Differential Blending for Expressive Sketch-Based Posing

A. Cengiz Öztireli[1] Ilya Baran[2,3,4] Tiberiu Popa[1] Boris Dalstein[5] Robert W. Sumner[2] Markus Gross[1,2]
[1] ETH Zurich [2] Disney Research Zurich [3] Belmont Technology, Inc. [4] Adobe Research [5] University of British Columbia

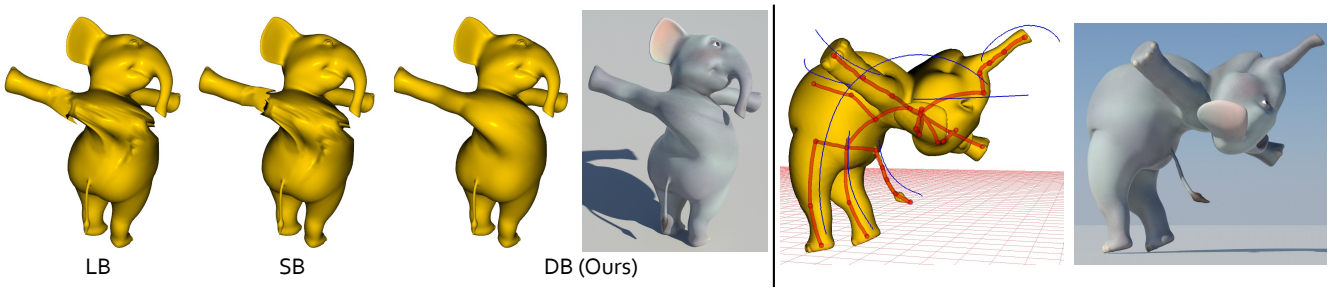

Figure 1: *(Left) Our differential blending supports poses that result in artifacts with existing skinning algorithms such as LB (linear blending of dual quaternions) and SB (screw linear blending of dual quaternions). (Right) By utilizing differential blending, our system allows artists to pose characters using familiar 2D cartoon animation concepts. The blue lines visualize the history of sketches used to create the expressive character poses.*

Abstract

Generating highly expressive and caricatured poses can be difficult in 3D computer animation because artists must interact with characters indirectly through complex character rigs. Furthermore, since caricatured poses often involve large bends and twists, artifacts arise with traditional skinning algorithms that are not designed to blend large, disparate rotations and cannot represent extremely large rotations. To overcome these problems, we introduce a differential blending algorithm that can successfully encode and blend large transformations, overcoming the inherent limitation of previous skeletal representations. Based on this blending method, we illustrate a sketch-based interface that supports curved bones and implements the *line-of-action* concept from hand-drawn animation to create expressive poses in 3D animation. By interpolating stored differential transformations across temporal keyframes, our system also generates caricatured animation. We present a detailed technical analysis of our differential blending algorithm and show several posing and animation results created using our system to demonstrate the utility of our method in practice.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Skeletal animation, posing, skinning, deformation

1 Introduction

Caricatured poses and expressive movement are the hallmark of hand-drawn animation, where pencil and paper afford the artist full creative freedom when crafting the shape and movement of animated characters. In three-dimensional animation, however, highly expressive and caricatured poses are more difficult to achieve, since the artist interacts with the character indirectly via the character's rigging controls. Two primary issues contribute to this problem. First, the character's rigging system, which defines the space of achievable poses, may not accommodate the range of expressiveness that the artist would like to achieve. Second, controlling a complex character rig can be unwieldy, requiring individual attention to tens or hundreds of rigging parameters just to achieve a single pose. These two problems are interconnected: in order to accommodate a wider range of expressiveness, character rigs become more and more complex, making them even harder to control. Even with this complexity, some poses are still not achievable, since the technical components of the rigging system are not designed to support them. As a result, expressive poses that come naturally from the fluid interaction of paper and pencil can be cumbersome or impossible to achieve using modern 3D animation tools.

Our research addresses these shortcomings in 3D animation by proposing a novel blending method for skeletal deformations and illustrating how it can be used to transfer the concept of *line-of-action* curves from 2D hand-drawn animation for creating highly expressive poses with intuitive sketch-based controls in 3D animation. In 2D animation, these curves serve as a guide to convey the composition, balance, energy, and dynamics of the character's pose (Figure 2) [Blair 1994]. By interpreting these curves for 3D skeletal deformations, our system allows fast and intuitive creation of highly expressive poses that are notoriously difficult to obtain with complex classical rigs (Figure 3).

The core technical challenge of developing such a system lies in blending skeletal transformations. Because highly expressive poses involve large bends and twists, the rigging system is forced to blend large, disparate rotations in complex regions such as the shoulder where vertices are influenced by multiple portions of the skeleton. Due to ambiguities inherent in the representation of rotations, blending algorithms used by existing rigging systems fail to give smooth and intuitive results in this case. For example, dual-quaternion blending [Kavan et al. 2008], accepted as one of the best algorithms, interpolates rotations along geodesic curves. Interpolation from a fixed rotation in the shoulder to different rotations along a highly deformed arm may follow very different geodesic paths, although the arm rotations are similar to one another. Because the interpolation paths are different, nearby vertices in the shoulder region may be transformed very differently. Refining the

Figure 2: *In 2D animation, line-of-action curves capture the character's overall motion and balance. These curves naturally match the character's skeleton and provide a guide to the artist when fleshing out the full drawing.* ©*Disney*

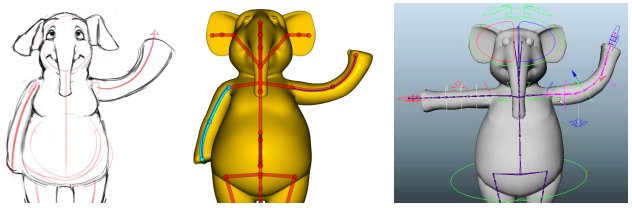

Figure 3: *(Left) 2D sketch using line of action. (Middle) Similar pose created using our sketch based system. (Right) A snapshot of a professional Maya rig (with around* 100 *degrees of freedom) of the same character. Our system offers an interaction very close to 2D sketching to create expressive skeleton-based poses, while state-of-the-art methods can only approximate such poses through sophisticated rigging techniques and cumbersome controls.*

skeleton with additional bones [Mohr and Gleicher 2003] does not address the problem, since transformations are concatenated along joint chains, leading back to large rotation disparities. This subtle problem results in significant, unacceptable artifacts when highly expressive poses are created using traditional rigging algorithms, as shown in Figure 1(left). As an added problem, existing methods are not designed to represent rotations greater than 2π, leading to additional artifacts when the user desires extremely large bending or twisting.

To solve these problems, we propose a new blending technique specifically designed for large and disparate transformations, as our main contribution. Our *differential blending* method represents all transformations in a differential manner and computes averages of the differential transformations, which are then composed to get the final blended transformation. Since averaging is used to combine only small transformations, geodesic averaging always gives the correct result and the artifacts that arise from blending disparate transformations or ones that are too large to be represented correctly are avoided. By interpolating the differential transformations between animation keyframes, our system naturally extends to animated deformations. We provide a detailed analysis of our new differential blending algorithm and show that it results in smooth and intuitive skeletal deformations.

After explaining various components of our posing and animation system in Section 3, we move on to the technical core of our work, where we introduce *differential blending* in Section 4, and show key mathematical properties of it in Section 5. Our figures use the following convention: underlying skeleton is rendered in red, the selected bones are rendered in cyan and, where applicable and unless otherwise stated, the sketched curves are shown in dark blue.

2 Related work

Skeleton-based deformations. Although invented in the 80s, skeleton-based skinning [Magnenat-Thalmann et al. 1988] remains the most popular method for animating articulated characters in games and films, due to its simplicity, efficiency, and intuitiveness. However, skinning methods have many problems, and over the years a large set of techniques have been developed to address various shortcomings of the original method [Alexa 2002; Kavan and Žára 2005; Yang et al. 2006; Forstmann and Ohya 2006; Forstmann et al. 2007; Kavan et al. 2008; Jacobson and Sorkine 2011] ranging from alleviating deformation artifacts such as *collapsing elbows* and *candy-wrapper joints* to adding robust support for more complex deformations such as *bone twisting* and *bone stretching*.

The relatively rigid structure of traditional skeletons with piecewise linear elements makes it difficult to create expressive poses common in 2D animation that exhibit larger, full-body curves. For more artistic control, more bones and joints can be added to the skeleton, but this results in additional effort and increased complexity. A solution to this limitation is a skeleton with *stretchable/twistable* [Jacobson and Sorkine 2011] or *curved bones* [Yang et al. 2006; Forstmann and Ohya 2006; Forstmann et al. 2007]. Even though curved bones can capture the expressiveness desired by the artist, the current skinning methods that deform the model to conform with the new pose often fail in the presence of large rotations leading to visual artifacts as illustrated in Figure 1. While this is also a limitation of traditional skeleton and skinning methods, it is more prevalent when applied to expressive cartoon like poses that are typically exaggerated for artistic purposes. Our novel differential skinning method overcomes the shortcomings of previous skinning methods allowing the user to transfer his or her full artistic talent and intention to the animation.

Sketch based modeling and animation. A sketching interface is an intuitive and familiar modeling paradigm for artists. Hence, it comes as no surprise that sketch based interfaces for modeling and deformation have received a lot of attention [Olsen et al. 2008]. Several works [Akeo et al. 1994; Zeleznik et al. 1996; Igarashi et al. 1999; Nealen et al. 2007; Gingold et al. 2009; Gingold et al. 2009] focus on creating 3D content from scratch, while others use sketching as a means of deforming existing shapes [Kho and Garland 2005; Nealen et al. 2005; Zhou et al. 2005; Zimmermann et al. 2008; Kraevoy et al. 2009]. These techniques deform shapes by mapping the sketched curves to some salient surface curves on the model [Olsen et al. 2008] such as silhouette or ridge curves and use these as geometric constraints. Although these methods are suited for direct mesh manipulation and detail editing, they do not follow the skeleton based animation pipeline, which is the most widely used and mature method for character posing and animation due to its predictable and intuitive nature and easy keyframing. For our purposes, a skeletal representation also matches well with the semantics of 2D curves.

Closer to our work, a further set of sketch based methods are tailored to character posing and animation [Hoshino and Hoshino 2001; Thorne et al. 2004; Mao et al. 2005; Mao et al. 2007; Davis et al. 2007; Mao et al. 2009]. However, most of these methods target animation prototyping, manipulating trivial models such as stick figures and not detailed 3D characters that have complex and rich geometry. In contrast, we aim to couple a sketch-based interface that intuitively maps the sketched curves to the underlying skeleton to produce expressive poses with a novel skinning method that can be used to deform rich and complex geometry without visual artifacts.

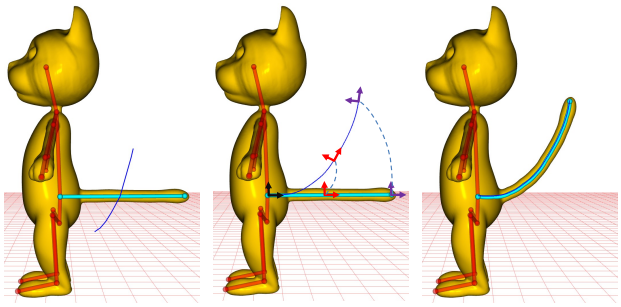

Figure 4: *(Left) The user draws a stroke to select a bone, (middle) transformations from the frames on the projected curve for the bone to those on the stroke are computed, and (right) are used to generate the final deformation of the bone and the model.*

3 The Posing and Animation System

Our system is designed for intuitive controlling of skeletal deformations via sketching, to mimic the 2D concepts familiar to artists. We integrate and adapt various existing components [Kho and Garland 2005; Nealen et al. 2005] in the context of skeletal deformations to achieve this goal.

The system accepts as input a 3D model with a traditional skeleton rig that can be created using any standard rigging method [Baran and Popović 2007] or commercial product available such as Maya, making it very flexible to use. It then places samples along the bones to obtain a skeleton with curved bones (please refer to Section 4.4 for details). At posing time, the strokes drawn by the artist are first mapped onto a subset of previously selected skeleton bones. The selected bones are subsequently deformed to match the user's curve. Finally, the 3D model itself is deformed to match the new pose. This process is illustrated in Figure 4 and the following sections presents these steps in more detail. The user can keyframe several poses to create expressive animations as shown in the accompanying video.

3.1 User Interface

The user interaction starts with a sketch-based selection of the bones that the user would like to deform (Figure 4 (left)). The user can choose to select a single bone, a connected network of bones or all bones in the hierarchy. Although automatic methods to infer selections from the strokes could be used, relying on the user for this task provides a more predictable and controllable behavior. Next, the user can sketch a curve to shape the bones (Figure 4 (middle)). The skeleton and model are deformed accordingly for instant feedback (Figure 4 (right)). If the result is not satisfactory, the artist can repeat the stroke until the desired result is reached. This selection and sketching process is then repeated for other parts of the character until the desired pose is realized. The accompanying video contains a recording of a modeling session.

To provide more flexibility to the user, we added a few further user interface features. We provide a tool for twisting selected bones (Figure 5 (c)) by simply right-clicking. As stretching of the bones depending on the stroke's length can be counter-intuitive in some cases, we provide the user with the option of preserving the original length of the bones (Figure 5 (a)). Furthermore, depending on the modeling intention of the current stroke, deforming the unselected children of selected bones is also provided as an option (Figure 5 (b)).

3.2 Stroke Representation

A user stroke obtained using a mouse or a sketching pen is represented by a series of points. The goal is to use these points to construct a parametric curve that can be easily used to compute the local coordinate frames along the curve in order to transfer them onto the selected bones. To accomplish this we fit a clothoid spline [Baran et al. 2010] to the stroke points. The clothoid spline has been shown to generate aesthetically pleasing curves due to the linear variation of curvature over arc length.

3.3 Stroke Mapping

A key ingredient of our system is the intuitive mapping and deformation transfer between the user strokes and the selected bones. Figure 4 illustrates this process on a simple example. The tail bone is mapped onto the sketched curve in 2D by projecting the bone onto the projection plane and performing a direct arc-length cross parametrization [Kho and Garland 2005]. We solve the ambiguity of the initial correspondence by mapping the first point on the sketched curve onto the closest end of the bone. The bone is then deformed to match the curve and these transformations are stored on the bones for the skinning step.

This mapping only works when the selected bones form a chain (Figure 6 (right)). However, in order to mimic the line-of-action sketching used in traditional 2D animation, the system needs to be more general and allow the user to select bone configurations which are trees and not just simple chains as shown in Figure 6 (left). Due to topological incompatibilities, the mapping between such a tree and the sketched curve is ill-posed. Inspired by the traditional 2D rendering pipeline, we developed a method to perform intuitive deformation transfer from a sketched curve to an arbitrary user selection of bones.

As illustrated in Figure 2, the line-of-action (rendered in red) is *an imaginary line extending through the main action of the pose*. If we lift the line-of-action concept from 2D to 3D, the 3D line of action (that we call the support curve) actually goes through the main body, or the part of the body that is selected. With respect to the rest pose, the support curve corresponds to the line that best fits, in the least square sense, to the joints of the selected bones. The exact positioning of the support line is not critical for the system. We use the support curve as a common parametrization domain for both the selected skeleton subtree and the sketched curve. To find the correspondence between a point on the skeleton and the support curve, the bones are projected onto the support curve. Next, we compute a direct arc-length cross-parametrization [Kho and Garland 2005] between the projected support curve and the sketched curve. Combining these steps, we can compute a map between the selected skeleton bones and the sketched curve. This method results in very intuitive deformations in practice as illustrated in the figures and the video.

In our system, the user can decide which mode he or she would like to use. While the line of action mode gives a more general control over the body, resembling line-of-action sketching, the direct mode that can only be applied to chains, enables to easily draw multiple bones in detail, as illustrated in Figure 6.

4 Differential Blending

Once the deformation from the sketched curved is transferred to the skeleton, the next step is to use a skinning method to deform the 3D model. Skinning methods compute the new location of a given vertex by blending the transformations of several bones. Blending

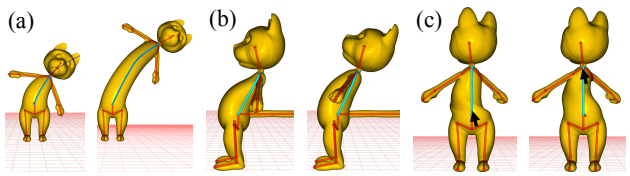

Figure 5: *The user can (a) (dis-)allow stretch and (b) rotation of children bones, and (c) twist bones according to the position clicked on the bone.*

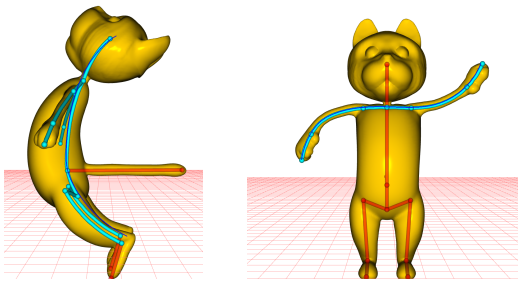

Figure 6: *Our system supports (left) line-of-action sketching, and (right) detailed bone shape sketching.*

transformations is non-trivial, particularly when the angular difference between rotations exceed π. However, exaggerated poses that occur frequently in expressive animations are very often large. In this section, we present a new method for blending transformations that avoids the artifacts when using existing methods. After explaining the problem, we introduce our method, and show how it can be used in practice in our posing and animation system.

4.1 Deformations with Skeletons

A traditional skeleton is represented as a tree of segments called bones, which store transformations. Each mesh vertex $\mathbf{x}$ is bound to one or more bones with skinning weights $w_b(\mathbf{x})$, non-negative real numbers that sum to one, $\sum_{b=1}^{n_b} w_b(\mathbf{x}) = 1$ for n_b number of bones. A skeletal deformation is given by the set of rigid (or more generally, affine) transformations $\mathbf{q}_b$ that takes the bone b from its rest pose to the current pose. The skeletal deformation is used to compute the deformed location of a mesh vertex by transforming the vertex with a weighted average of the $\mathbf{q}_b$'s with the weights $w_b(\mathbf{x})$, which we denote by $A(w_1(\mathbf{x}), \cdots, w_{n_b}(\mathbf{x}); \mathbf{q}_1(\mathbf{x}), \cdots, \mathbf{q}_{n_b}(\mathbf{x})) = A(w_b(\mathbf{x}); \mathbf{q}_b)$.

Producing the types of expressive poses we seek with rigid bones requires a large number of bones to accurately capture the highly deformed skeleton shapes common in 2D animations. A new set of skinning weights needs to be computed for each added bone, making the skinning process tedious. A more viable alternative is using *curved bone* based skeletons [Yang et al. 2006; Forstmann et al. 2007]. In these structures, the bones represent a continuous set of transformations. The transformations are parametrized along the bone curves, and each vertex is attached to a particular parameter $s_b(\mathbf{x}) \in [0, 1]$ for bone b. Hence, the transformations to be blended become functions of vertex positions, resulting in the blending formula $A(w_b(\mathbf{x}); \mathbf{q}_b(\mathbf{x})) = A(w_b(\mathbf{x}); \mathbf{q}_b(s_b(\mathbf{x})))$. Due to its flexibility, we employ such a skeleton with curved bones as the underlying representation for the deformations.

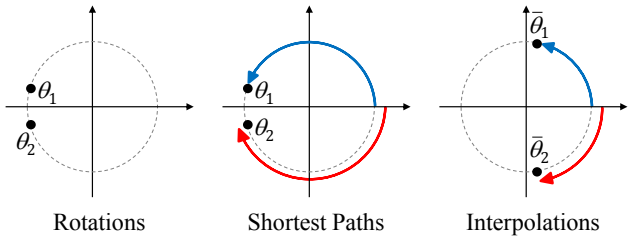

Figure 7: *Although rotations θ_1 and θ_2 for two close vertices are close (left), the shortest paths when interpolating them with the identity rotation ($\theta = 0$) are far apart (middle), leading to a discontinuity in the blended rotations (right).*

4.2 Existing Transformation Blending Methods

Once skinning weights and (for curved bones) parameters of vertices on bones are determined, the next step is to perform vertex skinning by blending the transformations stored on bones. The simplest existing approach to blending is linear blend skinning (LBS), where the transformations are represented as matrices and the blending operator A is a weighted linear combination of matrices. Although simple and efficient, LBS produces well-known artifacts on deformed surfaces [Kavan et al. 2008]. These artifacts stem from the fact that LBS averages transformations without respecting the manifold of rigid transformations, $\mathbf{SE(3)}$, by computing distances between transformations in the embedding space of this manifold. Instead, intrinsic averages on this manifold can be computed by representing transformations with dual quaternions [Kavan et al. 2008]. Since the distances are measured on $\mathbf{SE(3)}$, this method leads to interpolations along geodesics. Hence, the blended transformation minimizes the sum of intrinsic distances. Both blending methods exhibit two important limitations.

Discontinuity problem. Although using *shortest paths* for blending results in intuitive blended transformations for many cases, it can lead to discontinuities when the blended transformations are far apart. To illustrate this problem, we consider the simple case of 2D planar rotations in Figure 7. For this case, a rotation can be represented by a single angle θ and shortest path interpolation corresponds to simple averaging of the angles, $\theta(t) = \theta_0(1-t) + \theta_1 t$, $t \in [0, 1]$. Assume that we are given two close vertices $\mathbf{x}_1$ and $\mathbf{x}_2$, for which the rotations are given by interpolations between the identity ($\theta = 0$) and the angles θ_i, $i = 1, 2$. Hence, the interpolated rotations are $\bar{\theta}_1 = \theta_1 t$ and $\bar{\theta}_2 = \theta_2 t$, assuming the weight t for both are the same, since the vertices are close to each other on the mesh.

For a smooth variation of rotations stored on the bones, we can assume that θ_1 and θ_2 are close, and expect the blended rotations $\bar{\theta}_1$ and $\bar{\theta}_2$ for two vertices to be also close. Let the angles be given by $\theta_1 = \pi - \epsilon$ and $\theta_2 = \pi + \epsilon$, for a small angle ϵ (Figure 7, left), and $t = 0.5$. The interpolated value for the first vertex is simply half the angle $\bar{\theta}_1 = \frac{\pi - \epsilon}{2}$. However, since θ_2 is bigger than π, the interpolation will choose the shorter path from 0 to θ_2 (Figure 7, middle), leading to $\bar{\theta}_2 = \frac{3\pi}{2} + \frac{\epsilon}{2}$ (Figure 7, right). Hence, the two close vertices will have very different rotations, leading to discontinuities on the surface mesh. An example of this behavior is shown in Figure 8 for a $3D$ model.

Large rotations problem. Another problem of shortest path interpolation results from not being able to represent rotations bigger than 2π with dual quaternions. For angles $\theta > 2\pi$, the representation will reduce it to a smaller angle, leading to incorrect interpolations.

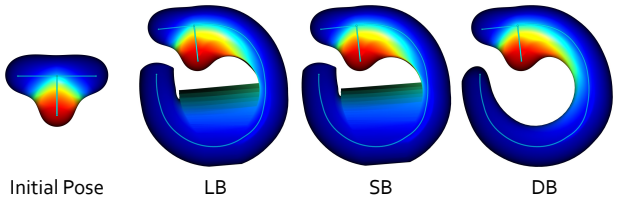

Figure 8: *Blending continuous rotations on the bones leads to discontinuities on the mesh with LB (linear blending of dual quaternions) and SB (screw linear blending of dual quaternions) due to their shortest path property, while DB (differential blending) generates a continuous blending. Skinning weights are plotted for the middle bone to indicate the affected vertices.*

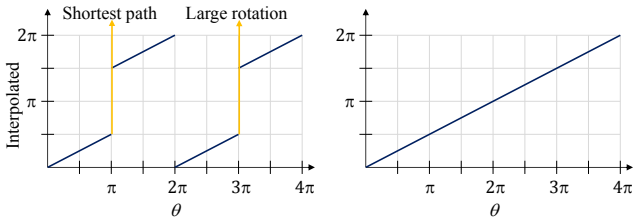

Figure 9: *Graphs of interpolation between θ and the identity transformation. Shortest path interpolation is discontinuous when the shortest path changes or when the rotation is too large to be represented (left), while differential blending correctly interpolates for all angles (right).*

A graph of the interpolated angle between 0 and θ is given in Figure 9 (left). The graph is discontinuous with jumps when the shortest path changes, or when the angle becomes too large to be correctly represented. These problems extend to rigid transformations in $3D$. They are especially pronounced when expressive poses with large rotations are needed. An example taken from a real animation is presented in Figure 10. This behavior of existing skinning algorithms severely limits the space of poses that can be generated via our interface and the expressive quality of the resulting animations. To overcome these problems, we develop a new blending technique, described in the next section.

4.3 Differential Blending of Transformations

The discontinuities while blending with existing methods stem from trying to average transformations that are far apart from each other, or too large to be represented correctly. Hence, if only small transformations are involved in the averages, the shortest path will always give the most intuitive result. Following this observation, our main idea is to represent all transformations to be blended as compositions of small transformations. These differential transformations are blended with shortest path methods, and the blends are composed to get the final weighted average.

Following the example in Figure 7, we illustrate how this idea avoids discontinuities and large rotation problems by restricting the transformations to rotations in a $2D$ plane in Figure 11. We assume that the rotation of angle θ is to be averaged with the identity transformation (i.e. rotation of 0 angle), which should simply give $\theta/2$. First, θ is divided into two smaller angles $\Delta\theta_1$ and $\Delta\theta_2$ (Figure 11 (b)). These are then interpolated with 0 in the shortest path to give $\Delta\theta_1/2$ and $\Delta\theta_2/2$ (Figure 11 (c)), which are finally composed to get the average rotation (Figure 11 (d)). The result is in contrast with that of direct shortest path interpolation of θ as illustrated in Figure 11 (e).

This idea can be extended to rigid transformations in $3D$. In this work, we represent transformations with dual quaternions and the multiplication $\mathbf{q}_1\mathbf{q}_2$ and blending operator A are defined according to [Kavan et al. 2008]. Given n transformations $\mathbf{q}_i$ to be averaged, their weights in the average w_i, and their decompositions into small transformations $\mathbf{q}_i = \mathbf{q}_{i1}\cdots\mathbf{q}_{il}$, the blended transformation is given by the following formula

$$A_\Delta(w_i;\mathbf{q}_i) = \prod_{j=1}^{l} A(w_i;\mathbf{q}_{ij}), \tag{1}$$

where $A(w_i;\mathbf{q}_{ij}) = A(w_1,\cdots,w_n;\mathbf{q}_{1j},\cdots,\mathbf{q}_{nj})$. Here, the intrinsic averaging function A gives the correct average since the transformations $\mathbf{q}_{ij}$ are small. By composing these averages, the final average is computed. Differential blending is illustrated in Figure 12 for the case of two transformations. Starting from the identity transformation, each small transformation in the composition of the transformations to be blended can be seen as a step on $\mathbf{SE(3)}$. When the number of differentials are not the same for all $\mathbf{q}_i$'s such that $\mathbf{q}_i = \mathbf{q}_{i1}\cdots\mathbf{q}_{il_i}$, we define A_Δ by taking $l = \max_i(l_i)$, and appending identity transformations to $\mathbf{q}_i$'s such that all l_i's become the same and equal to l. A detailed analysis and properties of differential blending is provided in Section 5.

4.4 Skinning with Differential Blending

In our system we use a curved bone based skeletal structure. The skeleton consists of a tree of bones. Unlike traditional skeletons, in a curved bone skeleton, each bone is implicitly subsampled into a sequence of nodes that store local transformations (Figure 13 (left and middle)). These transformations are updated based on the user sketch and used to determine the position of the vertices bound to their bone. In order to use our differential blending strategy that avoids blending large deformations, rather than storing the full transformation at each node, we store the differential transformation $\Delta\mathbf{q}$, which is the difference between the current sample transformation $\mathbf{q}$ and the previous $\mathbf{q}_{prev}$ on the path from the root to $\mathbf{q}$, $\Delta\mathbf{q} = \mathbf{q}\mathbf{q}^*_{prev}$ (Figure 13 (middle)).

Hence, the skeleton stores the root transformation $\mathbf{q}(0)$, and a number n_s of differential transformation samples $\Delta\mathbf{q}_{bi} = \Delta\mathbf{q}_b(s_i)$, $s_i \in [0,1]$ on each bone b. Here, $s_i \in [0,1]$ denotes the parameters of the samples on the bones. Since we assume the same sampling scheme for all bones, the set of s_i's for all bones is the same. We could reconstruct any sample transformation $\mathbf{q}_b(s_i)$ by simply composing all differential transformations on the path from the root as $\mathbf{q}_b(s_i) = \Delta\mathbf{q}_b(s_i)\cdots\mathbf{q}(0)$. This representation avoids the large rotation ambiguity by effectively tracking where each large rotation comes from and representing it as a composition of infinitesimal (in practice, finite but small) rotations.

Each vertex $\mathbf{x}$ on the mesh is assigned a parameter $s_b(\mathbf{x}) \in [0,1]$ for each bone b. In general, this parameter does not coincide with the sample parameters. To get the transformation at the parameter $s_b(\mathbf{x})$, we use linear interpolation of the transformation samples around it, which implies that $\Delta\mathbf{q}_b\left(s_b(\mathbf{x})\right) = \Delta\mathbf{q}_b(s_i)^u$. Here, s_{i-1} and s_i are the sample parameters before and after $s_b(\mathbf{x})$, respectively, when starting the path from the root, and $u = |s_b(\mathbf{x}) - s_{i-1}|/|s_i - s_{i-1}|$ (Figure 13 (right)).

We assume that initially all bones are straight and thus a classical skeleton and a set of vertex weights $w_b(\mathbf{x})$ are provided. Given these inputs, the bones are sampled, and the parameters $s_b(\mathbf{x})$ of the vertices on the bones are computed. Although more sophisticated methods [Jacobson and Sorkine 2011] can be used to compute these attachment points, we found that it is sufficient to use a simple pro-

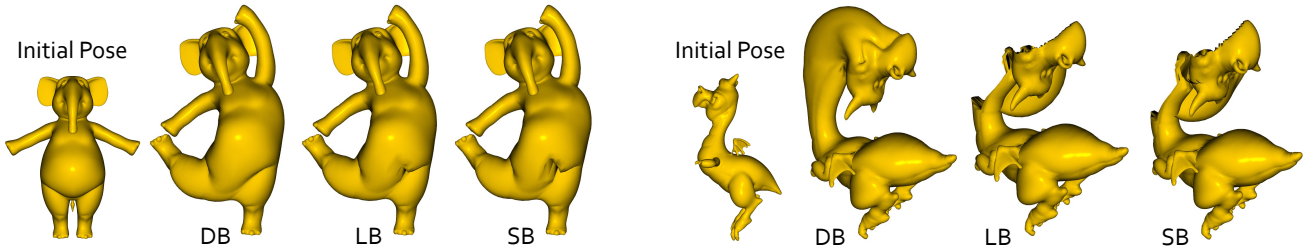

Figure 10: *The discontinuities due to blending with existing methods are avoided by differential blending. DB (differential blending), LB (linear blending of dual quaternions), SB (screw linear blending of dual quaternions).*

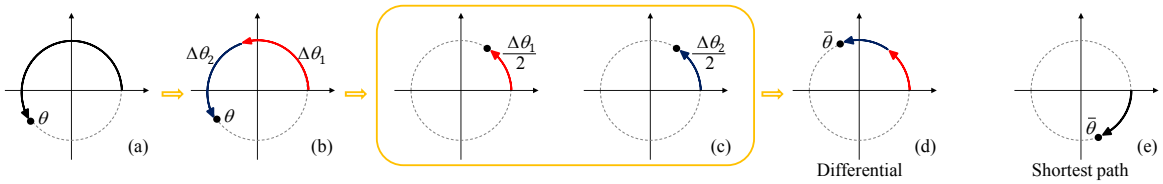

Figure 11: *(a) In order to compute the average of θ and 0, (b) θ is divided into two smaller rotations, (c) which are averaged with 0 using shortest path interpolation, and (d) then composed to get the correct interpolated angle $\bar{\theta}$. This is in contrast to (e) direct shortest path interpolation of θ and 0.*

jection of the vertex locations onto the bones in the rest pose. Once these are completed, the rig is ready to be used for posing.

4.5 Animation with Interpolation of Differentials

Apart from supporting differential blending for posing, the differential transformation samples can also be utilized for animation by interpolating between two or more poses, e.g., for keyframing. Because all of the $\Delta\mathbf{q}$'s represent small transformations, there is no rotation ambiguity, and we can simply interpolate the individual $\Delta\mathbf{q}$'s. With enough samples, we could even blend $\Delta\mathbf{q}$'s linearly as matrices: for small transformations, linear blending does not suffer from artifacts. However, to keep the number of samples small, we represent $\Delta\mathbf{q}$'s as dual quaternions and use dual quaternion linear blending [Kavan et al. 2008].

5 Analysis of Differential Blending

We have shown how differential blending can be used to eliminate discontinuities in blending transformations. However, this property in itself is not sufficient for differential blending to be useful in practice. In this section, we discuss some of the most important properties that have been shown to be essential for high quality blendings [Kavan et al. 2008]. Namely, we explain that differential blending produces unit dual quaternions that represents *valid rigid transformations*, is *coordinate invariant*, and gives results close to *shortest path interpolation* when shortest path is indeed the correct path for interpolation. The first property is trivial to show since composition and intrinsic averaging always generate valid rigid transformations. Hence, below we discuss the latter two properties.

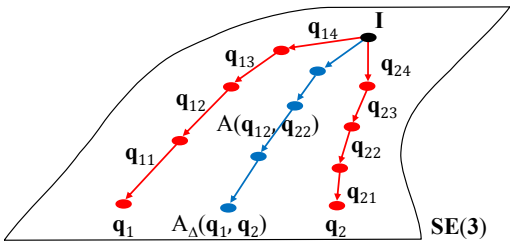

Figure 12: *A schematic representation of differential blending of the transformations $\mathbf{q}_1$ and $\mathbf{q}_2$. Each pair $\mathbf{q}_{1i}$, $\mathbf{q}_{2i}$ of differential transformations on the paths is averaged using shortest path interpolation (written as $A(\mathbf{q}_{1i}, \mathbf{q}_{2i})$). Composition of these differential averages gives the final interpolation, $A_\Delta(\mathbf{q}_1, \mathbf{q}_2)$ (the weights are omitted for brevity).*

5.1 Coordinate Invariance

For intuitive skinning, the coordinate system chosen should not have an effect on the deformed mesh other than a global rigid motion. This implies that changing the coordinate system in which the transformations are defined before or after blending should give the same result. Formally, this can be written as $A(w_i; \mathbf{m}\mathbf{q}_i\mathbf{m}^*) = \mathbf{m}A(w_i; \mathbf{q}_i)\mathbf{m}^*$. Hence, coordinate invariance can be proved by showing right and left invariance given by $A(w_i; \mathbf{q}_i\mathbf{m}) = A(w_i; \mathbf{q}_i)\mathbf{m}$ and $A(w_i; \mathbf{m}\mathbf{q}_i) = \mathbf{m}A(w_i; \mathbf{q}_i)$, respectively.

Differential blending depends on the intrinsic averaging function

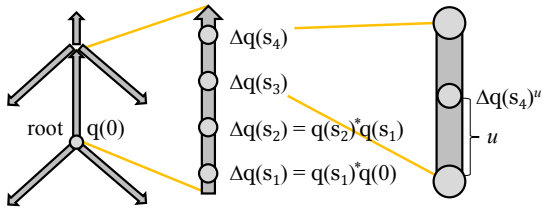

Figure 13: *We utilize (left) a hierarchical skeleton where (middle) each bone stores a set of differential transformation samples. These differentials are interpolated (right) to get the differential transformation for a parameter on a given bone.*

A, which can be shown to be right and left invariant [Kavan et al. 2008]. Denoting the differential representation of $\mathbf{m}$ as $\mathbf{m} = \mathbf{m}_1 \cdots \mathbf{m}_h$, right invariance of A_Δ can be proved as follows:

$$\begin{aligned} A_\Delta(w_i; \mathbf{q}_i\mathbf{m}) &= A_\Delta(w_i; \mathbf{q}_{i1} \cdots \mathbf{q}_{il}\mathbf{m}_1 \cdots \mathbf{m}_h) \\ &= \prod_{j=1}^{l} A(w_i; \mathbf{q}_{ij}) \prod_{k=1}^{h} A(w_i; \mathbf{m}_k) \\ &= A_\Delta(w_i; \mathbf{q}_i) \prod_{k=1}^{h} \mathbf{m}_k \\ &= A_\Delta(w_i; \mathbf{q}_i)\mathbf{m}. \end{aligned} \quad (2)$$

Left invariance can be proved analogously, implying that differential blending of transformations is coordinate invariant.

5.2 Deviation from Shortest Path Interpolation

For some sets of transformations, intrinsic averages where distances are measured along the shortest paths do not violate the continuity of skinning. In those cases, we would like the differential blending to give results close to shortest path based intrinsic averages. For shortest path interpolation of two transformations, there exists a simple closed form solution that we denote by $I(t; \mathbf{a}, \mathbf{b}) = (\mathbf{b}\mathbf{a}^*)^t \mathbf{a}$. The differential interpolation is then given as $I_\Delta(t; \mathbf{a}, \mathbf{b}) = \prod_{i=1}^{l} (\mathbf{b}_i\mathbf{a}_i^*)^t \mathbf{a}_i$.

To quantify the deviation of differential interpolation from shortest path interpolation, we consider $\mathbf{e} = I(t; \mathbf{a}, \mathbf{b})I_\Delta(t; \mathbf{a}, \mathbf{b})^*$. Note that due to right invariance of I and I_Δ, this can also be written as $I(t; \mathbf{I}, \mathbf{b}\mathbf{a}^*)\mathbf{a}\,(I_\Delta(t; \mathbf{I}, \mathbf{b}\mathbf{a}^*)\mathbf{a})^* = I(t; \mathbf{I}, \mathbf{b}\mathbf{a}^*)\mathbf{a}\mathbf{a}^* I_\Delta(t; \mathbf{I}, \mathbf{b}\mathbf{a}^*)^* = I(t; \mathbf{I}, \mathbf{b}\mathbf{a}^*)I_\Delta(t; \mathbf{I}, \mathbf{b}\mathbf{a}^*)^*$, where $\mathbf{I}$ denotes the identity transformation. Hence, we can equivalently consider the interpolation of a transformation $\mathbf{q} = \mathbf{q}_1 \cdots \mathbf{q}_l$ with the identity transformation, for which the difference becomes $\mathbf{e} = \mathbf{q}^t \left(\mathbf{q}_1^t \cdots \mathbf{q}_l^t\right)^*$.

Computationally analyzing or deriving bounds for the deviation $\mathbf{e}$ is very challenging due to the dimension of the space of variables to consider for the former, and the intractability of the solutions of the equations for the latter. In this section, we perform a computational analysis on rotational deviation for the case where two differentials make up the transformation such that $\mathbf{q} = \mathbf{q}_1\mathbf{q}_2$. In addition to being illustrative, the results of this analysis will be used to validate our theoretical analysis in Section 5.3.

When considering rotations, dual quaternions can be replaced by quaternions. Hence, below, $\mathbf{q}$ and $\mathbf{q}_i$ denote regular quaternions. Our goal is to computationally analyze the rotation angle of $\mathbf{e} = \mathbf{q}^t \left(\mathbf{q}_1^t\mathbf{q}_2^t\right)^*$ with the constraint that $\mathbf{q} = \mathbf{q}_1\mathbf{q}_2$. The first step of this analysis is deriving $\mathbf{e}$ such that it can be represented as a function of a few variables, which are sampled to get the maximum deviation. Denoting the quaternions by $\mathbf{q}_i = \cos(\theta_i/2) + \mathbf{s}_i \sin(\theta_i/2)$, it can be shown that $\mathbf{e}$ can be represented as a function of t, θ_1, θ_2, and $\|\mathbf{s}_1 \times \mathbf{s}_2\|$, where $\cdot \times \cdot$ denotes the cross product when the pure quaternions $\mathbf{s}_i$ are regarded as vectors in $\mathbb{R}^3$. Note that in order for the shortest path to give the correct interpolation, $\langle \mathbf{q}, \mathbf{I} \rangle = \cos(\theta/2) > 0$ [Kavan et al. 2008], since we are interpolating between $\mathbf{q}$ and $\mathbf{I}$. This is always satisfied for $\theta_1 + \theta_2 < \pi$.

Once we represent θ_e in terms of the desired variables, we can sample these variables for the deviation analysis. By sampling, we found out that the maximum of $\mathbf{e}$ occurs when $t = 1/2$ and $\|\mathbf{s}_1 \times \mathbf{s}_2\| = 1$. This is in accordance with the intuition that the deviation is maximum when the interpolated transformation $\mathbf{q}^t$ is furthest from both $\mathbf{I}$ and $\mathbf{q}$, and the axes of rotation for $\mathbf{q}_1$ and $\mathbf{q}_2$ are orthogonal. With these variables fixed, we plot $\mathbf{e}$ as a function

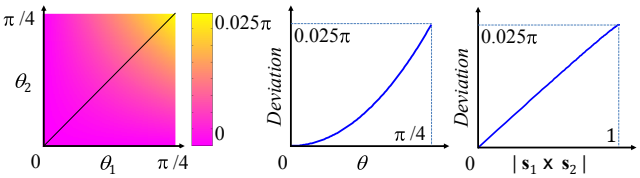

Figure 14: *Rotational deviation of differential interpolation from shortest path interpolation as a function of (left) angles θ_1, θ_2 of the two differentials used, (middle) $\theta = \theta_1 = \theta_2$ that corresponds to the black line on the left, and (right) the cross product of the axes $\mathbf{s}_1$ and $\mathbf{s}_2$ of rotations of the two differentials.*

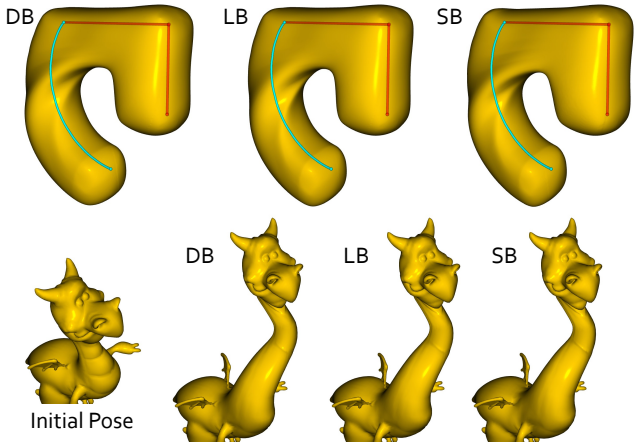

Figure 15: *Differential blending based deformations are close to those provided by shortest path based methods, when they generate smooth results without artifacts. We illustrate two difficult cases for differential blending, where (top) the rotation axis change along the bones, and (bottom) large translations along and rotations around different axes are applied. DB (differential blending), LB (linear blending of dual quaternions), SB (screw linear blending of dual quaternions).*

of θ_1 and θ_2 in Figure 14 (left). Decreasing angles leads to smaller deviations, with a quadratic tendency, as shown for $\theta_1 = \theta_2$ in Figure 14 (middle). The maximum angular deviation is around 4.5 degrees for $\theta_i = 45$. Finally, as illustrated in Figure 14 (right), the angular deviation depends approximately linearly on $\|\mathbf{s}_1 \times \mathbf{s}_2\|$.

5.3 Theoretical Derivation of the Deviation

The exact formula for $\mathbf{e}$ we derived in the last section is fairly complex to analyze analytically. However, it is possible to derive much simpler and very accurate expressions by utilizing the Baker-Campbell-Hausdorff formula. In this section, we derive the angular and translational deviation by a second order analysis.

We would like to find a closed form formula for $\|\log(\mathbf{e})\|$ that contains the angular and translational parts of $\mathbf{e}$. The deviation $\mathbf{e}$ can be written as $\mathbf{e} = e^{t\log(\mathbf{q}_1\mathbf{q}_2)} e^{-t\log(\mathbf{q}_2)} e^{-t\log(\mathbf{q}_1)}$. Hence, it is sufficient to find a formula of the form $e^{\mathbf{x}}e^{\mathbf{y}} = e^{f(\mathbf{x},\mathbf{y})}$ for some function f. For non-commutative Lie groups, it is in general not true that $e^{\mathbf{x}}e^{\mathbf{y}} = e^{\mathbf{x}+\mathbf{y}}$ for $\mathbf{x}$ and $\mathbf{y}$ in the corresponding Lie algebra. Instead, this multiplication is given by $e^{\mathbf{x}}e^{\mathbf{y}} = e^{BCH(\mathbf{x},\mathbf{y})}$, where BCH is the Baker-Campbell-Hausdorff formula (e.g. [Govindu 2004]):

$$BCH(\mathbf{x}, \mathbf{y}) = \mathbf{x} + \mathbf{y} + \frac{1}{2}[\mathbf{x}, \mathbf{y}] + h.o.t.. \quad (3)$$

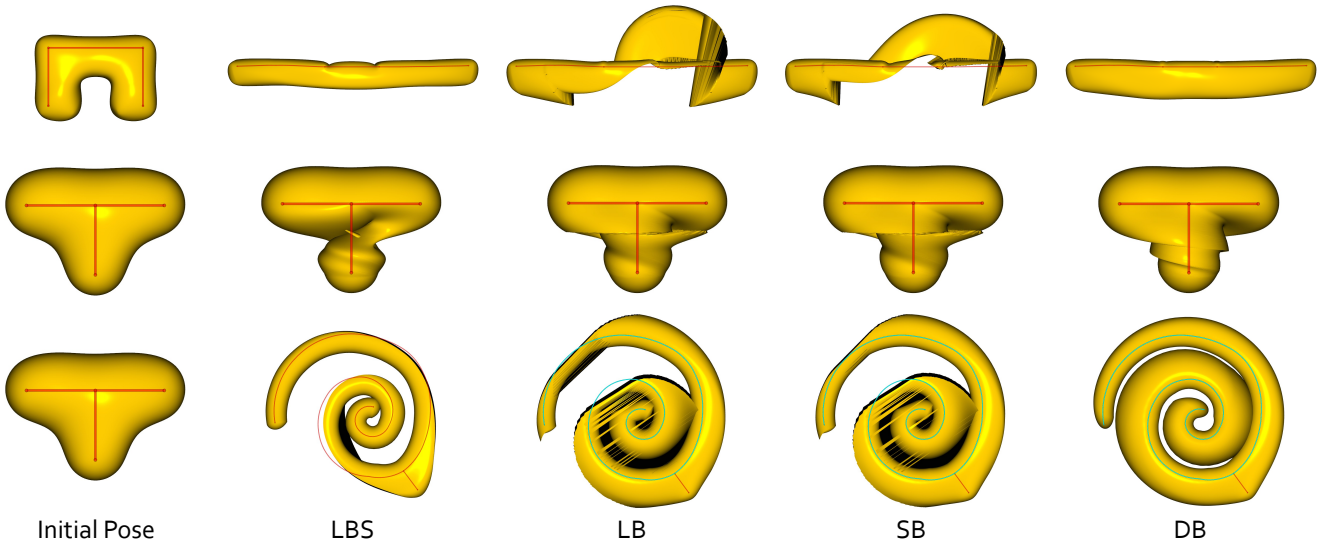

Figure 16: *The discontinuities due to blending with existing methods are avoided by differential blending. LBS (linear blend skinning), LB (linear blending of dual quaternions), SB (screw linear blending of dual quaternions), DB (differential blending).*

Here, $[\mathbf{x}, \mathbf{y}] = \mathbf{xy} - \mathbf{yx}$ denotes the commutator or Lie bracket and $h.o.t.$ stands for higher order terms. Defining $\mathbf{a} = \log(\mathbf{q})$, $\mathbf{a}_1 = \log(\mathbf{q}_1)$, and $\mathbf{a}_2 = \log(\mathbf{q}_2)$, this second order expansion can be applied to $\mathbf{e} = \mathbf{q}^t(\mathbf{q}_2^t)^*(\mathbf{q}_1^t)^*$ as follows:

$$\begin{aligned} \log(\mathbf{e}) &= \log\left(e^{t(\mathbf{a}_1+\mathbf{a}_2+\frac{1}{2}[\mathbf{a}_1,\mathbf{a}_2])}e^{-t\mathbf{a}_2}e^{-t\mathbf{a}_1}\right) \\ &= \frac{t(1-t)}{2}\left[\mathbf{a}_1, \mathbf{a}_2\right], \end{aligned} \tag{4}$$

keeping terms up to second order. Hence, it is sufficient to compute the commutator $[\mathbf{a}_1, \mathbf{a}_2] = \log(\mathbf{q}_1)\log(\mathbf{q}_2) - \log(\mathbf{q}_2)\log(\mathbf{q}_1)$. The logarithms of the dual quaternions $\mathbf{q}_i$ can be written as $\log(\mathbf{q}_i) = \frac{\theta_{i0}+\epsilon\theta_{i\epsilon}}{2}(\mathbf{s}_{i0} + \epsilon\mathbf{s}_{i\epsilon})$. Here, θ_{i0} is the rotation around and $\theta_{i\epsilon}$ is the translation along the screw axis $\mathbf{s}_{i0}$. Plugging these into $[\log(\mathbf{q}_1), \log(\mathbf{q}_2)]$, one can compute the following expressions for the angular and translational parts:

$$|\theta_{e0}| = \frac{1}{2}|t(t-1)\theta_{10}\theta_{20}|\|\mathbf{s}_{10} \times \mathbf{s}_{20}\| \tag{5}$$

$$|\theta_{e\epsilon}| = \frac{1}{2}\left|t(t-1)\left(\gamma\theta_{10}\theta_{20} + \theta_{1\epsilon}\theta_{20} + \theta_{2\epsilon}\theta_{10}\right)\right|\|\mathbf{s}_{10} \times \mathbf{s}_{20}\|,$$

where $\gamma = \langle \widehat{\mathbf{s}_{10} \times \mathbf{s}_{20}}, \frac{\mathbf{s}_{1\epsilon}\times\mathbf{s}_{20}+\mathbf{s}_{10}\times\mathbf{s}_{2\epsilon}}{\|\mathbf{s}_{10}\times\mathbf{s}_{20}\|}\rangle$ with $\hat{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}\|$. The equation for $|\theta_{e0}|$ is computationally shown to be very accurate in Figure 14. By utilizing vector identities, it can be shown that γ depends on $\langle \mathbf{s}_{10}, \mathbf{s}_{20}\rangle$ and hence when the two axes are orthogonal, $\gamma = 0$. Thus, when the rotation error is maximum, the bound for $|\theta_{e\epsilon}|$ simplifies to:

$$|\theta_{e\epsilon}| \leq \frac{1}{2}\left|t(t-1)\left(\theta_{1\epsilon}\theta_{20} + \theta_{2\epsilon}\theta_{10}\right)\right|. \tag{6}$$

Longer paths. So far, we have assumed that the transformations to be blended are composed of two differentials. Using the Baker-Campbell-Hausdorff formula (Equation 3) and writing the deviation in the form of Equation 4 for more than two differentials such that $\mathbf{q} = \mathbf{q}_1 \cdots \mathbf{q}_n$, it is possible to show that

$$\|\log(\mathbf{e})\| = \left|\frac{t(1-t)}{2}\right|\left\|\sum_{i=1}^{n}\sum_{j>i}[\log(\mathbf{q}_i), \log(\mathbf{q}_j)]\right\|. \tag{7}$$

6 Discussion and Results

To demonstrate and validate our method, we have created a set of expressive poses and animations using a number of pre-rigged models. The rigs are obtained using a standard skeleton with straight bones. We generated the vertex weights using Pinocchio [Baran and Popović 2007], and attachment points as described in Section 4.4. We use 10 differential transformation samples on each bone, uniformly spaced along the bones. Once these are computed, the model is ready to be posed using our framework.

The hardest obstacle we faced during modeling sessions was the artifacts when skinning with existing methods. We illustrate the skinning problems occurring with shortest path based methods, namely linear blending of dual quaternions (LB) [Kavan et al. 2008] and iterative screw linear blending of dual quaternions (dual quaternion iterative blending) (SB) [Govindu 2004] and how the new differential blending (DB) technique avoids these on toy examples in Figure 16. As shown on the top row, simple operations such as straightening a bar can easily create skinning artifacts. The artifacts also appear often when twisting or bending is required (Figure 16, second and third rows). Due to the gradual blending of differentials, our method gives smooth results without artifacts for all cases. Skinning comparisons for poses extracted from real animations are shown in Figures 1 and 10. Especially for poses with exaggerated deformations, these artifacts become unavoidable for many cases. In contrast, differential skinning provided smooth and intuitive results for all animations and poses created, with deformations close to shortest path blending for cases without artifacts, as illustrated in Figure 15. This behavior is also verified via computational and theoretical analysis, as shown in Figure 14 and explained in Section 5.

Differential blending based skinning avoids the artifacts caused by previous blending methods regardless of the underlying skeletal structure used. We illustrate this in Figure 17, where a classical skeleton with many straight bones are used to approximate the curved bones. The discontinuity problem persists in all cases due to blending transformations that are far apart.

Our results show that the line-of-action sketching metaphor popular in traditional 2D drawing can be intuitively adapted to 3D mod-

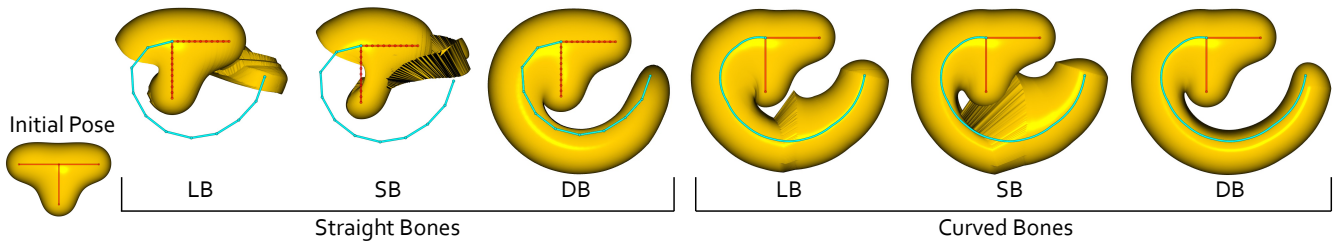

Figure 17: *Blending of large transformations with disparate rotations leads to discontinuities, irrespective of the underlying skeleton structure used. LB (linear blending of dual quaternions), SB (screw linear blending of dual quaternions), DB (differential blending).*

eling, circumventing the tedious rigging controls of conventional 3D modeling. Our system allows much more intuitive interaction with the skeleton, close to 2D sketching. Poses generated via our system were impossible or very difficult to obtain even with very sophisticated rigs and controls (Figure 3, please see the accompanying video for a modeling session comparison with our system and Maya). This allowed the artists to create expressive poses very efficiently and fluidly with a few simple curves as illustrated in Figures 1, 19, and 18. Please also refer to the accompanying video for the keyframing animations generated purely using our system.

Limitations. With our current implementation running on a multi-core CPU architecture, skinning a frame of a model with $11k$ vertices and 10 samples on each bone takes about 0.5 seconds. Since our goal was posing a single character, this performance was sufficient to provide the user nearly instant visual feedback. Significant speedups can be obtained by utilizing further parallelization along with caching strategies to avoid repeated computations, making the algorithm's performance competitive with simple methods such as linear blend skinning. We leave this for future work.

7 Conclusion

In this paper, we proposed a novel way of blending transformations that generates continuous and intuitive skeletal deformations for large and disparate transformations that are typical for expressive posing and animation. This method allowed us to extend the range of achievable poses in a sketch-based posing and animation system that incorporates tried-and-true techniques from traditional 2D illustration. By adapting these techniques to 3D character posing, our system can be used to intuitively create caricatured and expressive poses that are difficult or impossible to get with existing 3D skeletal methods.

We expect differential blending to be utilized in skeleton-based animation pipelines and hence to have a significant impact on 3D character animation, providing a new level of expressive power. The idea of differential blending can also be adapted for geometry interpolation and deformations without an underlying skeletal structure. Furthermore, our work shows the potential of re-interpreting proven concepts from 2D illustration in the context of 3D animation and points to many other areas of future work. While we have focused on *line of action* curves as inspiration, silhouette lines, guide strokes, reference shapes, and solid sketches [Blair 1994] could provide additional starting points for novel 3D animation representations and interfaces. Likewise, timing diagrams [Whitaker and Halas 2002], traditionally used in 2D animation, could lead to more fluid ways to interact with the timing of 3D animations. An even more far-reaching direction of future work lies in exploring hybrid representations that use 2D illustration concepts not only for posing but also for the visual representation of 3D animations by exploring the middle ground between the 2D and 3D worlds.

Acknowledgments

We would like to thank Maurizio Nitti and Alessia Marra for generating the animations and poses.

References

AKEO, M., HASHIMOTO, H., KOBAYASHI, T., AND SHIBUSAWA, T. 1994. Computer graphics system for reproducing three-dimensional shape from idea sketch. *Computer Graphics Forum 13*, 3, 477–488.

ALEXA, M. 2002. Linear combination of transformations. *ACM Trans. Graph. 21*, 3 (July), 380–387.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph. 26*, 3 (July), 72:1–72:8.

BARAN, I., LEHTINEN, J., AND POPOVIĆ, J. 2010. Sketching clothoid splines using shortest paths. *Computer Graphics Forum 29*, 2 (May), 655–664.

BLAIR, P. 1994. *Cartoon Animation*. Walter Foster Publishing.

DAVIS, J., AGRAWALA, M., CHUANG, E., POPOVIĆ, Z., AND SALESIN, D. 2007. A sketching interface for articulated figure animation. In *ACM SIGGRAPH 2007 courses*, ACM, New York, NY, USA, SIGGRAPH '07.

FORSTMANN, S., AND OHYA, J. 2006. Fast skeletal animation by skinned arc-spline based deformation. In *Proc. Eurographics 2006 Short-Papers*, 1–4.

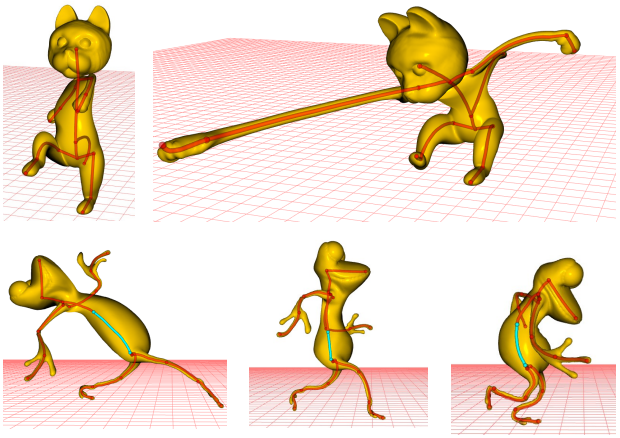

Figure 19: *Expressive poses created with our system. Please refer to the accompanying video for the animations.*

Figure 18: *A selection of frames from the set of animations created with our system. Please refer to the accompanying video for the complete animations and additional results.*

FORSTMANN, S., OHYA, J., KROHN-GRIMBERGHE, A., AND MCDOUGALL, R. 2007. Deformation styles for spline-based skeletal animation. In *Proc. SCA '07*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 141–150.

GINGOLD, Y., IGARASHI, T., AND ZORIN, D. 2009. Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph. 28* (December), 148:1–148:9.

GOVINDU, V. 2004. Lie-algebraic averaging for globally consistent motion estimation. In *Proc. CVPR 2004*, vol. 1, I–684 – I–691 Vol.1.

HOSHINO, J., AND HOSHINO, Y. 2001. Intelligent storyboard for prototyping animation. *Multimedia and Expo, IEEE International Conference on 0*, 96.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *Proc. SIGGRAPH '99*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 409–416.

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. In *Proc. SIGGRAPH Asia 2011*, ACM, New York, NY, USA, SA '11, 165:1–165:8.

KAVAN, L., AND ŽÁRA, J. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *Proc. I3D '05*, ACM, New York, NY, USA, 9–16.

KAVAN, L., COLLINS, S., ZARA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph. 27*, 4, 105:1–105:23.

KHO, Y., AND GARLAND, M. 2005. Sketching mesh deformations. In *Proc. I3D '05*, ACM, New York, NY, USA, 147–154.

KRAEVOY, V., SHEFFER, A., AND VAN DE PANNE, M. 2009. Modeling from contour drawings. In *Proc. SBIM '09*, ACM, New York, NY, USA, 37–44.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proc. Graphics interface '88*, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 26–33.

MAO, C., QIN, S., AND WRIGHT, D. 2005. A sketch-based gesture interface for rough 3d stick figure animation. In *Proc. SBIM. Dublin, 2005*, Eurographics.

MAO, C., QIN, S., AND WRIGHT, D. 2007. Sketch-based virtual human modelling and animation. In *Smart Graphics*, A. Butz, B. Fisher, A. Krüger, P. Olivier, and S. Owada, Eds., vol. 4569 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 220–223.

MAO, C., QIN, S. F., AND WRIGHT, D. 2009. A sketch-based approach to human body modelling. *Computers & Graphics 33*, 4, 521 – 541.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. In *Proc. ACM SIGGRAPH 2003*, ACM, New York, NY, USA, SIGGRAPH '03, 562–568.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3 (July), 1142–1147.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3 (July).

OLSEN, L., SAMAVATI, F., SOUSA, M., AND JORGE, J. 2008. A taxonomy of modeling techniques using sketch-based interfaces. In *Proc. Eurographics 2008, STAR*.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: an interface for sketching character motion. *ACM Trans. Graph. 23*, 3 (Aug.), 424–431.

WHITAKER, H., AND HALAS, J. 2002. *Timing for Animation*. Focal Press.

YANG, X., SOMASEKHARAN, A., AND ZHANG, J. J. 2006. Curve skeleton skinning for human and creature characters: Research articles. *Comput. Animat. Virtual Worlds 17*, 3-4 (July), 281–292.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. Sketch: An interface for sketching 3d scenes. In *Computer Graphics, Proc. Siggraph 1996*, 163–170.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3 (July), 496–503.

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2008. Sketch-based interfaces: Sketching contours. *Comput. Graph. 32*, 5 (Oct.), 486–499.