\mathbf{J}$
3. Invert $\mathbf{J}$
4. Using equation 7, calculate $\dot{\mathbf{q}}$
5. Integrate $\dot{\mathbf{q}}$ to obtain $\mathbf{q}$
6. Repeat until $\mathbf{f}(\mathbf{q})$ is close enough to $\mathbf{s}$

Typically $\mathbf{J}$ is non-square. $\mathbf{J}^{-1}$ then has to be replaced by a pseudo-inverse of $\mathbf{J}$, $\mathbf{J}^{\dagger}$. Using the Moore-Penrose pseudo-inverse ensures that joints rotate as little as possible to match the desired end effector position [BBZ91].

Redundancy can be exploited by defining additional tasks, subject to satisfying the primary positioning task. This can be done by modifying equation 7 to:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{s}} + (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})\Delta\mathbf{q}_{sec} \tag{8}$$

Where $\Delta\mathbf{q}_{sec}$ defines a secondary task by specifying a desired joint variation and $(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})$ is a projection operator which selects those components of $\Delta\mathbf{q}_{sec}$ that do not change the end effector location, the so-called *null space* of the Jacobian.

The main drawbacks of the Jacobian inverse method are that no stable solutions can be found near singularities of the Jacobian [Wel93] and that calculating the inverse Jacobian is computationally expensive. If the end effector location is specified using just positional constraints, rather than positional and rotational constraints, a solution for $\mathbf{q}$ can be found almost 2.5 times faster. Many IK-problems, including walking, can adapted so that just positional constraints are required to position the end effector [MM04]. The damped

least squares technique [Mac90] can be used to provide stable solutions near singularities. It trades convergence speed for solution stability.

3.1.5.2. Cyclic-Coordinate Descent method Cyclic-Coordinate Descent (CCD) [Lue84], introduced as IK solving mechanism by Wang and Chen [WC91] iterates through the joints, typically starting with the one the closest to the end effector, and varies one joint variable at a time based on a heuristic. An example of such a heuristic is to minimize the angle between the vector originating from the current joint toward $\mathbf{s}$ and the vector from the current joint towards $\mathbf{f}(\mathbf{q})$. Unlike the Jacobian inverse method, which distributes joint rotation changes equally along the chain, CCD has a preference of moving distal links first [Wel93]. The calculation costs per iteration are low, but this method can exhibit a poor convergence rate [Wel93]. The CCD method is completely immune to difficulties near singularities.

Because CCD typically results in unnatural poses, its direct application is not very suitable for VH animation. Kulpa and Multon [KM05c] propose an adaption of the CCD-algorithm to address this shortcoming. In every iteration, first a group of joints near an end effector (typically an arm or leg), are analytically positioned in such a way that the angle between the vector originating from the root joint of the group toward $\mathbf{q}$ and the vector from the root joint of the group toward $\mathbf{s}$ is minimized. Then the remaining joints in the kinematic chain are rotated using the CCD algorithm described above. This results in an algorithm that is computationally very cheap and that produces natural poses.

3.1.5.3. Optimization based methods Optimization-based methods cast the IK problem into a minimization problem. The distance between $\mathbf{f}(\mathbf{q})$ and $\mathbf{s}$ serves as an error measurement.

$$s_{err} = (\mathbf{f}(\mathbf{q}) - \mathbf{s})^T (\mathbf{f}(\mathbf{q}) - \mathbf{s}) \tag{9}$$

The goal is then to find the DoF vector $\mathbf{q}$ that minimizes the error. Upper and lower bounds q_{L_i} and q_{U_i} can be specified for each DoF i in the chain. The minimization of s_{err}, given $q_{L_i} \leq q_i \leq q_{U_i}, i = 1..n$ is a classic non-linear constraint optimization problem [Wel93], which can be solved by a number of standard numerical methods [GMW82], for which several toolkits are available [BDV, The, Sta].

3.2. Physical Simulation

Kinematic based systems are intuitive, but do not explicitly model physical integrity. As a result kinematic animation does not always seem to respond to gravity or inertia [MTT96]. Physical simulation models the body of the VH as a system of rigid bodies, connected by joints. Each of these rigid bodies has its own mass, inertia and possibly other physical properties. Movement is generated by manipulating torques on the joints. Several formulations of the dynamics formulations of such a system exist.

3.2.1. The Physical Equations of Motion

The equations of motion of a system of connected rigid bodies describe the relation between joint-torques and the linear and angular acceleration of the rigid bodies. The system has to be constrained so that it moves only in ways that the DoF of the joints of the body allow. The *constraint force* approach applies constraint forces on all rigid bodies to satisfy movement constraints. The *reduced coordinate* approach reshapes the equations of motion in such a way that only torques and accelerations on the DoF are allowed.

The constraint force approach is simple to understand and easy to design as a modular system in software [Bar96]. Furthermore, the constraint force approach can be used to specify non-holonomic constraints, or constraints that are hard to parameterize (for instance those in deformable rather than rigid bodies). However, because the constraints have to be enforced by forces, numerical errors can cause 'drifting': two rigid bodies connected by a joint have the tendency to drift apart. Constraint stabilization techniques have to be used to prevent this.

3.2.2. Forward Dynamics

Forward dynamics(FD), pioneered for skeletal animation by Armstrong and Green [AG85], is the animation process that moves a VH when torques on joints are provided. Efficient $O(n)$ algorithms, with n the number of DoF, exist to solve FD for systems of rigid bodies without loops, both by using constraint force methods [Bar96] and by using reduced coordinate methods [Fea07]. At the cost of computational speed, these methods can be extended to solve for loops.

3.2.3. Inverse Dynamics

Inverse dynamics (ID) is the process of finding the torques and forces on the joints in a body given the movement of its segments. It can be used to predict torques needed for kinematically specified movement and to check if joint torques exceed comfort or strength limits.

3.2.4. Friction and Impact

Collision Detection deals with finding the time of collision and the collision contact points, lines or surfaces between rigid bodies in a simulation. Several algorithms exist to detect collision between meshes. A simple representation of the bounds of the geometrical representation of a rigid body (for example: a bounding box, bounding sphere or bounding capsule) can be used to determine for what bodies the bounding shape overlaps and thus where a more extensive collision check is needed. The temporal or spatial coherency between rigid bodies in the simulation can be exploited to reduce calculation time. We refer the interested reader to [BW97] for a more thorough explanation of collision detection algorithms and an overview of specialized literature on this subject.

In rigid body simulation, no inter-penetration of bodies is

allowed. There are two types of contact that need to be dealt with. A *colliding* contact is defined as a contact between two bodies that have a velocity towards each other. Resolving these contacts requires an instantaneous change in the velocity of the bodies involved in the contact. The coefficient of restitution ε of the collision determines the amount of kinetic energy lost in the collision. If $\varepsilon = 0$, the collision is inelastic, which effectively stops the colliding objects at the collision point. If $\varepsilon = 1$, the collision is elastic, no kinetic energy is lost.

If two frictionless rigid bodies are in *resting* contact, they are resting at each other (with $\mathbf{0}$ velocity). In this case, a contact force has to be exerted on one or both bodies to prevent inter-penetration.

The Coulomb friction model (or an approximation of this model) is typically used in rigid body simulation to model the friction between contacts. In this model, the friction force $\mathbf{F}_T$ is linearly related to the contact force that acts in the direction of the normal of the friction surface $\mathbf{F_N}$.

$$|\mathbf{F}_T| \leq \mu |\mathbf{F}_N| \tag{10}$$

where μ is the friction coefficient. At a contact point with *static friction*, the relative tangential velocity between bodies is $\mathbf{0}$. If the relative tangential velocity is nonzero, *dynamic friction* occurs and $|\mathbf{F}_T| = \mu |\mathbf{F}_N|$.

Resolving these contact and friction forces is a complex numerical problem, we refer the interested reader to the vast literature on this subject (among many other publications: [BW97, Bar94, Ste00]).

3.2.5. Physical Simulation Software and Hardware

Several software toolkits can be used for FD and/or impact and friction handling, including the open source software such as the Open Dynamics Engine [Smi08] and the Bullet Open Source Physics Library [Cou08], and commercial packages like SD/Fast [SR01] and Havok Physics [Hav08b]. Dedicated physics hardware is becoming available to handle physical calculations, including rigid body dynamics: Nvidia's PhysX Physics Processing Unit (PPU) [Nvi08], which uses either dedicated hardware or the graphical processing unit on their videocards, and Sony's Playstation 3 cell processor [Son08]. Boeing and Bräunl [BB07] provide a recent comparison of physics engines. Their benchmark software is available online and kept up to date with the latest physics engines. For real-time VH simulation, the accuracy and stability of the constraints and the calculation time is important, but depending on the application the VH is used in, other simulation aspects, such as the accuracy of collision detection and friction handling could also play an important role.

4. Animation Techniques

We define animation techniques as techniques to construct *motion spaces*. Each motion space has a certain function (for example: the motion space of walk cycles, beat gestures, left hand uppercuts). They can define motion for the full body of a VH or on a subset of the joints of the VH. An instance of the motion space is a motion primitive. A motion primitive is selected from the motion space using a set of *parameters*. Exactly what parameters can be used to select the motion primitive differs per motion technique. Motion primitives can be split up in *phases*. For example, a running motion might contain phases for flight, left foot heel contact, left foot heel and toe contact, etc. We classify animation techniques by the information they use to construct a motion space (see Figure 1 and 2).

4.1. Motion Editing

Motion editing techniques aim to generalize motion spaces from recorded motion primitives. Motion modification methods construct the motion space by applying modifications to a single recorded motion primitive. Combination techniques, first proposed by Lamouret and van de Panne [LvdP96], make use of multiple motion primitives in a motion capture database to construct a motion space.

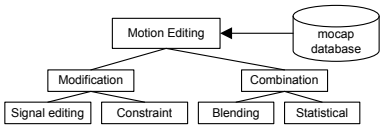

Figure 1: *Classification of motion editing techniques.*

4.1.1. Signal Editing

A motion primitive can be considered a continuous function that maps time to the DoF of a skeleton. So, the value of a DoF over time can be considered a *signal*. Therefore many techniques from the field of signal processing can be applied to create a motion space. Bruderlin and Williams [BW95] are the first that consider some motion editing problems as signal processing problems. One of the signal processing techniques they use is *displacement mapping*. With this technique it is possible to change the shape of a signal while maintaining continuity and preserving the global shape of the signal. To change the motion primitive, the animator just needs to set some additional keyframes (or have them determined by IK). From these keyframes, a displacement map can be calculated that encapsulates the desired displacement (offset) of the signal. Splines can be used to calculate the in-betweens. The displacement map then yields a displacement for every frame, which can be added to the original signal. Other techniques include *motion waveshaping* that makes it possible to limit joint ranges and introduce stylistic elements in the joint angles. The signal is directed through a *shape function* that alters the signal.

Witkin and Popovic [WP95] present motion warping, which is a combination of a displacement mapping and time

warping (see 4.1.3.2). Lee and Shin [LS99] present hierarchical displacement mapping. At each iteration, a finer displacement map (using splines with a higher knot density) is added to the motion until the error is small enough. This avoids using overdefined splines and hence, high computation times.

4.1.2. Constraint Based Motion Editing

Constraint based techniques create the motion space by editing or preserving (typically geometric) features in recorded motion primitives by explicitly stating them as constraints [Gle01]. Note that some signal editing techniques described above are also constraint based, such as [WP95, BW95].

The desired joint rotation changes can be calculated from geometric constraints, using IK. These constraints can then be enforced at keyframes [CPK99]. This does not guarantee constraint enforcement at the inbetweens. Alternatively, constraints can be enforced at every frame, as is done in [LS99]. To make sure the resulting motion in the motion space is smooth and propagated through non-constrained frames, the IK solution is 'filtered' using B-splines. Gleicher [Gle01] calls the family of solutions that uses such an approach 'Per Frame Inverse Kinematic + Filtering' (PFIK+F). To demonstrate the generality of PFIK+F, they implement it with a different IK solver and a convolution based linear filter. Boulic et al [BLCHB03] provide a PFIK+F framework that can handle multiple constraints. It resolves possible conflicts in constraints by satisfying those with the highest priority first. It uses inverse Jacobian IK solvers (see 3.1.5.1), using the null space of the Jacobian of the solver for the high priority constraints to restrict the domain of the Jacobian of lower priority solvers. An ease-in ease-out curve is used as a filter to smoothly activate and deactivate the constraints.

An alternative approach by Gleicher [Gle97] is to pose the constraint specification as a numerical constrained optimization problem:

$$\text{minimize} \quad R(\mathbf{q}) \quad \text{subject to} \quad \mathbf{C}(\mathbf{q}) = \mathbf{c} \qquad (11)$$

Where $R(\mathbf{q})$ is the objective function, $\mathbf{c}$ is a vector of desired constraint values and $\mathbf{C}$ is a vector function of the constraints. The objective is to minimize the distance between the motion capture data and the constrained motion. To allow real-time execution of this optimization, an objective function is chosen that evaluates the distance between the motion capture data and new motion efficiently and the constraints are only enforced at key frames. The optimization approach allows for the specification of any constraint that can be specified as a function of $\mathbf{q}$ and is thus more flexible than PFIK+F [Gle01]. The geometric constraints that can be solved with PFIK+F are a subset of those that can be solved using the optimization approach. Optimization can add (among many others) constraints for a region an end effector must stay in, fixed distances between end-effectors or inter frame constraints (for example: have the hand in the same position at different frames without having a specific location in mind). This flexibility comes at a cost: it is not ensured that the constraints are met at the inbetweens and the solution time of the optimization process is less predictable than that of a PFIK+F approach. We refer the reader to [Gle01] for a more thorough comparison of the two methods.

4.1.3. Blending

Blending methods, pioneered in [WH97a] construct the motion space using an interpolating of recorded motion primitives. Such an interpolation can be done using one of the techniques discussed in 3.1.3, or using specialized techniques to blend in, for instance, the PCA [IST02] or Fourier [UAT95] domain.

In order to correctly interpolate motion primitives, one needs to preprocess them such that they correspond in time (especially at key events such as foot plants) and space. Kovar and Gleicher present *registration curves* that automatically determine the time, space and constraint correspondences between a set of motion primitives.

4.1.3.1. Spatial aligning Before blending, motion primitives should be aligned in space. For example, when blending two walk cycles, the root translation must globally be in the same direction. One can obviously align the root orientation and position, but several other strategies exist. Kovar et al. [KGP02, KG03] present a technique that determines the 2D transformation by registering point clouds corresponding to the poses over a window of frames.

4.1.3.2. Time warping To align corresponding phases in motion primitives, one can apply *time warping*. This enforces the temporal correspondence and reduces unnatural motion artifacts. Specific key events, such as heel strikes, need to be time aligned. Several time warping algorithms exist. Some algorithms require the user to manually annotate these key events [RCB98, PSS02] after which they linearly interpolate the corresponding key times to do time warping. The timewarp technique of [BW95] aligns two motion primitives uses dynamic programming to minimize a global difference function. Kovar and Gleicher [KG03] extend this work by creating a time warp curve which applies for more than two motions primitives and is strictly increasing.

4.1.3.3. Constraint matching Motion primitives are often annotated with additional constraints that can be used for various postprocessing techniques. An example is the moments of heel strikes. These constraints can be set by the animator or be determined automatically [IAF06, BB98]. When two motion primitives are blended, the resulting motion primitive must also contain correct annotations. Kovar and Gleicher [KG03] present a technique to automatically find the corresponding constraints between two annotated motions.

4.1.3.4. Pose Distance Metrics In general, one can only interpolate between poses that "resemble" each other. When this is not the case, visual artifacts such as foot skating may appear. A *distance metric* quantifies the resemblance between poses. Van Basten and Egges [vBE09] present an overview and comparison of various distance metrics.

4.1.4. Statistical models

Statistical methods construct the motion space from statistical models learned from the statistical variation of recorded motion primitives. Several statistical models can be used, including Hidden Markov Models (HMM) [BH00], Linear Dynamic Systems [LWS02], Scaled Gaussian Process Latent Variable Models (SGPLMVM) [GMHP04], Principle Component Analysis (PCA) [EMMT04], or variogram functions [MK05].

4.2. Simulation

Simulation methods use parameterized physical or procedural models to construct the motion space (see Figure 2).

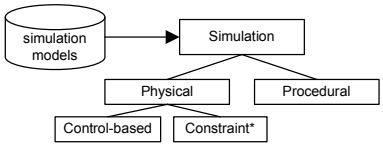

Figure 2: *Classification of simulation techniques. Constraint based physical animation is currently not a real-time technique.*

4.2.1. Physical Simulation

In physical simulation the motion space is constructed using a physical simulation model that applies torques on the joints of the VH.

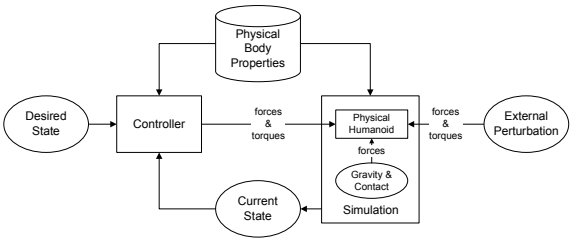

Figure 3: *A physical control system*

4.2.1.1. Physical Simulation using Controllers In this unconstrained control method, a dynamic controller applies the torques on the joints of the VH. Such a controller and the dynamic system it controls (the physical body of a VH) together form a control system [KMB96] (figure 3). The input to controller is the desired value of the system's variables. The output is a set of joint torques that, when applied to the system, should guide the system's variables closer to the desired system variables. The controller can make use of static physical properties (like mass, or inertia) of the physical body it controls to do this. Such a control system can, to a certain extend, cope with external perturbation, in the form of forces or torques exerted on the body, like those of a hit in a boxing match. The goal of the system is to minimize the discrepancy between the actual and desired system values. Forces and torques from the controller, forces from gravity and ground contacts, and, optionally, forces and torques caused by external perturbation are applied to the physical body. The body is then moved using FD. The new state of the body is fed back into the controller.

4.2.1.1.1. Proportional Derivative Control A simple to implement and often used [HWBO95, WH95, Woo98, ZH99, FvdPT01b, YLS04, ACSF07] controller is the Proportional Derivative (PD) controller. The output torque of the PD-controller is proportional to the difference in position and velocity between the desired state and the actual state:

$$\tau = k_p(x_d - x) + k_d(\dot{x_d} - \dot{x}) \tag{12}$$

in which x_d is the desired state, x is the actual state and k_p and k_d are the proportional and derivative gains. Note that the system reacts similarly as a springer-damper system, with spring gain k_p and damper gain k_d. Typically x_d is a desired DoF value, but other state variables are used in more complex PD-controllers (such as CoM position in balancing [Woo98]). Finding appropriate values for k_p and k_d is a manual trial-and-error process. They depend on characteristics of both the system and the motion.

4.2.1.1.2. Antagonist Control Neff and Fiume [NF02] use a slightly different formulation of the PD-control equation, that has more intuitive control parameters, but the same error response. It is based on agonist and antagonist muscle groups around joints, that are modeled as springs:

$$\tau = k_{p_L}(\theta_L - \theta) + k_{p_H}(\theta_H - \theta) - k_d\dot{\theta} \tag{13}$$

in which θ_L and θ_H are the spring set points, which serve respectively as desired lower and upper limit for the joint rotation θ. τ is the output torque. k_{p_L} and k_{p_H} are the spring gains. The animator can specify the desired amount of stiffness, defined as $k_{p_L} + k_{p_H}$. Equilibrium point control (see 2.3.3) is used to calculate k_{p_L} and k_{p_H}, given the desired stiffness and external forces (typically gravity). Movement is achieved by gradually moving the equilibrium position.

4.2.1.1.3. End Effector Control Rather than directly controlling joint torques, a virtual force is applied on an end effector. The relation between joint torques τ and the virtual force $\mathbf{f}$ is given by

$$\tau = \mathbf{J}^T\mathbf{f} \tag{14}$$

[Cra89]. Rather than setting up torques for all joints in the chain, only **f** needs to be set using a control technique. Redundancy (that is, multiple values of τ can realize **f**) can be exploited for secondary tasks, by using the null space of the Jacobian [SK05].

4.2.1.1.4. Automatic Controller Generation Searching techniques or evolution-based machine learning techniques have been employed to automatically generate controllers that map sensor inputs (joint angles, ground touch) to joint torques, in such a way that a certain measure (distance traveled, energy expended, distance from stylized reference pose) is optimized [vdP93, vdPKF94, Sim94, AFP*95, SvdP05]. Using such techniques, locomotion controllers for simple creatures with few DoFs can be created. However, so far automatic controller generation techniques have not proven to scale up to provide natural motion primitives for full-sized VHs.

4.2.1.1.5. Physical Controllers Toolkits The Dynamic Animation and Control Environment [Sha07] provides researchers with an open, common platform to test out and design physical controllers using scripting. Naturalmotion's Endorphin [Nata] is a commercial animation system that provides authors a predefined set of controllers. It offers animation authoring through controller parameterization, controller combination, physical constraint handling (e.g. lock hands to a bar for a 'hang on bar' motion) and several ways to integrate motion capture with physical simulation. Naturalmotion offers the Euphoria [Natb] toolkit to handle such functionality in real-time so that it integrates with a game engine. Details on how Naturalmotion software handles this functionality (as far as disclosed) are discussed in the appropriate sections.

4.2.1.2. Constraint Control Methods Constraint based methods calculate those torques on joints that satisfy certain animation constraints (pose at key time, ground contact, etc). In general, the problem of solving for these torques is overspecified. That it, there are many possible muscle torque paths that lead to the desired constraints. An objective function $R(\mathbf{q})$ (with $\mathbf{q}$ a vector containing joint rotations and applied torques) can be introduced to specify a certain preference for solutions. Typically, the objective functions are biomechanically based. Examples are the minimization of expended energy, used for relaxed swinging motion like walking, or the minimization of jerk of an effector, used in coordinated goal-directed motion or a weighted combination of those two [BBZ91]. The constraint control problem can be stated as a non-linear optimization problem (compare with 3.1.5.3): minimize $R(\mathbf{q})$ subject to n constraints $C_i(\mathbf{q}) = 0, i = 1..n$, in which $C_i(\mathbf{q})$ is the constraint function for constraint i. Spacetime optimization is introduced by Witkin and Kass [WK88], using this principle.

Several techniques have been proposed to speed up the calculation process of the optimization [Coh92, GTH98, LP02, FP03], typically at the cost of some physical realism. Even with those speedups, constrained based control methods are currently not a feasible option for real-time animation.

4.2.2. Procedural

Procedural simulation defines parameterized mathematical formulas to construct the motion space. Such formulas can describe joint rotation directly [Per95], or describe the movement path of end effectors (such as hands) through space. The latter is typically used to mathematically construct gesture motion spaces [CCZB00, KW02, HHL*05, NKAS08].

4.3. Strengths and Weaknesses of Different Motion Techniques

Motion editing techniques retain the naturalness and detail of recorded motion primitives or motion primitives generated by skilled artists. However, motion editing techniques produce natural motion only when the modifications to the recorded motion primitives are small. Techniques that make use of multiple recorded motion primitives to generate the motion space retain naturalness over larger modifications than techniques that adapt a single recorded motion primitive [Gra00]. However, both blending and statistical techniques suffer from the curse of dimensionality: the number of required recorded motion primitives grows exponentially with the number of control parameters [Gle08]. Furthermore, motion editing techniques do not provide physical interaction with the environment and motion editing can invalidate the physical correctness of motion (see 6.1). Motion editing is useful for creating animation in advance for non-interactive applications (like films), or for applications in which large modifications are not needed such as free dance [SNI06, hKPS03]. For other domains like games, naturalness can only be assured by using a huge database of recorded motion primitives.

Physically simulation provides physically realistic motion and (physical) interaction with the environment. Physical controllers can robustly retain or achieve parameters under the influence of external perturbation. This robustness comes with a disadvantage: precise timing and limb positioning using physical controllers is an open problem. While physical simulation provides physically correct motion, this alone is often not enough for motion to be natural. Therefore, physical simulation is mainly used to generate human motion that is physically constrained and in which interaction with the environment is important, such as motion by athletes [HWBO95, WJM06], stunts by stunt men [FvdPT01b], or falling motions [WH00, SPF03, Man04].

Procedural animation offers precise timing and limb positioning and can easily make use of a large number of

parameters. However, it is hard to incorporate movement details such as those found in recorded motion primitives into the mathematical formulas that steer procedural motion. Furthermore, to maintain physical naturalness, it has to be explicitly authored in the procedural model for all possible parameter instances. Expressive motion, as used in talking and gesturing VHs, requires many control parameters and precise timing to other modalities, such as speech. It is therefore typically the domain of procedural motion techniques [Per95, PG96, CCZB00, KW02, HHL*05, vWNRZ06, NKAS08].

The qualities of motion editing and motion simulation techniques can potentially be combined by taking into account which of the qualities is needed in a certain situation, or by determining what quality is needed on what body part. For example, a VH can be steered by motion editing until a physical interaction with the environment is needed, which then will be handled by physical simulation. The flexibility and precision of procedural motion can be used to generate arm gestures on a VH which retains balance using physical simulation on the lower body. Throughout the remaining sections, we will show several examples of such combinations that enhance naturalness and/or control, as we discus the control and naturalness provided by different motion techniques.

5. Control

Animation involves the creation of *animation plans* that typically span multiple motion spaces and are executed by multiple motion primitives. To be able to deal with interactive and changing environments, such plans need constructed and adapted in real-time.

Control involves the parameterization, combination and concatenation of motion spaces. Parameterization deals with selecting the motion primitive from a motion space that satisfies some desired properties (for example: select the motion primitive that hits the tennis ball from the forehand motion space). Motion spaces that are active on different body parts can be combined to cover a wider natural motion space (for example: a walk cycle motion space and a chew gum motion space can be combined to a walk while chewing gum motion space). Motion spaces are concatenated to form a more complex animation plan (for example: concatenate walk cycle primitives to form a walk on a path).

5.0.1. Parameterization in Procedural Motion

Procedural animation is very parameterizable by design, the parameters can be expressed in terms of variables of the motion functions. Pose constraints are typically satisfied by using the parameters of procedural functions to enforce IK positions or joint rotations. Authoring procedural motions requires specifying how each parameter influences the motion. For higher level parameters, this is not a very intuitive process. Typically the procedural animation techniques use a mapping of intuitive high level control parameters to the lower level parameters that select the motion primitive. A crucial issue in parameterization of procedural motion is parameter conflict resolution: the procedural model must be able to deal with parameter spaces that are unachievable or result in unnatural movement.

Neff and Fiume [NF05], design a hierarchical framework for procedural motion and provides a generic parameter mapping framework. Lower level parameters specify the motion on a single joint or group of joints (called an action in [NF05]). Higher level parameters map to lower level parameters through a script created by an animator. Motion primitives are constructed from various, possibly conflicting low level and high level parameters. Therefore, several mechanisms are in place to handle conflict resolution. Low level parameters (placed on a single DoF, rather than on the whole body) take precedence over high level parameters. Parameters defined on actions take precedence over default parameters defined in a *Sketch* that models the VHs style (see 6.3.2).

Densley and Willis [DW97] modify poses by mapping emotional parameters to adaptations in stance and joint rotation. The exact mapping is not disclosed in their paper.

Chi et al. [CCZB00] claim that Effort and Shape parameters from Laban Movement Analysis (LMA) not only provide means to parameterize gesture, but are necessary elements of gesture. Shape involves the changing forms that the body makes in space. Effort describes dynamic qualities of movement, like weight (light, for example dabbing paint on a canvas or strong, for example punching someone in the face in a boxing match) and flow (uncontrolled, for example shaking of water vs. controlled, for example carefully carrying a hot cup of tea). Their work provides a computational framework that maps abstract Effort and Shape parameters to lower level parameters that guide arm movement, specified as end effector key locations. Shape parameters influence the position of the hand in space on those key locations. Effort parameters influence the path and timing of the movement toward the end effector location. In later work, Badler et al. [BAZB02] achieve emotional parameterization by mapping emotion to LMA parameters.

Howe et al. [HHL*05], use a smaller but quite similar set of parameters. From a literature review they conclude that six parameters (activation, spatial extend, temporality, fluidity, power and repetivity) are sufficient to specify gesture expressivity [MHP04]. The parameter selection is based on what humans can observe and reliably recognize. In their system, gestures are generated by TCB splines [KB84] defining the trajectory of the hands. The six high level parameters are mapped to low level parameters that modifying the timing and position of the control points in the spline or set the tension, bias and continuity of the spline. Their high-level parameters are intuitive, but not independent, specif-

ically they mention an unresolved conceptual interdependence between the power and temporal extend (roughly duration) parameters.

5.1. Parameterization of Motion Spaces

Parameterization deals with selecting a motion primitive from the motion space, based on certain parameter values. One common parameterization is the specification of a pose constraint (for example, requiring the hand to be at a certain location) at a desired time. It can easily be checked if or how precise such constraint is achieved by checking the motion primitive at the frame the constraint is imposed on. Higher level parameterizations deal with parameters like emotion or physical state (such as tiredness).

5.1.1. Parameterization in Motion Modification

Recorded motion primitives can be modified to adhere to pose constraints, using motion modification techniques (see section 4.2.1.2 and 4.1.1). These techniques only allow small modifications. Larger modifications can be made by motion combination techniques, such as blending (see section 4.1.3) and statistical modeling (see section 4.1.4), that use multiple motion segments as a basis for the new motion.

5.1.2. Parameterization using Constraint Editing

Amaya et al. [ABC96] state that emotion is observed in motion timing and spatial amplitude. An emotion transform is applied on neutral motion using non-linear timewarping and a spatial amplitude transform technique based on signal amplifying methods. The required timewarp and amplification for such an emotion transform is obtained by determining the emotional transforms needed to get from recorded neutral movement to the same movement executed in an emotional style. Hsu et al. [HPP05] describe a similar method for style transform, using a Linear Time Invariant model [Lju98] rather than signal amplification for the spatial transform.

5.1.3. Parameterization using Blending

Blending techniques have to solve the inverse motion interpolation problem [SM01] to achieve the desired pose: a set of motion primitives and their interpolation weights have to be found so that after blending a motion primitive with the desired pose constraints at the desired time results. Many blending techniques have been developed to solve a subset of the pose constraint problem: positioning an end effector at a desired position $\mathbf{s}_{des}$, specified by three parameters. Blending does not yield a linear parameterization of the parameter space [RSC01]. That is, if $\mathbf{s}_{des}$ is perfectly inbetween $\mathbf{s}_1$ and $\mathbf{s}_2$, this does not mean that a blend with interpolation weights of 0.5 of the joint rotation vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, placing the endeffector at $\mathbf{s}_1$ and $\mathbf{s}_2$, will end up placing the end effector at $\mathbf{s}_{des}$.

Rose et al. [RCB98] use scattered data interpolation to compute a best linear map between blend weights and motion parameters. Radial basis functions are then created in this space, centered on each recorded motion primitive. The run time cost of the interpolation is $O(n)$, with n the number of recorded motion primitives. For desired parameters far from the examples, blend weights are based purely on the linear approximation and hence are effectively arbitrary [KG04]. Grassia [Gra00] approximates the end effector position using blending and uses a constrained based method (see 4.1.2) to exactly position the end effector at the goal position. Many other techniques make us of pseudo example motion primitives, created by setting predefined blend weights. Wiley and Hahn [WH97a] constructs an dense, regular grid in parameter space offline, in a pre-computing step that exhaustively searches through interpolation weights and motion primitives to find the desired end effector locations on the grid. The grid can then be used to efficiently select the motion primitives to be interpolated. The interpolation weights are assumed to vary linearly with the motion parameters in such a dense grid. In later work Rose et al. [RSC01] use the smoothness of the function that maps blend weights to parameter values to create pseudo examples online at selected positions. Kovar and Gleicher [KG04] create random pseudo samples online, in/near the bounding box of the parameter space. By using k-nearest neighbor interpolation rather than interpolating from all samples, the run-time cost of their algorithm is independent of the number of recorded and pseudo example motion primitives.

Using blending methods, the 'degree' of an emotion or physical state can be adapted. For example: by blending a happy walk with a normal walk, a slightly happy walk can be obtained [RCB98, IST02]. Unuma et al. [UAT95] introduces blending in the Fourier domain for cyclical motions (such as walking and running). Such a Fourier domain blend ensures that the motions that are to be blended are time-aligned automatically, so time-warping is not needed in the pre-processing steps. For walking and running, the Fourier description provides parameters to control the step size, speed, duration of the flight stage and maximum height during the flight stage. Similar motions with different emotional or physiological aspects (brisk, tired, happy, etc) can be blended in the Fourier domain, so that these aspects can be used as motion parameters. Fourier descriptions can also be used to transfer motion aspects: by applying the Fourier description of briskness from a brisk walk onto a normal run, a brisk run is created. Because the parameters are qualitative, strict accuracy is unneeded: the blending method described above do not ensure that the desired parameters are achieved by the blend weights.

Torresani et al. [THB07] provide numerically accurate parameterization of three of the LMA Effort parameters (see section 5.0.1). A blend is created between two recorded motion primitives with annotated LMA parameter values. The LMA parameter values of the blend are then again annotated. This annotated motion primitive is used to learn a function

that maps blend weights, input joint angle data and input LMA parameter values to the LMA parameter values of the blended motion. The style of a motion with unknown Effort parameters can then be adapted to a desired set of Effort parameters by blending. This entails finding its k-nearest neighbors in the database of annotated motion primitives and find the motion primitive pair that, with the optimal blend weight, approximates the desired LMA parameter values the best. The optimal blend weights are found by uniformly sampling the blend weights space for each pair. At the cost of computation time and annotation effort (by an LMA-expert), this method achieves the generation of motion that more precisely matches desired LMA Effort parameter values than the simpler linear interpolation schemes described above.

5.1.4. Parameterization in Statistical Models

Grochow et al. [GMHP04] search their SGPLMVM model representation of the motion space using optimization to find motion primitives with poses satisfying certain constraints.

Li et al.'s [LWS02] motion texton representation of the motion space allows the construction of motion primitives by specifying poses at selected frames.

Mukai and Kuriyama [MK05], create a geostatistical model of a set of recorded motion primitives with given pose parameters. Geostatistical interpolation is then used to obtain the motion primitive with the desired pose constraints. This method is more accurate in achieving the desired pose constraints than blending methods that use radial basis functions (provided that they do not employ pseudo examples). It is more efficient (in terms of calculation time and memory usage) than blending methods that do use pseudo examples.

Carvalho et al. [CBT07] introduce a constraint based editing method that uses the same IK solver as [BLCHB03] on a low-dimensional statistical motion model rather than on the full body. This low-dimensional model is a statistical model generated using principle component analysis (PCA) or probalistic PCA (PPCA). Their system takes less calculation time, and is, according to the authors, in some cases more natural than the PFIK+F approach used in [BLCHB03].

In human motion, there are many correlations between joint actions [PB02]. Statistical methods [EMMT04] and machine learning [BH00] have been employed to find orthogonal parameters in a set of recorded motion primitives. Because the parameters are independent, it is not necessary to resolve parameter conflicts. However, the movement parameters learned in such approaches are not very intuitive to use and are highly depended on the training data. For example, [BH00] reports having a parameter that sets both the speed *and* the global pose. Therefore, such parameters are typically used solely to create small variations on existing motion.

5.1.5. Parameterization using Physical Simulation

The desired state of a controller can be used as a set of motion parameters. Parameters like desired joint rotation, pelvis height or CoM position provide intuitive direct low-level control. However, many other physical parameters of controllers, such as stiffness and damping gains do not provide intuitive control and are typically tweaked by trail and error.

Satisfying pose constraints precisely and timely using physical controllers is still an open problem, since in general it is unknown if and when a controller achieves such a pose constraint. Some recent efforts attempt to address this issue. Neff et al. [NKAS08] uses empirically determined offsets on the pose time and angular span multipliers on the pose itself, so that their system achieves poses on time, for certain classes of movement (e.g. gesture). Other systems rely on critically damped controllers to achieve arm poses precisely and timely [ACSF07, KMB96]. These controllers can only generate movement in which the 'muscles' are critically damped and impose limited or no movement of the trunk.

Some techniques have been devised to map higher level parameters to low level controller parameters. Chao et al. [CYL06] provide a mapping from LMA-Effort parameters to parameters for a tracking controller, such as damping, stiffness and desired joint rotation. Yin et al. [YCBvdP08] apply an optimized learning strategy to adapt the parameters of a walking controller to new situations (for example: low friction as in walking on ice, step over an obstacle, push furniture). A continuation variable γ represents the parameterization of the change. The parameter space is searched for valid combinations (as in, those that do not make the VH fall) of γ and the controller parameters $\mathbf{w}$. There might be many viable solutions of $\mathbf{w}$ that achieve γ. An objective function evaluates $\mathbf{w}$ to help select a unique optimal solution. This function is hand-authored. It can be designed to prefer solutions that have a minimal deviation from the original parameters, a certain walking speed or step size, etc. The learning process is offline, but the learned parameterizations can be interpolated to achieve real-time control. It is yet to be seen if and how this method generalizes to more than one continuation variable.

5.2. Concatenating Motion Spaces and Primitives

To achieve a natural concatenation of two natural motion primitives, one needs to retain naturalness in the transition point. A possible way used to achieve a natural concatenation is to let all motion primitives start and stop in an idle pose. This is not very flexible, since transitions can only be generated after the motion primitive finishes. Such a method sacrifices naturalness on the motion plan to gain some naturalness on the motions physics. Another possible solution in a limited set of motion primitives, is to create transition motion primitives for every motion primitive to the motion primitives that could be concatenated to it, as done in the

computer game Prince of Persia, The Sands of Time [Ubi04]. However, this is quite a time consuming task, and again, this only provides concatenation opportunities at the end of each motion primitive. In computer games that require fast interaction, motion primitives (mocap clips) are often directly concatenated, preserving only the general shape (standing, lying, etc) [Lon07].

5.2.1. Concatenation using Motion Editing

Interpolation techniques, using ease-in ease-out, introduced in [Per95] use interpolation between two motion primitives to concatenate them. The first motion primitive is faded out as the second one is faded in. Transitions between different sets of motion primitives differ in naturalness. Ikemoto et. al. [IAF07] generate transitions by cached multi-way blends. They cluster recorded motion primitives using the distance metric by Kovar et al. [KGP02]. All mediods (central item of cluster) are representatives for the clips belonging to that cluster. During preprocessing, all possible 2, 3 or 4 multiway blends between representatives are evaluated by footskating and ZMP evaluation and the best blend recipe (containing a weight function and representatives) is stored. A transition is generate at runtime by matching the current and next motion primitives to mediods and applying the stored blend recipe. Treuille et al. [TLP07] define a cost metric for each transition that measures how much closer it brings a motion to a desired goal and how much naturalness is lost in the transition. Using offline reinforcement learning, they approximate a value function that measures the total cost of the optimum motion primitive transition sequence that reaches a desired goal. This value function can then be used to select the (near) optimal motion primitive sequence in real-time, given a start motion primitive.

5.2.1.1. Motion graphs In many applications, one requires a continuous stream of motion. A very common technique is to put all the possible transitions between animations in a graph like structure: a motion graph. A motion graph is a directed graph where all edges correspond to motion primitives. A trivial motion graph can be constructed where the original motion primitives are single edges, blends (or: edges) can then be added between poses that are similar enough. Note that a single edge can correspond to very small motion primitives.

In the game industry, these graphs, *move trees*, were originally created manually [MBC01]. Kovar et al. [KGP02] present an algorithm that automatically creates motion graphs. Good transition points are automatically detected using a geometrical distance metric. In order to avoid dead ends, they prune the graph by using only the largest strongly connected component. After the graph is created, control can then be gained by doing a graph search that searches for an animation that adheres to certain constraints. For example, one can concatenate motion primitives such that the resulting motion primitive follows a specific path.

Many variations of motion graphs exist which can be distinguished in off-line methods where the desired animation is known in advance [AF02, AFO03, KGP02, PB02, CLS03, TH00, SH07] and methods that work at interactive speed [GSKJ03, PSS02, PSKS04, KS05, LL04, LCR*02].

In order to speed up the search the graph is often restructured using, for instance, clustering [AF02, LCR*02] of the edges. Gleicher et al. [GSKJ03] present snap-together graphs where common poses are used as hubs in the graph. Lee and Lee [LL04] precompute the desired behavior and animation of a VH using reinforcement learning and dynamic programming. Choi et al. [CLS03] use a combination of a motion graph and probabilistic path planning techniques [KcLO97] to capture the connectivity of the free space which is then used for footplan-driven synthesis. Arikan and Forsyth [AF02] searches for a global solution by making local changes using a *local search* technique.

Methods that work at interactive speed only evaluate local properties, for they do not know the desired animation in advance, nor do they have time to evaluate global properties [FAI*06]. Local search, as used in [KGP02] evaluates only properties of a certain number of nodes ahead when choosing what node to transition to. This might lead to a horizon problem [FAI*06]: a choice made now might lead to trouble that is invisible because it is on the other side of the horizon, separating the future cases we consider from those we do not. Global search [AFO03] cannot be done in real-time, but is suitable for motion authoring purposes. Typically, multiple paths satisfy the desired motion constraints, this phenomena is called 'motion ambiguity' in [FAI*06]. The occurrence of motion ambiguity on a motion graph lessens the amount of occurring horizon problems.

Control and motion planning is limited by the available paths on the graph. Using motion graphs it is in general very hard to generate motion that needs tight coupling to the environment, like pointing to an object in the world, or walking up a stairs, unless exactly those motion are in the database. As more motion constraints are added, less paths will become available. Motion graphs are successfully used in applications in restricted domains that require few constraints. Examples are dancing [SNI06, hKPS03], gesturing feedback on a predefined snowboard tutorial in a game [SDO*04] or moving through small game like environments [LCR*02].

5.2.1.2. Concatenation of Motion Spaces Recently, several techniques have been developed that are able to concatenate motion spaces to generate a continuous stream of motion.

Shin and Oh [SO06] present *fat graphs*. These graphs are based on the snap-together graphs of Gleicher et al. [GSKJ03], see 5.2.1.1. The common poses (hubs) are the nodes of a fat graph. The edges that start and end at this common pose are grouped together in a motion space. The fat graphs suffer from the same disadvantage as the snap-

together graph. In order to transition from one motion to another, the VH first needs to transition to a common pose. Heck and Gleicher [HG07] introduce *parameteric motion graphs*. They use the method of Kovar and Gleicher [KG04] to automatically create motion spaces. The motion graph is constructed by sampling the motion spaces using different parameter values and creating edges for the parameter values that result in appropriate transitions between the motion spaces.

5.2.1.3. Concatenation using Statistical Methods Li et al [LWS02] models a motion space as a LDS. They define a distance metric for LDSs and construct a motion graph-like structure to support concatenation of similar LDSs. By setting the first two poses of the next LDS in the path to the last two poses of the current LDS (see 5.1.4), a fluent connection is achieved.

5.2.2. Concatenation of Physically Controlled Motion

In physical simulation using controllers, a different physical controller constructs each motion space. This implies that a transitions between motion spaces is a transition between controllers. If the exit state of one controller leaves the simulation in a valid entry state for the next controller, valid transitions can easily be attained [WH97b]. A transitional controller can be designed to facilitate transitions between controllers with incompatible exit and entry states. Wooten and Hodgins [WH97b] demonstrates this, by using a landing controller to take a VH from a flight to a state suitable for balancing on the ground. Faloutsos et al. [FvdPT01a] facilitates transitions between controllers by describing pre conditions and post conditions for each controller. The pre conditions in define the entry state region that leads to a successful execution of the controller. Specifying valid pre conditions for controllers is not always a trivial task (for example: what are valid pre conditions for balancing?). Support vector machine (SVM) classifiers are trained to predict the success or failure of a controller for an arbitrary starting state. The preconditions for a controller are then determined by what a trained SVM for that controller classifies as successful.

Different phases of a single motion space can be modeled by different controllers [HWBO95, WH95, LvdPF00, FvdPT01b, YLS04]. These phases are then typically concatenated using a state machine. For example, [HWBO95] shows a state machine that uses different phases (and thus, controllers) for the flight, loading, heel contact, heel and toe contact, toe contact and unloading phases of a running motion space.

5.2.3. Concatenation of Procedural Motion

Zeltzer [Zel82] models the different phases of a procedural walking motion by different procedures and concatenates them using a state machine. Some frameworks for the generation of procedural arm gestures concatenate the gestures using procedural techniques that allow a flexible start pose of the arm [KW02, HMP02]. The end pose of the previous gesture is then used as the start pose of the current gesture. Other procedural animation systems use blending to generate a transition motion primitive between two procedural motion spaces [PG96, KM05a].

5.2.4. Concatenating Physical Simulation and Motion Editing

Motion editing techniques provide natural motion, but it is hard to set them up to interact with the physical world. Physical simulation provides world interaction, but less naturalness. Several methods have been developed to take advantage of the strength of both techniques by switching between them depending on the type of interaction needed.

Shapiro et al. [SPF03] switches control from kinematics to physics on contact with active objects in the environment. Active objects are defined as those objects that apply physical forces upon the VH to be animated that are not part of the original motion. A transition from physical simulation to motion editing (in this system a motion graph) can be made if the VHs pose is similar to a pose in a motion primitive of one the motion editing motion spaces. It is not stated state how a suitable motion space and motion primitive therein is found. Presumably the number of motion primitives in the graph is low, so that an exhaustive search can be performed on all their poses.

Mandel [Man04] makes the transition from motion editing to PD-control, whenever some physical event is triggered, pushing the VH over. A PD-control system is then started in the pose last set by the motion capture animation. A fall controller lets the VH fall, while trying to break this fall with the hands. As soon as the hands hit the floor, the system attempts to return control to motion editing. To find a suitable animation, the motion capture database is searched for a motion primitive that has a similar pose to the pose the VH is in. This is done using the Approximate Nearest Neighbor Search algorithm. An intermediate physical controller then moves the VH to this pose. Once the VH is close enough to that pose, animation is played using motion capture again.

Zordan et al. [ZMCF05] search a motion capture database to find a suitable motion primitive to play after a physical impact. During the physical impact, a physical ragdoll motion is played for a short period of time (0.1-0.2s), then motion is steered by a physical tracking controller (see 6.1.3), that tracks a blend of the motion primitive before the impact and the selected motion primitive after the impact. In later work, Zordan et al. [ZMM*07] contribute an automatic, real-time, motion primitive search algorithm. Re-entry motion primitive candidates are classified offline, using machine learning. This significantly reduces the number of candidates to select from.

Naturalmotion's Euporia [Natb] and Endorphin [Nata] animation systems allow transitions between motion editing and physically simulation. Selecting a suitable motion primitive to play after physical simulation is left to the motion author.

5.3. Combining Motion Techniques on Different Body Parts

A VH can execute multiple tasks at the same time that each require motion, possibly on different parts of the body. Rather than using one animation technique for every possible combination of tasks, it is more efficient to combine techniques to form one larger motion space.

5.3.1. Combination using Motion Editing

A simple way to combine motion primitives is by using a direct 'ease-in ease-out' interpolation of the motion primitives [Per95, BBET97, KM05b]. The interpolation weights of the motion primitives to be combined is set per joint, so that certain motion primitives can be set up to affect certain joints more than others. This method can produce unrealistic results because it ignores both physical and stylistic correlations between various joints in the body [HKG06].

Heck et al. [HKG06], combine (splice) the upper body motion of one motion space with lower body motion of another. Both motion spaces contain a walk cycle. Temporal relations between the upper and the lower body are maintained by making use of the rhythmic nature of walking to time warp and align the motion spaces. The pelvis is rotated in such a way that the upper and lower body are aligned, while retaining the desired upper body posture.

Ha and Han [HH08] decouple the body and splice the two motion spaces (one for the upper and one for the lower part of the body). They present a double timewarping scheme where each recorded motion primitive is first synchronized with a reference motion primitive (for both the upper and lower body motion primitives). The upper and lower body motion space is then synchronized using these reference motions. An additional problem is that the configuration of the lower body changes the parameter space of the upper body. Therefore they preprocess the parameter space of the upper body and transform it to the current lower body configuration.

Bruderlin [BW95] introduces *multiresolution filtering*. The motion signal passes through a cascade of lowpass filters to produce a set of bandpass or lowpass signal components. Motions primitives can be stored as a pyramid of lowpass or bandpass filter bands where each level represents a different band of frequencies. Subtleties of a movement (and noise) are typically considered to be encoded by high frequencies, while general, gross motion aspects are encoded by low frequencies. Using this technique, it is possible to blend bands of various motion primitives. Using multiresoltion filtering, one can blend specific frequency bands of two or more motion primitives. High frequency movement can hence be transferred to another motion primitive.

5.3.2. Combination of Physical Controllers

Physical controllers can be combined by adding up the forces and torques applied by them on each joint [WH00]. Sentis and Khatib [SK05] introduce a prioritized-based control architecture. End effector controllers (see 4.2.1.1.3) with lower priority are controlled without violating the goals or constraints of higher priority end effector controllers. This is achieved by projecting the torques generated by the lower priority controllers onto the motion null-space (see 3.1.5.1) of the higher lever controllers.

5.3.3. Combination of Procedural Motion

Many procedural animation systems combine procedural motion on different body parts, by employing a procedural motion technique for each body part [KW04, HMP02]. Procedural motion on the same body part can be combined using ease-in ease-out blending (see 5.3.1) [PG96]. Thiebaux et al. [TMMK08] employ specialized blend controllers to combine motion primitives generated by different procedural animation techniques.

5.3.4. Combination of Kinematic Motion and Physical Simulation

Oore et al. [OTH02] present a mixed kinematics/physics animation puppeteering system. The physical model helps to reduce the amount of DoFs for the puppeteer, while full freedom of control is maintained. It acts on the knee and ankle joints. The physical model is coupled with the upper body through its mass displacement. The joint torques of the kinematically moved parts in the upper body are not taken into account in the physical movement of the lower body.

Isaacs and Cohen [IC87] show how ID and FD can be combined in a single physical simulation system, given that either the joint accelerations or the joint torques are known for each joint, at each frame. This way, if kinematic motion is known for every frame for some joints, the forces those joints exert on the other joints is taken into account when the remaining joints are moved using physical simulation. Combined solving of accelerations and torques can only be done if the physical system is formulated using Lagrange equations. More efficient recursive formulations can not deal with such a partial force and partial motion specification [Ott03]. Van Welbergen et al. [vWRV08, vWZR09] extend on this work by providing a simplification of the simulation model. Their system allows the use of efficient iterative techniques to calculate the torques exerted by the kinematically steered joints and provides easy integration with existing physics engines.

5.3.5. Combining Procedural Motion and Motion Editing

Lance and Marsella [LM08] combine a biomechanical model of eye movement (which is hard to motion capture) with a motion editing technique for neck and trunk movement. Heck [Hec07] notes that augmenting a motion editing technique with gaze would require a prohibitive amount of motion primitive recordings. Instead, she employs a biologically and psychologically inspired model for gaze that is layered on top of motion space generated by motion editing.

5.4. Aspects of Control

In the previous subsections we have looked at ways to parameterize, concatenate and combine motion spaces using various techniques. Here we discus how much control was gained using such techniques, by looking at various aspects of control.

Responsiveness deals with how fast a motion editing technique responds to new parameter value setting or how fast a motion plan is adapted to a new goal. For example, how fast does an animated soccer player respond to a gamer pressing the shoot button? Responsiveness is a major theme in the design of motion graphs, it might take a while to traverse to the graph to reach the desired node, especially if the graph is sparse. Forsyth et al. [FAI*06] introduce the diameter: the average path length of the shortest path connection two nodes on a motion graph as a measure for responsiveness. A denser graph (with a smaller diameter) can be created by sacrificing some naturalness (see 5.2.1.1). Physical simulation has high responsiveness to physical events (for example, being hit by a falling anvil), but lower responsiveness to parameter changes that effect the desired state of the VH. Procedural animation and motion editing techniques have higher responsiveness to parameters that change the desired of state the VH, but do not provide high responsiveness to physical events.

Precision deals with how good the desired parameters (like pose constraints, and timing constraints) are satisfied. Procedural motion is very precise. Motion editing techniques can provide precision at the cost of calculation time. Physical simulation is imprecise, it is unknown if and when desired pose and time constraints are met. Some precision can be gained by sacrificing naturalness and creating only motions in which the 'muscles' are critically damped (see 5.1.5).

Coverage deals with how much of a parameter space is covered and what goals can be satisfied by a motion plan. Motion graphs can suffer from bad coverage, for example: some parts in an environment cannot be reached from a certain position because no path on the graph will go there. Reitsma and Pollard [RP07] present quality metrics to evaluate the (global) quality of motion graphs, including the coverage of the environment and the distance between the shortest walk through an environment and a walking path generated by the motion graph. Physical simulation can suffer from bad coverage in parameters values near the edge of balance, a physically simulated VH can easily fall over for such parameter values. Less natural but more stable balancing mechanisms can increase coverage. While most motion editing techniques can cover a wide range of parameter values, only parameter values that select a motion primitive near a recorded motion primitive will yield natural motion. Procedural motion has good coverage, but not all parameter instances will provide natural motion.

Expressiveness is a measure for the size of a motion space. The expressiveness of an animation technique is determined by the number of parameters that can be used to select motion primitives using the technique and the coverage of the parameters. Motion techniques that can deal with a large set of parameters, such a procedural motion and to a lesser extend physical simulation have high expressiveness. Motion editing has a lower expressiveness.

Intuitiveness deals with how intuitive the parameter set of a motion technique is. For example, an amplitude parameter is a more intuitive parameter than a damping gain parameter. All techniques provide parameters that can set pose constraints. Higher level parameters (such as emotion, physical state) are typically mapped to low-level parameters that are then provided to an animation technique. An intuitive set of higher-level parameters might cause conflicts between parameters, but an orthogonal set of parameters is typically not intuitive (see 5.1.4).

By combining and concatenating motion spaces generated by different animation techniques, control on several aspects can be enhanced. For example, a concatenation of a motion space generated by motion editing by one generated using physical simulation enhances the responsiveness to physical events. Another example is the enhancement of expressiveness by combining procedural motion and motion editing.

6. Naturalness

For many animation systems, plausibility or naturalness rather than full realism is acceptable [ODGK03]. We define naturalness as observed realism of VH movement. Naturalness therefore partly depends on properties of human observation. For example, naturalness depends on the realism of the embodiment of the VH: as anthropomorphism of the embodiment increases, the same animation is more likely to be judged artificial, rather than biological [CHK07]. Human sensitivity to errors in animation can be different in horizontal direction than in the vertical direction and can be different for acceleration and deceleration [RP03].

In 3.2, we identify physical realism as one property of natural animation. Involvement of the whole body is also crucial to make an animation natural [BCZC00]. Other properties

are the plausibility of the motion is with respect to the cognitive and emotional state of the moving VH [BCZC00]. Furthermore, movement should be consistent with static (such as age, gender) and dynamic (like tiredness, anger) properties of the VH that is being animated [GRA*02]. How these properties affect movement is captured in a VH's style. Variability is a concern if a motion is to be repeated. In this section we will elaborate on different aspects of naturalness and show how naturalness can be enhanced and evaluated.

6.1. Physical and Biological Realism

Motion spaces constructed by physical animation techniques are physically realistic by design. It is relatively easy to consider joint strength and comfort in these methods. Keyframed animation created by a skilled animator, or motion captured animation is also physically realistic, since it origins from real humans moving or expert knowledge of human movement. However, even if this original motion was physically correct, when we modify this motions to gain control in non-physical ways, physical correctness might be invalidated. Typical artifacts that invalidate physical realism include foot skate, unnatural balance, or momentum changes in flight. We outline some methods to enhance physical and biological realism.

6.1.1. Physical filters

The physical naturalness of motion primitives can be improved by post processing motion primitives with a physical filter.

Pollard and Reitsma [PR01] propose a filter for physically correct ground contacts. A friction model is used to make the foot slide when appropriate. Their filter makes use of the fact that no forces or torques can be applied at the root of a VH, since that joint is not actuated. Each frame of motion is casted on a physical model of the VH. Then, per frame, the root forces and torques are determined. Contact forces on the feet that are necessary to eliminate or minimize such root torques and forces are then calculated. If any force/torque remains on the root, it is eliminated by modifying the rotational acceleration on all actuated joints and the rotational and transitional acceleration on the root.

Shin et al. [SKG03] employ a constraint based motion editing method (see 4.1.2) to enhance the physical and biomechanical correctness of edited motion. During flight stages, the angular momentum is conserved and the center of mass is constrained to follow a parabolic path. The ZMP is constrained to fall into the support polygon. The corrections are applied to a user-selected set of joints during the flight stage, ZMP correction is applied on one user selected joint.

Footskate is a typical artifact caused by motion editing. The VH's foot slides on the floor after the VH plants it, rather than remaining tightly in place [IAF06]. If it is known when a foot is planted, then a constraint based motion editing method (see 4.1.2) can be used as a motion filter, to constrain the movement of the planted foot [KSG02]. Fully automatic reliable detection of footskate in real time is still an open problem. Existing methods have to be trained for each motion [IAF06] or refine roughly estimated contact times and durations [GBT06]. Alternatively, foot contact can be annotated in the recorded motion primitives, and motion editing techniques can be set up to retain these annotations [KG03].

6.1.2. Physical Correctness of Blending

Because the difficulties and large computation time associated with physical filters, some blending approaches deal with physical realism during the blend stage, rather than using a post-hoc method. A number of simple modifications can be used to improve the physical correctness of interpolation of motion primitives that are physically correct on their own [SH05]. By interpolating the center of mass, rather than the root and by setting the total interpolation time as $T = \sqrt{T_1^2 w + T_2^2 (1-w)}$, rather than $T = T_1 w + T_2 (1-w)$ (with T_1 the duration of animation 1, T_2 the duration of animation 2, w the blend weight), the net force during flight is equal to gravity. If, during ground contact, only the non redundant DoF (that is: the center of mass, the foot positions, two 'knee-circle' parameters and all joints angles except the legs) are interpolated, rather than directly interpolating joint rotations, the feet will not slide, balance will be retained and the ground friction will be within the same friction cones as the source motion primitives. Ménardais et al. [MKMA04] use a simple technique to avoid or reduce footskate. Motion primitives are annotated with support phase information (left foot, right foot, double support, no support). A time-warp then assures that the support phases of the motion primitives are compatible during the blend. Treuille et al. [TLP07] prevent footskate in support phases where only one foot is on the ground by first aligning the support feet and then interpolating the motion primitives with the support feet as the root.

6.1.3. Improving Physical Correctness using Tracking

A tracking controller tracks the rotation of joints specified in a motion primitive. This is done by specifying these rotations as the desired state for a physical controller for each joint. The resulting motion obeys Newtonian physics and thus (among others) can respond to collisions with the environment and its own body in a physically realistic way. Physical tracking recently became a component of some commercial high level animation toolkits [Art08, Hav08a].

Motion capture noise, retargetting errors, tracking errors and environmental changes can easily disturb the balance of a VH that is controlled by tracking. For early tracking methods [KMB96, ZH99] this was not an issue because they only track the upper body. Other tracking methods enforce balance by constraining the root to the translation specified in

the motion primitive [Nata, Natb, YCP03]. Zordan and Hodgins [ZH02] experiment with both a virtual actuator [Pra95] pulling up the root and a PD-controller for balancing like that in [Woo98]. These realistic balance controllers limit the application domain to non-locomotive motion. Wrotek et al. [WJM06] use a less realistic balancing method that does allow locomotion: a weak root spring connects the root of the VH to the universe. This spring can 'break' if too much force is exerted on it, causing the VH to lose balance.

A tracking PD-controller neceseraly has very high PD-gains, which results in stiff reaction to the environment. The PD-gains can be reduced on impact, to decrease such undesired stiffness [ZH02, WJM06]. Yin et al. [YCP03] track motion capture movement using a controller based on a simple human neuromotor model: ID acts as a feedforward mechanism and small perturbations are corrected using a low gain feedback PD-controller. The gains on the feedback controller increase with the amount of exerted force. This corresponds with perturbation movement in real humans, where muscle stiffness (corresponding with PD-gains) increases upon perturbation.

Oshita and Makinouchi [OM01] track an input motion using an acceleration controller, that exerts joint accelerations rather than joint forces, in a customly designed physical simulation environment. Their system is aimed at dynamically changing physically realistic input motion perturbated by impulses or forces, and provides some biomechanical enhancement of the motion by reducing stress on joints. The balance controller used in their system is designed for standing with double-support.

6.1.4. Physical Correctness through Procedural Techniques

Physical simulation can greatly enhance expressive procedural motion. It can help model important nuances of VH motion, such as realistic balance, force transference between limbs and momentum effects like overshoot [Nef05]. Physical controllers can explicitly address muscle strength and comfort limits. Some of these effects have, to some extent, been reproduced by procedural models.

Inverse kinetics [BK07] is a kinematic technique that can be used to position the CoM of a VH. This does help in creating balanced poses, but to generated realistically balanced *movement*, these methods need to be augmented with a model that provides a path of the CoM over time. Neff [Nef05] devises a feedback-based procedural balance system based on the physical controller of [WH00]. Unlike this physical balance controller, the procedural system works only on a single supporting foot and takes just the position of the CoM and velocity of the CoM, but not the forces generated by upper body movement into account.

ID can be used to analyze the muscle power used in procedural motion. The motion can then be adapted to adhere to muscle strength and comfort limits [LWZB90, KB96].

6.2. Whole Body Involvement

Procedural gesture animation techniques typically steer the head and the arms starting at the shoulder and leave the rest of the body relatively stiff. Naturalness can be enhanced by providing automatic, coherent movement of the rest of the body. Some of the techniques used to enhance physical realism also help engaging the whole body. For example, a physically based balance model can be used to automatically generate lower body movement (see 6.1.4 and [Nef05]).

Egges and Magnenat-Thalmann [EMT05] propose a statistical model to enhance the naturalness of procedurally generated gesture movement on the arms. PCA is performed on a mocap database of gesture animation. Using this PCA analysis, the procedural animation is filtered in PCA-space, in such a way that only movement similar to that in the database (and thus assumed natural) remains. Because the PCA components involve multiple joints, this automatically engages the full body.

Both Chi et al. [CCZB00] and Neff et al. [Nef08] aim to involve the torso automatically in gesture movement. The Effort and Shape parameters used to enhance the expressiveness of procedural gesture in [CCZB00] (see 5.0.1) are also used to enhance their procedurally generated torso movement. Neff [Nef08] show that 'drives', such as hand position and gaze direction can be used to automatically generate torso movement. This is done by defining a linear mapping between the drives and movement parameters of a procedural torso movement model.

6.3. Style

Style denotes the particular way in which a motion is performed. Stylistic differences of motion with the same function are caused by certain more or less static personal characteristics of the subject, like age, gender and personality [RP02, THN04]. It is important to endow VHs with style. Style contributes to naturalness, and, even more importantly, expresses information about the VH as cultural identity, as well as his relationship to other (virtual) humans, such as role and power relationship. Style is reflected by the motions a VH performs and the manner in which these motions are performed [NR05, NKAS08]. In this article we focus solely on the latter.

6.3.1. Style using Motion Editing

Urtasun et al. [UGB*04] employ blending from recorded motion primitives from different subjects (and thus with different styles) in PCA space for style transition. A motion capture database is constructed, containing recorded motion primitives of several subjects, with different parameter values for one parameter (walking with variable speed, jumping with variable height). A motion in the style of a new subject is created from one recorded motion primitive of this subject. First, the recorded motion primitive is modeled as a

blend of motion primitives from the different subjects in the database that have the same parameter value. The weights of this blend are then used to construct motion primitives with a new parameter values using a blend of motions in the database with these new parameter values. This system can create motion in the style of a user in an online application, by tracking the users movement using a cheap computer vision system.

Egges et al. [EMMT04] generate different styles of idle motion using recorded motion primitives of different individuals. On top of the posture shift motions, variation of movement is generated by applying a noise function on principal components derived from recorded motion primitives. This noise function is defined by a probabilistic model of recorded variations in motion. Individualized variations can be synthesized by determining the parameters of the probabilistic model for a given individual.

6.3.2. Style in Simulation

Procedural animation applies style by mapping high level style characteristics to lower level animation parameters, using parameterization. Perlin [Per95] models personality and emotion using noise functions on top of motion generated by an existing procedural model. Ruttkay and Pelachaud [RP02] model style as a mapping from static characteristics, such as age or sex and dynamic characteristics, such as emotion to gesture animation parameters. Neff and Fiume [NF05], models style using a *Character Sketch*. Such a sketch defines modifications to be made to motion parameters, can be designed to automatically insert new actions to an animation script and can provide a default stance.

6.4. Variability

Variability is a measure of the differences in a motion which is repeated many times by the same person [BSH99]. If, using a specific motion technique, the same parameters produce the same motion primitive, the motion will look unnatural if those parameters occur several times in the motion performance. Several methods can be used to avoid this invalidation of naturalness.

6.4.1. Procedural Generation of Variability

Perlin [Per95] simulates variability by adding noise to the rotation of some of the joints in the skeleton of a VH. This method is not scalable on all joints, because relations exist between rotation of one joint and rotation of another. If these relations are not captured, the resulting animation will look unrealistic [EMMT04].

Bodenheimer et al. [BSH99] apply variability by using a biomechanically inspired method. Since the amount of variability is usually correlated to larger movements of the body, the noise has its largest amplitude at the extrema of a DoF of a moving joint. The noise is scaled with the distance the joint travels, thus obeying Fitts' Law. Since the shape of the noise is based upon the movement of the joints, this approach somewhat implicitly models inter-joint variability relations. However, reciprocally covarying movement variability between joints (like, when elbow moves to compensate shoulder variability on an aiming task) is not captured by this approach.

6.4.2. Generating Variability using Statistical Models

Statistical methods that capture orthogonal components of motion also capture the relation between joint movements [BH00, EMMT04]. Since these components are independent, they can be modified separately. Small posture variations are generated by adjusting the components using perlin noise [Per95]. In Li et al.'s [LWS02] LDS model, variability is generated by sampling noise.

6.4.3. Generating variability in Physical Simulation

Motion generated by physical simulation often looks 'sterile', because variation caused by small details is not taken into account [BHW96]. Such details, for example small bumps on a floor, or the non-rigidness of human body parts are not simulated because it would not be possible to do so in real time or because simulation methods for this are yet unknown. Barzel et al. [BHW96] propose some techniques to model some of these details in a physically plausible (but not physically realistic) ways. For example: the inherent variability and instability of a physical simulation system can be exploited to generate motion variability by slight variations in its starting state, or a physical form of bumpmapping can be used to create slight variations in the normal of a physically modeled flat floor.

Another cause of variability in human movement is noise in the control signals that steer our limbs [HW98]. The variability of the noise increases with the torque to be exerted. Bodenheimer et al. [BSH99] model this type of variability by adding noise to joint torques in a physical simulation in a similar way as described above for kinematic motion.

Naturalmotion's Euphoria [Natb] and Endorphin [Nata] toolkits provide an undisclosed method to add noise to physical controllers. The amount of noise can be set as a parameter value for the controllers.

6.5. Evaluation of Naturalness

VHs usually do not have a photo-realistic embodiment. Therefore, if the naturalness of animation of VHs is evaluated by directly comparing moving humans with a moving VH, the embodiment could bias the judgment. A motion captured human movement can be projected onto the same embodiment as the VH. This projection is then compared with generated animation. Typically this is done in an informal way. A motion Turing Test is used to do this more

formally [HWBO95, EVMT04, CHK07, vBE09]. Some metrics have been defined that aim to measure certain aspects of naturalness objectively.

6.5.1. User Testing

In a motion Turing test, subjects are shown generated movement and similar motion captured movement, displayed on the same VH. Then they are asked to judge whether this was a 'machine' moving or a real human. The size of the natural parameter space, that is, the space of parameters that select natural motion primitives from a motion space can be explored by having subjects directly set and evaluate the parameter values, as done in [BSH99, vBE09].

However, such a human judgment is not sufficient to measure the naturalness of motion. Even if a certain movement is judged as natural, an invalidation of naturalness that is not noticed consciously can still have a social impact [RN96]. Unnatural moving VHs can be evaluated as less interesting, less pleasant, less influential, more agitated and less successful in their delivery. So, while a VH Turing test is a good first measure of naturalness (at least it looked human-like), further evaluation should determine if certain intended aspects of the motion are delivered. Such aspects could include showing emotion, enhancement of the clearness of a spoken message using gesture, showing personality, etc.

6.5.2. Comparing with Motion Invariants

Some qualitative comparisons have been made by comparing motion invariants of record motion with those of generated motion. Graphs of end effector speed, end effector square jerk, end effector position and motion curvature can be used to compare human motion to generated motion, to evaluate how well the systems model invariants such as the bell shaped velocity profile, minimum jerk, Fitts' law and the two-third power law [GLM01, MZW99, KW04].

6.5.3. Metrics for Naturalness

Intuitively, physical correctness can be measured directly from the animation. Reitsma and Pollard [RP03] evaluate physical correctness is by checking and evaluating perceptual metrics for allowable errors in horizontal and vertical velocities and the effective gravity constant for ballistic movement.

Metrics such as the average amount of foot gliding [Ahm04] and the number of frames in which the ZMP is outside the support polygon [JLLL08] address the anomalies in motion editing and can be used to compare the naturalness of different motion editing techniques.

Some attempts have been made to evaluate naturalness automatically. Ren et al. [RPE*05] argue that evaluation of the naturalness of human motion is not intrinsically subjective, but instead, an objective measure is imposed by the data as whole. In other words, movements that we have seen often are judged as natural, and movements that occur rarely are not. They make use of machine learning techniques, trained with statistical properties of human motion to classify new animations as natural or unnatural, and to point out the parts that invalidate natural movement. The system is still outperformed by human observers in recognizing natural or unnatural movement.

6.5.4. On Motion Demand

An unnatural motion that occurs often in an application degrades the naturalness of the overall motion more than an equally unnatural motion that occurs rarely. Therefore, the naturalness of motion should be tested considering a valid motion demand for the application using the motion. In [IF04] Unreal Tournament game sessions are used to model such a motion demand.

7. Conclusion

In this STAR we have discussed a variety of techniques that all can contribute to an 'ultimate' fully-controllable animation system producing natural motions in real-time. Current techniques offer trade-offs between control, naturalness and calculation time. The selected trade-off depends on the application domain. Motion editing techniques employ the detail of captured motion or the talent of skilled animators, but they allow little deviation from the captured examples and can lack physical realism. Procedural motion offers detailed and precise control using a large number of parameters, but lacks naturalness. Physical simulation provides integration with the physical environment and physical realism. However, physical realism alone is not enough for naturalness and physical simulation offers poor precision in both timing and limb placement.

Motion editing is typically used for current computer games (requiring huge motion capture databases), films and cartoons. Physical simulation is typically used for the animation of physically constrained motion, such as motion by athletes and stuntmen or 'ragdoll' animation. Expressive animation, such a gesturing while speaking requires tight synchronization to speech, precise limb placement and a large number of control parameters. Therefore, expressive animation is the domain of procedural animation techniques.

A big challenge in the animation domain is finding an integrated way of generating natural motions that interact with the environment and provide detailed control. We show that hybrid systems that combine and concatenate motion generated by different paradigms can enhance both naturalness and control. These systems could provide a starting point for such an integration.

Another issue deals with simulating physiological processes, such as emotions, sleepiness, hungriness and so on. We hope that this STAR will provide a useful starting point for researchers who are interested in animation of VHs on

the basis of such concepts, thus bringing us one step closer to a 'real' VH.

Acknowledgements

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

References

[ABC96] AMAYA K., BRUDERLIN A., CALVERT T.: Emotion from motion. In *Proceedings of the conference on Graphics interface* (Toronto, Ont., Canada, Canada, 1996), Canadian Information Processing Society, pp. 222–229.

[ACSF07] ALLEN B., CHU D., SHAPIRO A., FALOUTSOS P.: On the beat!: timing and tension for dynamic characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 239–247.

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. In *SIGGRAPH* (New York, NY, USA, 2002), ACM Press, pp. 483–490.

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3 (2003), 402–408.

[AFP*95] AUSLANDER J., FUKUNAGA A., PARTOVI H., CHRISTENSEN J., HSU L., REISS P., SHUMAN A., MARKS J., NGO J. T.: Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Trans. Graph. 14*, 4 (1995), 311–336.

[AG85] ARMSTRONG W. W., GREEN M. W.: The dynamics of articulated rigid bodies for purposes of animation. In *Graphics Interface 1985* (May 1985), pp. 407–415.

[Ahm04] AHMED A. A. H.: *Parametric Synthesis of Human Animation*. PhD thesis, University of Surrey, Surrey, UK, April 2004.

[Art08] ARTIFICIAL LIFE SOLUTIONS: Massive prime, 2008. http://www.massivesoftware.com/prime/.

[Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH* (New York, NY, USA, 1994), ACM, pp. 23–34.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH* (New York, NY, USA, 1996), ACM, pp. 137–146.

[BAZB02] BADLER N. I., ALLBECK J., ZHAO L., BYUN M.: Representing and parameterizing agent behaviors. In *Computer Animation* (2002), pp. 133–143.

[BB98] BINDIGANAVALE R., BADLER N. I.: Motion abstraction and mapping with spatial constraints. In *CAPTECH '98: Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments* (London, UK, 1998), Springer-Verlag, pp. 70–82.

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* (New York, NY, USA, 2007), ACM, pp. 281–288.

[BBET97] BOULIC R., BÉCHEIRAZ P., EMERING L., THALMANN D.: Integration of motion control techniques for virtual human and avatar real-time animation. In *VRST '97: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 1997), ACM Press, pp. 111–118.

[BBZ91] BADLER N. I., BARSKY B. A., ZELTZER D. (Eds.): *Making them move: mechanics, control, and animation of articulated figures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[BCZC00] BADLER N. I., COSTA M., ZHAO L., CHI D. M.: To gesture or not to gesture: What is the question? In *Computer Graphics International* (2000), pp. 3–10.

[BDV] BILOTI R., D'AFONSECA L., VENTURA S.: Open optimization library. http://ool.sourceforge.net/.

[BH00] BRAND M., HERTZMANN A.: Style machines. In *SIGGRAPH* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192.

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (New York, NY, USA, 1996), Springer-Verlag New York, Inc., pp. 183–197.

[BK07] BOULIC R., KULPA R.: Inverse kinematics and kinetics for virtual humanoids. *Eurographics tutorials 2* (2007), 643–742.

[BLCHB03] BOULIC R., LE CALLENNEC B., HERREN M., BAY H.: Experimenting Prioritized IK for Motion editing. In *Annual Conference of the European association for Computer Graphics* (2003).

[Boe82] BOEHM W.: On cubics: A survey. *Computer Graphics and Image Processing 19*, 3 (1982), 201 – 226.

[BPW93] BADLER N. I., PHILLIPS C. B., WEBBER B. L.: *Simulating humans: computer graphics animation and control*. Oxford University Press, Inc., New York, NY, USA, 1993.

[BSH99] BODENHEIMER B., SHLEYFMAN A. V., HODGINS J. K.: The effects of noise on the perception of

animated human running. In *Computer Animation and Simulation* (September 1999).

[BW76] BURTNYK N., WEIN M.: Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Commun. ACM 19*, 10 (1976), 564–569.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *SIGGRAPH* (New York, NY, USA, 1995), ACM Press, pp. 97–104.

[BW97] BARAFF D., WITKIN A.: Physically based modeling: Principles and practice. SIGGRAPH Course, 1997.

[CBT07] CARVALHO S. R., BOULIC R., THALMANN D.: Interactive low-dimensional human motion synthesis by combining motion models and pik. *Comput. Animat. Virtual Worlds 18*, 4-5 (2007), 493–503.

[CCZB00] CHI D. M., COSTA M., ZHAO L., BADLER N. I.: The EMOTE model for effort and shape. In *SIGGRAPH* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 173–182.

[CHK07] CHAMINADE T., HODGINS J. K., KAWATO M.: Anthropomorphism influences perception of computer-animated characters' actions. *Social Cognitive and Affective Neuroscience 2*, 3 (May 2007), 206–216.

[CLS03] CHOI M. G., LEE J., SHIN S. Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph. 22*, 2 (2003), 182–203.

[Coh92] COHEN M. F.: Interactive spacetime control for animation. In *SIGGRAPH* (New York, NY, USA, 1992), ACM Press, pp. 293–302.

[Cou08] COUMANS E.: Bullet open source physics library, 2008. http://www.bulletphysics.com/.

[CPK99] CHOI K.-J., PARK S.-H., KO H.-S.: Processing motion capture data to achieve positional accuracy. *Graphical Models and Image Processing 61*, 5 (1999), 260–273.

[Cra89] CRAIG J. J.: *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[CYL06] CHAO S.-P., YANG S.-N., LIN T.-G.: An LMA-Effort simulator with dynamics parameters for motion capture animation. *Computer Animation and Virtual Worlds 17*, 3-4 (2006), 167–177.

[DW97] DENSLEY D. J., WILLIS P. J.: Emotional posturing: a method towards achieving emotional figure animation. In *Computer Animation* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 8–14.

[EMMT04] EGGES A., MOLET T., MAGNENAT-THALMANN N.: Personalised real-time idle motion synthesis. In *Pacific Graphics* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 121–130.

[EMT05] EGGES A., MAGNENAT-THALMANN N.: Emotional communicative body animation for multiple characters. In *Workshop on Crowd Simulation* (November 2005), pp. 31–40.

[EVMT04] EGGES A., VISSER R. M., MAGNENAT-THALMANN N.: Example-based idle motion synthesis in a real-time application. In *CAPTECH Workshop* (Dec. 2004), pp. 13–19.

[FAI*06] FORSYTH D. A., ARIKAN O., IKEMOTO L., O'BRIEN J. F., RAMANAN D.: Computational studies of human motion: part 1, tracking and motion synthesis. *Foundation and Trends in Computer Graphics Vision 1*, 2 (2006), 77–254.

[Fea07] FEATHERSTONE R.: *Rigid Body Dynamics Algorithms*. Springer, October 2007.

[FH85] FLASH T., HOGAN N.: The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience 5*, 7 (July 1985), 1688–1703.

[FP03] FANG A., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (July 2003), 417–426.

[FvdPT01a] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *SIGGRAPH* (New York, NY, USA, 2001), ACM Press, pp. 251–260.

[FvdPT01b] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics 25*, 6 (2001), 933–953.

[GBT06] GLARDON P., BOULIC R., THALMANN D.: Robust on-line adaptive footplant detection and enforcement for locomotion. *The Visual Computer 22*, 3 (2006), 194–209.

[GKP04] GIBET S., KAMP J.-F., POIRIER F.: Gesture analysis: Invariant laws in movement. In *Gesture-Based Communication in Human-Computer Interaction, 5th International Gesture Workshop, GW 2003, Selected Revised Papers* (Apr. 2004), vol. 2915 of *Lecture Notes in Computer Science*, pp. 1–9.

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics* (New York, NY, USA, 1997), ACM Press, pp. 139–148.

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *SIGGRAPH* (New York, NY, USA, 1998), ACM Press, pp. 33–42.

[Gle01] GLEICHER M.: Comparing constraint-based motion editing methods. *Graph. Models 63*, 2 (2001), 107–134.

[Gle08] GLEICHER M.: More motion capture in games -

can we make example-based approaches scale? In *Motion in Games* (2008), pp. 82–93.

[GLM01] GIBET S., LEBOURQUE T., MARTEAU P.-F.: High-level specification and animation of communicative gestures. *J. Vis. Lang. Comput. 12*, 6 (2001), 657–687.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. In *SIGGRAPH* (New York, NY, USA, 2004), ACM Press, pp. 522–531.

[GMW82] GILL P. E., MURRAY W., WRIGHT M. H.: *Practical Optimization*. Academic Press, January 1982.

[Gra98] GRASSIA F. S.: Practical parameterization of rotations using the exponential map. *J. Graph. Tools 3*, 3 (1998), 29–48.

[Gra00] GRASSIA F. S.: *Believable automatically synthesized motion by knowledge-enhanced motion transformation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2000. Chair-Andrew Witkin.

[GRA*02] GRATCH J., RICKEL J., ANDRÉ E., CASSELL J., PETAJAN E., BADLER N. I.: Creating Interactive Virtual Humans: Some Assembly Required. *IEEE Intelligent Systems 17*, 4 (2002), 54–63.

[GSKJ03] GLEICHER M., SHIN H. J., KOVAR L., JEPSEN A.: Snap-together motion: assembling run-time animations. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM, pp. 181–188.

[GTH98] GRZESZCZUK R., TERZOPOULOS D., HINTON G.: Neuroanimator: fast neural network emulation and control of physics-based models. In *SIGGRAPH* (New York, NY, USA, 1998), ACM Press, pp. 9–20.

[Hav08a] HAVOK: Havok behavior, 2008. http://www.havok.com/.

[Hav08b] HAVOK: Havok physics, 2008. http://www.havok.com/.

[Hec07] HECK R. M.: *Automated authoring of quality human motion for interactive environments*. PhD thesis, University of Wisconsin at Madison, Madison, WI, USA, 2007.

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM Press, pp. 129–136.

[HH08] HA D., HAN J.: Motion synthesis with decoupled parameterization. *Vis. Comput. 24*, 7 (2008), 587–594.

[HHL*05] HOWE N. R., HARTMANN B., LEVENTON M. E., MANCINI M., FREEMAN W. T., PELACHAUD C.: Implementing expressive gesture synthesis for embodied conversational agents. In *Gesture Workshop* (2005), Gibet S., Courty N., Kamp J.-F., (Eds.), vol. 3881 of *Lecture Notes in Computer Science*, Springer, pp. 188–199.

[HKG06] HECK R., KOVAR L., GLEICHER M.: Splicing upper-body actions with locomotion. *Computer Graphics Forum 25*, 3 (2006), 459–466.

[hKPS03] HOON KIM T., PARK S. I., SHIN S. Y.: Rhythmic-motion synthesis based on motion-beat analysis. In *SIGGRAPH* (New York, NY, USA, 2003), ACM Press, pp. 392–401.

[HMP02] HARTMANN B., MANCINI M., PELACHAUD C.: Formational parameters and adaptive prototype instantiation for mpeg-4 compliant gesture synthesis. In *CA '02: Proceedings of the Computer Animation* (Washington, DC, USA, 2002), IEEE Computer Society, p. 111.

[HPP05] HSU E., PULLI K., POPOVIĆ J.: Style translation for human motion. *ACM Trans. Graph. 24*, 3 (2005), 1082–1089.

[Hum05] HUMANOID ANIMATION WORKING GROUP: H|Anim, 2005. http://www.h-anim.org/.

[HW98] HARRIS C. M., WOLPERT D. M.: Signal-dependent noise determines motor planning. *Nature 394* (August 1998), 780–784.

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *SIGGRAPH* (New York, NY, USA, 1995), ACM Press, pp. 71–78.

[IAF06] IKEMOTO L., ARIKAN O., FORSYTH D. A.: Knowing when to put your foot down. In *Proceedings of the symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM Press, pp. 49–53.

[IAF07] IKEMOTO L., ARIKAN O., FORSYTH D.: Quick transitions with cached multi-way blends. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 145–151.

[IC87] ISAACS P. M., COHEN M. F.: Controlling dynamic simulation with kinematic constraints. In *SIGGRAPH* (New York, NY, USA, 1987), ACM Press, pp. 215–224.

[IF04] IKEMOTO L., FORSYTH D. A.: Enriching a motion collection by transplanting limbs. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 99–108.

[IST02] IK SOO L., THALMANN D.: Construction of animation models out of captured data. In *International Conference on Multimedia and Expo* (2002), pp. 829 – 832.

[JLLL08] JANG W.-S., LEE W.-K., LEE I.-K., LEE J.: Enriching a motion database by analogous combination of partial human motions. *Vis. Comput. 24*, 4 (2008), 271–280.

[Kaw99] KAWATO M.: Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology 9* (1999), 718–727.

[KB84] KOCHANEK D. H. U., BARTELS R. H.: Interpolating splines with local tension, continuity, and bias control. In *SIGGRAPH* (New York, NY, USA, 1984), ACM Press, pp. 33–41.

[KB96] KO H.-S., BADLER N. I.: Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications 16*, 2 (1996), 50–59.

[KcLO97] KAVRAKI L. E., CLAUDE LATOMBE J., OVERMARS M. H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *in IEEE International Conference on Methodsand Models in Automation and Robotics, 2005* (1997), MorganKaufmann, pp. 566–580.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 214–224.

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3 (2004), 559–568.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *SIGGRAPH* (New York, NY, USA, 2002), ACM Press, pp. 473–482.

[KM05a] KALLMANN M., MARSELLA S.: Hierarchical motion controllers for real-time autonomous virtual humans. In *Interactive Virtual Agents* (London, UK, 2005), Lecture Notes in Computer Science, Springer-Verlag, pp. 253–265.

[KM05b] KALLMANN M., MARSELLA S.: Hierarchical motion controllers for real-time autonomous virtual humans. In *Proceedings of the 5th International working conference on Intelligent Virtual Agents (IVA'05)* (Kos, Greece, September 12-14 2005), pp. 243–265.

[KM05c] KULPA R., MULTON F.: Fast inverse kinematics and kinetics solver for human-like figures. In *Humanoid Robots* (December 2005), IEEE, pp. 38–43.

[KMB96] KOKKEVIS E., METAXAS D. N., BADLER N. I.: User-controlled physics-based animation for articulated figures. In *Computer Animation* (Washington, DC, USA, 1996), IEEE Computer Society, pp. 16–26.

[KS05] KWON T., SHIN S. Y.: Motion modeling for online locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM, pp. 29–38.

[KSG02] KOVAR L., SCHREINER J., GLEICHER M.: Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM Press, pp. 97–104.

[KW02] KOPP S., WACHSMUTH I.: Model-based animation of coverbal gesture. In *Computer Animation* (2002), pp. 252–260.

[KW04] KOPP S., WACHSMUTH I.: Synthesizing multimodal utterances for conversational agents: Research articles. *Comput. Animat. Virtual Worlds 15*, 1 (2004), 39–52.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. In *SIGGRAPH* (New York, NY, USA, 2002), ACM Press, pp. 491–500.

[Lju98] LJUNG L.: *System Identification: Theory for the User (2nd Edition)*. Prentice Hall PTR, December 1998.

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 79–87.

[LM08] LANCE B. J., MARSELLA S. C.: A model of gaze for the purpose of emotional expression in virtual embodied agents. In *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)* (2008), pp. 199–206.

[Lon07] LONNEE J.: Transitions in simulated human motion. In *Proceedings of the 6th Twente Student Conference on IT* (February 2007), University of Twente, pp. 251–260.

[LP02] LIU K. C., POPOVIĆ Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (2002), 408–416.

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 39–48.

[Lue84] LUENBERGER D.: *Linear and nonlinear programming*. Addison-Wesley, Reading, MA, 1984.

[LvdP96] LAMOURET A., VAN DE PANNE M.: Motion synthesis by example. In *Computer Animation and Simulation* (New York, NY, USA, 1996), Springer-Verlag New York, Inc., pp. 199–212.

[LvdPF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. In *SIGGRAPH* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 201–208.

[LWS02] LI Y., WANG T., SHUM H.-Y.: Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 465–472.

[LWZB90] LEE P., WEI S., ZHAO J., BADLER N. I.:

Strength guided motion. In *SIGGRAPH* (New York, NY, USA, 1990), ACM, pp. 253–262.

[Mac90] MACIEJEWSKI A. A.: Motion simulation: Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications 10*, 3 (1990), 63–71.

[Man04] MANDEL M. J.: *Versatile and Interactive Virtual Humans: Hybrid use of Data-Driven and Dynamics-Based Motion Synthesis*. Master's thesis, Carnegie Mellon University, August 2004.

[MBC01] MIZUGUCHI M., BUCHANAN J., CALVERT T.: Data driven motion transitions for interactive games. In *EUROGRAPHICS 2001 short papers* (2001).

[MHP04] MANCINI M., HARTMANN B., PELACHAUD C.: *Non-verbal behaviors expressivity and their representation*. PF-star report 3, University of Paris 8, 2004.

[Mir96] MIRTICH B.: Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools 1*, 2 (1996), 31–50.

[MK05] MUKAI T., KURIYAMA S.: Geostatistical motion interpolation. *ACM Trans. Graph. 24*, 3 (2005), 1062–1070.

[MKMA04] MÉNARDAIS S., KULPA R., MULTON F., ARNALDI B.: Synchronization for dynamic blending of motions. In *Symposium on Computer Animation* (2004), pp. 325–335.

[MM04] MEREDITH M., MADDOCK S.: Using a half-jacobian for real-time inverse kinematics. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education* (2004), pp. 81–88.

[MTSC04] MAGNENAT-THALMANN N., SEO H., CORDIER F.: Automatic modeling of virtual humans and body clothing. *Journal of Computer Science and Technologie 19*, 5 (2004), 575–584.

[MTT96] MAGNENAT-THALMANN N., THALMANN D.: Computer animation. *ACM Computing Surveys 28*, 1 (1996), 161–163.

[MZW99] MATARIĆ M. J., ZORDAN V. B., WILLIAMSON M. M.: Making complex articulated agents dance. *Autonomous Agents and Multi-Agent Systems 2*, 1 (1999), 23–43.

[Nata] NATURALMOTION: Endorphin. http://www.naturalmotion.com/.

[Natb] NATURALMOTION: Euphoria. http://www.naturalmotion.com/.

[Nef05] NEFF M.: *Aesthetic Exploration and Refinement: A Computational Framework for Expressive Character Animation*. PhD thesis, Department of Computer Science, University of Toronto, 2005.

[Nef08] NEFF M.: Automatic torso engagement for gesturing characters. In *Intelligent Virtual Agents* (2008), vol. 5208 of *Lecture Notes in Computer Science*, Springer, pp. 522–523.

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), ACM Press, pp. 81–88.

[NF05] NEFF M., FIUME E.: Aer: aesthetic exploration and refinement for expressive character animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), ACM Press, pp. 161–170.

[NKAS08] NEFF M., KIPP M., ALBRECHT I., SEIDEL H.-P.: Gesture modeling and animation based on a probabilistic re-creation of speaker style. *ACM Trans. Graph. 27*, 1 (2008), 1–24.

[NR05] NOOT H., RUTTKAY Z.: Variations in gesturing and speech by gestyle. *International Journal of Human-Computer Studies 62*, 2 (2005), 211 – 229.

[Nvi08] NVIDIA: Nvidia physx, 2008. http://www.nvidia.com/.

[ODGK03] O'SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. *ACM Transactions on Graphics 22*, 3 (2003), 527–536.

[OM01] OSHITA M., MAKINOUCHI A.: A dynamic motion control technique for human-like articulated figures. *Computer Graphics Forum 20*, 3 (2001), 192–203.

[OTH02] OORE S., TERZOPOULOS D., HINTON G. E.: Local physical models for interactive character animation. *Computer Graphics Forum 21*, 3 (2002), 337–346.

[Ott03] OTTEN E.: Inverse and forward dynamics: models of multi-body systems. *Philosophical Transactions of the Royal Society 358*, 1437 (September 2003), 1493–1500.

[PB02] PULLEN K., BREGLER C.: Motion capture assisted animation: texturing and synthesis. In *Proceedings of the annual conference on Computer graphics and interactive techniques* (2002), pp. 501–508.

[Per95] PERLIN K.: Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1*, 1 (1995), 5–15.

[PG96] PERLIN K., GOLDBERG A.: Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 205–216.

[PM54] P. M. F.: The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 6 (June 1954), 381–391.

[PR01] POLLARD N. S., REITSMA P.: Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale Workshop on Adaptive and Learning Systems* (2001).

[Pra95] PRATT J. E.: *Virtual Model Control of a Biped Walking Robot*. Master's thesis, Massachusetts Institute of Technology, 1995.

[PSKS04] PARK S. I., SHIN H. J., KIM T. H., SHIN S. Y.: On-line motion blending for real-time locomotion generation. *Comput. Animat. Virtual Worlds 15*, 3-4 (2004), 125–138.

[PSS02] PARK S. I., SHIN H. J., SHIN S. Y.: On-line locomotion generation based on motion blending. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM, pp. 105–111.

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5 (1998), 32–40.

[RN96] REEVES B., NASS C.: *The media equation: how people treat computers, television, and new media like real people and places*. Cambridge University Press, New York, NY, USA, 1996.

[RP02] RUTTKAY Z., PELACHAUD C.: Excercises of style for virtual humans. In *Proceedings of Animating Expressive Characters for Social Interaction Symposium* (London, 2002), Imperial College, pp. 85–90.

[RP03] REITSMA P. S. A., POLLARD N. S.: Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Transactions on Graphics 22*, 3 (2003), 537–542.

[RP07] REITSMA P. S. A., POLLARD N. S.: Evaluating motion graphs for character animation. *ACM Trans. Graph. 26*, 4 (2007), 18.

[RPE*05] REN L., PATRICK A., EFROS A. A., HODGINS J. K., REHG J. M.: A data-driven approach to quantifying natural human motion. *ACM Transactions on Graphics 24*, 3 (2005), 1090–1097.

[RSC01] ROSE C. F., SLOAN P.-P. J., COHEN M. F.: Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum 20*, 3 (September 2001), 239–250(12).

[Sch75] SCHMIDT R.: A schema theory of discrete motor skill learning. *Psychological Review*, 83 (1975), 225–260.

[SDO*04] STONE M., DECARLO D., OH I., RODRIGUEZ C., STERE A., LEES A., BREGLER C.: Speaking with hands: creating animated conversational characters from recordings of human performance. *ACM Transactions on Graphics 23*, 3 (2004), 506–513.

[SH05] SAFONOVA A., HODGINS J. K.: Analyzing the physical correctness of interpolated human motion. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), ACM Press, pp. 171–180.

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph. 26*, 3 (2007), 106.

[Sha07] SHAPIRO A.: Dynamic animation and control environment, 2007. http://www.arishapiro.com/dance/.

[Sho85] SHOEMAKE K.: Animating rotation with quaternion curves. *SIGGRAPH 19*, 3 (1985), 245–254.

[Sim94] SIMS K.: Evolving virtual creatures. In *SIGGRAPH* (New York, NY, USA, 1994), ACM, pp. 15–22.

[SK05] SENTIS L., KHATIB O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics 2*, 4 (2005), 505–518.

[SKG03] SHIN H. J., KOVAR L., GLEICHER M.: Physical touch-up of human motions. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 194–203.

[SM01] SUN H. C., METAXAS D. N.: Automating gait generation. In *SIGGRAPH* (New York, NY, USA, 2001), ACM Press, pp. 261–270.

[Smi08] SMITH R.: Open dynamics engine, 2008. http://www.ode.org/.

[SNI06] SHIRATORI T., NAKAZAWA A., IKEUCHI K.: Dancing-to-music character animation. *Computer Graphics Forum 25*, 3 (September 2006), 449–458.

[SO06] SHIN H. J., OH H. S.: Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 291–298.

[Son08] SONY: Playstation 3, 2008. http://www.playstation.com/.

[SPF03] SHAPIRO A., PIGHIN F. H., FALOUTSOS P.: Hybrid control for interactive character animation. In *Pacific Graphics* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 455–461.

[SR01] SHERMAN M., ROSENTHAL D.: Sd/fast, 2001. http://www.sdfast.com/.

[Sta] STANFORD BUSINESS SOFTWARE: Optimization software. http://www.sbsi-sol-optimize.com/index.htm.

[Ste00] STEWART D. E.: Rigid-body dynamics with friction and impact. *SIAM Rev. 42*, 1 (2000), 3–39.

[SvdP05] SHARON D., VAN DE PANNE M.: Synthesis of

controllers for stylized planar bipedal walking. In *International Conference on Robotics and Animation* (2005), pp. 2387–2392.

[TGB00] TOLANI D., GOSWAMI A., BADLER N. I.: Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models and Image Processing 62*, 5 (2000), 353–388.

[TH00] TANCO L. M., HILTON A.: Realistic synthesis of novel human movements from a database of motion capture examples. In *HUMO '00: Proceedings of the Workshop on Human Motion (HUMO'00)* (Washington, DC, USA, 2000), IEEE Computer Society, pp. 137–142.

[THB07] TORRESANI L., HACKNEY P., BREGLER C.: Learning motion style synthesis from perceptual observations. In *Advances in Neural Information Processing Systems 19*, Schölkopf B., Platt J., Hoffman T., (Eds.). MIT Press, Cambridge, MA, 2007.

[The] THE MATHWORKS: Matlab optimization toolbox. http://www.mathworks.com/products/optimization/.

[THN04] THEUNE M., HEYLEN D., NIJHOLT A.: Generating Embodied Information Presentations. In *Multimodal Intelligent Information Presentation*, Stock O., Zancanaro M., (Eds.). Kluwer Academic Publishers, 2004, pp. 47–69.

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. *ACM Trans. Graph. 26*, 3 (2007), 7.

[TMMK08] THIEBAUX M., MARSHALL A. N., MARSELLA S., KALLMANN M.: Smartbody: Behavior realization for embodied conversational agents. In *Proc. 7th International Conference on Autonomous Agents and Multiagent Systems* (2008), pp. 151–158.

[UAT95] UNUMA M., ANJYO K., TAKEUCHI R.: Fourier principles for emotion-based human figure animation. In *SIGGRAPH* (New York, NY, USA, 1995), ACM Press, pp. 91–96.

[Ubi04] UBISOFT: Prince of persia: The sands of time, 2004. http://www.princeofpersiagame.com/.

[UGB*04] URTASUN R., GLARDON P., BOULIC R., THALMANN D., FUA P.: Style-based motion synthesis. *Computer Graphics Forum 23*, 4 (2004), 799–812.

[UKS89] UNO Y., KAWATO M., SUZUKI R.: Formation and control of optimal trajectory in human multijoint arm movement - minimum torque-change model. *Biological Cybernetics 61* (1989), 89–101.

[vBE09] VAN BASTEN B. J., EGGES A.: Evaluating distance metrics for animation blending. In *Proceedings of the 4th International Conference on the Foundation of Digital Games (to appear)* (2009).

[vdP93] VAN DE PANNE M.: Sensor-actuator networks. In *SIGGRAPH* (New York, NY, USA, 1993), ACM Press, pp. 335–342.

[vdPKF94] VAN DE PANNE M., KIM R., FIUME E.: Virtual wind–up toys for animation. In *Graphics Interface '94* (May 1994), pp. 208–215.

[VT82] VIVIANI P., TERZUOLO C.: Trajectory determines movement dynamics. *Neuroscience 7*, 2 (1982), 431–437.

[vWNRZ06] VAN WELBERGEN H., NIJHOLT A., REIDSMA D., ZWIERS J.: Presenting in virtual worlds: Towards an architecture for a 3D presenter explaining 2D-presented information. *IEEE Intelligent Systems 21*, 5 (2006), 47–99.

[vWRV08] VAN WELBERGEN H., RUTTKAY Z., VARGA B.: Informed use of motion synthesis methods. In *Motion in Games*, Egges A., Kamphuis A., Overmars M., (Eds.), vol. 5277 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 2008, pp. 132–143.

[vWZR09] VAN WELBERGEN H., ZWIERS J., RUTTKAY Z.: Real-time animation using a mix of dynamics and kinematics. *Submitted to Journal of Graphics Tools* (2009).

[WC91] WANG L. C. T., CHEN C. C.: A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *Robotics and Automation, IEEE Transactions on 7*, 4 (1991), 489–499.

[Wel93] WELMAN C.: *Inverse kinematics and geometric constraints for articulated figure manipulation*. Master's thesis, Simon Fraser University, September 1993.

[WH95] WOOTEN W. L., HODGINS J. K.: Simulation of human diving. In *Graphics Interface '95* (May 1995), pp. 1–9.

[WH97a] WILEY D. J., HAHN J. K.: Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl. 17*, 6 (1997), 39–45.

[WH97b] WOOTEN W. L., HODGINS J. K.: Transitions between dynamically simulated motions: Leaping, tumbling, landing, and balancing. In *In ACM SIGGRAPH Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH* (1997), ACM Press, p. 217.

[WH00] WOOTEN W. L., HODGINS J. K.: Simulating leaping, tumbling, landing, and balancing humans. In *International Conference on Robotics and Animation* (2000), pp. 656–662.

[Win04] WINTER D. A.: *Biomechanics and Motor Control of Human Movement*. Wiley, August 2004.

[WJM06] WROTEK P., JENKINS O. C., MCGUIRE M.: Dynamo: Dynamic, data-driven character control with adjustable balance. In *Proceedings of the Sandbox Symposium on Video Games* (July 2006), ACM, ACM.

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *SIGGRAPH* (New York, NY, USA, 1988), ACM Press, pp. 159–168.

[Woo98] WOOTEN W. L.: *Simulation of leaping, tumbling, landing, and balancing humans*. PhD thesis, Georgia Institute of Technology, March 1998.

[WP95] WITKIN A., POPOVIC Z.: Motion warping. In *SIGGRAPH* (New York, NY, USA, 1995), ACM Press, pp. 105–108.

[WP00] WATT A., POLICARPO F.: *3D Games: Volume 1: Real-Time Rendering and Software Technology*. Addison-Wesley, 2000.

[WTT92] WOODSON W. E., TILLMAN B., TILLMAN P.: *Human Factors Design Handbook*. McGraw-Hill, 1992.

[YCBvdP08] YIN K., COROS S., BEAUDOIN P., VAN DE PANNE M.: Continuation methods for adapting simulated skills. *ACM Trans. Graph. 27*, 3 (2008).

[YCP03] YIN K., CLINE M. B., PAI D. K.: Motion perturbation based on simple neuromotor control models. In *Pacific Graphics* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 445–449.

[YLS04] YANG P.-F., LASZLO J., SINGH K.: Layered dynamic control for interactive character swimming. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 39–47.

[Zel82] ZELTZER D.: Motor control techniques for figure animation. *Computer Graphics and Applications, IEEE 2*, 9 (Nov. 1982), 53–59.

[ZH99] ZORDAN V. B., HODGINS J. K.: Tracking and modifying upper-body human motion data with dynamic simulation. In *In Proceedings of Computer Animation and Simulation* (September 1999), pp. 13–22.

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), ACM Press, pp. 89–96.

[ZMCF05] ZORDAN V. B., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. In *SIGGRAPH* (2005), ACM Press, pp. 697–701.

[ZMM*07] ZORDAN V. B., MACCHIETTO A., MEDINA J., SORIANO M., WU C.-C.: Interactive dynamic response for games. In *Sandbox: Proceedings of the ACM SIGGRAPH symposium on Video games* (New York, NY, USA, 2007), ACM Press, pp. 9–14.

[ZvdP05] ZHAO P., VAN DE PANNE M.: User interfaces for interactive control of physics-based 3d characters. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM Press, pp. 87–94.