Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006), pp. 1–9
M.-P. Cani, J. O'Brien (Editors)

Spatial Pose Trees: Creating and Editing Motions Using a Hierarchy of Low Dimensional Control Spaces

Igor Mordatch, Patrick Coleman, Karan Singh, and Ravin Balakrishnan[†]

University of Toronto

Abstract

Spatial pose trees are a representation of motion that organizes motion data as a hierarchy of components. Each component has an associated set of target poses, as well as a user–editable 3D control space for creating motion by pose interpolation. This hierarchical partitioning, along with a selective display of poses relevant to a particular motion component, allows users to easily navigate large sets of poses that would be impractical to manage with a single layout. In addition to providing a system for creating motion with this representation, we present techniques for embedding and subsequently editing existing motions, as control curves within a control space. Users can introduce new poses to a control space, either to edit a motion for aesthetic reasons or to meet specific constraints, or to reduce interpolation error. Edited motions introduce new control curves, with associated poses created by interpolation or displacement mapping. This results in a region of control space that produces motions that meet changing constraints. To assist users with system interaction, we introduce tools for simplifying hierarchy construction and 3D control space navigation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

In creating a 3D character animation, animators often take an approach of first constructing a basic motion layout and then adding refinements and subtleties [TJ81, Wil01]. Existing tools do not explicitly facilitate such a workflow, however, as they represent motion as a set of splines that can be organized into a hierarchy of degrees of freedom, without consideration of motion content.

This paper presents *spatial pose trees*, a new motion representation that addresses the motion refinement problem by representing motions as a set of multi–target pose interpolations, which are organized in a hierarchy of low dimensional control spaces. Each control space in the hierarchy represents a motion component, a basic motion that can be blended with others to produce a refined motion. As each child node provides its parent node with a changing pose, the characteristic poses of a repetitive motion can be varied in time as a motion refinement.

For example, if the top level node contains a generic walk cycle, the child nodes can contain the time–varying change of the characteristic poses of that walk cycle. This also allows for motion reuse with parameterized individuality, as child nodes can provide a node containing a generic motion with pose refinements for specific characters. A more general example would be a set of child nodes, each containing motions of particular actions. The parent node would then control the transition among such actions.

Spatial pose trees are more amenable than existing tools to workflows involving refinement and multi–user creation. A user can create a motion sketch in a top level node, with refinements applied in child nodes. The independence of the motion components allows each to be authored by a different user, if desired. Spatial pose trees generalize the one–dimensional time–based parameterization that users manipulate in current motion spline editors. We choose to use three dimensional control spaces, as we wish to eventually use devices that take advantage of human skills in 3D perception and manipulation [GWB04]. 3D interpolation spaces also al-

[†] igor@mordatch.com, {patrick | karan | ravin }@dgp.toronto.edu

low for greater creative freedom, as a given motion can revisit a target pose without duplication.

Spatial pose trees also generalize systems in which users can edit poses in low dimensional control spaces [IMH05, GMHP04]. In these systems, users navigate 2D control spaces to interactively select and blend poses, using a regression model built from the set of examples. In Igarashi et al. [IMH05], users explicitly position poses. Radial basis functions define the control space to pose mapping. As spatial pose trees include user control of pose layout, we incorporate their control space model. The system of Grochow et al. [GMHP04], uses a scaled GPLVM[†] to both lay out a set of poses and define a continuous mapping. As their system focuses on inverse kinematics optimization on a smooth manifold, the automatic layout is preferable. We also support automatic pose sequence layout, but we retain user control over editing this layout. In addition, one of our layout techniques has been chosen specifically to provide visual meaning to the layout of poses from an embedded motion. Our user–editable control space model, hierarchical structure, and tools for navigation and organization allow for better user control as data sets scale in size, a necessity for creating refined character motion.

As spatial pose trees allow users to create and edit motion with a new representation, we consider and address design challenges in the following areas:

Control Space Creation: The spatial pose tree system targets the creation of complex, refined motion. As the number of target poses in a control space increases, data management becomes a time–consuming challenge to the user. This, in part, motivates the use of a hierarchy, as it reduces the complexity of the data presented to the user at any given moment. Furthermore, we provide a tool that assists users with positioning new poses in a control space, while minimizing changes to the pose mapping.

Control Space Navigation: While prefer 3D control spaces, we recognize the difficulty of visualizing and exploring 3D space using 2D input and visualization devices. To simplify this complexity, we provide a tool that assists users in modifying 3D control space layouts for easy visualization and navigation with these devices.

Hierarchy Editing: Large pose data sets for refined motion can be difficult and time–consuming to organize. To assist users with this process, the system provides a tool for automatically arranging a set of poses into a hierarchy, such that each node contains similar poses and different nodes contain dissimilar poses.

Motion Creation: Most animation systems allow animators to set full body keyframe poses at specific times. Non–temporally aligned degree of freedom extrema, a characteristic of both natural motion and high quality character animation, have to be explicitly added by the user. The independence of timing in spatial pose tree motion components allows for this to be created as refinements are added, without changing the representation of the basic motion.

Motion Layout: It can be time–consuming for users to position existing motions in control spaces for subsequent editing. We provide tools to automate this layout.

Motion Editing: Users should be able to edit an existing motion, both for creative exploration and to meet specific constraints. The spatial pose tree system allows users to make sparse edits to a motion that then define new motion variations, while preserving the original. The results of all iterations remain available for reuse.

1.1. Approach

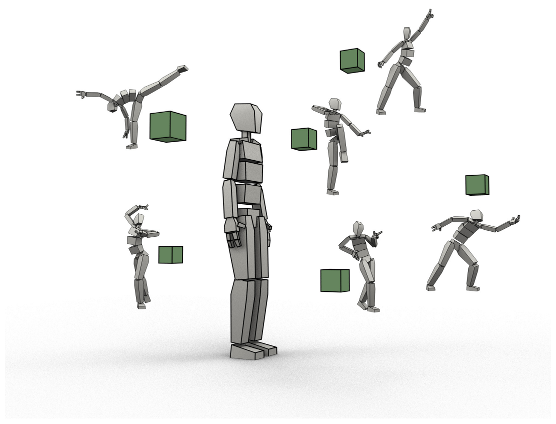
To work with the system, users create or import a set of target poses (Figure 1a) for each node. For each pose, users then position an associated control target—a cube icon—in the 3D control space of that node (Figure 1b). The position of an interpolation cursor—a cross hairs icon—in the same control space determines the desired blend of the example poses. Each node in the hierarchy has an independent control space with an independent set of target poses, as well as its own interpolation widget. Parent nodes contain the interpolated poses of child nodes as additional target poses. To create an animation, users keyframe animate the interpolation widgets, typically in a top down order. Additional poses and nodes can be added as desired. This allows for a workflow in which motion can be refined from a rough sketch to detailed, nuanced animation.

Typically, top–level nodes contain a small set of poses corresponding to major body actions. For example, one or two will indicate arm extensions, one or two will indicate leg extensions, etc. Child nodes parameterize these parent node poses as an interpolation among more detailed poses. These child node poses can represent stylistic variations or changing constraints. There is no restriction on the type of pose variation, breadth of the tree, or depth of the tree; it is adaptable to different animation styles and problems and different individual user workflows.

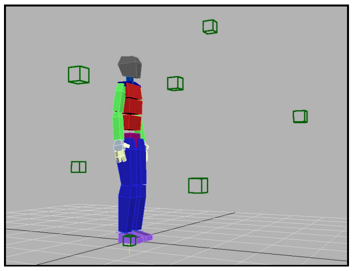
As we seek to provide a system for interacting with complex data sets, we include tools to help users navigate among and organize many poses more effectively. First, we provide a tool to help users partition a node with many poses into a set of child nodes, each of which will contain a set of poses from the original node that are similar. Second, we provide an approach to changing the layout of a control space, such that when overlaid with the 3D scene, users can navigate among as many poses as possible with a 2D input device.

In addition to working with poses to create new motions, we provide a set of techniques for importing existing mo-

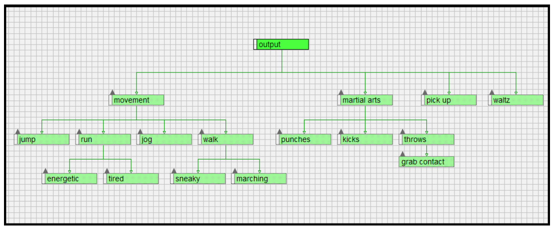
[†] Gaussian process latent variable model



(a)



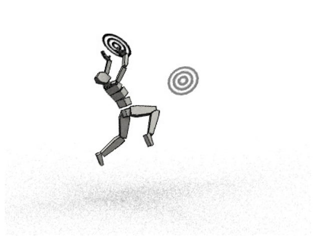
(b)



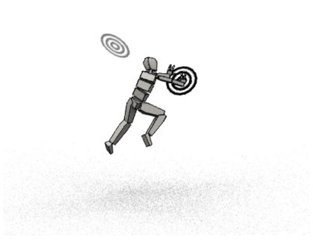
(c)

Figure 1: *For each node, the user positions example poses (a) in a 3D control space with corresponding control targets, which overlay the 3D scene (b). The position of the 3D cursor defines an interpolated pose using the poses associated with the selected node of the pose tree (c).*

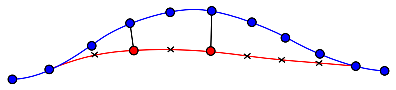
tions and then editing them within a control space. Considering the motion as a pose sequence, a motion control path is created within the low dimensional control space. Users can then independently edit either the motion path or the individual poses. New paths in control space define new motions, allowing users to quickly create new variations of an existing motion, or to edit a motion to meet specific constraints. For example, a change in a single pose (as shown in Figure 2a and b), results in the creation of a new motion following a new control path, which smoothly varies to meet the pose constraint (Figure 2c).



(a)



(b)



(c)

Figure 2: *Motions can be edited in the 3D control space by changing one or more poses (a, b) and their associated control targets (red, in c). The change in each is propagating to adjacent frames ('x,' in c) by altering a control space embedding of the motion.*

2. Related Work

Our representation of motion that allows users to create a rough motion sketch and then add refinements is similar in spirit to Neff and Fiume's AER system [NF05]. Their system is complementary to ours, in that they provide a set of tools for procedural refinements, while our refinements are under complete user control. The system of Laszlo et al. [LNS05] adaptively predicts possible future poses of a simulated character. It displays these as icons in a control space that maps to simulation parameters. In contrast, our system directly maps control space to specific poses.

The editing of motion paths has been investigated using world space techniques [Gle01]. We also introduce approaches to editing with motion paths; however, our paths exist in an abstract control space. Our path based editing is intended to fundamentally change the motion, rather than to reuse the motion details on a new world space path.

Multi–target pose interpolation is a commonly used technique for facial animation [Par72], and the use of radial basis functions has been adapted to full body pose interpolation [LCF00]. Radial basis function interpolation has also been used to interpolate among example motions to meet specific constraints [RSC01, ES03].

Low dimensional data embeddings created with regression models can be useful for both compressing motions and optimizing motions based on example data. Safanova uses a low dimensional linear embedding of example motions to efficiently compute spacetime optimization solutions for new motions of similar style [SHP04]. Grochow et al. estimate latent space pose probability distributions using a scaled Gaussian process latent variable model, and character poses are optimized to meet inverse kinematics constraints using the latent space likelihood function [GMHP04]. They also allow users to navigate the latent space to select poses, although the lack of pose layout control makes such navigation unpredictable. Wang et al. build upon this approach, using Gaussian processes to model the temporal dependencies among adjacent poses when constructing their embedding [WFH05].

The interpolation of example motions has been addressed with a number of techniques. Frequency band interpolation can capture the transition among cyclic motions such as locomotion [BW95, UAT95]. Nearest neighbor searches combined with linear interpolation allow for similar new motions to be created that approximately meet given constraints [WH97]. Radial basis functions better capture the nonlinearities of arbitrary motion parameterizations with respect to control parameters [RBC98]. The two–component statistical model of Mukai and Kuriyama takes a different approach for accounting for the nonlinearity, using control parameters alone to predict it [MK05]. Robust time alignment among examples [KG03] and automatic extraction of similar motions for interpolation [KG04] have also been investigated.

We also create edited motions by interpolation, but construct each new frame using a combination of poses at neighboring frames and displacement–mapped motion variations created from new poses provided by the user. The use of displacement maps to edit motions to meet specific constraints has been investigated using both kinematic techniques [LS99] and spacetime optimization [Gle97, Gle98].

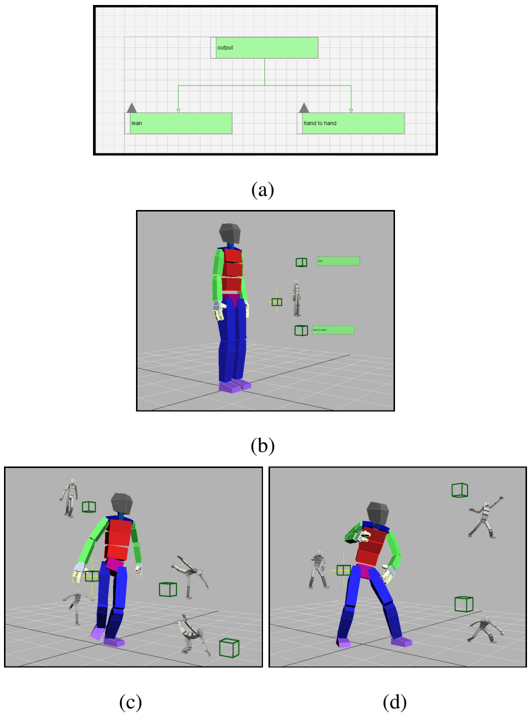


Figure 3: *Contents of an example pose tree (a): Each node contains references to either user–created poses or interpolated poses, which child nodes create (b). Leaf nodes contain only example poses; these sets are typically task specific. This example decouples leaning poses (c) from hand contact poses (d).*

3. Spatial Pose Trees

Spatial pose trees allow users to organize target poses in a hierarchy such that each node can be uniquely specialized to a particular degree of fidelity in creating a motion. Within each node, the pose layout defines an interpolation space such that any point $\mathbf{c}$ in the control space maps to a pose $\mathbf{p}(\mathbf{c})$. By moving a cursor in control space (we use three dimensional control spaces), users blend among example poses $\mathbf{p}_i$ with associated *control targets* $\mathbf{c}_i$. These control targets—locations in control space that map exactly the given poses—are either positioned by the user or automatically laid out using one of the techniques described in Section 4.1. By casting this problem as function approximation, we define a continuous mapping from control pace to pose space using sparse data interpolation, as was done by Igarashi et al. [IMH05].

The hierarchical structure of the pose tree allows a child node to compute an interpolated pose, which the parent node includes as a pose $\mathbf{p}_i$ and control target $\mathbf{c}_i$. For example, the top level node will typically have a small set of target poses that are sufficient to capture a simple animation. This might include the different poses of a walk cycle or different limb extensions. Each of these poses can then have a corresponding child node, which will include variations of the pose. These can be stylistic variations, constraint–based variations, or semantically important, but more detailed poses (for example, posing the hand). An arbitrary degree of refinement is possible; the final complexity of the pose tree is dependent on both the application and required motion quality.

To create an animation using spatial pose trees, users first lay out poses within each node of the control space hierarchy. We keyframe animate widgets in each node to capture the variation of pose over time; the location of these widgets defines the interpolated pose. This can be thought of as the child node contributing a motion component to the overall motion of the parent node. Thus, a top level node can be animated to blend among a punch, a kick, a rest pose, and a block, while child nodes which specialize each of these poses can have their control widgets animated to provide variation within the sequence of punches, kicks, etc. This representation also provides the benefit that individual motion components can be reused, retimed, or warped, independently of the remainder of the motion.

At a technical level, we wish to approximate, for each node, a pose approximation function $\mathbf{p}(\mathbf{c})$, in which the example poses $\mathbf{p}_i$ and associated control targets $\mathbf{c}_i$ are considered as evidence for the overall shape of the function. Note that a subset of these poses are computed by interpolation in a child node; the remainder are explicitly specified by the user. As in Igarashi et al., we use radial basis functions with Euclidean distance metric kernel functions. This results in the functional form

$$\mathbf{p}(\mathbf{c}) = \sum_i \mathbf{w}_i |\mathbf{c} - \mathbf{c}_i| + \mathbf{a}(\mathbf{c}), \tag{1}$$

where $\mathbf{a}(\mathbf{c})$ is a hyperplane in pose space that accounts for the linear components of the function. The basis function coefficients $\mathbf{w}_i$ are found by enforcing the constraints $\mathbf{p}(\mathbf{c}_i) = \mathbf{p}_i$. This leads to a linear system; see Igarashi et al. for a description of the computation. As in their system, we choose this kernel function for its intuitive mapping between change in cursor position and change in pose.

3.1. Pose Location Suggestion

When a control space contains many poses, it can be difficult for the user to select locations for the control target of a new pose that do not significantly modify the overall control space. We provide an assistance mode, in which the user can create a pose, and a control target location is automatically determined that minimally modifies the control space. We cast this as an optimization problem, seeking to minimize the objective function $L(\mathbf{c}) = |\mathbf{p}(\mathbf{c}) - \mathbf{p}'|$. Here, $\mathbf{p}'$ is the new user–provided pose, and the optimization solution $\mathbf{c} = \mathbf{c}'$ gives us the suggested location for the control target. We use a pose distance metric similar to that of Kovar et al., which measures the aggregate distance among corresponding joint positions relative to the root [KGP02].

To minimize $L(\mathbf{c})$, we use particle swarm optimization, a stochastic global optimization algorithm [KE95]. Candidate solutions $\mathbf{s}_i$ are created at the locations of each of the current control targets $\mathbf{c}_i$. Each solution is then treated as a particle in control space, with associated velocity $\mathbf{v}_i$. We track the best solution values $\mathbf{s}'_i$ for each candidate particle, as well as a global best solution $\mathbf{s}_g$. Each candidate solution is iteratively updated using a flocking–like update rule that advances it toward a stochastic interpolation between $\mathbf{s}'_i$ and $\mathbf{s}_g$. We sample two uniformly distributed random numbers r_1 and r_2, each $\in [0,1]$, and apply these particle update rules:

$$\mathbf{v}_i = \mathbf{v}_i + w_1 r_1 (\mathbf{s}'_i - \mathbf{s}_i) + w_2 r_2 (\mathbf{s}_g - \mathbf{s}_i)$$

$$\mathbf{s}_i = \mathbf{s}_i + \mathbf{v}_i.$$

We set the relative weighting coefficients to $w_1 = w_2 = 0.15$.

To update the best candidate solutions $\mathbf{s}'_i$, we evaluate $e(\mathbf{s}_i) = \mathbf{p}' - \mathbf{p}(\mathbf{s}_i)$. Each $\mathbf{p}(\mathbf{s}_i)$ is computed using Equation 1. We also track best error evaluation values e_i associated with $\mathbf{s}'_i$, and update these best candidate solutions when $e(\mathbf{s}_i)$ is less than e_i. We similarly update $\mathbf{s}_g$, and accept the best solution after a fixed number of iterations (100). While this does not guarantee the best possible location will be found precisely, in practice, an exact minimum is not as important as interactive manipulation that allows concurrent display of the suggested control target as the user modifies the new pose.

3.2. Automatic Hierarchy Construction

We allow users to automatically construct a set of child nodes from a set of poses in a given node, as a form of automatic hierarchical organization. This approach is intended for nodes containing a large set of sparsely spaced example poses, and not for a dense set of poses such as motion data, in which categorization is not intuitively clear. For example, a set of poses containing various kick poses, various resting poses, and various punching poses would be more amenable, as the data better admits categorization. The user selects a desired number of child nodes (one for each recognizable category), and we use k–means clustering to partition the selected poses into a set of new child nodes. We retain the corresponding control target locations. For clustering, we again use a pose–to–pose distance metric similar to that of Kovar et al. [KGP02]. Figure 4 illustrates this process; each resulting child node contains poses that have similar stance.

3.3. Control Space View Selection

Given a preferred view for working with a given motion, it is possible that the control targets $\mathbf{c}_i$ are laid out such that it is difficult to select desired poses. This arises due to the loss of a degree of freedom in using a 2D interface to explore a 3D control space. For example, the group of control targets to the left of the character in Figure 5a has a great deal of screen space overlap. To reduce this loss of control, we can transform the control spaces such that the screen space projection captures maximal variance among the control target locations. We first partition the poses into a set of clusters using k–means clustering. We then compute a principle components analysis basis on the control target positions $\mathbf{c}_i$ for each cluster. Each cluster of targets is repositioned such that the two eigenvectors whose corresponding eigenvalues are larger than the third are orthogonal to the image plane's normal vector. We also center and scale the data relative to the center and dimensions of the image space projection. We find that clustering works well, as users have a strong tendency to spatially organize data into groups of related targets. If this is not the case, users can request single cluster processing, in which case all targets are considered as a single group. An example layout from three nodes is shown Figure 5b.

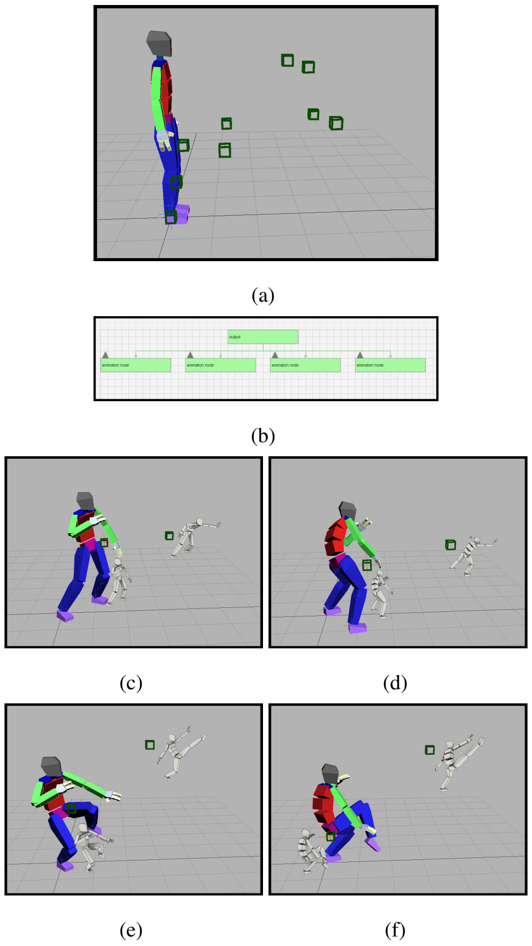


Figure 4: *Automatic hierarchy construction: We cluster the target poses of a single node (a) into a set of child nodes (b), using a pose space distance metric. Control targets retain their positions, but the simpler control spaces are easier for users to recall (c–f). We overlay target poses for illustration.*

4. Motion

For motion, the hierarchical structure provides a rather powerful editing system. Motion refinements applied only to certain limbs can be stored in child specialization nodes. Building on this, the entire motion can be broken down into a hierarchy of motion components to assist animators in organizing motion data. Furthermore, any given node can store similar variations of a motion, allowing users the freedom to interactively select among stored options for different motions that can be applied to different parts of a character.

To edit a motion, a user first embeds the motion into a low–dimensional control space. Rather than requiring the user to do this manually, we provide two automated tools for embedding a motion. By modeling a motion $\mathbf{M}$ as a sequence of poses $\{\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_n\}$ with an associated sequence

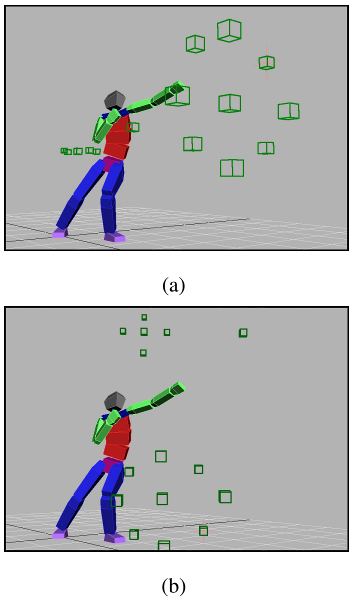


Figure 5: *As control targets exist in 3D space, some pose sets can be difficult to navigate from a particular view, for example the set to the left of the character (a). Repositioning each set of poses relative to the current view alleviates this difficulty (b).*

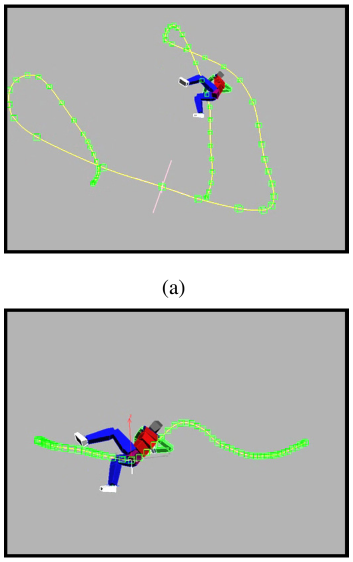
of control targets $\{\mathbf{c}_1, \mathbf{c}_2, \ldots \mathbf{c}_n\}$, we can explicitly connect adjacent poses in control space to create an abstract control path. By doing so, we can apply an edit to a motion and also create a new associated abstract control path in control space. We supply tools for motion playback, such that a user can interactively select among multiple paths in a control space to create variations of a motion. The approach of keyframe animating a control space widget then treats the per–frame poses of the motion paths as interpolation targets for generating new motions.

4.1. Embedding Motions

We provide two approaches for embedding a motion as a path in control space: one attempts to capture maximal pose variance in the layout and the other attempts to provide a visual metaphor by tracking an important character position. The latter approach is particularly useful for motions that follow a path, such as locomotion, as the control path provides a guideline for the character's position in space. For in–place motions such as gestures or dance, it can be difficult to work with; in this case, the maximal variance embedding is more appropriate.

To capture maximal variance, we use principle components analysis to reduce the dimensionality of the set of poses $\mathbf{p}_i \in \mathbf{M}$. We allow users to select either two–dimensional or three–dimensional PCA projections of the poses, depending on whether screen space or 3D control is preferred. We choose the 2 or 3 eigenvectors with corresponding eigenvalues of maximal magnitude, and project the poses onto the basis they form. This embedding tends to follow low–frequency curves in the projected spaces, as seen in figure Figure 6a, maintaining our metaphor of providing an abstract motion path. We assign each pose $\mathbf{p}_i$ to a control target $\mathbf{c}_i$ located at the pose's projection into the reduced–dimensional space.

To provide a visual metaphor, we allow the user to select a joint on the character. For each frame i, we then assign the pose $\mathbf{p}_i$ to a control target $\mathbf{c}_i$. This is illustrated in Figure 6b.


(a)

(b)

Figure 6: *Users can embed existing motions as control curves for editing, visualization, and playback. The layout of an embedding either maximizes variance among the poses (a) or follows the path of a joint (b).*

4.2. Editing Motions

We currently support two approaches for editing motions: creating deformations by interpolation that meet sparse constraints provided by the user, and creating displacements that propagate such constraints along the motion in time such that an interpolatable space of motions is created. In each case, we place the edited region of the motion along an alternate branch of the control path.

Each technique allows users to edit motions to meet specific constraints. The first approach, which includes interpolated poses, works well when the alternate path is placed near the original. This distance is approximately proportional to how broadly in time the effects of the new pose are apparent. As distance from the original path increases, believability reduces, however, as the interpolation acts as a low pass filter. Should this become objectionable, the user can select any non–explicitly provided pose and edit it, populating the interpolation space to increase believability. The second technique contains no pose computed as an interpolation of other poses. Only changes to the degrees of freedom are interpolated. However, this populates the control space with many more basis functions, which can decrease efficiency when many motions are stored in a node.

Given edited branches such as these, users can interpolate among the branch locations to create motion blends. They can also keyframe animate interpolation widgets to follow a path near the branches, allowing for retiming and changing interpolation weights. This also allows a single constraint–based edit to provide information for a continuously varying constraint, and users can provide as many branches as necessary to model the relevant constraint space. Note that users populate this space by specifying only a few poses per branch, regardless of how the branch is evaluated, and this is a relatively quick process.

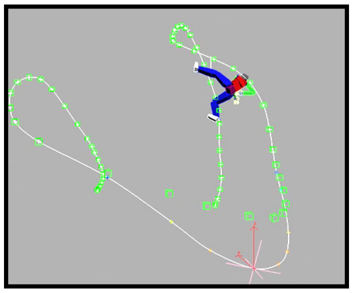


Figure 7: *After embedding a motion, users can alter the location of the control path to create interpolated variations that pass through poses other than those originally in the motion.*

To deform a motion, a user selects a starting pose, an ending pose, and a set of poses to edit. Let these be $\mathbf{p}_s, \mathbf{p}_e, \{\mathbf{p}_j\} \in \mathbf{M}$, respectively. Each edited pose $\mathbf{p}'_j$ has a corresponding new control target $\mathbf{c}_j$, which the user positions along the desired path of the edited branch. We compute the control path of the new branch by fitting an interpolating spline through $\mathbf{p}_s$, each $\mathbf{p}'_j$, and $\mathbf{p}_e$. We add new intermediate control points along the branch to maintain the total number of frames, although these intermediate control points need not have associated poses that introduce new interpolation constraints.

In the interpolation approach, each of the edited poses $\mathbf{p}'_j$ and control targets $\mathbf{c}_j$ act as new constraints on the control space, and we simply re–solve for the coefficients and linear component of Equation 1. The intermediate control targets do not have new associated poses, and as playback passes through these locations in control space, we evaluate the pose with interpolation. Figure 7 illustrates this technique.

The second approach applies motion displacement mapping [BW95], or warping [WP95], to generate new poses along the path. We fit an endpoint–interpolating spline through the pose differentials $\mathbf{p}'_j - \mathbf{p}_j$ and sample this spline to generate the new data points. We then re–solve for the coefficients and linear component of Equation 1.

5. Implementation Details and Results

Spatial pose trees have been implemented as a set of plugins and scripts to Autodesk's 3D Studio Max animation system. We allow the system to work with an arbitrary parameterized character, as well as whatever interpolation parameterization is appropriate to a given character rig. The 3D control spaces coexist in world space along with the 3D scene the user works with, although this is not necessary. Our graphic tree representation is edited using an internal node editing utility, and we display the poses associated with the control spaces of all currently selected nodes. Animation of control space interpolation widgets is accomplished using the available set of animation tools.

A number of example pose trees have been created for interactive manipulation. Figure 1c contains a pose tree organized for everyday activities, while the particular control space shown in Figures 1a and 1b contains a collection of stylized character poses. Figure 3 is a simpler, more illustrative example. As the system can control arbitrary parameterized models, we illustrate its application to facial animation in Figure 8, in which the user interactively drags among five target facial poses to create expression combinations. In Figure 9, the original motion (a) has the character make contact with the dark target. By editing the contact pose, the motion smoothly varies to meet the new contact constraint (b). This example uses motion displacement mapping.

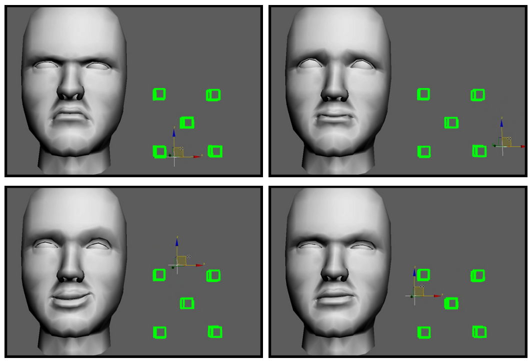
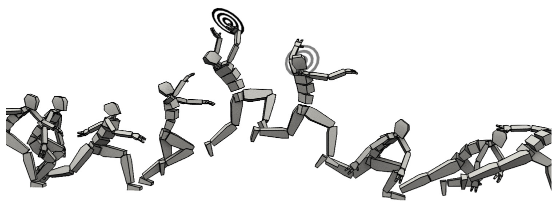
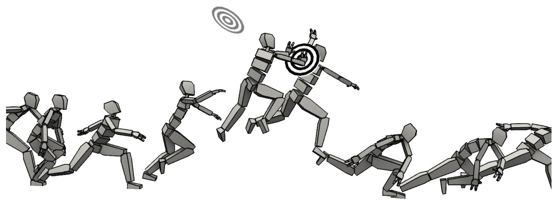


Figure 8: *Spatial pose trees can control any parameterized model. In this example, the user creates four new facial expressions by interactively blending among sculpted example poses.*



(a)



(b)

Figure 9: *In this example, the user changes a single pose of a given motion (a) such that it makes contact with a new target (b). Adjacent frames smoothly vary to meet the constrained pose.*

6. Conclusion and Future Work

Spatial pose trees provide a system in which animators can create and edit motion using a top–down approach. As users refine a motion, they explicitly organize the data they create. The tree representation also provides an interface for navigating, selecting, and editing the user–created components of a motion; existing systems do not provide this. Our representation is amenable to multi–user workflows, in which a team of animators can refine a motion, as the motion refinements are independently represented. The view selection and hierarchy creation tools assist users with interacting with this representation, reducing the time spent organizing and navigating among pose data. The motion embedding system generalizes spatial pose trees to facilitate interactive blending, editing, and deformation of motions to meet both aesthetic goals and explicit motion constraints.

While our three dimensional control space facilitates the layout of more complex sets of poses, interaction based on a two dimensional interface and display does decrease the user's perception of data and their control accuracy. While the control target layout tool decreases this difficulty, we aim to explore the use of fully 3D navigation and visualization interfaces to refine the system interaction [GWB04].

While our motion embedding tools allow visualization and editing among many example motions, the playback system could be extended to allow for interactive visualization and editing of complex motions graphs [KGP02]. Our system currently addresses constraints by introducing new motions at user–positioned paths; an alternative would be to search for a path that best meets the constraints, similar to our pose suggestion system. To avoid populating the control space too densely, constraints could be layered on top of motions using an inverse kinematics system.

Finally, we would like to investigate the use of visual metaphors to aid users in navigating a control space. Igarashi et al. address this by making the control cursor an integral part of the scene being created [IMH05]. However, this is not appropriate for many animation applications, as many motions are not focused on object manipulation. A suggestive navigation interface that provides hints as to potential future states, such that presented by Laszlo et al. [LNS05], might be more appropriate for general animation.

References

[BW95] Bruderlin A., Williams L.: Motion Signal Processing. In *SIGGRAPH 1995: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), pp. 97–104.

[ES03] ElKoura G., Singh K.: Handrix: Animating the Human Hand. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003).

[Gle97] Gleicher M.: Motion Editing with Spacetime Constraints. In *I3D 1997: Proceedings of the 1997 Symposium on Interactive 3D Graphics* (1997).

[Gle98] Gleicher M.: Retargetting Motion to New Characters. In *SIGGRAPH 1998: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998).

[Gle01] Gleicher M.: Motion Path Editing. In *I3D 2001: Proceedings of the 2001 Symposium on Interactive 3D Graphics* (2001).

[GMHP04] Grochow K., Martin S. L., Hertzmann A., Popović Z.: Style–Based Inverse Kinematics. *ACM Trans. Graph. 23*, 3 (2004), 520–531.

[GWB04] Grossman T., Wigdor D., Balakrishnan R.: Multi–Finger Gestural Interaction with 3D Volumetric Displays. In *UIST 2004: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (2004), pp. 61–70.

[IMH05] Igarashi T., Moscovich T., Hughes J. F.: Spatial Keyframing for Performance–Driven Animation. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on ComputerAnimation* (2005), pp. 107–115.

[KE95] Kennedy J., Eberhart R. C.: Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks* (1995), pp. 1942–1948.

[KG03] Kovar L., Gleicher M.: Flexible Automatic Motion Blending with Registration Curves. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003).

[KG04] Kovar L., Gleicher M.: Automatic Extraction and Parameterization of Motions in Large Data Sets. *ACM Trans. Graph. 23*, 3 (2004).

[KGP02] Kovar L., Gleicher M., Pighin F.: Motion Graphs. In *SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (2002), pp. 473–482.

[LCF00] Lewis J. P., Cordner M., Fong N.: Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton–Driven Deformation. In *SIGGRAPH 2000: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), pp. 165–172.

[LNS05] Laszlo J., Neff M., Singh K.: Predictive Feedback for Interactive Control of Physics–Based Characters. In *Proceedings of Eurographics 2005* (2005).

[LS99] Lee J., Shin S. Y.: A Hierarchical Approach to Interactive Motion Editing for Human–Like Figures. In *SIGGRAPH 1999: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (1999), pp. 39–48.

[MK05] Mukai T., Kuriyama S.: Geostatistical Motion Interpolation. *ACM Trans. Graph. 24*, 3 (2005).

[NF05] Neff M., Fiume E.: AER: Aesthetic Exploration and Refinement for Expressive Character Animation. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005).

[Par72] Parke F. I.: Computer Generated Animation of Faces. In *ACM 1972: Proceedings of the ACM Annual Conference* (1972), pp. 451–457.

[RBC98] Rose C., Bodenheimer B., Cohen M. F.: Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions. *Computer Graphics and Applications 18*, 5 (1998), 32–40.

[RSC01] Rose C. F., Sloan P.-P. J., Cohen M. F.: Artist–Directed Inverse Kinematics Using Radial Basis Function Interpolation. In *Eurographics 2001* (2001).

[SHP04] Safonova A., Hodgins J. K., Pollard N. S.: Synthesizing Physically Realistic Human Motion in Low–Dimensional, Behavior–Specific Spaces. *ACM Trans. Graph. 23*, 3 (2004), 514–521.

[TJ81] Thomas F., Johnston O.: *The Illusion of Life: Disney Animation*. Abbeville Press, 1981.

[UAT95] Unuma M., Anjyo K., Takeuchi R.: Fourier Principles for Emotion–Based Human Figure Animation. In *SIGGRAPH 1995: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), pp. 91–96.

[WFH05] Wang J., Fleet D., Hertzmann A.: Gaussian Process Dynamic Models. In *NIPS 2005: Proceedings of the 2005 Conference on Neural Information Processing Systems* (2005).

[WH97] Wiley D. J., Hahn J. K.: Interpolation Synthesis of Articulated Figure Motion. *Computer Graphics and Applications 17*, 6 (1997), 39–45.

[Wil01] Williams R.: *The Animator's Survival Kit*. Faber and Faber, 2001.

[WP95] Witkin A., Popovic Z.: Motion Warping. In *SIGGRAPH 1995: Proceedings of the 22nd Annual Conference on*

Computer Graphics and Interactive Techniques (1995), pp. 105–108.