EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

Layered Performance Animation with Correlation Maps

Michael Neff[†1], Irene Albrecht[‡2] and Hans-Peter Seidel[2]

[1]University of California, Davis, U.S.A.
[2]MPI Informatik, Saarbrücken, Germany

Abstract

Performance has a spontaneity and "aliveness" that can be difficult to capture in more methodical animation processes such as keyframing. Access to performance animation has traditionally been limited to either low degree of freedom characters or required expensive hardware. We present a performance-based animation system for humanoid characters that requires no special hardware, relying only on mouse and keyboard input. We deal with the problem of controlling such a high degree of freedom model with low degree of freedom input through the use of correlation maps *which employ 2D mouse input to modify a set of expressively relevant character parameters. Control can be continuously varied by rapidly switching between these maps. We present flexible techniques for varying and combining these maps and a simple process for defining them. The tool is highly configurable, presenting suitable defaults for novices and supporting a high degree of customization and control for experts. Animation can be recorded on a single pass, or multiple layers can be used to increase detail. Results from a user study indicate that novices are able to produce reasonable animations within their first hour of using the system. We also show more complicated results for walking and a standing character that gestures and dances.*

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

1. Introduction

In traditional animation, two main approaches are used: keyframing and straight-ahead animation [TJ81]. With keyframing, poses are set at particular points in time and in-between poses are added later to create continuous motion. In straight-ahead animation, no keyframes are used, rather the final frames are simply drawn in sequence. Keyframing is considered preferable for planned, controlled motion, while straight-ahead animation often produces more free, spontaneous and exciting movement. Most computational animation tools are based on the key-frame approach and the research community has paid comparably less attention to providing a computational equivalent to straight-ahead animation.

Performance animation provides a good computational parallel to straight-ahead animation, yet it is difficult to build such systems, particularly when the aim is to control a complex character with limited input degrees of freedom (DOFs). This paper presents a performance based animation system in which an animator uses a mouse to interactively control the movements of a 3D humanoid character. We tackle this problem by using a meaningful, reduced DOF parameterization of character pose and employing *correlation maps* to link several of these DOFs to a single input parameter. Multiple correlation maps may be active at one time. Correlation maps encode two kinds of relationships: correlation between movements in input space and movements of the character, and correlations between various character pose parameters. For example, a downward movement in input space might cause a character to hunch over (correlation between input space and character space), and the rotation of the spine may also cause the collar bones to rotate and knees to bend (correlation within character space). The system allows correlation maps to be easily built and combined so that limited 2D mouse input can be used to control complex character movement. Correlation maps are normally defined

† neff@cs.ucdavis.edu
‡ {albrecht|hpseidel}@mpi-inf.mpg.de

only over a subset of the pose parameters. The parameterization consisting of balance adjustments, IK handles, spine curves, joint DOFs, etc. (Table 2) is a key component of the efficacy of the system.

One of the key observations of this work is that correlations in movement often hold for only a short duration. For this reason, the system allows users to rapidly switch between correlation maps while interacting, essentially changing the character rig on the fly. It is this ability to rapidly change control that allows a low DOF input device to control a high DOF character, providing for a variety of movement changes during a single interaction. As well, the tool is highly configurable, allowing the user to define new maps, specify arbitrary combinations of maps and define how she wishes to invoke the maps during interaction.

Animations are normally recorded in multiple layers, where the main structure of the motion is specified during a first interaction and refinements are added during subsequent passes. An innovative feature of the system is the introduction of *overlays*, which allow the animator to define correlations across layers, avoiding the need to synchronize performance passes. An overlay can be viewed as a dynamic character filter that will adjust the warps it makes to character motion based on the already recorded data.

All system input is via a mouse and keyboard, providing near universal accessibility. A mouse is used here not because it is necessarily the ideal input device for animation, but because it offers a particularly difficult test case, with only 2-DOF of input. Techniques that work with such an impoverished input device will hopefully extend gracefully to higher DOF input devices such as game controllers that provide additional DOFs. The main movements targeted include a range of standing motions including gesturing, night-club dancing, and walking. The system is particularly effective for rapidly exploring the movement space and improvising spontaneous animations.

To evaluate the system, we performed a novice user study that indicated that people with no previous animation experience were able to create two reasonable gesture animations of short video sequences within their first hour of using the system, including training time. The quality of these animations ranged from rough to quite good, which seems reasonable for such short exposure to a new instrument (cf. a piano).

Key contributions of this work include:

- The identification of a good set of movement parameters for character control.
- An effective interface design for real-time interaction and a discussion of the trade offs involved in the different types of mappings.
- The introduction of overlays for modifying style of recorded performance motion.
- A design that both provides appropriate defaults for novices and allows experts to extend the power of the tool.

2. Background

Researchers have developed a number of interesting, useful and fun tools for performance animation. In comparing these approaches, it is useful to consider the range of movement that can be modeled, the number of degrees of freedom of the character that is controlled, and the input device used. Our current prototype maintains the character in a standing or walking position, although this is not a general restriction of the approach. Compared to other approaches, our system generally features a larger range of movement on a more complex humanoid model while using minimal input DOFs.

Perlin [Per95] introduced a computer puppet system in which the animator invokes predefined actions that are defined in script files using sine and noise based interpolation functions and can be smoothly combined at run time. The system offers a flexible range of movement, but the control is at a higher level than in our approach.

Laszlo et al. [LvdPF00] and van de Panne [vdP01] use mouse input to directly control simple two dimensional characters that are physically simulated. The use of physical simulation amplifies the two DOF mouse input because the speed of input will change the motion due to momentum. [LNS05] extend this work with the use of predictive look-aheads that are displayed in the interaction space, making it easier to predict the outcome of control input. The animator can essentially select the desired position of the character, rather than needing to generate an abstract input curve.

Oore et al. [OTH02a, OTH02b] present a performance animation system based on a novel interface consisting of two six DOF trackers embedded in cylinders. Like our system, they use a layered approach to animation, but map layers to particular body regions with a normal performance ordering. Oore et al. opt for a literal input mapping, aligning the cylinders with the character's bones, while we employ a more complex set of user definable mappings that can be rapidly switched between at runtime.

In the work of Dontcheva et al. [DYP03], users control a performance animation system by moving props in space which are mapped to character DOFs. Mappings are either defined explicitly or inferred from similarities between prop and character movement. By layering several passes of acting, the user adds detail to the animation. The differences between this system and our system stem mainly from differences in character parameterization: Dontcheva et al. map user input directly to rotational and translational character DOFs, while we use more abstract correlations. This allows us to animate complex characters. We provide tools that allow an animator to explicitly define mappings, but we do not support inference. Our system is highly configurable and the

user can switch between different mappings within one animation pass.

Motion doodles [TBvdP04] use a scriptive interface to control a two DOF character that can move through a three dimensional environment. The path of the character can be drawn and cursive gestures are used to specify different character movements. They use a preset mapping and deal with a different range of movements than our system.

Our system has some basic similarity to the blend shape method (e.g. [JTDP03]) commonly used in facial animation. While both methods blend input to create output, blend shapes work with full versions of the final target output (i.e. a mesh) whereas our system operates on a subset of parameters, specified over multiple layers, that are then used to calculate the final pose in a separate algorithmic stage. Blend shapes have no inherent mapping from input space to parameter value(s), while this is fundamental to our approach.

Spatial keyframing [IMH05] embeds whole body poses at particular locations in input space. The system continuously interpolates between these poses as the pointer is moved. This provides simple, intuitive control, but the range of possible movement is limited. By tying spatial location to (interpolated) poses, spatial keyframing uses absolute mappings. In contrast, our system also offers both relative mappings, where the input mouse location is interpreted relative to the current character pose, and control over any subset of character parameters. This allows for intuitive layering of animation passes and results in greater flexibility through the reuse of partial body correlations that can be found in multiple movements, versus more restrictive full body poses.

Numerous "computer puppetry" systems have been developed that use motion capture technology to drive the movement of a character in real time (e.g. [Stu98]). These systems use expensive, high DOF input systems whereas we are interested in employing the cheap, low DOF input technology people already have access to. In addition, when these technologies use literal mappings, which is the norm, they require the animator to be able to actually make the movements he wishes to animate. We are interested in using simpler mappings to allow an animator to create movements he may or may not be personally capable of performing.

Other approaches are based on replaying motion capture data. FootSee [YP03] uses a floor pressure sensor as input, while marker-based [CH05] and makerless [RSH*05] motion capture systems have been developed that measure a performers movement and then reconstruct it using prerecorded motion capture data. Such approaches have the potential of creating high quality movement, but rely on the user's ability to perform the desired movements and potentially limit the user's control through reliance on prerecorded motion clips.

Terra and Metoyer [TM04] present a system in which the desired key values are defined separately, but the user can interactively define the timing for translatory aspects (body translation, IK handles) of the motion. In our work, the angles and timing are both defined through performance and both rotational and translatory data are controlled.

Yamane and Nakamura [YN03] present a fast IK system that allows arbitrary points on a character to be pinned or dragged. While not a performance system per se, it allows for rapid pose manipulation while authoring animations. Neff and Fiume [NF06] developed a system for modeling expressive posture based on the arts literature that has also been applied to gesture modeling [NKAS07]. We find their parameterization of character pose provides useful handles for interactively controlling a character and use it here (see Table 2 and Section 4).

3. Basic Interaction and Workflow

A user creates animation interactively by moving his mouse pointer in the screen space of the character. The mouse movements are converted to changes in body pose through *correlation maps*. Each correlation map defines how mouse movement in a particular dimension (x or y) varies one or more parameters of a character's pose. IK and balance algorithms convert these parameters into a final pose. Often, multiple correlation maps will be active at any given time. The *interaction palette* defines what correlation maps are currently active and allows different active sets to be mapped to three different mouse buttons. Switching mouse buttons during a drag allows the user to very quickly change the correlation maps he is using without introducing any discontinuities in the motion. The interaction palette allows the user to specify up to ten correlation maps that can be triggered at the same time, define which mouse button activates which subset of maps, and alter other parameters such as map gain and the use of relative or absolute mappings. More details are given in Section 5 and an example is shown in Figure 1. Keyboard hotkeys are used to switch between correlation maps and even switch whole palettes and mouse mappings.

3.1. Correlation Map Definition

Formally, a correlation map consists of a set of correlation entries where each entry is a linear map from one dimensional mouse input (either x or y) to a scalar parameter that is used in defining the state of the skeleton. A correlation entry is defined by four values per dimension: two mouse coordinates in either the x or y dimension, m_A and m_B, and two character parameter values, p_A and p_B, corresponding to the input mouse values. A simple linear function transforms an input value, m, to a character parameter value, p:

$$p = f(m) = p_A + (p_B - p_A)\frac{m - m_A}{m_B - m_A} \qquad (1)$$

Correlation entries are quick to specify and simple to invert. The latter property is important for implementing relative mappings as discussed below.

Correlation Map	Mouse DOF	Description
RHand	XY	Position constraint on right hand.
Two Hands	XY	Position of left (or right) hand and other hand mirrors it.
LHand	XY	Position constraint on left hand.
Two Hand Vert	Y	Vertical movement of the two hands.
Twist	X	Full body rotation.
Twist	X	Second copy (allows different scale).
Crunch	Y	Downward C bend of spine.
Beauty Line	X	S-curve through body; coronal plane.
Lean	X	Sideways lean.
Shoulders Vert	Y	Up and down movement of the collar bones.

Table 1: *A sample palette from our user study.*

3.2. Defining Correlation Maps

A number of correlation maps have been defined for the system. These provide a generic, flexible range of input useful for gesturing, night club-style dancing and walking. Different sets of these maps have been defined on different palettes, which the user can freely switch between using hotkeys or GUI menus. The user can also change which correlation maps are included on any particular palette. An example palette designed for gesturing and provided to novice users in our test scenario is detailed in Table 1 and will be discussed in Section 5.1.1.

While flexible, the set of pre-built correlation maps may not meet all of a user's needs. A second interface, named the *mapping definition palette*, is provided to allow experienced users to interactively define new correlation maps. The interface allows all the low-level body parameters to be controlled through a GUI. To define a correlation map, the user first decides on the set of pose parameters he wishes to control. He then places a marker in input space to define the m_A and m_B input values and uses the GUI to define the correlated pose parameters, which can be previewed on the character. A correlation map is thus defined in a few clicks and saved for future use. The effective pose parameterization and realtime updates of the skeleton pose make this a simple task. Previously defined correlation maps can be activated while designing a new map to allow the animator to ensure that the new map combines effectively with previous maps.

Once defined, the new correlation maps can be loaded into the interaction palette and used with all the pre-existing maps. Advanced users will thus move back and forth between the two palettes, first experimenting with control mappings, then extending the set of control mappings, then authoring more animation, etc. The highly configurable nature of the tool allows it to adapt to different users' needs and skill levels.

3.3. Workflow

The animator must first decide on a set of correlation maps to use for the motion sequence. This can be done by either experimenting with the tool or reflecting on the nature of the movement. The animator configures the interaction palette so that it contains the desired maps, they are triggered by his preferred mouse buttons and any keyboard hot switching has been specified. All of this configuration data can be saved. The animator can then rehearse the motion to become familiar with the mappings and hotkey layout.

To record the animator clicks the record button and lays down a base layer of motion. Although not restricted to this, the base layer normally consists of hand movements in space, often combined with posture deformations and possibly head movements. This reflects the definitional nature of hand movement in determining gestures and the fact that posture changes are often correlated with the movement of the hands. Some participants in our study preferred to first specify posture deformation and this remains an option. During any recording phase, the system can play either video or audio in the background to allow the animator to align movements with character text or example video. The animator makes multiple takes of the initial layer until satisfied.

Once specified, the base layer can be replayed and additional movements added on top. These movements may include further arm and posture movement, adjustment of arm swivel and hand depth, (additional) head movement, rotation of forearm and hand angles, adjustments to balance and pelvic twist, etc. Additional movements can be added in two ways. The animator can perform new mouse input, or the animator can invoke overlays - correlation maps which are driven by the recorded mouse input from a previous layer(s). Overlays allow additional body movements to be automatically synchronized with previously recorded movements, avoiding the challenging problem of trying to perform a new motion in synchrony with a previous pass, and will be described in detail in Section 5.

The animator can also reduce the speed of playback to make it easier to combine new features with the timing of previously recorded layers, and to reduce the need for fast mouse movements. The combination of the new layer and the previous motion are updated and displayed in real-time.

Each loop of input is recorded on a separate layer. The animator can turn on or off any of the recorded layers when generating the animation. This represents a more flexible form of undo, allowing the animator to use only the best interaction passes. It also allows him to examine the different components of the animation in isolation by turning on and off layers at will.

Description	# params
Spinal deformations in the coronal and sagittal plane following S and C curves.	1 per dimension
Spinal twist.	1
Right and left hand positions.	3 per hand
Swivel angle of arms.	1 per arm
Forearm rotation.	1 per arm
Hand rotation.	2 per hand
Vertical and horizontal movement of the collar bones.	2
Gaze direction and tilt control for head.	3
Lateral and forward/backward centre of mass shifts.	2
Knee bends.	1 per knee
Pelvic twists.	1
Foot positions	3 per foot

Table 2: *Character Parameterization: This is the reduced set of parameters used to characterize character pose.*

4. Mapping Design

4.1. Character Parameterization

A critical issue in making effective correlation maps is ensuring that the controlled parameters are expressively relevant. Directly controlling the character's DOFs is not ideal both because this requires too many parameters in order to specify a character's pose (48 for our skeleton not including hand shape) and more importantly, individual DOFs do not have clear expressive meanings. At the same time, parameters that are too high level, such as a full character pose, limit the animator's control over the movements that can be expressed in the system. Also worth considering, linear combinations of joint angles will not in general produce certain desirable outputs, such as a particular end effector path in space. We adopted the low-level parameter set specified in [NF06] which has a maximal set of 33 DOFs to define a skeleton pose and provides parameters that are expressively salient, based on research in the arts literature. Most interactions rely heavily on a subset of eleven of these parameters: six DOFs for hand positions, three DOFs for spine configuration and two DOFs for collar bones. The full parameter set is summarized in Table 2. The use of automatic balance adjustment and the availability of balance offset parameters has proved very important in creating lively motions.

4.2. Absolute vs. Relative Mappings

By definition, every correlation map has an absolute embedding in input space. This means that a particular location in input space corresponds to a particular parameter value in character space. If absolute mappings are used, the starting location of a mouse drag defines the initial configuration of the character. While using the system, an animator can switch between absolute and relative mappings. A relative mapping takes the current character position as the starting point for a movement and uses changes in mouse movement to deform the character from there. Formally, an absolute mapping is defined by Equation 1 and a relative mapping is defined as:

$$p = g(m) = f(m - f^{-1}(p_0)) \tag{2}$$

where p_0 is the value of the parameter being controlled at the start of interaction.

	Relative	Absolute
Average	$p_0(t_0)+$ $\frac{1}{k}\sum_{i=0}^{k-1}(p_i(t)-p_i(t_0))$	$\frac{1}{k}\sum_{i=1}^{k} p_i(t))$
Add	$p_0(t)+$ $\sum_{i=1}^{k-1}(p_i(t)-p_i(t_0))$	$\sum_{i=0}^{k-1} p_i(t)$

Table 3: *Blend rules for different inputs controlling the same parameter. Each formula is used to calculate $p(t)$, the cumulative result of the k different requested values for the parameter. $p_i(t)$ represents the i-th requested value for the parameter at time t. p_0 is from the first input layer.*

Absolute mappings require the user to be aware of the location of their mouse in input space. An advantage of absolute mappings is that they blend well with other absolute mappings with similar spatial relationships. Invoking an absolute mapping can cause a jump in character state, which is sometimes useful. For instance if the entire input space corresponds to variations of a severely hunched back, invoking this mapping will instantly hunch a previously erect character. Relative mappings, conversely, work to adjust the character from it's current position and do not require the animator to be aware of their input location. They work well for interacting on top of already recorded motion and are also used as the default for general interaction.

In practice, we blend in absolute mappings when they are activated to avoid motion jumps:

$$p = f'(m) = f(m - c(t)f^{-1}(p_0)) \tag{3}$$

where $c(t)$ linearly transitions from 1 to 0 over a specified blend duration, currently ten frames.

A given parameter value may be varied by multiple inputs. This can occur if the same input is varied on multiple animation layers, or in occasional cases, it is useful to have both the x and y input dimensions of a particular correlation map vary the same character parameter. These various parameter adjustments must be combined. We define two blending rules, one that averages input and one that adds it. These can also be defined in a relative or absolute sense. Table 3 summarizes the four update rules. A useful addition to the system would be allowing a new stretch of input to replace a previous section.

4.3. Correlation Map Design

Correlation maps can be defined at varied levels and there is a trade-off between easy to use, high level control, and

more flexible, lower level control. For instance, it is possible to build high level correlation maps that make it very easy to control specific gestures, such as a shrug, or generate specific movements, like walking, but most of the correlation maps provided in the system are closer to the low-level parameters. Such low-level maps are generic and can easily be combined with each other to control a wide range of movement. Using simple maps and then combining them to effect more complex control also allows the scale of each component to be varied independently (more on scale below). In practice, it is very common to combine multiple correlation maps that are driven by the same input DOF, for instance mapping lateral hand movement, torso twists and balance adjustments to horizontal input.

4.4. Mapping Categories

The input-to-character space mappings fall into three categories: direct spatial, spatially based and abstract. A direct spatial mapping connects an input parameter to a character parameter such that the screen location of the input is the same as the location of the character parameter (e.g. grabbing a hand and moving it about). Our head tracking and wrist position controls come close to this category, but in each case we chose to violate the exact constraint to provide more intuitive control. Hand movement is defined in chest space, and the direct mapping will be violated as the chest rotates. The adjustment of head movement is scaled to make it easier to control.

The second category is spatially based. For instance a mouse move to the left can cause a character to twist to the left, but there is no direct alignment between the mouse pointer and the location of a body part. Most mappings in the system fit into this category as they provide intuitive control and tend to layer well with other spatially based mappings.

In abstract mappings, there is no direct relationship between the spatial movement of the character and the movement of the mouse. Forearm rotation is an example of this and it could be associated with either input dimension. These mappings are rare.

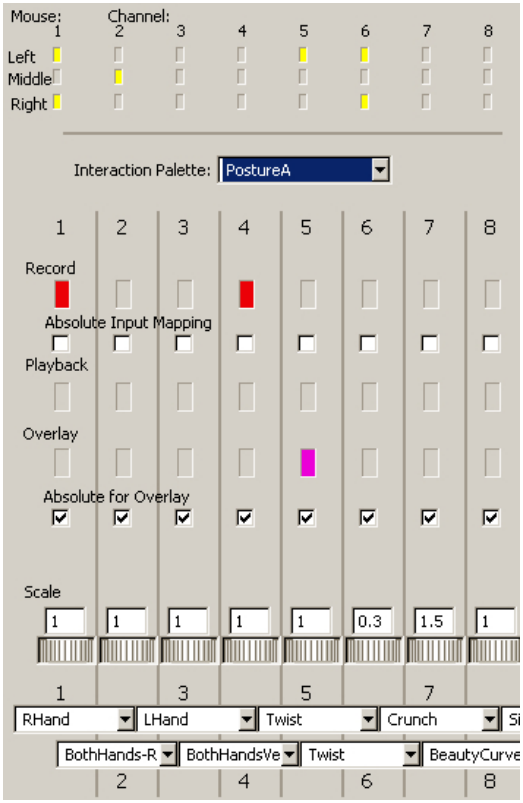

Figure 1: *Correlation maps are arranged in the palette interface in columns. The name of the map is at the bottom of the column. The middle portion of the column controls how the map behaves and the top portion determines which mouse buttons activate the correlation map.*

5. Advanced Input

5.1. Configuring the Animation Interface

5.1.1. Palettes

In the interaction palette, a section of which is shown in Figure 1, the correlation maps are arranged on vertical *channels*, one map per channel. We refer to the set of currently available correlation maps as a *palette*, which contains up to ten channels. Palettes can be predefined and an animator can switch between them as needed during animation. When defining the palette, the entry on any channel can be changed by selecting any of the available correlation maps from a drop down menu. Table 1 shows a sample palette from our user study. It contains a combination of hand position controls and posture controls, which is typical for early interaction passes. Other palettes may contain additional postural deformations, head movement, fine tuning of hand and forearm rotation, arm swivel, etc.

A key feature of the system is that an animator will not normally use a single channel at a time, but combine multiple channels during a single interaction. As well, when switching between input mappings during an animation sequence, the animator will normally switch between sets of channels. For instance, an interaction run might begin with the animator combining right hand movement with a large twist and a slight beauty line. Part way through the animator might switch to two hand movement with a smaller twist and a torso crunch by switching mouse buttons, and then switch to a third mapping or even back to the first. The power of the system comes to a large degree by being able to overload multiple channels during a single interaction pass.

5.1.2. Switching Control Mappings

The palette interface is used by the animator to specify which subset of channels can be invoked with each mouse button during an interaction run. The top three rows of each channel contain checkboxes corresponding to each of the three mouse buttons (left, middle, right), as can be seen in Figure 1. Enabling one of these checkboxes means that when the user drags over the character window with the selected button, the corresponding channel mapping will be invoked. The user can map as many channels as desired to a particular mouse button and associate different channels with different buttons. This is useful for very quickly switching control mappings while interacting with the character. By changing the button depressed, the nature of the control can be changed during a *single* mouse drag. As an example, this can be used to have a character rotate with the first portion of an arm movement, then stop rotating while seamlessly continuing the arm movement, and finally adding a torso crunch to accompany a later arm movement. A set of correlation maps are normally ideal for controlling a very short segment of motion, so switching rapidly between them is essential for providing adequate control of longer sequences.

Keyboard keys can also be used to switch mouse button mappings. By default, the first four palette channels are used for different combinations of hand movements. These are often combined with posture deformations, which can be stored in the latter six channels. The number keys can be used to switch the active hand(s), while maintaining the same posture controls. The key to the left of the “1" switches off all hand control. The keys 1 through 4 enable the corresponding hand mapping. All five of these keys do not change the button set up for the right six channels. This allows the keyboard to be used for rapid switching between hands, without affecting the selected posture channels, while the mouse buttons are used to switch between postural control. Any rig consisting of a palette, mouse maps, scale values (Sec. 5.2.1) and relative or absolute mappings can also be tied to a keyboard hotkey. This allows rapid tool switching during interaction by pressing a single key. Quick mapping changes are what allows two DOF input to be used to control a wide range of high DOF character movement.

5.2. Advanced Channel Options

The behaviour of the individual channels can be adjusted through the interface using the parameters and controls summarized in Table 4. Scale and overlays are explained below.

5.2.1. Movement Scale

The scale value acts as a gain on the correlation map. A scale s is applied directly to the parameter values used to define a correlation map in Eq. 1, such that $p'_A = sp_A$ and $p'_B = sp_B$. Changing the scale alters the amount of movement in input space required for a given pose change. Scale values can also be used to vary the contribution of different channels that are combined together on a given mouse button, allowing for instance, small and large amounts of twist on different buttons. Specifying a negative scale inverts a mapping. A hand movement combined with a normal “crunch" will have the spine curl down in synchrony with a downward arm movement; with a negative crunch, the spine will curl up. Each conveys a different, important expressive intent.

Record	Highlighted when the channel is being recorded. Can turn on to record an overlay.
Absolute	Check box to allow the channel to be switched between absolute and relative mapping.
Overlay	Highlighted when the channel is applied as an overlay. Can turn on and off.
Absolute Overlay	Check box to specify whether the overlay is absolute or relative.
Scale	Adjusts the gain on the channel.
Name	Dropdown displaying the current correlation map and allowing other correlation maps to be selected.

Table 4: *Properties associated with each channel.*

5.2.2. Overlays

One of the most challenging tasks in a layered approach to interactive animation is to perform a new layer so that it synchronizes with the movements on a previous layer. Overlays are a recognition of the need to correlate additional body parameters with previously recorded ones. Any combination of channels can be invoked as an overlay. During playback, an active overlay channel will modify the character’s movement based on the channel’s correlation map, but rather than using interactive data from the user as the input, it will use the mouse input from a specified, previously recorded layer(s). An animator can try an overlay and then decide whether or not to record it. The channel scale can also be adjusted interactively during the playback, allowing it to be varied continuously over different portions of the animation. By default, overlays use an absolute mapping, but relative mappings are also possible.

Overlays allow very different posture deformations to be applied to a given sequence of motion to create different characterizations. Unlike a static default posture, overlays are dynamic, changing based on the previously recorded motion. Such overlays act essentially as dynamic character filters, useful for making broad, stylistic changes.

5.3. Editing Operations

Editing occurs at multiple levels within the system. At the most coarse level, an animator can turn on or off any of the layers that have been recorded. This allows multiple takes

of any interaction to be recorded and the best performances to be selected. Individual channels within a layer can also be disabled after recording. Overlays represent an additional form of editing.

Consistent with the performance paradigm, the most difficult types of edits are those that would require changes to the input mouse data (i.e. the *performance*). If the timing is off, we offer no performance based way of editing this. It is best handled with off-line time warping, or by redoing the performance. If a recorded value needs to be changed, such as a hand position at a certain point in time, the user can record an offset to this data making use of the blending rules defined in Table 3. Motion complexity can also be increased by layering posture deformations in this way. Another method for changing recorded data would be to allow short sections to be re-performed, replacing the previous data and blending at the ends. This "replace and blend" edit has not been implemented, but is a straightforward addition.

6. System

The set of parameters used to define a character's pose includes both world space and joint space values. The underlying animation engine satisfies these constraints using a combination of feedback based balance adjustment, fast inverse kinematics and forward kinematics, based on an implementation of the system described by Neff and Fiume [NF06]. Specifically, an analytic IK routine is used to solve for the angles in the lower body kinematic chain. Balance is adjusted by feeding back error values to adjust the ankle angles and move the character to a desired balance point. Simple two limb IK is used to position the wrists at desired constraint points. Our implementation differs from Neff and Fiume's in two significant ways. We use forward kinematics instead of optimization to control the torso. This means that a character will not adjust his spine to reach a target beyond his grasp, but simply point in the direction of it. This restriction performs very naturally in the movement tests we have performed. The second difference is that wrist constraints are defined in the character's chest frame, rather than the world frame. This allows the character's hands to move with him if twists or other deformations are applied in later layers. The system also implements automatic collar bone adjustment based on the height of the hands.

6.1. Data Recording

Determining which data to store is a significant technical decision. We elect to store the original mouse data for each interaction run, along with related data needed to reconstruct the motion such as which correlation maps were active. This design decision provides maximum flexibility in editing the motion after it has been recorded. A hierarchical structure is used to organize the data. An *interaction manager* is responsible for recording and playing back all data. The interaction manager contains a set of *interaction records*. Each interaction record corresponds to one record/playback loop (one *layer*). These records store the 2D mouse samples for all interactions during the loop, and a sequence of *interaction runs*. A run corresponds to the period from one mouse click to a release. Runs store the correlation maps that are active during the run, offset values for each correlation entry ($f^{-1}(p_0)$) and scale samples for each correlation map at each time step during the run. Storing this data allows any interaction run to be turned on or off and also any correlation map or channel to be turned on or off. Maintaining this data also allows the blending rules to be arbitrarily changed after the data has been recorded.

7. System Evaluation

The system is evaluated with a novice user study and by using the system to create a range of animations.

7.1. User Study

The system is designed foremost for the creation of spontaneous, improvised animation that has the free, chaotic feel of classical straightahead animation. We wished to perform a user study to evaluate the system, but it is difficult to measure free improvisation. We decided instead to use a task that is outside of the sweet spot of our system: the recreation of specific performances from video clips using the original audio. This task is particularly challenging in a performance based system as it requires precise synchrony with the source audio. This task has several advantages, however. It is well defined and easy to explain to subjects. It also decouples creativity from system usability as subjects were not required to be creative. Having everybody animate the same movements also made it easier to compare user results.

In the user study, 11 subjects (4 female, 7 male) with different levels of animation experience recreated brief performances based on video of two actors. In order to compare our approach to animation to traditional keyframing, we asked them to animate the two sequences using both our system and Curious Labs™ Poser®. Half of the participants started with Poser, the others used our system first.

After a brief training session on a given system (5-20 min), every subject created two animations using each tool. Due to time constraints, creation time for each animation was limited to roughly 20 min. Participants were allowed to view the movies as often as they wished in a movie player and also play the audio in both systems (but not the video). Users of our system were not allowed to define new correlation maps, but were asked to use those predefined on some simple palettes such as the sample included previously. These palettes were created before the actors were recorded and were not customized to suit the actors' movements. After the experiment, participants filled out a questionnaire and compared the two systems with regard to several aspects

Figure 2: *Frames from a dance animation that was generated with the system in real-time, in a single pass.*

such as ease of use, level of detail or satisfaction with animations produced.

The evaluation only produced a small number of statistically significant statements due to the small number of participants and the fact that personal preferences seem to play a strong role in the type of tool a user prefers. The statistically relevant results were: Subjects felt that our system encourages creativity more than Poser and strongly preferred it for sketching out the initial performance. Several results fell just below statistical relevance, but users seemed to favour Poser for fine-tuning and for adding detail. There was also a tendency to prefer an interactive as opposed to an offline approach to animation. Participants appeared to believe that our system produces more natural animations and is better suited for modeling style and expressiveness.

Typical beginner problems included using too large amplitudes and a certain jerkiness of mouse movements as seen in the resulting animations. Amplitude can be regulated by scaling the mappings, and trajectories could be easily smoothed. More experienced animators appreciated the possibility to define their own mouse button occupancies and what they considered to be the greater naturalness of the movements from our system, particularly with respect to timing. Worth noting, the well defined task did not inhibit personal preferences in tool use. In our system, some users would lay down an initial layer quite quickly and then try to refine it, while other users would work to define a good mapping and then rehearse the movements multiple times before arriving at a final recording.

Some of the best and worst results from both systems are included in the accompanying video. In every case, we show the results of the same user in each system. The quality of results varies more across users than across systems.

7.2. Sample Animations

The accompanying video also includes some short clips made by an intermediate user of the system who was also involved in the system design. These include gesture animations, a bow, walking and a dance sequence. Several frames from the dance sequence are shown in Figure 2, although the motion can be better evaluated in the accompanying video. In the first gesture animation, hand position and posture deformation were modeled together and forearm rotation was added on a separate layer. A second gesture animation shows how a given hand movement track can be given very different style by overlaying different posture deformations. For the bow, posture deformation was recorded on one layer and arm movement was added on a second layer.

The walking sequence is performed using three pairs of mirrored correlation maps: one pair controls the foot movement and balance adjustments, one the accompanying torso twists, and one the arm movements. In each pair, one member corresponds to the right step and one the left. Walking is controlled by assigning the right step related maps to one mouse button and the left step maps to another. The horizontal directions of each mapping are reversed so that holding down one mouse button and making a forward arc will cause the character to take a right step forward, and holding the other mouse button and making a backwards arc will cause the character to take a left step forward. Thus, by making back and forth arcs in input space, the character can be made to walk forwards or backwards. Ankle bends and toe rolls are not currently supported in the system, which reduce the realism of the foot movement.

All motions in the extended dance sequence were recorded in real-time on one layer. The snippet shows the beginning of a 1.5 minute animation that was recorded in a single take on the fourth try. While not a scientific result, it is worth noting that these types of free movement are especially fun to create in the system. It is also interesting to note how exploratory the process is: interesting movement patterns were often discovered by defining a mapping for one purpose and then interacting with it and finding new possibilities.

8. Discussion and Conclusion

Our system, and likely performance animation in general, performs very well for certain tasks while other tasks are more difficult. It is easiest to use our system when creating free, spontaneous motion, a task that is difficult in approaches like keyframing. Controlling the motion envelope (the timing of transitions) also appears to be easier using direct control. Overlays provide a useful method for making whole scale changes to the style of a motion in a very controlled way. This feature of our system is not an aspect of performance animation in general. Layering reduces the

number of DOFs that need to be controlled in one pass and allows detail to be added.

Precise editing of already recorded values is more difficult in our system and precisely synchronizing movements with prerecorded audio is a challenging performance task. The latter is due to the difficulty of predicting the time at which the alignment point will come, so preview techniques may make this significantly easier. Ghost previews [DYP03], that show the already recorded motion slightly ahead of time are an example. Non-performance based tools, such as those used for processing motion capture, should combine well with performance data. This would allow a performance to be time warped to align with particular phrasing and also allow particular values to be adjusted more easily.

Worthwhile extensions to the system include allowing filtering of mouse input to smooth out unwanted jerkiness and adding a “replace and blend" editing option.

In summary, we have presented a flexible and highly configurable tool for performance animation of complex characters that requires only a mouse and keyboard as input. The system can perform a good range of character movements. It is particularly useful for roughing out motions, creating quick prototypes and and exploring the movement space. As users become more skilled, we feel they will also be able to produce a range of quality results. The system also provides a way to create free, spontaneous animations, such as the dance sequence, that would be difficult to generate in any other way. We believe such systems occupy an important niche in the animation tool range.

References

[CH05] CHAI J., HODGINS J. K.: Performance animation from low-dimensional control signals. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 686–696.

[DYP03] DONTCHEVA M., YNGVE G., POPOVIĆ Z.: Layered acting for character animation. *ACM Transactions on Graphics 22*, 3 (2003), 409–416.

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J.: Spatial keyframing for performance-driven animation. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005* (July 2005), pp. 107–116.

[JTDP03] JOSHI P., TIEN W. C., DESBRUN M., PIGHIN F.: Learning controls for blend shape based realistic facial animation. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Aug. 2003), pp. 187–192.

[LNS05] LASZLO J., NEFF M., SINGH K.: Predictive feedback for interactive control of physics-based characters. In *Eurographics* (2005).

[LvdPF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. *Proceedings of SIGGRAPH 2000* (2000), 201–208.

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

[NKAS07] NEFF M., KIPP M., ALBRECHT I., SEIDEL H.-P.: Gesture modeling and animation based on a probabilistic recreation of speaker style. *ACM Transactions on Graphics* (2007). to appear.

[OTH02a] OORE S., TERZOPOULOS D., HINTON G.: A desktop input device and interface for interactive 3d character animation. *Graphics Interface ’02* (2002), ?

[OTH02b] OORE S., TERZOPOULOS D., HINTON G.: Local physical models for interactive character animation. *Computer Graphics Forum 21*, 3 (2002), 337–346.

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

[RSH*05] REN L., SHAKHNAROVICH G., HODGINS J. K., PFISTER H., VIOLA P.: Learning silhouette features for control of human motion. *ACM Transactions on Graphics 24*, 4 (Oct. 2005), 1303–1331.

[Stu98] STURMAN D. J.: Computer puppetry. *Computer Graphics and Applications* (1998), 38–45.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics 23*, 3 (2004), 424–431.

[TJ81] THOMAS F., JOHNSTON O.: *The Illusion of Life: Disney Animation*. Abbeville Press, New York, 1981.

[TM04] TERRA S. C. L., METOYER R. A.: Performance timing for keyframe animation. In *2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (July 2004), pp. 253–258.

[vdP01] VAN DE PANNE M.: Motion playground, 2001. http://www.motionplayground.com.

[YN03] YAMANE K., NAKAMURA Y.: Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 352–360.

[YP03] YIN K., PAI D. K.: FootSee: an interactive animation system. In *Proceedings of the 2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2003), Eurographics Association, pp. 329–338.