Volume xx (200y), Number z, pp. 1–23

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. M. 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 motion naturalness and the control that can be exerted over the motion. We show mechanisms to parameterize, combine (on different body parts) and concatenate motions generated by different animation techniques. 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): Three-Dimensional Graphics and Realism [I.3.7]: 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 perceived realism.

Many techniques exist that achieve real-time animation. These techniques differ in the trade-off they offer between the control that can be exerted over the motion, the motion naturalness, and the required calculation time. Choosing the right technique depends on the needs of the application. This paper 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 discuss models of the VH's body that are steered by animation. In Section 3 we classify animation techniques that are used to generate motion primitives and discuss their strengths and weaknesses. In Section 4 we show how to parameterize, combine (on different body parts) and concatenate motion generated by these techniques to gain control. We elaborate on 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. 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 common to work with the underlying *skeleton* instead. A skeleton is an articulated structure: a hierarchy of segments connected by joints. A *pose* of a VH is set by rotating the joints of the skeleton. How the skeleton deforms the mesh is beyond the scope of this paper, 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 knee joint has only one DoF, while a shoulder joint has three. The global translation of the skeleton

is represented by the translation of the root joint. The pose of a skeleton with n rotational DoFs can therefore be described by an $n+3$ dimensional vector $\mathbf{q}$. For an overview of rotation representations we refer the reader to the work of Lee [Lee08].

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

2.2. Physical Model of the VH

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

Most physical animation systems 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 (see [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 physical realism 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) controls our muscles, making use of input gained via sensors. 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 is readily available from the representation of the virtual world. Sensors used in computer animation therefore do not necessarily need to correspond to the sensors found in humans, but merely represent a convenient higher level presentation of VH state information that can be shared among different motion controllers [FvdPT01]. Examples of such sensors are the center of mass (CoM) of the VH, contact information (are the feet or other body parts in contact with the ground?), the location of the support polygon (the convex hull of body parts touching the ground), 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. In all physically realistic motion with ground contact, the ZMP is inside the support polygon. If the ZMP is outside the support polygon, the VH is perceived as being unbalanced [SKG03].

2.3.2. Modeling Muscles

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 hypothesized that the CNS exerts control at a joint and joint synergy 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]. In real-time animation, such spring and damper systems are often designed using PD-controllers or variants thereof (see 3.3.2.1,3.3.2.2). Joint rotation limits and maximum joint strength can be obtained from the human factors literature (see for example: [WTT92, KB96]).

2.3.3. Motor Control

Motor control is the process that steers 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 accurate 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*, joint torque paths are not explicitly programmed, but emerge 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 automatically guide the hand to an equilibrium point. Movement is achieved by a succession of equilibrium points along a trajectory [Fel86]. Feedback control (see 3.3.2), GMPs (explicitly in [Zel82, KW04], implicitly in 3.2, 3.1.2) and equilibrium point control (see 3.3.2.2) are all 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 [Fit54]. 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. The variability in muscle output increases with the strength of the command. For maximum accuracy it is therefore desirable to keep the strength of motor commands low during the whole movement trajectory, thus producing smooth movement. Faster movement requires stronger motor commands, thus higher variability which leads to reduced precision. In computer animation, movement invariants have been used both in motion synthesis models (for example: [GLM01, KW04]) and as evaluation criteria for the naturalness of animation (see 5.5.2). The notion of signal dependent noise has been exploited in the generation of motion variability (see 5.4.3).

3. Animation Techniques

Animation techniques create *motion primitives* from *motion spaces* on the basis of *animation parameter* values (see Figure 1). A motion space is a (continuous) collection of motions that can be produced by a technique. A motion primitive is an element of such a motion space. Motion primitives can define motion for the full body of a VH or on a subset of the joints of the VH. The motion primitives in a specific motion space typically have a certain semantic function (for example: walk cycles, beat gestures, left hand uppercuts). The animation parameters needed to create motion primitives differ per technique. Note that animation parameters are not necessarily intuitive parameters to control motion, but merely the parameters a specific animation technique requires to create a motion primitive. We discuss how to map more intuitive *control parameters* into animation parameters in Section 4.1.

We classify animation techniques by the mechanism they use to create motion spaces (see Figure 1 and 2). *Motion editing* techniques generate motion primitives within a motion space spanned by one or more specific example motion primitives. In *simulation* techniques, the motion space contains all motion primitives that can be created using a parameterized physical or procedural model. Animation parameters in simulation techniques are the parameters used in the simulation model. In Sections 3.1, 3.2 and 3.3 we briefly discuss the inner working of each technique and discuss the the nature of the animation parameters and motion spaces produced by them. Figure 2 provides a summary of the latter. Section 3.4 discusses the strengths and weaknesses of each technique in terms of naturalness and control and gives an overview of application domains in which each techniques is typically used.

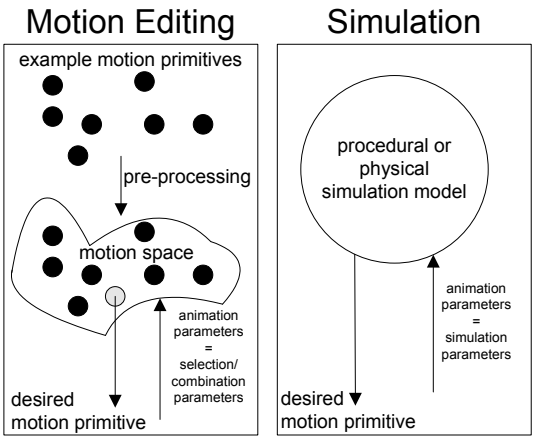

Figure 1: *Motion primitives, motion spaces and animation parameters in motion editing and simulation.*

3.1. Motion Editing

Motion editing techniques generate motion primitives within a motion space spanned by one or more specific example motion primitives. Often, this motion space is explicitly constructed in a pre-processing stage. The example primitives originate from motion captured movement of actors or are created by hand by an animator. We define *motion modification methods* as methods that generate new motion primitives by applying modifications to a *single* example motion primitive. *Combination techniques* create motion primitives using a database of *multiple* example primitives.

3.1.1. Motion Modification

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 around a example motion primitive. Bruderlin and Williams [BW95] considerer 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 make local modifications to the signal while maintaining continuity and preserving the global shape of the signal. This is done by specifying some additional keyframes, or have them determined by inverse kinematics (IK, see Appendix for an overview of techniques), within a example motion primitive. 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 inbetween displacements. The displacement map then yields a displacement for every frame, which is added to the original signal. Satisfying constraints only at key frames does not guarantee constraint enforcement at inbetween frames. Alternatively, a constraint can be enforced at *every* frame on which it is

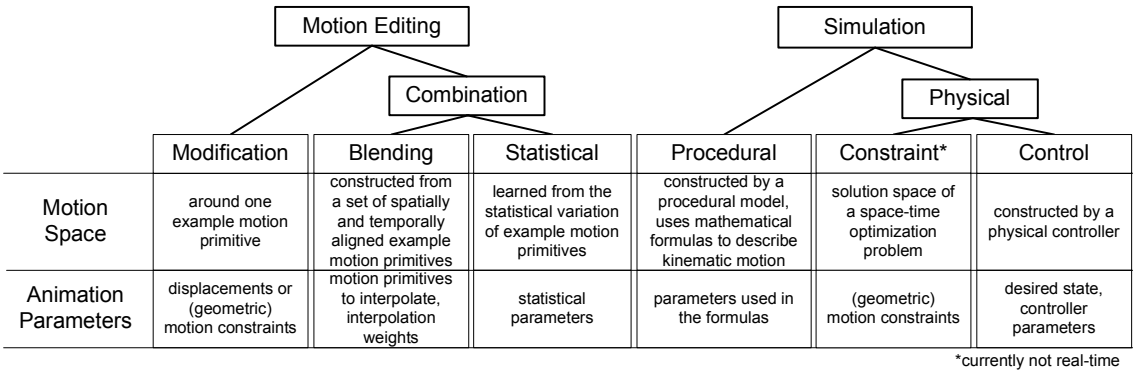

Figure 2: *Classification of animation techniques and an overview of their animation parameters and motion spaces.*

desired as proposed by Lee and Shin [LS99]. To make sure the resulting motion is smooth and propagated through non-constrained frames, it is 'filtered' using a hierarchy of 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, he implements it with a different IK solver and a convolution based linear filter.

An alternative approach by Gleicher [Gle97] is to pose the constraint specification as a numerical constraint optimization problem: an objective function measuring the distance between the example motion primitive and the resulting motion is minimized subject to any constraint that can be specified as a function of the vector of DoF $\mathbf{q}$. To allow real-time execution of this optimization, an efficient objective function is chosen and the constraints are only enforced at key frames. 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. 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.

3.1.2. Blending

Blending [WH97a] creates a motion primitive by interpolating a family of example similar motion primitives. The animation parameters are interpolation weights and a selection of the example motion primitives to interpolate. The interpolation does not need to take place in the Euler space, but can also be done in, for example, the principal component [IST02] or Fourier [UAT95] domain. 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.

The blend motion space is created by pre-processing "similar" example motion primitives, typically such that they correspond in time (especially at key events such as foot plants) and space (e.g. root rotation and position). The process of time-aligning corresponding phases in motion primitives, is called time warping [BW95]. Kovar and Gleicher [KG03] present an integrated method called registration curves to automatically determine the time, space and constraint correspondences between a set of motion primitives and provide an overview of previous methods used for this.

3.1.3. Statistical models

Statistical methods create a motion space using statistical models learned from the statistical variation of example 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].

3.2. Procedural Simulation

Procedural simulation uses parameterized mathematical formulas to create motion primitives. The parameters of such formulas are the animation parameters. The formulas can describe joint rotation directly (as done in [Per95]), or describe the movement path of end effectors (such as hands) through space. The latter is typically used to design procedural models that create gesture motion primitives (see for example [CCZB00, KW04, HMP06, NKAS08]).

3.3. Physical Simulation

A physical simulation model applies torques on the joints of the VH, on the basis of animation parameters. The resulting

motion primitive is calculated using forward dynamics (see Appendix).

3.3.1. Constraint Control Methods

Constraint Control Methods use (geometric) constraints as animation parameters. There are typically many possible muscle torque paths that achieve the constraints. An objective function can be introduced to specify a certain preference for solutions. Typically, the objective functions are biomechanically based: minimize the expended energy, minimize end effector jerk, or use a weighted combination of those two. The constraint control problem can be stated as a non-linear optimization problem [WK88]. Several techniques have been proposed to speed up the calculation process of the optimization (for example: [LP02, FP03]), typically at the cost of some physical realism. Even with those speedups, constraint based control methods are currently not a feasible option for real-time animation.

3.3.2. Physical Simulation using Controllers

A physical controller and the physical system it controls (the physical body of a VH) together form a control system [KMB96]. The input to the controller is the desired value of the system's state. This desired state is part of the animation parameter set. The output is a set of joint torques that, when applied to the system, guides its variables towards their desired values. The controller can make use of static physical properties (like mass, or inertia) of the physical body performing the motion. Such a control system can, to a certain extent, cope with external perturbation, in the form of impulses, forces or torques exerted on the body. The goal of the controller is to minimize the discrepancy between the actual and desired state. Forces and torques set by the controller, gravity and ground contacts forces, and forces and torques caused by external perturbations are applied to the physical body. The body is then moved using forward dynamics. The new state of the body is fed back into the controller.

3.3.2.1. Proportional Derivative (PD) Control is an easy to implement and frequently used control method (for example in [HWBO95, ZH99, WH00, FvdPT01, ACSF07]). The output torque of the PD-controller is proportional to the difference in position and velocity between the desired state and the actual state:

$$\tau = k_p(x_d - x) + k_d(\dot{x_d} - \dot{x}) \quad (1)$$

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 [WH00]). The animation parameters that have to be used to create a motion primitive are k_p, k_d, x_d and $\dot{x_d}$. Finding appropriate values for k_p and k_d that result in achieving x_d and $\dot{x_d}$ is a manual trial-and-error process. They depend on characteristics of both the system and the motion.

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

$$\tau = k_{p_L}(\theta_L - \theta) + k_{p_H}(\theta_H - \theta) - k_d\dot{\theta} \quad (2)$$

in which animation parameters 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. Stiffness, defined as $k_{p_L} + k_{p_H}$, is used as another animation parameter. Equilibrium point control (see 2.3.3) is used to calculate k_{p_L} and k_{p_H}, given the provided stiffness and external forces (typically gravity). Movement is achieved by gradually moving the equilibrium position.

3.3.2.3. Local Optimization PD-controllers typically do not generalize well beyond the specific physical body, environment and contact conditions they were designed for. Controllers using local quadratic optimization provide better generalization. They optimize the control objectives for the current frame, subject to certain constraints (e.g. the physical equations of motion). Unlike space-time approximation approaches (see 3.3.1), these controllers cannot anticipate the long term minimization of their objective, given constraints at certain time frames, but do allow real-time execution. The computation cost of local optimization is far higher than the computation cost of PD or similar controllers.

Stewart and Cremer [SC92] introduce a custom physics simulator that can optimize objectives (which are required to be second order derivatives of system variables), subject the physical equations of motion and optionally specific constraints that can be added on the fly. Abe et al. [AdSP07] extend on this work by designing controllers that optimize objectives, subject to not only the physical equations of motion, but also contact and friction dynamics and maximum joint strengths. Their system is designed to work with any physics simulator. The objectives regulate the values of certain kinematic quantities $f(\mathbf{q})$, by minimizing the difference between the desired acceleration of f and its current acceleration. Abe et al. [AdSP07] show some strategies to find the desired acceleration of $f(\mathbf{q})$ for balancing controllers and controllers that track a prescribed joint rotation trajectory.

3.3.2.4. 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 an animation parameter (distance traveled, energy expended, distance from stylized reference pose) is optimized (see for example [Sim94,SvdP05]). Using

such techniques, locomotion controllers for simple creatures with few DoFs can be created. However, so far automatic controller generation techniques have not scaled up to provide natural motion for full-sized VHs.

3.3.2.5. Physical Controllers Toolkits The Dynamic Animation and Control Environment [SFNTH05] provides researchers with an open, common platform to test out and design physical controllers using scripting. NaturalMotion's Endorphin [Nat] is a commercial animation system that provides 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 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.

3.4. Strengths and Weaknesses of Different Animation Techniques

Motion editing techniques retain the naturalness and detail of example motion primitives or motion primitives generated by skilled artists. However, motion editing techniques produce natural motion only when the modifications to the example motion primitives are small. Techniques that make use of multiple example motion primitives retain naturalness over larger modifications than techniques that use a single example motion primitive [Gra00]. However, both blending and statistical techniques suffer from the curse of dimensionality: in practice the number of required example motion primitives grows exponentially with the number of animation 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 5.1). Motion editing is useful for creating animation in advance for non-interactive applications (like films). For other domains, like games, naturalness and controllability can only be assured by using a huge database of example motion primitives.

Physical simulation provides physically realistic motion and (physical) interaction with the environment. Physical controllers can robustly retain or achieve animation 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 (see 4.1.5). 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 (for example in [HWBO95, WJM06]), stunts by stunt men [FvdPT01], or falling motions (for example [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 example motion primitives into the mathematical formulas that create 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 animation techniques like [Per95, PG96, CCZB00, KW04, HMP06, 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, or the flexibility and precision of procedural motion can be used to generate arm gestures on a VH which retains balance in a physically realistic manner using a balance controller 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 animation techniques.

4. 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 to be constructed and adapted in real-time. Control enables the expression and adaptation of such plans by means of parameterization, combination and concatenation.

Parameterization (see Figure 3) is the process of selecting *animation parameter* values (like blend weights, stiffness gains, Principal Component values, etc) that, when provided to an animation technique, create a motion primitive that satisfies some *control parameters* (for example: create a gesture motion primitive that exhibits a certain tension and amplitude).

Concatenation (see Figure 4) deals with the generation of a sequences of motion primitives, to form a natural motion that satisfies certain control parameters. The motion primitives can be generated by different techniques (or the same technique that is initialized differently). For example: using a walk controller and a blending technique that uses preprocessed sit down motion primitives as its input, a sequence of motion primitives can be generated to achieve a 'walk to the chair and sit on it' motion.

Rather than explicitly constructing new motion primitives for each combination of motions acting on a separate body

part, different motion primitives, possibly constructed by different animation techniques, can be *combined* (see Figure 5) in such a way that a coherent whole body motion results. For example: a walk cycle motion primitive and a chew gum motion primitive can be combined to obtain walking while chewing gum.

Controllability is determined by various aspects. *Responsiveness* determines how fast a desired change in the motion plan is achieved. For example, how fast does an animated soccer player respond to a gamer pressing the shoot button? *Precision* is the accuracy with which control parameters (like end effector position or timing constraints) are achieved. *Coverage* deals with how much of the control parameter space is covered (for example: what positions can be kicked at within a kick motion space). We define *expressiveness* as the number of control parameters that can be used in the motion plan. *Intuitiveness* deals with how intuitive the control parameters that can be used in the specification of the motion plan are.

4.1. Parameterization

Parameterization deals with selecting the animation parameters, that, when provided to an animation technique, create a motion primitive with some desired control parameter values. One common control parameter is a pose constraint (for example: requiring the hand to be at a certain location) at a desired time. Other more abstract parameterizations deal with control parameters like emotion or physical state (such as tiredness). Some animation techniques provide intuitive animation parameters that can directly be used as control parameters (for example: geometric constraints in motion modification). Other techniques (for example blending) do not provide intuitive animation parameters. For such techniques and for the more abstract parameterizations mentioned above, some mapping of control parameters to animation parameters is needed (see Figure 3). If multiple desired

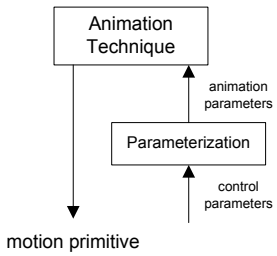

Figure 3: *Parameterization maps control parameters to animation parameters to create a motion primitive that satisfies control parameter values.*

control parameter values are specified, it might not be possible to satisfy them all. Several parameterization methods therefore include strategies to deal with conflicting control parameter values.

4.1.1. Parameterization in Procedural Motion

In procedural animation, the control parameters can directly be expressed in terms of variables of the motion functions (and thus animation parameters). Pose constraints are typically satisfied by setting animation parameters that specify end effector positions or joint rotations. Authoring procedural motions requires specifying how each parameter influences the motion. For control parameters such as emotion or physical state, this is not a very intuitive process. Therefore, such control parameters are typically mapped to to animation parameters instead. This mapping can result in parameter conflicts if control parameter values select different values for the same animation parameter.

Neff and Fiume [NF05], design a hierarchical framework for procedural motion and provide a generic parameter mapping framework. Lower level control parameters specify the motion on a single joint or group of joints (called an action in [NF05]). Higher level control parameters are mapped to animation parameters through a script created by an animator. Motion primitives are created using various, possibly conflicting low level and high level control parameters. Therefore, several mechanisms are in place to handle conflict resolution: low level control parameters (placed on a single DoF, rather than on the whole body) take precedence over high level control parameters, which take precedence over the default values defined in a Sketch that models the VHs style (see 5.3.2).

Chi et al. [CCZB00] claim that Effort and Shape parameters from Laban Movement Analysis (LMA) not only provide means to parameterize gesture, but are essential features of a 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 vs. 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 Effort and Shape control parameters to animation parameters that guide arm and torso movement. The arm movement is specified by 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 control parameters.

Hartmann et al. [HMP06], use a smaller but quite similar set of control parameters. From a literature review they conclude that six control parameters (activation, spatial extent, temporality, fluidity, power and repetivity) are sufficient to specify gesture expressivity. The control parameter selection is based on what humans can observe and reliably recognize. In their system, gestures are generated by TCB splines defining the trajectory of the hands. The six control parameters

are mapped to animation 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 control parameters are intuitive, but not independent, specifically they mention an unresolved conceptual interdependence between the power and temporal extend (roughly duration) control parameters.

4.1.2. Parameterization using Constraint Editing

Recorded motion primitives can be modified to adhere to a pose constraint, using a motion modification technique (see section 3.1.1). In this case, the animation parameters are directly used as control parameters. Le Callennec et al. [CB04] provide a PFIK+F framework that can handle multiple pose constraints. It resolves possible conflicts in constraints by satisfying those with the highest priority first.

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 emotion transform, using a Linear Time Invariant model rather than signal amplification for the spatial transform.

4.1.3. Parameterization using Blending

To achieve a desired pose at a desired time, a set of motion primitives to interpolate and their interpolation weights have to be found. Many parameterization methods 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 control parameters. Unfortunately, blending does not yield a linear parameterization of this control parameter space [RSC01]. That is, if $\mathbf{s}_{des}$ is exactly 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 end effector at $\mathbf{s}_1$ and $\mathbf{s}_2$, will end up placing the end effector at $\mathbf{s}_{des}$. Several methods have been developed to solve this discrepancy.

Rose et al. [RCB98] uses a sum of the best linear map between blend weights and end effector positions and radial basis functions, centered on each recorded motion primitive to determine blend weights. For desired poses that are far from the examples, blend weights are based purely on the linear approximation and hence are effectively arbitrary [KG04]. Grassia [Gra00] uses a linear approximation of the blend weights in an initial positioning step and then exactly positions the end effector at the goal position using a constraint based method (see 3.1.1). Van Basten et al. [vBSE10] linearize part of the posture representation and interpolate *positions* of joints instead of *rotations*. This will result in end effectors that are exactly on the desired position.

Many other techniques make use of pseudo example motion primitives. Wiley and Hahn [WH97a] *resample* the examples to a dense regular grid in a pre-computing step that exhaustively searches through interpolation weights and recorded motion primitives. The grid can then be used to efficiently select the pseudo examples to be interpolated. Note that, compared to Rose et al. [RCB98], only a subset of the example motion primitives are blended. Also, the number of required example motion primitives is $O(2^d)$, where d is the dimensionality of the parameter space, whereas Rose et al. [RCB98] only require $O(d)$ samples. Rose et al. [RSC01] use the smoothness of the function that maps blend weights to end effector position values to create pseudo examples online at selected positions, iteratively improving the accuracy of the parameterization. Kovar and Gleicher [KG04] randomly create random pseudo examples online. 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 intensity 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 cannot be attained by the blending methods described above.

Torresani et al. [THB07] provide parameterization of three of the LMA Effort control parameters (see section 4.1.1). Recorded motion primitives are annotated by a LMA expert. These annotations are translated to numerical values. By annotating the Effort of blends of motion primitives with known Effort values, a function that maps blend weights, input joint angle data and input Effort values to the output Effort values is learned. A motion primitive with unknown Effort values can then be adapted to have desired Effort values by blending. This entails finding its k-nearest neighbors in the database of annotated motion primitives and finding the motion primitive pair that, with the optimal blend weight

(found by uniform sampling), approximates the desired Effort values the best. At the cost of computation time and annotation effort (by an LMA-expert), this method achieves a more accurate Effort parameterization than simple linear blending.

4.1.4. Parameterization in Statistical Models

Satisfying control parameters using statistical models requires specialized methods for each statistical model. Grochow et al. [GMHP04] search their SGPLMVM model representation of the motion space using optimization to create motion primitives that satisfy pose constraints. Li et al.'s [LWS02] motion texton representation of the motion space allows the creation of motion primitives by directly specifying poses at selected frames. Mukai and Kuriyama [MK05], create a geostatistical model of a set of recorded motion primitives. Geostatistical interpolation is used to create the motion primitive with desired pose constraints. This method is more accurate in achieving the desired pose constraints than blending methods that use radial basis functions. It is more efficient (in terms of calculation time and memory usage) than blending methods that use pseudo examples. Carvalho et al. [CBT07] introduce a constraint based editing method that uses the same prioritized IK solver as [CB04] (see also 4.1.2) on a low-dimensional statistical motion model, generated using PCA or probalistic PCA. Their system is computationally more efficient, and is, according to the authors, in some cases more natural than the PFIK+F approach used in [CB04].

In human motion, there are many correlations between joint actions. Statistical methods [EMMT04] and machine learning [BH00] have been employed to find orthogonal control parameters in a set of recorded motion primitives. Because the parameters are independent, it is not necessary to resolve parameter conflicts. However, the control parameters learned in such approaches are not very intuitive to use and are highly depended on the training data.

4.1.5. Parameterization using Physical Simulation

The desired state of a controller can directly be used as a control parameter. Animation parameters like desired joint rotation, pelvis height or CoM position provide intuitive control. However, 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.

Abe and Popović [AP06] show how to set up a physical controller that satisfies control parameter values in order of their priority. They report that prioritization of balance control interferes with posture control, which makes it difficult to combine these two control objectives in a natural manner. In later work, Abe et al. [AdSP07] use a weighted combination of control objectives to achieve compromise between their control parameter values. The weights can be used to smoothly move from one objective to the next.

Some techniques have been devised to map control parameters to animation parameters used in controllers. Chao et al. [CYL06] provide a mapping from LMA-Effort parameters to animation parameters for a PD-controller, such as damping, stiffness and desired joint rotation.

Yin et al. [YCBvdP08] apply an optimized learning strategy to adapt the animation parameters $\mathbf{w}$ of a physical walking controller to a new situation parameterized by the control variable γ (for example: step over an obstacle of height γ, push a piece of furniture with weight γ, walk on slippery terrain with friction coefficient γ). The animation parameter space is searched for valid values of $\mathbf{w}$ (as in, those that do not make the VH fall) that achieve γ. There might be many viable solutions of $\mathbf{w}$ that achieve γ. A hand-authored objective function evaluates $\mathbf{w}$ to help select a unique optimal solution that achieves γ. It can be designed to prefer solutions that have a minimal deviation from the original animation parameters, a certain walking speed or step size, etc. The learning process is offline, but the learned animation parameter values can be interpolated to achieve real-time control. It is yet to be seen if and how this method generalizes to more than one control parameter.

4.2. Concatenating

Concatenation (see Figure 4) deals with the generation of a sequences of motion primitives, to form a natural motion that satisfies certain control parameters and assures the naturalness and smoothness of the resulting motion. The motion primitives can be generated by different techniques (or the same technique that is initialized differently).

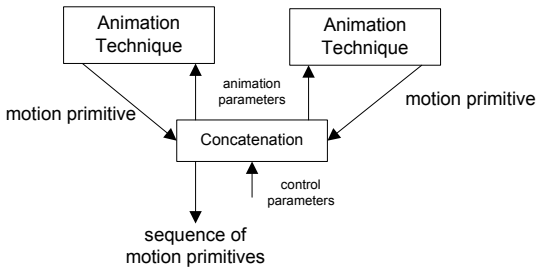

Figure 4: *Concatenation generates a sequence of motion primitives using (possibly different) animation techniques. The resulting motion satisfies the control parameter values.*

4.2.1. Concatenation using Motion Editing

Ease-in ease-out interpolation between two motion primitives can be used to concatenate them. The first motion primitive is faded out as the second one is faded in. Displacement maps (see Section 3.1.1) can also be used to transition from one motion primitive to another, as is done in [LCR*02]. Transitions between different pairs of motion primitives concatenated in this manner 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 motion primitives belonging to that cluster. During preprocessing, the naturalness of all possible 2, 3 or 4 multiway blends between representatives is evaluated (using footskate and ZMP position as evaluation criteria) and the best blend recipe (containing a weight function and representatives) is stored. A transition is generated at runtime by matching the current and next motion primitives to mediods and applying the stored blend recipe.

4.2.1.1. Motion Graphs In many applications, one requires multiple concatenated motion primitives to satisfy a longer term control parameter (for example: walk to a certain position). A very common technique is to put all the possible transitions between motion primitives in a graph-like structure: a motion graph. A motion graph is a directed graph where all edges correspond to motion primitives and nodes correspond to poses. Interpolations between poses from different (or the same) motion primitives that are 'similar enough' are added as new edges. In the game industry such 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. After the graph is created, the graph can then be searched to find a sequence of motion primitives that adheres to control parameters values (for example: walking along a specified path).

Many variations of motion graphs exist, which can be distinguished in off-line methods where the desired control parameters are known in advance and the motion is generated offline (for example: [KGP02, AF02]), and methods that work at interactive speed (for example: [GSKJ03, LL06, LCR*02, vBPE10]). These techniques mainly differ in the graph structure or the search strategy. Here we mainly discuss the inherent naturalness-control-calculation time trade-offs in motion graphs, for an exhaustive literature overview we refer the reader to a recent article featuring motion graphs (e.g. [ZS09]) or Forsyth et al.'s survey ([FAI*06], p184-194). Throughout this article, we make use of the terminology for naturalness, control and calculation time aspects of motion graphs introduced in [FAI*06].

At interactive speed, a global search on the graph is infeasible [FAI*06]. Local search evaluates only the values of control parameters in a path through a limited number of nodes when choosing the next sequence of motion primitives. Even if a path on the graph is available that satisfies control parameter values, a local search method might not find it, because it cannot look far enough ahead. This is called the horizon problem in [FAI*06].

Reinforcement learning (first proposed by Lee and Lee [LL06]) can be used to learn (near) optimal long-term plans for specific control parameter *values* that are specified as a reward function. The learning process is offline. It provides a (near) optimal path on the motion graph from every state of the VH and its environment achieving the learned control parameter values. Some flexibility can be gained by a smart selection of state and objective function. For example: if the state is set as the angle between the current walk direction and the goal direction, walking in any goal direction can be learned by learning how to walk forward. Walking to a desired 2D location can be learned in a similar manner. One can also learn a grid of control parameter values [LL06]. Because of its discretization of control parameter values, reinforcement learning sacrifices some accuracy for long term goal satisfaction. Furthermore, it can be hard to coordinate multiple goals and is typically very memory intensive [LL06]. Recent approaches using reinforcement learning aim to address the latter [TLP07, LZ08, LLP09].

Control and motion planning is limited by the available paths on the graph. As more control parameters are added, less paths will become available that satisfy all their desired values. It is possible to extend the graph (and hence, gain more control) by adding more transitions. Unfortunately, at some point the added transitions become unnatural [vBE09]. This is a typical trade-off when using motion graphs. More transitions will result in more control but also more visual artifacts such as footskating. Another disadvantage is that motion graphs are, in general, not able to generate motions that require tight coupling to the environment unless exactly those motions are in the database.

Several techniques have been developed that are able to automatically identify natural transitions between motion spaces. Shin and Oh [SO06] present fat graphs. In these fat graphs, blend spaces are constructed using edges that start and end at a common pose (or hub) of a motion graph. These blend spaces allow more flexible parameterization than traditional motion graphs. However, in order to transition from one motion to another, the VH must always first move through one of the common poses. Heck and Gleicher [HG07] introduce parameteric motion graphs. A parameteric motion graph encodes an edge as a mapping from a control parameter value that creates a motion primitive in the source motion space to a subspace of the control parameter values in a target motion space that create motion primitives to which a natural transition from the source motion primitive is achieved. Transitions are selected by specifying the current motion space and control parameter values and

the target motion space and desired target control parameters values. If a natural transition satisfying the target control parameters exists, they are achieved precisely. If not, a transition that provides the closest match to the target control parameter values can be selected. This either sacrifices accuracy for naturalness (for example, for a punch motion space with target punch position as a control parameter), or it can sacrifice responsiveness for naturalness if control parameter values are achieved by subsequent transitions (for example for a walk motion space with a direction control parameter).

4.2.1.2. Concatenation using Statistical Methods Li et al. [LWS02] models a motion space as a linear dynamic system (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 4.1.4), a fluent connection is achieved.

4.2.2. Concatenation of Physically Controlled Motion

In physical simulation using controllers, concatenation implies a switch to a different controller. 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]. Predefined transitions between controllers that satisfy this condition can be encoded in 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.

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 state to a state suitable for balancing on the ground. Faloutsos et al. [FvdPT01] facilitates transitions between controllers by describing pre-conditions and post-conditions for each controller. The pre-conditions define the sensor values (see 2.3.1) that lead 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 given sensor values. The pre-conditions for a controller are then determined by what a trained SVM for that controller classifies as successful.

Coros et al. [CBvdP09] show how to create control policies that satisfy longer term goals. The policy selects a near optimal sequence of locomotion controllers given a certain control parameter *value* offline. Each controller executes one locomotion cycle. After each walk cycle, the control policy selects the controller that will achieve a new global state (=world state $\times$ VH state) that maximizes the reward. Rewards are explored for 'trusted' global states (those close to states achieved 'normally' in the controllers), using off-line reinforcement learning, in a similar manner as discussed earlier for motion graphs (4.2.1.1).

4.2.3. Concatenation of Procedural Motion

Zeltzer [Zel82] models the different phases of a procedural walking motion by different procedural models 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 [KW04, HMP02]. The end pose of the previous gesture is then used as the start pose of the current gesture. Other procedural animation systems use interpolation to generate a transition motion primitive between two procedural motions [PG96, KM05].

4.2.4. 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] switch control from kinematics to physics on contact with physical objects in the environment. A transition from physical simulation to motion editing (in their 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 how such a suitable motion primitive 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 occurs that makes the VH fall over. A PD-control system is then started in the pose last set by the motion editing motion primitive. 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 motion primitive, 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, control is returned to motion editing. NaturalMotion's [Nat] Euporia and Endorphin animation systems allow transitions between motion editing and physical simulation. Selecting a suitable motion primitive to play after physical simulation is left to the motion author.

Rather than using recovery controllers, Zordan et al. [ZMCF05] search a motion capture database to find a suitable recovery 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 5.1.3), 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.

4.3. Combination

A VH can execute multiple tasks at the same time that each require motion, possibly on different parts of the body. Rather than explicitly setting up new motion primitives for each combination of motion acting on a separate body part, different motion primitives, possibly created by different animation techniques, can be combined in such a way that a coherent whole body motion results (see Figure 5).

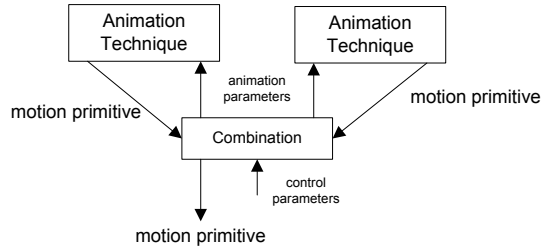

Figure 5: *Different motion primitives, each acting on a subset of the joints of the VH can be combined to form full body motion.*

4.3.1. 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 (as done in [Per95, KM05]). 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 the motion on various joints in the body [HKG06].

Heck et al. [HKG06], aim to combine (splice) one motion primitive on the upper body with one acting on the lower body. Both motion primitives 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 primitives. 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] generalize Heck's splicing method by constructing a time warp between any upper body and lower body motion space that can be used to splice motion primitives of the two spaces. Note that these two splicing methods only enforce coherence of the upper and lower body in the temporal domain.

4.3.2. Combination of Physical Controllers

Physical controllers can be combined by adding up the forces and torques applied by them on each joint (as done in [WH00]). Because an articulated rigid body system models the force transferences through the joints, such a combination automatically provides *physically* coherent whole body motion.

4.3.3. 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 from different procedural models, acting on the same body part can be combined using interpolation (see 4.3.1, [PG96]). Thiebaux et al. [TMMK08] employ specialized blend mechanisms to combine motion primitives generated by different procedural animation techniques.

4.3.4. Combination of Kinematic Motion and Physical Simulation

The requirements of physical integrity and accuracy are often of different importance for different body parts. For example, for a gesturing VH, positional and timing accuracy is primarily important on movement of a gesturing arm or head. At the same time, a physically valid balancing motion of the whole body could be achieved by moving only the lower body, where precise timing is less important. Combining kinematic motion with physical simulation on different body parts allows one to combine the accuracy of motion generated by procedural animation or motion editing with the physical realism of physical simulation.

Oore et al. [OTH02] present a system that mixes physical simulation, acting on the knee and ankle joints, with kinematical upper body motion. 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 inverse and forward dynamics (see Appendix) can be combined in a custom designed 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. Van Welbergen et al. [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.

4.3.5. Combining Procedural Motion and Motion Editing

By augmenting motion editing with procedural motion, expressiveness can be enhanced without requiring a prohibitive amount of motion primitive examples. Some examples: a biomechanical model of eye movement (which is hard to motion capture) can be combined with a motion editing technique for neck and trunk movement [LM08]. Heck [Hec07] employs a biologically and psychologically inspired model for gaze that is layered on top of motion primitives created by motion editing. Shapiro et al. [SKF07] combine lower-body motion capture with arm movement determined by planning-techniques from robotics.

4.4. 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 can be gained using such techniques, by looking at the various aspects of control.

4.4.1. Responsiveness

Responsiveness determines how fast a desired change in the motion plan is achieved. Responsiveness is a major theme in the design of motion graphs, it might take a while to traverse 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 connecting 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 4.2.1.1). Physical simulation has high responsiveness to physical events (for example, being hit by a falling anvil), but lower responsiveness to control 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 state of the VH, but direct reaction to physical events that occur in the world is not build-in.

4.4.2. Precision

Precision is the accuracy with which control parameters (like end effector position or timing constraints) are achieved. 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 4.1.5).

4.4.3. Coverage

Coverage deals with how much of the control parameter space is covered. Motion graphs can suffer from bad coverage. For example: some parts in an environment cannot be reached from certain nodes in a motion graph because no path starting in this nodes will go there. Reitsma and Pollard [RP07] present an algorithm to determine the environment coverage of a given motion graph. Note that the coverage of a motion graph is also greatly influenced by the used search algorithm. Even if a path on the graph is available that satisfies control parameter values, a local search method might not find it. Physical simulation can suffer from bad coverage whenever control parameters values create motion primitives that put the VH on the edge of balance. The coverage of physical simulation can be increased by sacrificing some balancing naturalness (see [WJM06]) or by using better balance methods that require more calculation time (see [dSAP08a]). While most motion editing techniques can cover a wide range of control parameter values, only the control parameters that result in a motion primitive near a example motion primitive will yield natural motion. Procedural motion has good coverage, but again not all control parameter values will provide natural motion.

4.4.4. Expressiveness

We defined expressiveness as the number of control parameters that can be used in the motion plan. Procedural and physical simulation techniques have high expressiveness. The number of parameters that can effectively be used in motion editing is low.

4.4.5. Intuitiveness

Intuitiveness deals with how intuitive the control parameters that can be used in the specification of the motion plan are. All techniques allow the use of control parameters that can set pose constraints. Other control parameters (such as emotion, physical state) can often be mapped to animation parameters. An intuitive set of control parameters might cause conflicts between animation parameters, but an orthogonal set of control parameters is typically not intuitive (see 4.1.4). For example, [BH00] reports having a parameter that sets both the speed *and* the global pose. Therefore, orthogonal control parameters are typically used solely to create small variations on existing motion (see 5.4.2).

4.4.6. Control Enhancement with Multiple Animation Paradigms

By combining and concatenating motion primitives created by different animation techniques, several aspects of control can be enhanced. For example, a concatenation of a motion primitive created by motion editing by one created using physical simulation enhances the responsiveness to physical events (see 4.2.4). Another example is the enhancement of expressiveness by combining procedural motion and motion editing (see 4.3.5).

5. Naturalness

For many animation systems, plausibility or naturalness rather than full realism is acceptable. We define naturalness as perceived realism of VH's movement. Naturalness therefore partly depends on properties of human observation.

Physical realism is one property of natural animation (see also the Appendix), but physical realism alone is not enough for motion to be perceived as natural. Involvement of the whole body is crucial to make an animation natural [CCZB00]. Furthermore, movement should be consistent with static (such as age, gender) properties of the VH that is being animated [GRA*02, RP02]. Variability is a concern if a motion is to be repeated. In this section we will elaborate on these different aspects of naturalness and show how naturalness can be enhanced and evaluated. Naturalness effects related to animation planning, such as the plausibility of the motion with respect to the cognitive and emotional state of the moving VH [GRA*02] are beyond the scope of this survey.

5.1. Physical and Biological Realism

Motion primitives created by physical simulation techniques are physically realistic by design. It is relatively easy to consider muscle strength in these methods. Motion captured animation is also physically realistic, since it originates from real humans moving. However, motion editing might invalidate its physical correctness, introducing artifacts such as foot skate, unnatural balance, or momentum changes in flight. We outline some methods to correct or prevent these artifacts and enhance physical and biological realism.

5.1.1. 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 to filter motion primitives to obtain physically correct ground contact. A friction model is used to make the foot slide when appropriate. Their filter makes use of the fact that no force or torque can be applied at the root of a VH, since that joint is not actuated. Each frame of motion is cast on a physical model of the VH. Then, per frame, the net root forces and torques are 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 3.1.1) 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. During ground contact, 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 3.1.1) 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 rough estimations of 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].

5.1.2. Retaining Physical Correctness in Interpolation

Because of the difficulties and large computation time associated with physical filters, some interpolation techniques deal with physical realism during the interpolation stage, rather than using a post-processing 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 clever selection of the interpolation duration, the net force during flight is equal to gravity. If, during ground contact, the center of mass, the foot positions, knee-swivel angles and all joints angles except the legs are interpolated, rather than directly interpolating joint rotations, the feet will not penetrate the ground, 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 synchronizes the support phases of motion primitives so that they are compatible during the interpolation. 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. Oshita [Osh08] contributes a method to generate transitions between two motion primitives based on their support phase that does not require re-aligning them and can handle a wider range of support phase combinations. It uses Treuille et al.'s method for the connection of motion primitives in which the same foot is moved. Flying motions are connected by aligning their pelvis directions and interpolating some frames of the start of the second motion with some frames of the end of the first. A transition from a motion primitive with single support to one with double support is created by interpolating from some frames of the end of the first motion primitive with the start *pose* of the second motion primitive. Transition between double support motions primitives are created by modifying the lower body of the second motion primitive, so that its feet positions matches the first motion primitive. The upper bodies are then interpolated. As soon as a foot

is lifted in the second motion primitive, the lower body is interpolated with the second motion primitive too.

5.1.3. Improving Physical Correctness using Tracking

A physical tracking controller tracks the joint rotation path specified in a motion primitive. This is done by setting this path as the desired state for the controller. The resulting motion obeys Newtonian physics and allows physical interaction with the environment. Physical tracking recently became a component of some commercial high level animation toolkits [Art08, Hav08].

A tracking PD-controller necessarily has very high PD-gains, which results in stiff reactions to the environment. The PD-gains can be reduced on impact, to decrease such undesired stiffness (as done in [ZH02, WJM06]). More sophisticated controllers use a predictive model that determines joint torques and typically correct small perturbations using a low gain feedback PD-controller (for example: [YCP03, YLvdP07, dSAP08b, MLPP09]).

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 such as [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 [Nat, YCP03]. Zordan and Hodgins [ZH02] use a balance controller specialized for standing with double support contact. 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 world. This spring can 'break' if too much force is exerted on it, causing the VH to lose balance. Yin et al. [YLvdP07] use a custom balance controller for locomotion. Da Silva et al. [dSAP08a] use a linear time varying system that learns (in an offline process) a balance strategy from reference motions, which allows them to track both cyclic and non-cyclic motions. Muico et al. [MLPP09] do not make use of a balance controller, but make use of a more precise torque prediction model instead. Their non-linear predictive model takes contact forces into account and tracks the input motion precisely. It allows the creation of controllers for agile motions, including running and sharply turning.

Using offline learning from a given motion primitive to construct a balance strategy (as in [dSAP08a]) or a forward model (as in [YLvdP07, MLPP09]) enhances the naturalness of the resulting motion generated by a controller. However, by using such offline strategies some control is lost, since they no longer allow the tracking of unknown (for instance: generated by a motion editing technique) motion primitives.

5.1.4. Physical Correctness in 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. 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 and Fiumi [NF06] 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.

Inverse dynamics can be used to analyze the muscle power used in procedural motion. The motion can then be adapted to adhere to muscle strength limits (as done in [LWZB90, KB96]).

5.2. Whole Body Involvement

Procedural gesture animation techniques typically steer the head and the arms 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 5.1.4 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. This method sacrifices some control -exact limb positioning is no longer garanteed- for a more natural full body motion.

Both Chi et al. [CCZB00] and Neff et al. [NK09] 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 4.1.1) are also used to enhance their procedurally generated torso movement. Neff [NK09] show that 'drives', such as hand position and gaze direction can be used to automatically generate torso movement. This is done by defining a mapping between the drives and movement parameters of a procedural torso movement model.

5.3. 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, GRA*02]. It is important to endow VHs with style. Style contributes to naturalness, and, even more importantly, expresses information about the VH such 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 paper we focus solely on the latter. We discus techniques that can either adapt generated motion primitives in real-time or that can determine (adaptations of) control or animation parameters to achieve a certain style in real-time.

5.3.1. Style using Motion Editing

Motion capture also captures the style of the motion captured actor. Ideally, this style could be isolated and be used to replace or define the style of other recorded or generated motions primitives. Here we focus on techniques that aim to do this automatically (in contrast to methods that require the animator to select the style component to transfer, e.g. [SCF06]) and in real-time.

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 values for one control parameter (for example: jumping with different heights). A motion in the style of a new subject is created from a single 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.

5.3.2. Style in Physical/Procedural Simulation

Procedural animation applies style by mapping 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 to gesture animation parameters in a procedural animation system. Neff and Fiume [NF05] model style using a *Character Sketch*. Such a sketch defines modifications to be made to control parameters, can be designed to automatically insert new actions to an animation script and provides a default stance.

5.4. Variability

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

5.4.1. 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 the amplitude of the movements of the body (see Section 2.3.3), 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.

5.4.2. Generating Variability using Statistical Models

Statistical methods that capture orthogonal components of motion (such as [BH00, EMMT04]) also capture the relation between joint movements. 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. Lau et al. [LBJK09] learn a motion space for the specific purpose of generating spatial and temporal variations of similar motion primitives, using a Dynamic Bayesian Network.

5.4.3. 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, biological 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.

5.5. Evaluation of Naturalness

Measuring naturalness is a daunting task. It depends both on the properties of motion and on properties of human observation. Often, some control can be sacrificed to gain more naturalness, or some naturalness can be sacrificed to gain some needed control. We discus how this naturalness-control trade-off can be quantified, provide some motion invariants and metrics that can be used to measure certain aspects of naturalness and discus the setup of user tests that can measure naturalness as a whole.

5.5.1. Exploring the Naturalness-Coverage Trade-off

The naturalness of motion primitives created from the same motion space can vary with the control parameters that were used to create them. The relation between the size of the parameter space (coverage, see 4.4.3) and naturalness can be explored by having subjects directly set and evaluate parameter values, as done in [BSH99]. Such an evaluation provides direct insight on the naturalness cost of a certain parameterization, or the control lost (specifically: reduction of coverage) if a certain level of naturalness is enforced. Clearly, having the subject determine the natural control parameter set is only feasible with a limited set of parameters.

5.5.2. Comparing with Motion Invariants

Some comparisons have been made by comparing motion invariants (see 2.3.3) of recorded motion with those of generated motion. 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 invariants such as the bell shaped velocity profile, minimum jerk, Fitts' law and the two-third power law are modeled in the generated motion. So far such comparisons have been solely qualitative and were applied only in arm gesture domains; graphs of invariants in recorded motion were put side-to-side with graphs of generated motion (see [GLM01, KW04]).

5.5.3. Automatic Evaluation of Naturalness

Intuitively, physical correctness can be measured directly from the animation. Reitsma and Pollard [RP03] evaluate physical correctness 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 physical 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.

5.5.4. User Evaluation

One can (semi-)automaticly evaluate certain naturalness properties of motion using automatic testing or motion invariant checking. However, most evaluation metrics check for a single naturalness artifact that only occurs within a specific animation technique. They can therefore not be used to compare different techniques. For example: it does not make much sense to evaluate the naturalness of physically simulated motion using a foot gliding metric, or to measure the naturalness of a procedural model that is specifically designed keeping a certain motion invariant in mind for adherence to that same motion invariant. Most naturalness metrics do not take human observation properties into account. User evaluation is invaluable for measuring naturalness as a whole and for providing between-technique naturalness comparisons.

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 (see [HWBO95, EVMT04, CHK07, vBE09, JvW09]).

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. Methods from Signal Detection Theory [MC04] provide a bias-independent sensitivity metric d' that can be compared among different test setups, observers and motions. This metric indicates how well two motions can be discriminated. The d' found by comparing motion captured motion with the generated motion is used as a naturalness measure for the model based motion. This approach is followed in [RP03, CHK07, RAP08, JvW09]. We refer the interested reader to [JvW09] for an overview of test paradigms for the evaluation of naturalness of animation that can be used with Signal Detection Theory and their advantages and disadvantages.

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]. Unnaturally 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. Discussion

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.

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 showed 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. In Figure 6 we provide an indication of the control and naturalness of the different hybrid systems discussed in this paper. Note that the control provided by a hybrid system is typically not better than the best control of the two techniques it combines (in theory this is possible if the two techniques that are good in non-overlapping control aspects). Similarly, the naturalness of a hybrid system is typically not greater than the naturalness of its most natural technique. The intersection of two dotted lines starting in an animation technique in Figure 6 indicates this best control and naturalness. Theoretically, very good naturalness and control could be achieved by combining techniques with high naturalness with those with great control. However, since techniques with great control also have low naturalness, it is hard to combine such techniques in a consistent manner.

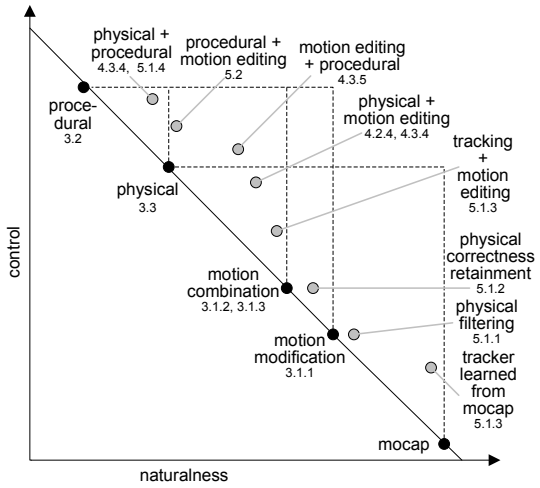

Figure 6: *Control and naturalness of methods used in this paper. Black dots indicate the animation techniques discussed in Section 3. Grey dots indicate hybrid methods.*

We have shown different methods that execute motion plans generated by some higher level planning process. Such plans could be constructed by higher-level behavior generation mechanisms. In embodied conversational agent applications, the animation plan is typically constructed from intentions (like greet the partner, indicate a location) and states (emotional, physical, cognitive). Typically the animation plan is embedded in a multi-modal behavior script, describing the synchronization between speech and gesture. Recent efforts aim to unify the multi-modal behavior scripts designed by different research groups into the Behavior Markup Language [KKM*06]. Another domain of applicability of a flexible motion generation system is crowd simulation. Here physical characteristics of the environment (obstacles, quality of the ground) as well as physical and social behavior rules (e.g. strategy to avoid collision with objects and other people) serve as basis for generating motion plans.

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). We would also like to thank the reviewers whose detailed comments helped improving the structure and content of this paper.

References

[ABC96] AMAYA K., BRUDERLIN A., CALVERT T.: Emotion from motion. In *Graphics Interface* (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 *Symposium on Computer Animation* (2007), Eurographics Association, pp. 239–247.

[AdSP07] ABE Y., DA SILVA M., POPOVIĆ J.: Multiobjective control with frictional contacts. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 249–258.

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

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

[AP06] ABE Y., POPOVIĆ J.: Interactive animation of dynamic manipulation. In *Symposium on Computer animation* (2006), Eurographics Association, pp. 195–204.

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

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

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. In *Computer graphics and interactive techniques* (2007), ACM, pp. 281–288.

[BH00] BRAND M., HERTZMANN A.: Style machines. In *SIGGRAPH* (2000), ACM, pp. 183–192.

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation* (1996), Springer-Verlag, pp. 183–197.

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

[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* (1999).

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

[CB04] CALLENNEC B. L., BOULIC R.: Interactive motion deformation with prioritized constraints. In *Symposium on Computer Animation* (2004), Eurographics Association, pp. 163–171.

[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.

[CBvdP09] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust task-based control policies for physics-based characters. In *SIGGRAPH Asia* (2009), ACM, pp. 1–9.

[CCZB00] CHI D. M., COSTA M., ZHAO L., BADLER N. I.: The EMOTE model for effort and shape. In *SIGGRAPH* (2000), ACM, 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 (2007), 206–216.

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

[dSAP08a] DA SILVA M., ABE Y., POPOVIĆ J.: Interactive simulation of stylized human locomotion. In *SIGGRAPH* (2008), ACM, pp. 1–10.

[dSAP08b] DA SILVA M., ABE Y., POPOVIĆ J.: Simulation of human motion data using short-horizon model-predictive control. *Comput. Graph. Forum 27*, 2 (2008), 371–380.

[EMMT04] EGGES A., MOLET T., MAGNENAT-THALMANN N.: Personalised real-time idle motion synthesis. In *Pacific Graphics* (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* (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* (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.

[Fel86] FELDMAN A. G.: Once more on the equilibrium-point hypothesis (lambda model) for motor control. *Journal of Motor Behaviour 18*, 1 (1986), 17–54.

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

[Fit54] FITTS P. M.: The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 6 (1954), 381–391.

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

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *SIGGRAPH* (2001), ACM, pp. 251–260.

[GBT06] GLARDON P., BOULIC R., THALMANN D.: Robust on-line adaptive footplant detection and enforcement for locomotion. *Vis. Comput. 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* (2004), vol. 2915 of *Lecture Notes in Computer Science*, pp. 1–9.

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *Symposium on Interactive 3D graphics* (1997), ACM, pp. 139–148.

[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* (2004), ACM, pp. 522–531.

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

[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 *Symposium on Interactive 3D graphics* (2003), ACM, pp. 181–188.

[Hav08] HAVOK: Havok behavior and 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, 2007.

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *Symposium on Interactive 3D graphics and games* (2007), ACM, pp. 129–136.

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

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

[HMP02] HARTMANN B., MANCINI M., PELACHAUD C.: Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In *Computer Animation* (2002), IEEE Computer Society, pp. 111–119.

[HMP06] HARTMANN B., MANCINI M., PELACHAUD C.: *Gesture in Human-Computer Interaction and Simulation*, vol. 3881 of *Lecture Notes in Computer Science*. Springer, 2006, ch. Implementing Expressive Gesture Synthesis for Embodied Conversational Agents, pp. 188–199.

[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* (1998), 780–784.

[HWBO95] HODGINS J., WOOTEN W., BROGAN D., O'BRIEN J.: Animating human athletics. In *Computer graphics and Interactive Techniques* (1995), ACM, pp. 71–78.

[IAF06] IKEMOTO L., ARIKAN O., FORSYTH D. A.: Knowing when to put your foot down. In *Symposium on Interactive 3D graphics and games* (2006), ACM, pp. 49–53.

[IAF07] IKEMOTO L., ARIKAN O., FORSYTH D.: Quick transitions with cached multi-way blends. In *Symposium on Interactive 3D graphics and games* (2007), ACM, pp. 145–151.

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

[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.

[JvW09] JANSEN S., VAN WELBERGEN H.: Methodologies for the user evaluation of the motion of virtual humans. In *Intelligent Virtual Agents* (Berlin, 2009), vol. 5573 of *Lecture Notes in Computer Science*, Springer, pp. 125–131.

[Kaw99] KAWATO M.: Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology 9*, 6 (1999), 718–727.

[KB96] KO H.-S., BADLER N. I.: Animating human locomotion with inverse dynamics. *Comput. Graph. Appl. 16*, 2 (1996), 50–59.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *Symposium on Computer Animation* (2003), Eurographics Association, pp. 214–224.

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

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

[KKM*06] KOPP S., KRENN B., MARSELLA S., MARSHALL A. N., PELACHAUD C., PIRKER H., THORISSON K. R., VILHJÁLMSSON H. H.: Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent Virtual Agents* (2006), vol. 4133 of *Lecture Notes in Computer Science*, Springer, pp. 205–217.

[KM05] KALLMANN M., MARSELLA S.: Hierarchical motion controllers for real-time autonomous virtual humans. In *Interactive Virtual Agents* (2005), vol. 3661 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 253–265.

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

[KSG02] KOVAR L., SCHREINER J., GLEICHER M.: Footskate cleanup for motion capture editing. In *Symposium on Computer Animation* (2002), ACM, pp. 97–104.

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

[LBJK09] LAU M., BAR-JOSEPH Z., KUFFNER J.: Modeling spatial and temporal variation in motion data. In *SIGGRAPH Asia* (New York, NY, USA, 2009), ACM, pp. 1–10.

[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. *ACM Trans. on Graph. 21*, 3 (2002), 491–500.

[Lee08] LEE J.: Representing rotations and orientations in geometric computing. *Comput. Graph. Appl. 28*, 2 (2008), 75–83.

[LL06] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. *Graph. Models 68*, 2 (2006), 158–174.

[LLP09] LEE Y., LEE S., POPOVIĆ Z.: Compact character controllers. *ACM Trans. Graph. 28*, 5 (2009), 1–8.

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

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

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH* (1999), ACM, pp. 39–48.

[LWS02] LI Y., WANG T., SHUM H.-Y.: Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH* (2002), ACM, pp. 465–472.

[LWZB90] LEE P., WEI S., ZHAO J., BADLER N. I.: Strength guided motion. In *SIGGRAPH* (1990), ACM, pp. 253–262.

[LZ08] LO W.-Y., ZWICKER M.: Real-time planning for parameterized human motion. In *Symposium on Computer Animation* (2008), ACM, pp. 29–38.

[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, 2004.

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

[MC04] MACMILLAN N. A., CREELMAN D. C.: *Detection Theory: A User's Guide*, 2 ed. Lawrence Erlbaum, 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.

[MLPP09] MUICO U., LEE Y., POPOVIĆ J., POPOVIĆ Z.: Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph. 28*, 3 (2009), 1–9.

[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.

[Nat] NATURALMOTION: Euphoria and Endorphin. 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.

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *Symposium on Computer Animation* (2002), ACM, pp. 81–88.

[NF05] NEFF M., FIUME E.: AER: aesthetic exploration and refinement for expressive character animation. In *Symposium on Computer Animation* (2005), ACM, pp. 161–170.

[NF06] NEFF M., FIUME E.: Methods for exploring expressive stance. *Graph. Models 68*, 2 (2006), 133–157.

[NK09] NEFF M., KIM Y.: Interactive editing of motion style using drives and correlations. In *Symposium on Computer Animation* (2009), ACM, pp. 103–112.

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

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

[Osh08] OSHITA M.: Smart motion synthesis. *Comput. Graph. Forum 27*, 7 (2008), 1909–1918.

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

[Per95] PERLIN K.: Real time responsive animation with personality. *IEEE Trans. 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* (1996), ACM, pp. 205–216.

[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).

[RAP08] REITSMA P. S. A., ANDREWS J., POLLARD N. S.: Effect of character animacy and preparatory motion on perceptual magnitude of errors in ballistic motion. *Comput. Graph. Forum 27*, 2 (2008), 201–210.

[RCB98] ROSE III C. F., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *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, 1996.

[RP02] RUTTKAY Z. M., PELACHAUD C.: Excercises of style for virtual humans. In *Proceedings of Animating Expressive Characters for Social Interaction Symposium* (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 Trans. Graph. 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 Trans. Graph. 24*, 3 (2005), 1090–1097.

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

[SC92] STEWART A. J., CREMER J. F.: Beyond keyframing: an algorithmic approach to animation. In *Graphics Interface* (1992), Morgan Kaufmann Publishers Inc., pp. 273–281.

[SCF06] SHAPIRO A., CAO Y., FALOUTSOS P.: Style components. In *Graphics Interface* (2006), Canadian Information Processing Society, pp. 33–39.

[Sch75] SCHMIDT R.: A schema theory of discrete motor skill learning. *Psychological Review*, 82 (1975), 225–260.

[SFNTH05] SHAPIRO A., FALOUTSOS P., NG-THOW-HING V.: Dynamic animation and control environment. In *Graphics Interface* (2005), Canadian Human-Computer Communications Society, pp. 61–70.

[SH05] SAFONOVA A., HODGINS J. K.: Analyzing the physical correctness of interpolated human motion. In *Symposium on Computer Animation* (2005), ACM, pp. 171–180.

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

[SKF07] SHAPIRO A., KALLMANN M., FALOUTSOS P.: Interactive motion correction and object manipulation. In *Symposium on Interactive 3D graphics and games* (2007), ACM, pp. 137–144.

[SKG03] SHIN H. J., KOVAR L., GLEICHER M.: Physical touch-up of human motions. In *Pacific Graphics* (2003), IEEE Computer Society, pp. 194–203.

[SO06] SHIN H. J., OH H. S.: Fat graphs: constructing an interactive character with continuous controls. In *Symposium on Computer animation* (2006), Eurographics Association, pp. 291–298.

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

[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. *Graph. Models 62*, 5 (2000), 353–388.

[THB07] TORRESANI L., HACKNEY P., BREGLER C.: Learning motion style synthesis from perceptual observations. *Advances in Neural Information Processing Systems*, 19 (2007), 1393–1400.

[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 *Autonomous Agents and Multiagent Systems* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 151–158.

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

[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*, 2 (1989), 89–101.

[vBE09] VAN BASTEN B. J. H., EGGES A.: Evaluating distance metrics for animation blending. In *Foundations of Digital Games* (2009), ACM, pp. 199–206.

[vBPE10] VAN BASTEN B. J. H., PIETERS P. W. A. M., EGGES A.: The step space: example-based footprint-driven motion synthesis. *Computer Animation and Virtual Worlds 21*, 3 (2010).

[vBSE10] VAN BASTEN B. J. H., STÜVEL S. A., EGGES A.: Exact parameterization for precise foot placement. In *Poster proceedings of Symposium on Computer animation* (2010).

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

[vWZR09] VAN WELBERGEN H., ZWIERS J., RUTTKAY Z. M.: Real-time animation using a mix of physical simulation and kinematics. *Journal of Graphics, GPU, and Game Tools 14*, 4 (2009).

[WH97a] WILEY D. J., HAHN J. K.: Interpolation synthesis of articulated figure motion. *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 *SIGGRAPH Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH* (1997), ACM, 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, 2004.

[WJM06] WROTEK P., JENKINS O. C., MCGUIRE M.: Dynamo: dynamic, data-driven character control with adjustable balance. In *Proceedings of the SIGGRAPH symposium on Videogames* (2006), ACM, pp. 61–70.

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

[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), 81.

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

[YLvdP07] YIN K., LOKEN K., VAN DE PANNE M.: SIMBICON: simple biped locomotion control. *ACM Trans. Graph. 26*, 3 (2007), 105.

[Zel82] ZELTZER D.: Motor control techniques for figure animation. *Comput. Graph. Appl. 2*, 9 (1982), 53–59.

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

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *Symposium on Computer Animation* (2002), ACM, pp. 89–96.

[ZMCF05] ZORDAN V. B., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. In *SIGGRAPH* (2005), ACM, 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* (2007), ACM, pp. 9–14.

[ZS09] ZHAO L., SAFONOVA A.: Achieving good connectivity in motion graphs. *Graph. Models 71*, 4 (2009), 139–152.

Appendix: Kinematics and Physics

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. *Forward Kinematics* (FK) is the process of determining the position and/or rotation of an *end effector* (the joint at the end of a chain of joints) **s** given the rotations and translations **q** of all joints in the chain. *Inverse Kinematics* (IK) specifies the inverse problem: finding **q**, given **s**. Often this problem of finding joint rotations is underspecified, that is, there are multiple combinations of joint DoF values **q** that put the end effector in the right location. Several techniques exist to solve this problem. The IKAN toolkit [TGB00] 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 pose in which the end effector is already close to the goal, a natural pose will often be achieved. Boulic and Kulpa [BK07] provide an overview of commonly used numerical methods that solve the IK-problem and discus the trade-offs made in naturalness and calculations speed in these different methods.

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. 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. *Forward dynamics* is the animation process that moves a VH when the torques on its joints and its contact forces and impulses with the environment (through friction and collision) are provided. *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 strength limits. Several software toolkits can be used for forward dynamcics and/or impact and friction handling. 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 constraint satisfaction 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.