EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne (Guest Editors)

Volume 20 (2001), Number 3

Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation

Charles F. Rose, III Peter-Pike J. Sloan Michael F. Cohen

{chuckr,ppsloan,mcohen}@microsoft.com
Microsoft Research

Abstract

One of the most common tasks in computer animation is inverse-kinematics, or determining a joint configuration required to place a particular part of an articulated character at a particular location in global space. Inverse-kinematics is required at design-time to assist artists using commercial 3D animation packages, for motion capture analysis, and for run-time applications such as games.

We present an efficient inverse-kinematics methodology based on the interpolation of example motions and positions. The technique is demonstrated on a number of inverse-kinematics positioning tasks for a human figure. In addition to simple positioning tasks, the method provides complete motion sequences that satisfy an inverse-kinematic goal. The interpolation at the heart of the algorithm allows an artist's influence to play a major role in ensuring that the system always generates plausible results. Due to the lightweight nature of the algorithm, we can position a character at extremely high frame rates, making the technique useful for time-critical run-time applications such as games.

1. Overview

A talented animator can create believable characters that spark a desired response in an audience. Believable characters, a story, and a setting can transport a viewer for a time to another world. Animators, together with modelers, editors, sound engineers, and others can through tens of thousands of hours of human labor create theatrical masterpieces as seen in film. While these endeavors may produce the highest quality animation, current techniques are costly and are not appropriate for many problems.

Interactive (unscripted) animation must adapt and change as needed to create a unique run-time experience each time such a system is executed. The most commercially successful interactive animation experiences are found in computer games, such as in *Microsoft's Asheron's Call* or *MechWarrior*. Avatar-mediated virtual conferencing, 3D chat, and other virtual reality applications also require interactive animation.

In this paper we focus on providing tools to solve the inverse-kinematics (IK) problem. Inverse-kinematics generally refers to positioning an end effector such as a hand or foot to some goal location. IK is needed for crucial tasks. When a character needs to lift, touch, or otherwise manipulate a virtual object, inverse-kinematics is required. The reason IK problems are interesting and difficult is that there are, in general, an infinite number of ways one can position the joint angles of a complex figure such as a human to place the hand in some location. Finding a configuration that solves the IK problem and also looks "natural" remains an open research problem. Additionally, most applications do not require simply a static position of the character to place the hand in some location, but in fact a complete sequence to move the character to and from such a static position. We examine both the static and dynamic needs for inverse-kinematics. Finally, we wish to place much of the decision of what looks "natural" to the artist.

Artist-centered interactive animation techniques have been an important topic of recent research. Bringing the artist back into focus is due to a realization that an artist's talent is hard to surpass and that artist supplied source material can be molded and changed as needed while respecting much of the original aesthetic. Research programs that seek to supplant, rather than support, the artist deny themselves a powerful ally. Furthermore, methods that leverage the artist's talent fit more easily into existing production pipelines, making that research more relevant in the near-term.

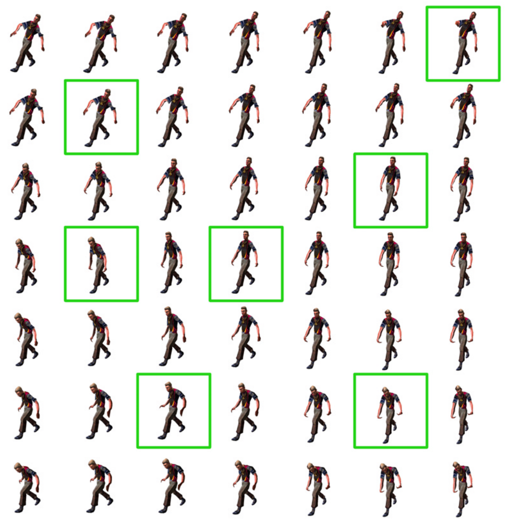

Figure 1: *Emotional control with the Verbs & Adverbs system. Examples are surrounded by a box. There are an infinite number of interpolated motions between and around these examples.*

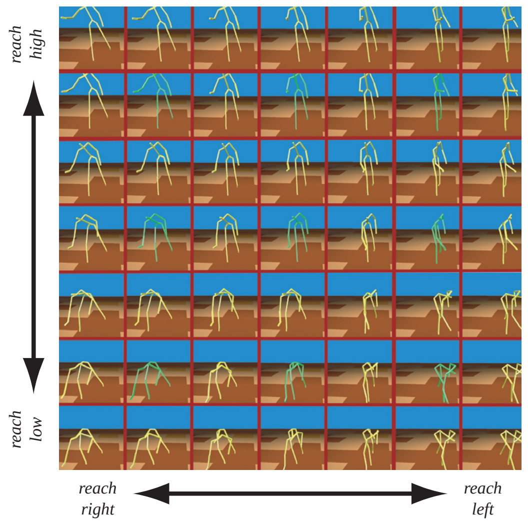

Figure 2: *An early reach from the Verbs & Adverbs system. Errors were often introduced with raw RBF interpolation, so the end-effector often strayed from its desired location. Highlighted characters show example poses and the others are a sampling of interpolated motions.*

The "Verbs & Adverbs" system [14], presented an efficient technique to control animation through interpolation using a combination of linear functions augmented with non-linear radial basis functions. The primary focus of that work was emotion control, such as varying a walk cycle from sad to happy (see Figure 1). One of the other *verbs* shown was a reach, indicating that inverse-kinematic control was possible, as shown in Figure 2. This paper expands upon that work, presenting a number of IK tasks solved through interpolation. The results are quantified and improved through a number of mathematical enhancements over the initial "Verbs & Adverbs" work, including use of a cardinal basis and automated methods for improving the accuracy of the inverse-kinematic tasks.

We present two major enhancements to the "Verbs & Adverbs" system. First, we replace the interpolation per degree-of-freedom (DOF) with an interpolation per example by using cardinal basis functions. The resulting animation is equivalent but the use of the cardinal basis greatly reduces the number of interpolations that need to be performed, greatly increasing the efficiency of the system at design time and modestly increasing its runtime efficiency. Secondly, we show how the non-linearities in *example*-based IK solutions can be automatically corrected by insertion of *pseudo-examples* derived directly from the initial interpolation space designed by the artist. The resulting interpolation-based animation is highly efficient, running at many thousands of frames per second.

The remainder of the paper will progress as follows. After a brief synopsis of related work, the construction of *verbs* will be covered in Section 3. This section will also detail the new cardinal basis construction and how it differs from the initial work in [14]. Section 4 will discuss the specifics of the inverse kinematics problem and how automatically generated *pseudo-examples* provide accurate and efficient results. Some results and conclusions will follow.

The video figures referenced in this paper can be found with the accompanying conference materials and can also be found online in compressed form at *http://research.microsoft.com/graphics/hfap/RBFIKVid.zip*.

2. Related work

Inverse-kinematics is one of the oldest problems in computer animation, tracing its origins back to the problem of positioning industrial robots. Positioning a character with rotational degrees of freedom is non-linear, but for small changes of an articulated figure's DOFs (i.e., the starting configuration is near the desired one), it is sufficiently smooth to enable an iterative linear IK solver. Girard and Maciejewski overview this technique in [5].

For complex motions, however, simple IK will often prove insufficient. Zhao and Badler [21] introduced a robust IK solution based on non-linear optimization. Rose, et al. [15] expanded this formulation to perform IK over entire motion sequences, mostly to enforce IK constraints during motion transitions. Non-linear IK is a powerful enough technique for complex motion capture analysis in the presence of noisy data, as shown by Bodenheimer, et al. [3]. Unfortunately, it suffers from being computationally expensive and by having a run-time that is highly dependent on the difficulty of the task and the quality of the initial guess. For real-time applications, this unpredictability reduces the technique's attractiveness.

Motion retargetting [6] makes great use of IK over sequences of motions to adapt motions from one character to another with great success. Gleicher makes use of a much faster sequential quadratic programming approach. While it behaves less reliably than the method of [21], it is much faster and can be used when the initial guesses are sufficiently good.

Both these techniques make little use of the large body of human biomechanics knowledge. Synergy-based IK [7] uses a measure of how much a particular joint contributes to a given IK task (the synergy), in order to reduce the dimensionality of the problem and to help ensure that the results are human-like.

Rather than relying on complex models for how creatures move, our methodology primarily takes advantage of existing motion data and interpolates these to create new motions that solve the given IK constraint. A considerable amount of work has been undertaken with respect to editing existing animations and blending between two segments [15, 6, 9].

Methods that have addressed modifying existing animation include Fourier techniques as used in [19] to modify periodic motions. Amaya, et al. [1] used information from one motion to modify the emotional content of another, primarily through changes in timing and intensity of motion. This technique also worked exclusively for periodic motions.

Bruderlin and Williams [4] use multi-target interpolation with dynamic time-warping to blend between motions. Linear interpolation over a set of example motions is used by Wiley and Hahn [20] and Guo and Robergé [8]. Both these techniques use $O(2^D)$ *examples* (where D represents the dimension of the space) and fail to address the usefulness of *examples* with a limited region of influence. Our work is most similar to [20] but has distinct advantages. That work used a dense sampling of the parameter space, so $O(2^D)$ actually *understates* their *example* requirements. While their runtime is efficient since multi-target linear interpolation over a regular grid is straightforward, their memory requirements make their technique unattractive for most real-world applications. In contrast, we use far fewer *examples*.

"Verbs & Adverbs" [14] presented an interpolated animation technique requiring a minimum of $O(D)$ examples and which also allowed for local refinement of the interpolation space. These results were applied to shape interpolation in [17] with the key mathematical improvement being the use of cardinal basis functions. We further extend this work by bringing it into the animation domain, and in particular show how to apply this technique to inverse-kinematic problems. The efficiency of this interpolation is such that it can be reasonably used to control a large cast of characters acting simultaneously in conjunction with a high-level control mechanism such as in Musse, et al.[11].

3. Verbs & Adverbs

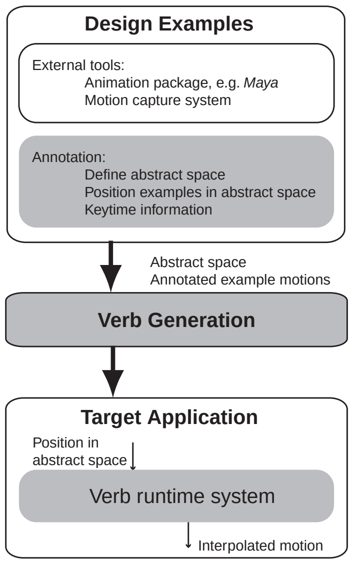

Figure 3: *Key parts of the Verbs & Adverbs system.*

The Verbs & Adverbs system is divided into three main parts, as is shown in Figure 3. Those areas shown in gray are the key elements of the Verbs & Adverbs system. The others represent external programs to prepare data for the system and the target application (e.g., the game) itself. Constructing *verbs* begins with designing *example* motions with an external animation system or a motion capture system. The *example* motions define the scope of variation that the interpolator can produce.

As detailed in [14] before a raw animation can be used as an *example*, it must be annotated with some auxiliary information. For animation blending, keytime and *adverb* information are both required. In short, keytimes, help to time-warp *example* motions so blending can occur between corresponding poses. Full details about the benefits of time-warping can be found in [14, 13, 4].

Adverbs, the other key annotation, are simply the different control axes for the motion. In [14] typical *adverbs* encoded non-kinematic values like a measure of the character's perceived happiness. This paper concentrates on kinematic goals, thus the *adverbs* represent quantities such as the *x*-, *y*-, *z*-goals for an end-effector, or surface-slope to position a foot. The set of *adverbs* for a given *verb* define an *abstract-space* of all possible motions that can be formed by the *verb* runtime system by adjusting the *adverb* values. For immeasurable quantities like happiness, the judgment of the *verb*

Object	Variable	Subscript	Range
Example	X	i	$1..N$
DOF	x	i	$1..N$
		j	$1..M$
Point in Adverb Space	$\mathbf{p}$	i	
Radial basis	R	i	
Radial Coefficient	r	i,j	
Linear Basis	A	l	$0..D$
Linear Coefficient	a	i,l	
Distance	d	i	
Keytime	K	m	$1..$*NumKeys*

Table 1: *Terminology*

designer was used to set the *adverb* values. For the IK motions discussed here, the *adverbs* can be set automatically.

Once *verbs* are designed, they are available to an interactive application. The application chooses at each instant, desired values for the *adverbs*, thus selecting a specific location in the abstract space. If a character were to track a moving ball with their hand, for instance, the *adverbs* might be the *x*-, *y*-, and *z*-offsets from the root of the character to the position of the ball. The *verb* runtime system handles this request very efficiently, enabling the Verbs & Adverbs system to be used in time-critical applications such as a game.

The number of *adverbs* defines the dimension of the abstract space, which we denote with D. We denote the number of *examples* included in the construction of a *verb* by N. Each example motion in a *verb* is required to have the same overall structure. That implies that each *verb* must include the same number of curves that define the trajectory of the character's joints. Use of a cardinal basis in this paper frees us from the further restriction in [14] that all curves have the same number of control points and same curve encoding scheme. All *example* motions in a particular *verb* must also represent the same basic motion. A set of *example* reaches, for instance, must all use the same arm to perform the reach. In terms of the keytimes mentioned earlier, each *example* must have the same number of keytimes in the same order, though the relative timing of the keytimes and the overall duration of each *example* may be different.

Our method uses a number of subscripts and symbols. Please refer to Table 1 for a complete list of these symbols and their meanings. We denote an *example* as X_i, where

$$X_i = \left\{x_{ij}, \mathbf{p}_i, K_m\right\} \tag{1}$$

where i is fixed for a particular example x_i, j ranges from $1..M$, the number of DOFs in the system, and m ranges from $0..$*NumKeys*.

Each x_{ij}, the $j^{\rm th}$ DOF for the $i^{\rm th}$ *example* represents a DOF curve represented in some fashion such as a B-spline, Fourier decomposition, etc. K is the set of keytimes that describe the phrasing (relative timing of structural elements) of the *example*. Based on the keytime annotation for each *example*, the curves are all time-warped into a common time frame. Keytimes are, in essence, additional DOFs and are interpolated at runtime to undo the time-warp. See [14] for details on the use of keytimes.

3.1. Verb generation

Given a set of *examples*, a continuous interpolation over the abstract space is generated for the *verb*. The goal is to produce at any $\mathbf{p}$ in the abstract space, a new motion $X(\mathbf{p})$ derived through interpolation of the *example* motions. When $\mathbf{p}$ is equal to the position $\mathbf{p}_i$ for a particular *example* i, then $X(\mathbf{p})$ should equal X_i. In between the *examples*, smooth intuitive changes should take place. This is the classic multivariate interpolation problem.

In [14], each B-spline coefficient was treated as a separate interpolation problem. The reach *verb* from Figure 2, for instance, had approximately 1200 separate sub-problems. This leads to inefficiencies in both the offline and run-time stages. Sloan [17] introduced the *cardinal basis*, where one interpolation is formed for each *example*. This function determines the *weight* given to that particular *example* over the entire space and is defined to have a value of 1 at the *example* location and 0 at the locations of all the other *examples*. Given this, it is clear that this will form an exact interpolation. For each DOF, the bases are simply scaled by the *example* DOF values and then summed. This approach is not only equivalent to [14], but is more efficient both at design time and runtime and also decouples the DOF representation for the *examples* from the interpolation scheme. Before this, each *example* had to have DOFs of equivalent structure, i.e. same curve types and same number of coefficients.

The shape of the individual cardinal basis function is something we can select. Our problem is essentially one of scattered data interpolation, as we have *example* motions, in a relatively high-dimensional space. Most published scattered data interpolation methods focus on one- and two-dimensional problems. For these, relatively simple schemes can be used, but these often do not scale well in high-dimensions. Linear interpolation using Delauney triangulation, for instance, does not scale well in high dimensions. Based on our need to work in high dimensions with sparse samples, we adopt a combination of radial basis functions and low order (linear) polynomials.

The cardinal basis weighting functions we use are defined as

$$w_{i_1}(\mathbf{p}) = \sum_{i_2=1}^{N} r_{i_2,i_1} R_{i_2}(\mathbf{p}) + \sum_{l=0}^{D} a_{i_1,l} A_l(\mathbf{p}) \qquad (2)$$

where the r_{i_2,i_1} and R_{i_2} are the radial basis function weights and radial basis functions themselves and $a_{i_1,l}$ and A_l are the linear coefficients and linear bases. The subscripts i_1 and i_2 both indicate *example* indices.

Given these bases, the value of each DOF is computed at runtime based on the momentary location, **p**, in the abstract space. The value for each DOF, x_j, at location **p** is defined as

$$x_j(\mathbf{p}) = \sum_{i_1=1}^{N} w_{i_1}(\mathbf{p}) x_{i_1,j} \qquad (3)$$

The linear function provides an overall approximation to the space defined by the *examples* and permits extrapolation outside the convex hull of the locations of the *examples*. The radial bases locally adjust the solution to exactly interpolate the *examples*. We discuss the linear approximation and radial bases in the following two sub-sections.

3.1.1. Linear approximation

The first step to forming our interpolation is, for each *example* i, to fit (in a least squared sense) a hyperplane to the values 1 at the location of the i^{th} *example* location, and 0 at all other *example* locations. This defines N separate least squares problems of the form

$$\mathbf{p}_h \mathbf{a} = \mathbf{F} \qquad (4)$$

where $\mathbf{p}_h$ is a matrix of homogeneous points in the abstract space (i.e., each row is a point location followed by a 1), **a** are the unknown linear coefficients, and **F** is a *Fit* matrix expressing the values we would like the linear approximation to fit. In this case **F** is simply the identity matrix since we are constructing a cardinal basis.

Figure 4 shows the linear approximation and the three radial basis functions associated with the first of three *examples* for a simple one-dimensional abstract space The three *examples* are located at $\mathbf{p} = 0.15, 0.30, 0.75$. The straight line labeled A_1 is the line that fits best through $(0.15, 1), (0.3, 0), (0.75, 0)$. In the one dimensional problem, a best fit line for any particular DOF could then be evaluated by simply scaling the individual lines associated with each example by the DOF values and summing.

While we have chosen to use a linear approximation, other choices could certainly be made. When a particular motion has an understood underlying structure which can be well approximated by a fixed-order polynomial, then the linear approximation step should be replaced with a step based on that particular polynomial.

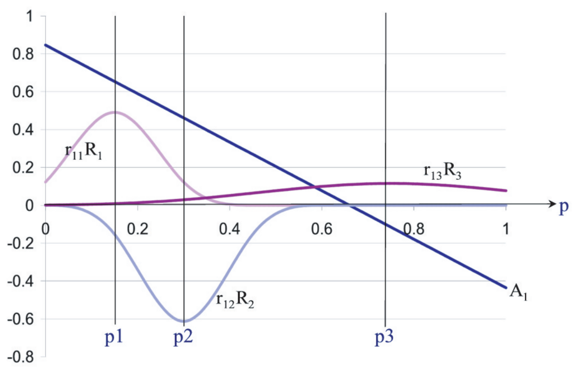

Figure 4: *Linear and radial parts of the cardinal basis function for the first example*

3.1.2. Radial basis

Radial basis interpolation is discussed in Micchelli [10] and by Powell [12]. Radial basis functions have been used in computer graphics for image warping by Ruprecht and Müller [16], Arad *et al.* [2], for 3D interpolation by Turk *et al.* [18], and for skinned shape interpolation by Sloan *et al.* [17].

Since the linear approximations cannot, in general, fully define a cardinal basis by passing through the values 1 and 0, there still remain residuals between the *example* values x_{ij} and the scaled and summed hyperplanes. Rather than cast our interpolations over DOFs, we continue to create the cardinal bases. The residuals in the cardinal bases are given by

$$q_{i_1,i_2} = \delta_{i_1,i_2} - \sum_{l=0}^{D} a_{i_2,l} A_l(\mathbf{p}_{i_1})$$

In the one dimesional problem illustrated in Figure 4, these are just the distances from the line to the desired values of 1, 0, and 0 at p_1, p_2, and p_3.

To account for these residuals, we associate N radial basis functions with each *example*. Since there are N *examples*, we need N^2 radial basis functions. We solve for the weights of each radial basis, r_{i_2,i_1}, to account for the residuals such that when the weighted radial bases are summed with the linear approximation, they complete the cardinal bases.

This leaves us with the problem of choosing the specific shape of the radial bases and determining the radial coefficients. Radial basis functions have the form:

$$R_i(d_i(\mathbf{p}))$$

where R_i is the radial basis associated with X_i and $d_i(\mathbf{p})$ is a measure of the distance between **p** and $\mathbf{p_i}$, most often the Euclidean norm $\|\mathbf{p} - \mathbf{p}_i\|$. There are a number of choices for this basis. As reported in [17], we found using a basis with a cross section of a cubic B-spline a good choice. Amongst

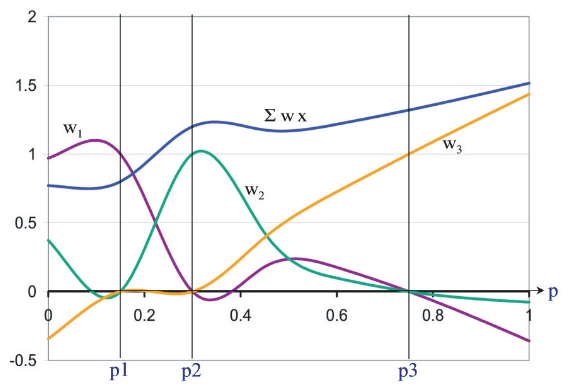

Figure 5: *Cardinal basis functions and scaled sum for a particular DOF.*

other properties, its compact support makes it attractive so as to allow for local refinement of the adverb space.

The radial basis weights, r_{i_2,i_1} can now be found by solving the matrix system,

$$\mathbf{Qr} = \mathbf{q}$$

where $\mathbf{r}$ is an N by N matrix of the unknown radial basis weights and $\mathbf{Q}$ is defined by the radial bases such that $\mathbf{Q}_{i_1,i_2} = R_{i_2}(\mathbf{p}_{i_1})$, the value of the unscaled radial basis function centered on *example* i_2 at the location of *example* i_1. The diagonal terms are all 2/3 since this is the value of the generic B-spline at its center. Many of the off-diagonal terms are zero since the B-spline cross-section drops to zero at twice the distance to the nearest other *example*.

Referring back to Figure 4, we see three radial basis functions associated with the first of the three *examples*. If these three are summed with the linear approximation, we get the first cardinal basis, the line labeled w_1 in Figure 5. Note that it passes through 1 at the location of the first *example* and is 0 at the other *example* locations. The same is true for the other two cardinal bases w_2 and w_3.

Given the solutions for the radial basis weights, we now have all the values needed to evaluate Equations 2 and 3 at runtime. The upper line in Figure 5 is simply the three cardinal bases scaled by the *example* values and summed as in Equation 3.

4. Inverse-kinematic approximations

Our goal is highly efficient, highly accurate IK using blending. As discussed in Section 2, there are many ways to solve the IK problem. IK that adheres to human-like motion is a more constrained problem yet. Commercial systems use biomechanical hints, or analytically solved sub-problems to keep their IK subsystems on track. Biomechanical plausibility can be encoded into the objective function of [21], though this will add further complexity to the nonlinear optimization stage. None of these addresses more subtle concerns of human figure animation, such as making the character reach for something in a particular way, such as lifting a light vs. heavy object. Animation blending provides a solution to all these problems. The animator or motion capture artist can spice the IK with hard to quantify subtlety and ensure that the motion looks realistic. Best of all, IK done using blending runs very efficiently: at a little over 20,000 fps on a 733 MHz PC.

A further benefit of using blending for IK is that the IK process can then be used to create entire motions, rather than just single poses. Some of the examples we present will be for single poses, but others will be for entire motions such as walk cycles over slopes, punching sequences, and reaching heavy or light objects. While IK can be used to fix up existing sequences, such as altering the location where a punch would land, it requires that IK be done on many frames before and after the punch hits the target since the IK change needs to be both blended in from and blended back out to the initial sequence [15].

The reaching motion, shown in Figure 2, demonstrated that motion blending techniques can be applied to inverse-kinematic problems as well as abstract spaces defined by emotional qualities. Unlike emotion, inverse-kinematic positioning can be quantifiably verified.

Even given a completely accurate *adverb*-assignment for a *verb*'s *examples*, in general, a linear continuous change in the *adverbs* will not produce a similar linear change in the resulting motion. This is due to the fact that both the interpolation method and the underlying forward-kinematics process are both non-linear. In general, the desired location of a reach and the actual location will only match exactly at the *example* locations. As reported in [13], the average, and max sampled errors over the bounding volume were 3.12%, and 9.68% respectively. Correcting for this error is the topic of this section.

4.1. Error of initial interpolation spaces

To illustrate the problem of 3D human figure IK, we will consider the simpler problem of positioning the end-effector of a 3-link arm embedded in the plane. Clearly, the 3-link arm example has a much simpler closed form solution but allows us to demonstrate the algorithm. Unlike the closed form solution that can only handle up to 6 degrees-of-freedom, our algorithms are useful for highly complex figures and yield efficient runtime processing.

Given the goal of placing the end-effector at a point (x, y), determine the DOF angles, $(\theta_1, \theta_2, \theta_3)$ that will yield that desired position. To use blending for IK, we need some *example* poses to be blended. Since blending will not yield a linear parameterization of the space, there will be a discrep-

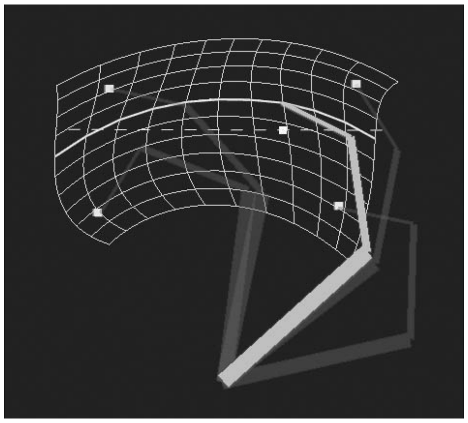

Figure 6: *Regular sampling of the adverb space yields irregular results. Parameters (adverbs) sampled along the dashed line yield results measured along the solid curve. Example poses are superimposed on the image.*

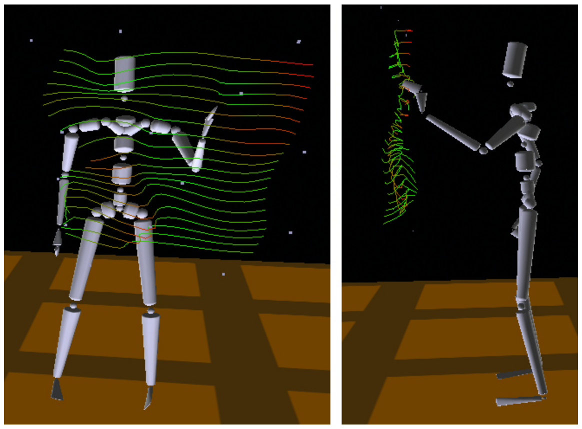

Figure 7: *Sampling the board draw yields results away from the plane of the examples.*

ancy between the location in the IK abstract space, **p** and a measured resulting position of the end effector $x_e(\mathbf{p})$.

Figure 6 shows the warped grid formed by regularly sampling the *verb* defined by the *examples* and plotting the resulting end-effector positions, $x_e(\mathbf{p})$. The warped grid has an average and maximum error, $\|x_e(\mathbf{p}) - \mathbf{p}\|$ of 6.85% and 14.15%, respectively over the sampled locations. Video Figure 1 also shows this arm's distorted *adverb* space.

A more interesting example shows the distorted grid in 3D for a 2D drawing motion (Figure 7). In this *verb*, all *example* poses were on the same plane in 3D, such as one would expect when drawing on a whiteboard. This type of IK problem would be equally relevant to positioning the character's end-effector on a 2-D manifold in 3-space, such as on the surface of a ship's hull [7].

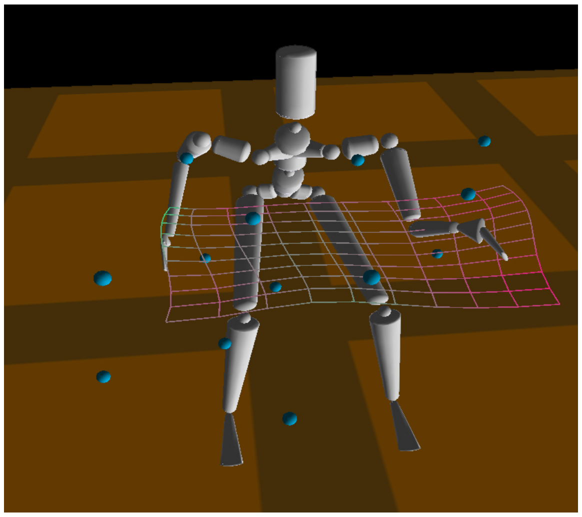

Figure 8: *A planar sampling of the adverb space for a 3D reach verb yields a non-planar, non-regular measured result.*

The least restricted problem is positioning the end effector in 3D, such as shown in Figure 8. This figure shows the distorted results for samples taken on one plane in the *adverb* space. A planar slice through the *adverb* space has no guarantee to result in the end effector remaining in a plane.

4.2. Finding the optimal setting of $\mathbf{p}'$

One thing to note about Figures 6, 7, and 8 is that the distortions are for the most part smooth. Improving the IK results can be done in one of two main ways: through modification of the *adverb* space to reduce distortion of the mapping from adverbs to resulting motion, or by on-the-fly modification of the desired point to achieve the desired goal. With very smooth distortions, this latter approach can provide a good solution.

Consider the 2D 3-link arm. When a desired location **p** is specified, the end-effector is measured at $x_e(\mathbf{p})$, yielding an error vector of $\mathbf{v}_e = x_e(\mathbf{p}) - \mathbf{p}$. Since the error function is smooth, (i.e., locally approximately linear), the inverse of the error vector should point towards a "better" point in the *adverb* space in order to reach the desired goal. We can search for a new point, $\mathbf{p}'$ such that $x_e(\mathbf{p}') = \mathbf{p}$. We perform a line search along the direction from **p** opposite the error vector. Performing this line search improves the IK results markedly. For the 3-link arm, average and maximum error were reduced from (6.86%,14.125%) to (1.125%,4.725%) over the working volume of the *verb*. Figure 9 shows the less distorted parameterization space.

Likewise, the board drawing and 3D reach were improved from (8.11%,23.86%) to (3.9%,11.9%) in the case of board drawing (Figure 10) and from (5.27%,8.98%) to (1.03%,3.22%) for the 3D reach.

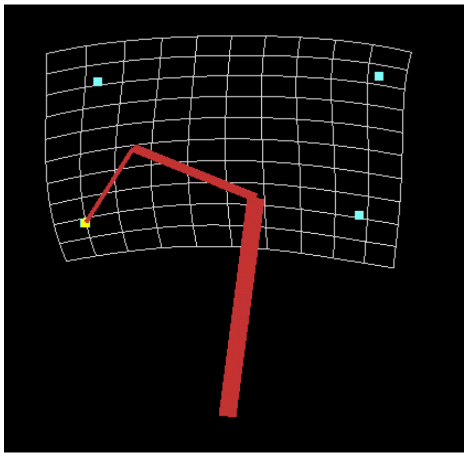

Figure 9: *Searching for optimal **p'** greatly improves the mapping from adverb to end effector location.*

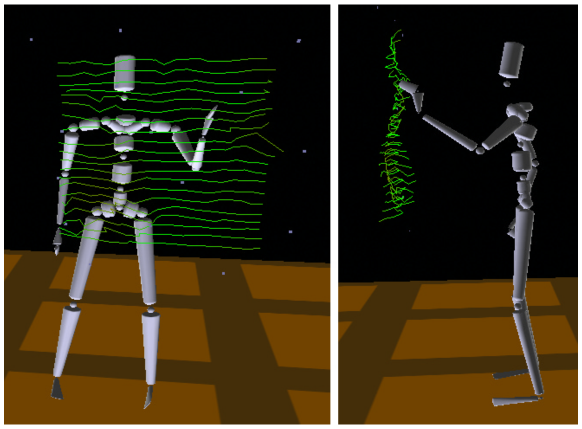

Figure 10: *Searching for optimal $\mathbf{p}'$ greatly improves the mapping from adverb to end effector location.*

The gradient decent line search can be iterated to increase the accuracy albeit at a computational cost. Searching for a new point in the adverb space to perform the blending fails when the error function is not smooth. It also suffers from being slow at runtime as it is an iterative solution. The inner loop requires an evaluation of the character's transformation hierarchy, very similar to traditional IK. An alternative we discuss next is to improve the abstract space itself.

4.3. Improving the abstract spaces

A second option, more costly at design-time, results in more efficiency at run-time. Through the addition of new *examples* or *pseudo-examples* [17], we can in essence "fix" the distortion from the *adverb* space to end-effector. We describe an automated process to improve the mapping from *adverb* to end-effector. The basic algorithm is as follows:

While the *adverb* space exhibits too much distortion:

1. Find the worst $\mathbf{p}$ of the adverb space, i.e., $max_{\mathbf{p}}\|x_e(\mathbf{p})-\mathbf{p}\|$.
2. Search to find the best $\mathbf{p}'$ to minimize $\|x_e(\mathbf{p}')-\mathbf{p}\|$.
3. Insert a *pseudo-example* at $\mathbf{p}$ drawn from the location $\mathbf{p}'$. In other words insert the *pseudo-example* $x(\mathbf{p}')$ at $\mathbf{p}$.
4. Solve for the w_i of the new interpolation space.

A *pseudo-example* carries no additional DOF positioning data, but is rather a weighted sum of the original *examples* as defined by Equation 3. Thus solving for the basis functions associated with a *pseudo-example* is slightly modified from before. Instead of having values 1 at the *example* location and 0 at all other *examples*, the new basis is constrained to have the values equivalent to the weights for each *example* found at $\mathbf{p}'$ before the new *pseudo-example* is inserted. See [17] for details. After the insertion of the new *pseudo-examples*, the new interpolation space is used as usual at runtime.

There is some flexibility in the last two steps of the algorithm above. When inserting a new *pseudo-example*. One could solve for new hyperplanes for each previous *example* and *pseudo-example*, given the newly inserted *pseudo-example* or simply keep the old ones. One could reset all the radii of the radial basis functions given the new nearest neighbors or keep the old one and only select a new radius for the new *pseudo-example*. Essentially these are decisions about how local or global the effect of the newly inserted *pseudo-example* should be. We tried each of these and have found keeping the effect of each new *pseudo-example* local works best and is of course simpler to implement and compute. One problem with each new *pseudo-example* having a global effect is that parts of the abstract space that worked well may get worse.

Thus, once a radius is established for an *example* or *pseudo-example* it remains fixed. Additionally, the hyperplanes also remain fixed for the original *examples*. Thus, as each new *pseudo-example* is inserted we only solve for the radial basis function weights needed to maintain interpolating cardinal bases. As the algorithm progresses, new radii will be smaller and smaller as finer and finer adjustments are made to the space.

These methods have two potential downsides: potential increases in memory footprint and decreases in run-time performance. As detailed in [17], *pseudo-examples* are made up of linear-combinations of real *examples*. As such, they add only modest memory needs above those of the initial verb.

The execution time is $O(N_e+N_p)$ where N_e is the number of real *examples* and N_p the number of *pseudo-examples*. If N_p grows large, it may prove too expensive at run-time. At some point, it could cross over the cost of performing on-the-

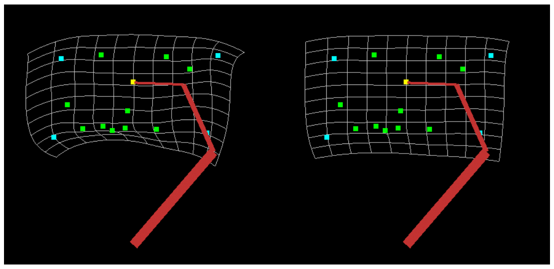

Figure 11: *Simple addition of examples vs. hybrid approach combining addition of pseudo-examples with optimal* $\mathbf{p}'$.

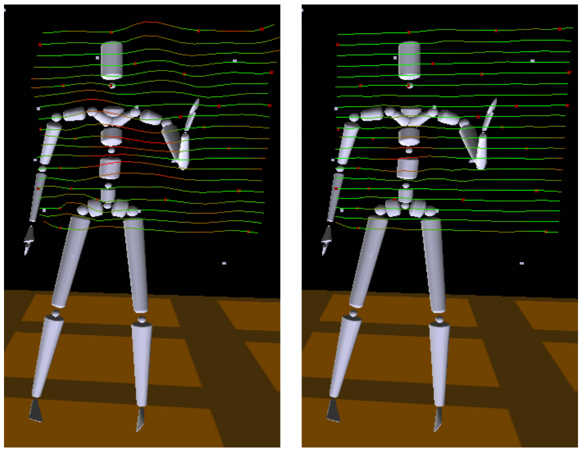

Figure 12: *Auto-refinement improves the interpolation spaces of the drawing. The left figure shows the pseudo-example-only technique, the right the hybrid technique.*

fly $\mathbf{p}$ optimization or traditional IK. A multiresolution RBF with recursive clustering, however, can mitigate this as reported in [13]. The run-time performance of a multiresolution RBF can be made to be $O(\log N)$.

What of the last few millimeters, however? The errors are driven towards zero, but are not driven to zero. For many applications, the level of accuracy will prove sufficient. For others, however, it will not. Hybrid approaches can ensure the final accuracy. One can use local search to find optimal $\mathbf{p}'$, though this is not guaranteed to have zero error. Traditional IK can be employed, or a combination of all three techniques. The *verb* will provide a very accurate initial guess, so the IK solution will converge in a stable manner in few iterations. Through use of the *verb*, traditional IK's downsides are alleviated. Note that this would only need to be employed in situations where extreme accuracy is required.

Figure 11 show the improvements of the space when the two variant auto-refinement algorithms (*pseudo-examples* only vs. hybrid *pseudo-examples* + optimal $\mathbf{p}'$) are run our our initial spaces. Figure 14 shows a plot of error improvements vs. added *pseudo-examples*. You can see that the initial spaces are greatly improved by these methods. Average error dropped from 6.85% to 2.39% for the *pseudo-example*-only approach and from 1.13% to 0.4% for the hybrid approach. Video Figure 1 shows *pseudo-example*-only process in motion for the 3-link arm. Likewise, Figures 12 and 14 show the improvements for the board drawing, improving the average and maximum error to (0.25%,0.69%) in the *pseudo-example*-only approach and (0.1%,0.48%) in the hybrid approach. Figures 13 and 14 shows the improvements for the 3D reach, improving the average and maximum error to (0.51%,1.41%) in the *pseudo-example*-only approach and (0.053%,0.96%) in the hybrid approach. Video Figure 2 shows a visualization of this improvement together with animated results. Figure 19* shows the improved board and reach plots in color, where color is used to show the relative quality of the various regions. The improved motions are mostly green, the color used to indicated low-error.

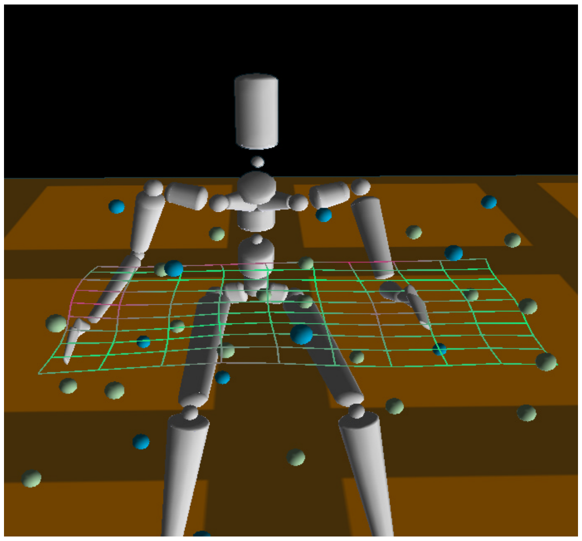

Figure 13: *Auto-refinement improves the interpolation spaces of the reaching motion. Blue spheres show the original examples and green the pseudo-examples.*

5. Results

All the results described so far and shown below run at many thousands of frames per second independent of rendering. So far, we have covered reaching motions for simple manipulators in the plane, human-like characters reaching to 2D manifolds in 3-space, and also reaching unrestricted in 3-space. There are many more types of IK motions achievable with *verbs*, many of which have both IK and non-IK control parameters.

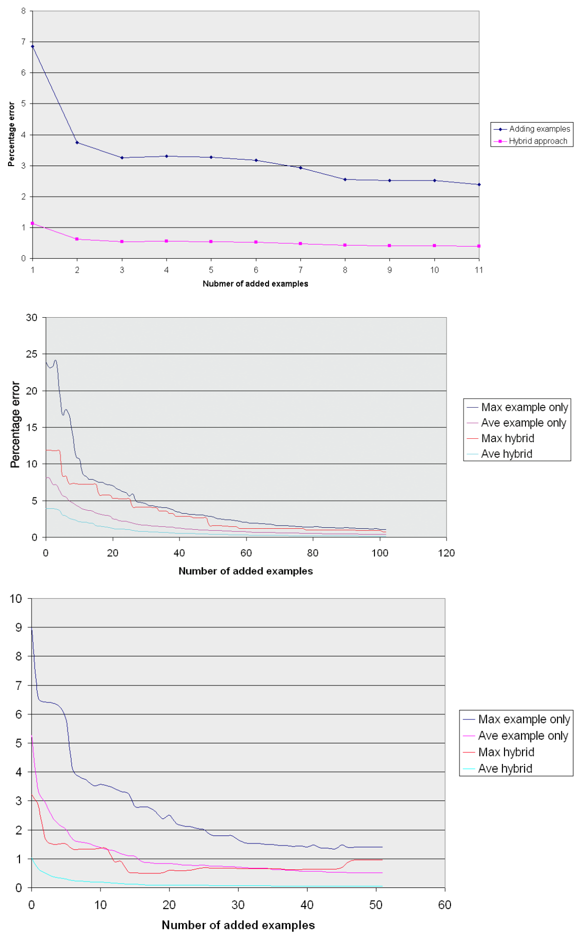

Figure 14: *A plot of improvements with respect to number of additional pseudo-examples for the 3-link, board, and reach.*

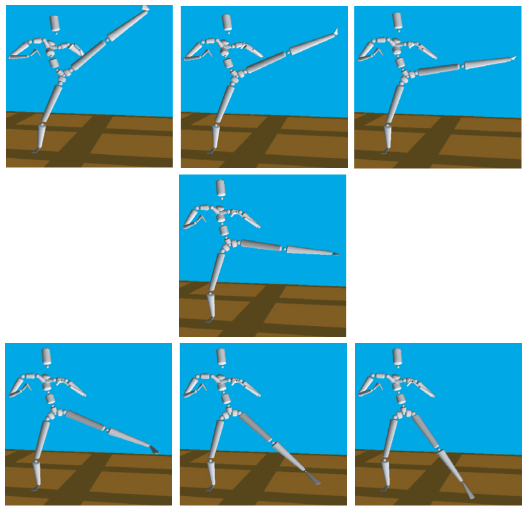

Figure 15: *A kicking verb. Example motions are shown in the center column.*

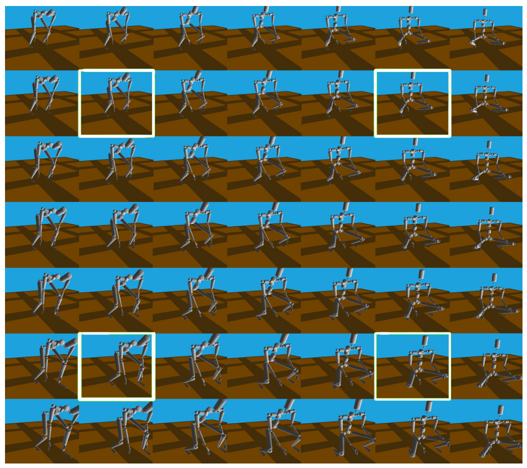

Figure 16: *The lifting motion combines IK control with visual cues indicating the weight of the box / difficulty the character has lifting it. Boxed poses are example poses.*

5.1. Action motions

The punch *verb* was constructed from 6 animated sequences. The height and direction of the character's punch can be controlled using this verb as seen in Video Figure 3. Likewise, the kick *verb* (Figure 15 and Video Figure 5) provides control over the kick's strike point.

The lifting motion was constructed from 4 animated *examples*, close-light, close-heavy, far-light, and far-heavy. A sampling of the lift is shown in Figure 16 and in Video Figure 4. This motion exhibits IK control combined with non-IK control.

5.2. Locomotion cycles

Locomotion cycles are often the most important animation sequences to an application. Combining IK with other concerns, such as a character's health level, can bring these sequences to life. Figure 17 and Video Figure 6 show a *Mechwarrior "Shadowcat"*, a character from a *Microsoft* produced game title. These figures shows a sampling of single poses from a sequence exhibiting control of walking slope and character damage.

The jogging motion shown in Figure 18 was created from 3 jogging sequences motion captured with a magnetic motion capture system. The control axis is turning radius and this motion, combined with motion transitioning, can be used to navigate a virtual character around a scene.

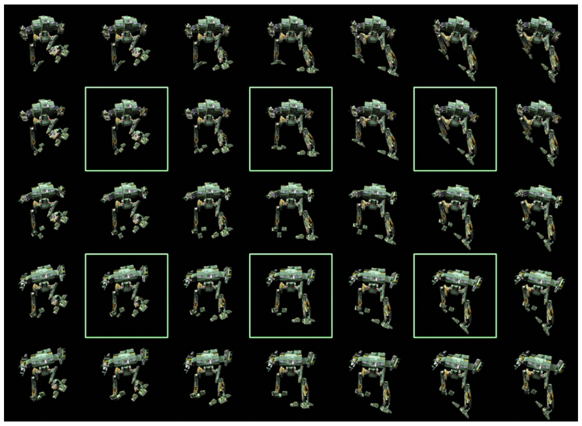

Figure 17: *Mechwarrior "Shadowcat" walk. Figures inside boxes are examples, the others are sampled interpolated and extrapolated motions.*

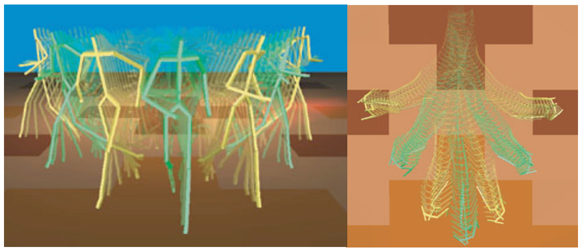

Figure 18: *Jogging motion using motion capture data. Example motions are indicated with a white dot.*

6. Conclusions

Interpolating motions has been proven to be an effective way to combine inverse-kinematic control alongside less quantifiable control axes such as perceived difficulty, health, etc. Unlike biomechanically driven systems, *example*-based motion interpolation fits into existing production pipelines for commercial animation houses due to its artist-centered focus. Its runtime memory and performance characteristics make it ideal for today's interactive real-time application domain.

Acknowledgements

Without an artist, artist-centered techniques cannot work. We owe a debt of thanks to Emmanuel Athans, our talented animator, for working with us on this project. You can view his work at *http://www.manopolis.com* and contact him by email at *eathans@manopolis.com*. Thanks also to David Thiel for help producing our video segments. Additional thanks to the *Mechwarrior* group at *Microsoft* for allowing us use of their data and models.

References

1. Amaya, K., Bruderlin, A., and Calvert, T. Emotion from motion. In Graphics Interface (May 1996), pp. 222–229. Proceedings of Graphics Interface 1996.
2. Arad, N., Dyn, N., Reisfeld, D., and Yeshurun, Y. Image warping by radial basis functions: Applications to facial expressions. Computer Vision, Graphics, and Image Processing 56, 2 (Mar. 1994), 161–172.
3. Bodenheimer, B., Rose, C. F., Rosenthal, S., and Pella, J. The process of motion capture: Dealing with the data. In Proceedings of the Eurographics Workshop on Computer Animation and Simulation (Sept. 1997), pp. 3–18.
4. Bruderlin, A., and Williams, L. Motion signal processing. In Computer Graphics (Aug. 1995), pp. 97–104. Proceedings of SIGGRAPH 1995.
5. Girard, M., and Maciejewski, A. Computational modelling for the computer animation of legged figures. In Computer Graphics (July 1985), pp. 263–270. Proceedings of SIGGRAPH 1985.
6. Gleicher, M. Retargetting motion to new characters. In Computer Graphics (July 1998), pp. 33–42. Proceedings of SIGGRAPH 1998.
7. Gullapalli, V., Gelfand, J. J., and Lane, S. H. Synergy-based learning of hybrid position/force control for redundant manipulators. In Proceedings of IEEE Robotics and Automation Conference, Minneapolis MN (June 1996), IEEE Press, Piscataway, NJ, pp. 3526–3531.
8. Guo, S., and Robergé, J. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In Proceedings of the 6th EuroGraphics Workshop on Animation and Simulation (Aug. 1996), pp. 95–107.
9. Lee, J., and Shin, S. Y. A hierarchical approach to interactive motion editing for human-like figures. In Computer Graphics (Aug. 1999), pp. 39–48. Proceedings of SIGGRAPH 1999.
10. Micchelli, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation 2 (1986).
11. Musse, S. R., Garat, F., and Thalmann, D. Guiding and interacting with virtual crowds. In Proceedings of the Eurographics Workshop on Computer Animation and Simulation (Sept. 1999), pp. 23–33.
12. Powell, M. J. D. Radial basis functions for multivariable interpolation: A review. In Algorithms for Approximation, J. C. Mason and M. G. Cox, Eds. Oxford University Press, Oxford, UK, 1987, pp. 143–167.
13. Rose, C. F. Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions. PhD thesis, Princeton University, Princeton, NJ, June 1999.
14. Rose, C. F., Cohen, M. F., and Bodenheimer, B. Verbs and adverbs: Multidimensional motion interpolation. IEEE Computer Graphics and Applications 18, 5 (Sept. 1998), 32–40.
15. Rose, C. F., Guenter, B., and Bodenheimer, B. Efficient generation of motion transitions using spacetime constraints. In Computer Graphics (Aug. 1996), pp. 147–154. Proceedings of SIGGRAPH 1996.
16. Ruprecht, R., and Müller, H. Image warping with scattered data interpolation. IEEE Computer Graphics And Applications 15, 2 (Mar. 1995), 37–43.
17. Sloan, P.-P., Rose, C. F., and Cohen, M. F. Shape by example, Mar. 2001. Proceedings of Symposium on Interactive 3D Graphics.
18. Turk, G., and O'Brien, J. F. Shape transformation using variational implicit functions. In Computer Graphics (Aug. 1999), pp. 335–342. Proceedings of SIGGRAPH 1999.
19. Unuma, M., Anjyo, K., and Tekeuchi, R. Fourier principles for emotion-based human figure animation. In Computer Graphics (Aug. 1995), pp. 91–96. Proceedings of SIGGRAPH 1995.
20. Wiley, D. J., and Hahn, J. K. Interpolation synthesis of articulated figure motion. IEEE Computer Graphics and Applications 17, 6 (Nov. 1997), 39–45.
21. Zhao, J., and Badler, N. I. Inverse kinematics positioning using non-linear programming for highly articulated figures. ACM Trans. Gr. 13, 4 (Oct. 1994), 313–336.

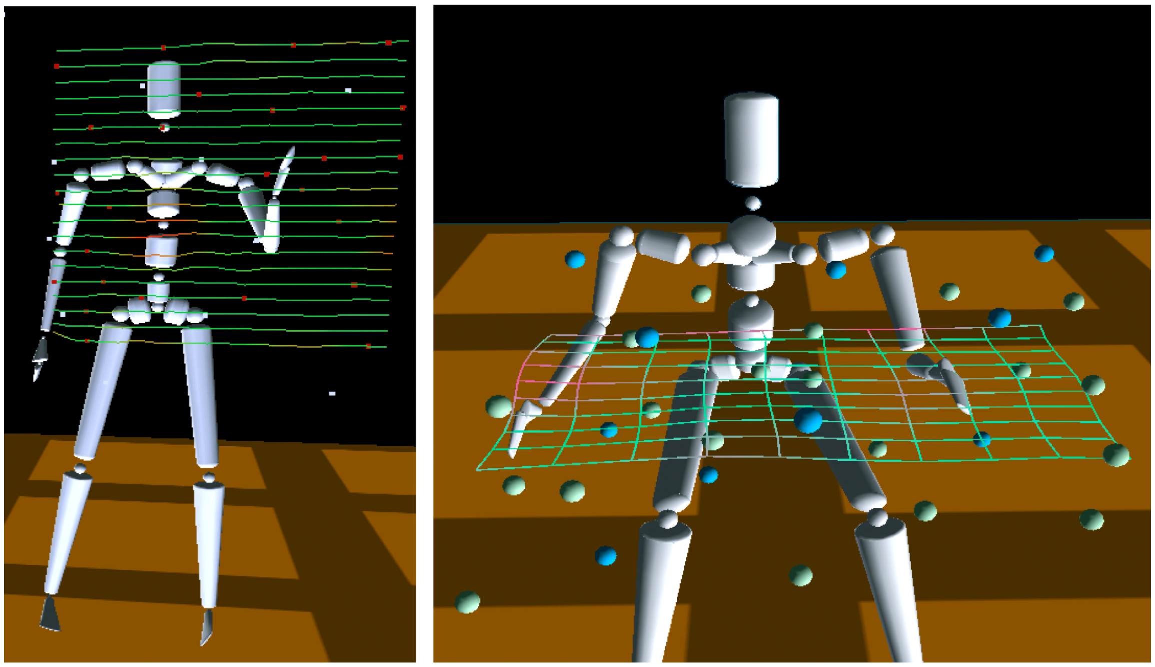

Figure 19: *This figure shows the 2- and 3-D improved reaches. The parameterization exhibits a high degree of fidelity. Green areas have lowest error and red the highest.*