SIGGRAPH Course 2014 — Skinning: Real-time Shape Deformation

Part I: Direct Skinning Methods and Deformation Primitives

Ladislav Kavan

University of Pennsylvania

1 Introduction

Skinning is the process of controlling deformations of a given object using a set of deformation primitives. A typical example of an object often subjected to skinning is the body of a virtual character. In this case, the deformation primitives are rigid transformations associated with bones of an animation skeleton. Later we will see that this is not the only possibility. Also, skinning can be well defined even without any skeleton. At a high level, the landscape of skinning algorithms can be divided into two main streams: direct and variational methods. Variational methods pose the task as an optimization problem, minimizing an objective function (deformation energy). Variational methods are rooted in continuum mechanics and the theory of elasticity and typically require iterative solvers. The focus of this part is on direct skinning methods, which have been developed in order to sidestep the high computational requirements of variational methods. Direct methods compute the resulting deformations using closed-form expressions, i.e., without any numerical optimization. Direct methods are often very fast and embarrassingly parallel, which makes them particularly attractive for interactive, real-time applications and GPU implementations.

2 Linear blend skinning

Linear blend skinning, also known as *skeleton-subspace deformation*, (single-weight-)*enveloping*, or *matrix-palette skinning*, is the basic and most well known algorithm for direct skeletal shape deformation. It is difficult to trace the roots of linear blend skinning. Some of the early ideas appeared in the pioneering works [Badler and Morris 1982] and [Magnenat-Thalmann et al. 1988]. Perhaps the first paper that gives an exact mathematical description of linear blend skinning is due to Lewis [2000], but it mentions the algorithm is well-known and implemented in commercial software packages. Linear skinning assumes the following input data:

- **Rest pose shape**, typically represented as a polygon mesh. The mesh connectivity is assumed to be constant, i.e., only vertex positions will change during deformations. We denote the rest-pose vertices as $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^3$. It is often convenient to assume that $\mathbf{v}_i$ are in fact $\mathbb{R}^4$ vectors with the last coordinate equal to one, according to the common convention of homogeneous coordinates.
- **Bone transformations**, represented using a list of matrices $\mathbf{T}_1, \dots, \mathbf{T}_m \in \mathbb{R}^{3\times 4}$. The matrices $\mathbf{T}_i$ can be conveniently defined using an animation skeleton; in this case they corresponds to spatial transformations aligning the rest pose of bone i with its current (animated) pose. Bone transformations are typically the only quantity that is allowed to vary during the course of an animation.
- **Skinning weights.** For vertex $\mathbf{v}_i$, we have weights $w_{i,1}, \dots, w_{i,m} \in \mathbb{R}$. Each weight $w_{i,j}$ describes the *amount of influence* of bone j on vertex i. A common requirement is that $w_{i,j} \geq 0$ and $w_{i,1} + \cdots + w_{i,m} = 1$ (partition of unity).

These concepts are best illustrated with an example; see Figure 1 for an example of bone transformations and Figure 2 for influence weights. Linear blend skinning computes deformed vertex positions $\mathbf{v}'_i$ according to the following formula:

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_{i,j} \mathbf{T}_j \mathbf{v}_i = \left(\sum_{j=1}^{m} w_{i,j} \mathbf{T}_j \right) \mathbf{v}_i \qquad (1)$$

The latter form highlights the fact that the rest pose vertex $\mathbf{v}_i$ is transformed by a linear combination (blend) of bone transformation matrices $\mathbf{T}_j$. These matrices are the *deformation primitives* of linear blend skinning, i.e., elementary building blocks of deformations. While arbitrary affine transformations are allowed, sometimes it is convenient to assume that $\mathbf{T}_j$ are rigid body transformations, i.e., $\mathbf{T}_j \in SE(3)$. Note that many implementations assume that most of the weights $w_{i,1}, \dots, w_{i,m}$ are zero. Due to graphics hardware considerations, it is common to assume there are at most four non-zero weights for every vertex; different limits can be found in some systems. Some older games used a variant dubbed *rigid skinning* which corresponds to allowing only one influencing bone per vertex. With increasing polygon budgets, linear blend skinning quickly replaced rigid skinning because it allowed for smooth transitions between individual transformations (some systems used the term *smooth skinning*). The design of high quality skinning weights is far from trivial and will be covered in the second part of these notes by Alec Jacobson.

Linear blend skinning works very well when the blended transformations $\mathbf{T}_j$ are not very different. Issues arise if we need to blend transformations which differ significantly in their rotation components. It is a well known fact that a linear combination of rotations is no longer a rotation [Alexa 2002]. Geometrically, this is a consequence of the fact that the Lie group of 3D transformations, $SO(3)$, is not a linear ("flat") space, but a curved manifold. Just like the curvature of the Earth rarely bothers us in everyday life, this is not a big problem if the blended transformations $\mathbf{T}_j$ are close. However, consider what happens if we need to blend these two rotations:

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

i.e., $\mathbf{R}_1$ is the identity and $\mathbf{R}_2$ is rotation about the z-axis by 180 degrees (translations are not relevant in this example). Linear blend $0.5 \cdot \mathbf{R}_1 + 0.5 \cdot \mathbf{R}_2$ results in a rank-1 matrix, projecting the 3D space onto the z-axis. Transformation of a shape by this matrix typically results in undesirable deformations. Large relative rotations are not uncommon in skinning, because joints such as shoulders, wrists, or even elbows often exhibit a rather large range of motion. This problem is so common it earned a name: a *candy-wrapper* artifact, see Figure 3. In addition to this problem, it is obvious that linear blend skinning has only a limited number of parameters. One possibility to expand the expressive power of linear blend skinning is by using more transformations, typically calculated from the original bone transformations by non-linear procedures [Weber 2000; Mohr and Gleicher 2003; Kavan et al. 2009]. Alternatively, it is possible to enrich the space of skinning weights, leading to methods which are still linear, but feature more parameters than linear blend skinning. We call these techniques *multi-linear* and we explore them in the following section.

3 Multi-linear skinning methods

While the candy-wrapper artifacts of linear blend skinning (Figure 3) cannot be fixed by changing weights, it turns out that introducing *more weights* helps, at least to some extent. This indicates that linear blend skinning is not the most general linear model – so, what is? We assume that the rest-pose shape and the skinning weights are properties of the object and are not changing during the course of an animation. The only quantity which changes during animation are the transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m \in \mathbb{R}^{3\times4}$. Therefore, the most general linear skinning model is simply a linear function which consumes all of the transformations ($12m$ scalars) as input, and

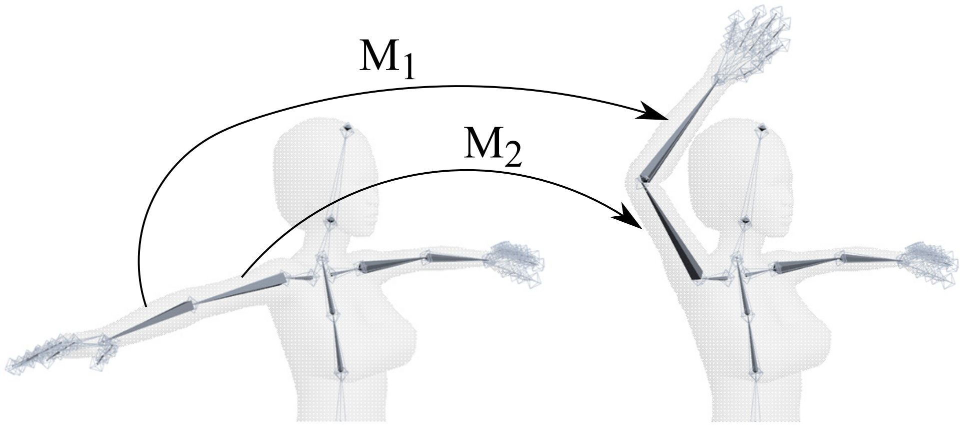

Figure 1: *Bone transformations (lower and upper arm bones) for one example deformed pose.*

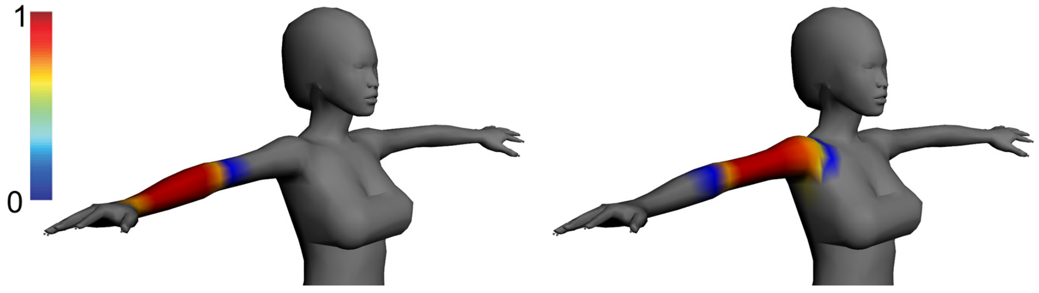

Figure 2: *Influence weights corresponding to lower and upper arm bones.*

produces the deformed vertex positions ($3n$ scalars). A convenient tool here is *vectorization*, i.e., an operator $\mathrm{vec}(\mathbf{T}_j) = \mathbf{t}_j$ which converts a $\mathbb{R}^{3\times 4}$ matrix into a $\mathbb{R}^{12\times 1}$ vector by stacking the individual columns. For more details, we recommend the Wikipedia page `http://en.wikipedia.org/wiki/Vectorization_(mathematics)`. Subsequently, we stack all of the $\mathbf{t}_j$ vectors into $\mathbf{t} \in R^{12m\times 1}$. Following the same convention, we can stack all of the deformed vertex positions $\mathbf{v}'_i \in \mathbb{R}^3$ into one long vector $\mathbf{v}' \in \mathbb{R}^{3n}$. Now, we can define a general linear skinning model:

$$\mathbf{v}' = \mathbf{X}\mathbf{t} \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{3n\times 12m}$ is a matrix of parameters. It will be useful to split this matrix into blocks $\mathbf{X}_{i,j} \in \mathbb{R}^{3\times 12}$ where $i = 1,\ldots,n$ and $j = 1,\ldots m$. Each $\mathbf{X}_{i,j}$ corresponds to a vertex-bone pair and has, in its most general form, 36 degrees of freedom. It is interesting to think how linear blend skinning relates to this form. Specifically, what is the $\mathbf{X}$ matrix corresponding to linear blend skinning (Equation (1))? This question can be elegantly answered using Kronecker products – a special type of matrix multiplication which is closely related to the vectorization operator. In general, the Kronecker product of matrices $\mathbf{A} \in \mathbb{R}^{p\times q}$ and $\mathbf{B} \in \mathbb{R}^{r\times s}$ is:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,q}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{p,1}\mathbf{B} & \cdots & a_{p,q}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{pr\times qs} \tag{3}$$

Kronecker products exhibit a number of interesting properties, summarized at `http://en.wikipedia.org/wiki/Kronecker_product`. Here, we will need the following property, which assumes that matrices $\mathbf{A}$ and $\mathbf{B}$ can be multiplied together, i.e., $q = r$. In that case,

$$\mathrm{vec}(\mathbf{AB}) = (\mathbf{I}_s \otimes \mathbf{A})\,\mathrm{vec}(\mathbf{B}) = (\mathbf{B}^{\mathsf{T}} \otimes \mathbf{I}_p)\,\mathrm{vec}(\mathbf{A}) \tag{4}$$

where $\mathbf{I}_s \in \mathbb{R}^{s\times s}$ and $\mathbf{I}_p \in \mathbb{R}^{p\times p}$ are identity matrices. Let us apply this to the linear blend skinning formula (Equation (1)):

$$\mathbf{v}'_i = \mathrm{vec}(\mathbf{v}'_i) = \sum_{j=1}^{m} \mathrm{vec}(\mathbf{T}_j(w_{i,j}\mathbf{v}_i)) = \sum_{j=1}^{m} (w_{i,j}\mathbf{v}_i^{\mathsf{T}} \otimes \mathbf{I}_3)\mathbf{t}_j = \sum_{j=1}^{m} \mathbf{X}_{i,j}^{LBS}\mathbf{t}_j = \mathbf{X}_i^{LBS}\mathbf{t} \tag{5}$$

where $\mathbf{X}_{i,j}^{LBS} = w_{i,j}\mathbf{v}_i^{\mathsf{T}} \otimes \mathbf{I}_3 \in \mathbb{R}^{3\times 12}$ and $\mathbf{X}_i^{LBS} = [\mathbf{X}_{i,1}^{LBS},\ldots,\mathbf{X}_{i,m}^{LBS}] \in \mathbb{R}^{3\times 12m}$. We can write out each $\mathbf{X}_{i,j}^{LBS}$ block even more explicitly: if we denote the coordinates of $\mathbf{v}_i = (v_{i,1}, v_{i,2}, v_{i,3})^{\mathsf{T}}$, we have

$$\mathbf{X}_{i,j}^{LBS} = [w_{i,j}v_{i,1}\mathbf{I}_3 \mid w_{i,j}v_{i,2}\mathbf{I}_3 \mid w_{i,j}v_{i,3}\mathbf{I}_3 \mid w_{i,j}\mathbf{I}_3] \tag{6}$$

This reveals that the classical linear blend skinning (Equation (1)) is a rather special form of general linear skinning. We are not aware of any work exploring the fully general 36-parameter linear skinning model (Equation (2)). The closest instance is Multi-Weight Enveloping (MWE) [Wang and Phillips 2002], which advocates a linear model with twelve weights per each vertex-bone pair:

$$\mathbf{X}_{i,j}^{MWE} = \left[\begin{array}{ccc|ccc|ccc|ccc} w_{i,j}^1 v_{i,1} & 0 & 0 & w_{i,j}^4 v_{i,2} & 0 & 0 & w_{i,j}^7 v_{i,3} & 0 & 0 & w_{i,j}^{10} & 0 & 0 \\ 0 & w_{i,j}^2 v_{i,1} & 0 & 0 & w_{i,j}^5 v_{i,2} & 0 & 0 & w_{i,j}^8 v_{i,3} & 0 & 0 & w_{i,j}^{11} & 0 \\ 0 & 0 & w_{i,j}^3 v_{i,1} & 0 & 0 & w_{i,j}^6 v_{i,2} & 0 & 0 & w_{i,j}^9 v_{i,3} & 0 & 0 & w_{i,j}^{12} \end{array}\right]$$

Notice that one weight of classical linear blend skinning, $w_{i,j}$ is replaced by $w_{i,j}^1,\ldots,w_{i,j}^{12} \in \mathbb{R}$. As demonstrated by Wang and Phillips [2002], this results in a more powerful linear method which can avoid candy-wrapper artifacts. The problem is that there are twelve times as many weights. Their design is more complicated, because these more general weights no longer have a simple intuitive interpretation as in classical linear blend skinning (see Figure 2). Instead, the MWE weights are optimized using a set of example shapes [Wang and Phillips 2002]. This assumes that these example shapes have been prepared by artists, which makes this method related to example-based skinning techniques, covered in more detail in the third section of this course by J. P. Lewis.

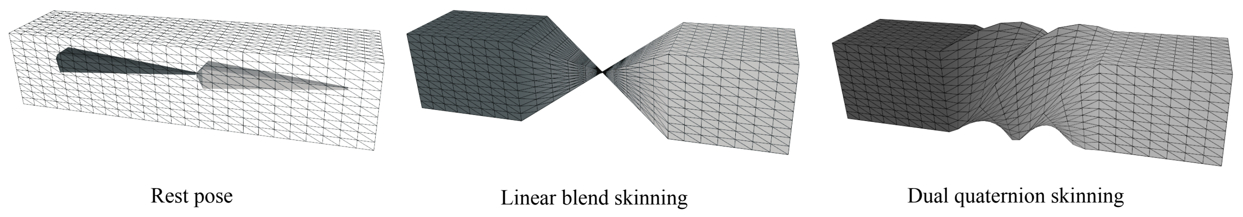

Figure 3: *An elastic bar in the rest pose (left) with two bones. A 180 degrees twist of the right bone (lighter gray) results in a candy-wrapper artifact (middle). The central cross-section is transformed by a projection matrix, which collapses the entire edge loop to a single point. This behavior can be avoided with dual quaternion skinning (right).*

Another linear skinning model is *Animation Space* [Merry et al. 2006a]. Merry and colleagues observed that not all of the 12 weights of MWE are useful and proposed a linear model which uses only 4 weights per vertex-bone pair. While this makes Animation Space seemingly less powerful than MWE, it turns out that Animation Space enforces world-space rotation invariance, i.e., it deliberately looses the ability to treat individual world coordinates differently. This feature does not seem to be very practical, because it is natural to assume that a world-space rotation of all transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m$ will rotate *but not otherwise deform* the resulting vertices $\mathbf{v}'_i$. This world-space rotation invariance corresponds to the classical requirement from elasticity, i.e., the total elastic energy should not depend on global world-space rotations. Note that it is perfectly reasonable for elastic deformations to depend on material-space rotations; only isotropic materials are invariant also to material-space rotations. In the following, we derive the general form of rotation invariant linear skinning.

If we denote $\mathbf{T} = [\mathbf{T}_1, \ldots, \mathbf{T}_m] \in \mathbb{R}^{3\times 4m}$, we can express the requirement of world-space rotation invariance as:

$$\forall \mathbf{T} \in \mathbb{R}^{3\times 4m},\ \forall \mathbf{R} \in SO(3) : \mathbf{X}_i \operatorname{vec}(\mathbf{RT}) = \mathbf{R}\mathbf{X}_i \operatorname{vec}(\mathbf{T}) \tag{7}$$

Applying Equation (4) transforms this equation into:

$$\mathbf{X}_i(\mathbf{I}_{4m} \otimes \mathbf{R})\mathbf{t} = \mathbf{R}\mathbf{X}_i\mathbf{t} \tag{8}$$

where $\mathbf{t} = \operatorname{vec}(\mathbf{T})$ as before. Because Equation (8) must be satisfied for arbitrary $\mathbf{t}$, it follows that:

$$\mathbf{X}_i(\mathbf{I}_{4m} \otimes \mathbf{R}) = \mathbf{R}\mathbf{X}_i \tag{9}$$

If we look at $\mathbf{X}_i$ as a collection of 3×3 blocks, Equation (9) requires that each such block $\mathbf{Y} \in \mathbb{R}^{3\times 3}$ commutes with $\mathbf{R}$, i.e., $\mathbf{YR} = \mathbf{RY}$. Because this must be true for all $\mathbf{R} \in SO(3)$, it is not difficult to prove that this implies that $\mathbf{Y}$ is a scaled identity, i.e., $\mathbf{Y} = \alpha\mathbf{I}_3$, $\alpha \in \mathbb{R}$ (sketch of a proof: choose a few specific matrices $\mathbf{R}$ and massage the resulting system of linear equations). Therefore, if a general linear skinning method (Equation (2)) is world-space rotation invariant, its corresponding $\mathbf{X}_{i,j}$ blocks must have the following structure:

$$\mathbf{X}_{i,j} = [y_{i,j,1}\mathbf{I}_3 \mid y_{i,j,2}\mathbf{I}_3 \mid y_{i,j,3}\mathbf{I}_3 \mid y_{i,j,4}\mathbf{I}_3] \tag{10}$$

i.e., the original 36 degrees of freedom reduce to four: $y_{i,j,1}, y_{i,j,2}, y_{i,j,3}, y_{i,j,4} \in \mathbb{R}$. An additional restriction will follow from considering translation invariance: global world-space translations can only translate but not otherwise deform the resulting vertices $\mathbf{v}'$. Formally, if $\mathbf{d} \in \mathbb{R}^{12m\times 1}$ is a global translation, i.e., stack of vectors $(0, 0, 0, 0, 0, 0, 0, 0, 0, x, y, z)^\mathsf{T} \in \mathbb{R}^{12\times 1}$, we require:

$$\mathbf{X}_i(\mathbf{t} + \mathbf{d}) = \mathbf{X}_i\mathbf{t} + \mathbf{d} \tag{11}$$

Combined with Equation (10), it follows that rotation and translation invariance requires

$$\sum_{j=1}^{m} y_{i,j,4} = 1 \tag{12}$$

The total number of degrees of freedom per vertex is therefore $4m - 1$. A linear skinning model satisfying Equation (10) and Equation (12) is exactly the Animation Space introduced by Merry and colleagues [Merry et al. 2006a]. From the discussion above it follows that Animation Space is the most general linear skinning method invariant to world-space rotations and translations. Similarly to multi-weight enveloping, Animation Space weights are also learned from a set of examples and are able to suppress candy-wrapper artifacts.

Some intuition can be gained by considering this formulation which is practically equivalent to Animation Space:

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_{i,j}\mathbf{T}_j\mathbf{v}_{i,j} \tag{13}$$

The only difference to the classical linear blend skinning (Equation (1)) is that we allow a *different rest-pose for each transformation*, i.e., we have $\mathbf{v}_{i,j} \in \mathbb{R}^3$ instead of just $\mathbf{v}_i \in \mathbb{R}^3$. Classical linear blend skinning (Equation (1)) can be interpreted as transforming the rest-pose mesh by each transformation, and blending the results using vertex weights. In Animation Space, we allow each transformation to have a different rest pose. The individual "imaginary" rest poses are never revealed to the user directly, because the resulting $\mathbf{v}'$ always blends them together, even if all of the transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m$ are identities. This indicates that Animation Space is in fact a special case of example-based skinning methods, featuring a particularly elegant mathematical formulation.

Further generalizations of linear skinning methods, going beyond Equation (2), are possible by considering more general transformations than the classical affine transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m \in \mathbb{R}^{3\times 4}$. One specific example is presented by Park and Hodgins [2006], who consider quadratic transformations represented by 3×9 matrices (see also [Müller et al. 2005]). With this more liberal mindset, clustered PCA applied to animated sequences [Sattler et al. 2005] can be also viewed as a special type of a linear skinning method. The idea of clustered (local) PCA is to split the input mesh into several disjoint components, each of which is subjected to classical (global) PCA [Alexa and Müller 2000]. Even closer to skinning are linear techniques which allow for overlaps between the individual components, such as the SPLOCs model [Neumann et al. 2013]. Data-driven skinning techniques are discussed in more detail in the final part of this course by Zhigang Deng.

4 Nonlinear skinning methods

Linear skinning methods are popular due to their efficient implementations and well understood mathematical properties, making them well suited for use as building blocks in more complex algorithms, e.g., in physics-based simulation [Faure et al. 2011]. However, as we have seen in the previous section, candy-wrapper artifacts can only be avoided using more parameters, which have to be learned, stored, and retrieved at runtime. Even then, some amount of undesired shrinking can still remain. The fundamental problem is that linear blending of rotations, i.e., elements of $SO(3)$, does not respect the fact that $SO(3)$ is a curved manifold. The idea of replacing linear blending with manifold-intrinsic averages leads us to nonlinear skinning methods.

A manifold-intrinsic interpolation method between two 3D rotations is SLERP (Spherical Linear Interpolation), introduced by Ken Shoemake [1985]. Even though SLERP can be formulated purely with matrices, Shoemake recognized the advantages of quaternions: 3×3 matrices contain 9 degrees of freedom and therefore require 6 constraints to represent rotations. Quaternions, on the other hand, feature only 4 degrees of freedom and therefore require only one constraint to represent rotations. Specifically, rotations correspond to unit quaternions, i.e., quaternions with length one. Formally, we denote the set of unit quaternions as $Q_1 = \{\mathbf{q} \in Q : ||\mathbf{q}|| = 1\}$. Geometrically, unit quaternions form a 3D unit hyper-sphere in 4D Euclidean space. While this is difficult to visualize, it is often sufficient to appeal to the intuition of the familiar 2D sphere in 3D Euclidean space (or even a unit circle in 2D).

Mathematicians are hardly impressed by the fact that quaternions require only 4 scalars as opposed to 9; worse, quaternions are sometimes incorrectly assumed to be just a different semantics to describe the group of 3D rotations. This is incorrect because the groups Q_1 and $SO(3)$ are not homomorphic. The catch is the fact that the two quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent *exactly the same* rotation, i.e., the group Q_1 is covering $SO(3)$ twice. This is called the "double cover" property [Hanson 2005] and it is the main reason why quaternions are very practical in skinning; let us explain this in more detail.

Consider uniform rotation of a rigid body about the z-axis (for example). Starting from the identity, i.e., no rotation, we eventually perform one full revolution, i.e., a 360 degrees rotation. Rotation matrices do not distinguish between 0 and 360 degrees rotation. Interestingly, quaternions do: a 360 degrees rotation will correspond to quaternion -1. The sign distinguishes between 0 degrees rotation (corresponding to $+1$) and 360 degrees rotation (corresponding to -1). Note that the formulas for converting from quaternion to matrix [Shoemake 1985] are quadratic and cancel the sign, i.e., after conversion to a matrix, the distinction between $\mathbf{q}$ and $-\mathbf{q}$ disappears. If we keep rotating our object beyond 360 degrees, we will eventually reach two full revolutions, i.e., 720 degrees rotation. Two full revolutions are not remembered by quaternions, i.e., a 720 degrees rotation is equivalent to 0 degrees rotation in both rotation matrices and quaternions. This interesting feature of quaternions can be observed in the real world; see, for example, the "Dirac's belt trick" [Hanson 2005].

In terms of blending rotations and, consequently, skinning, the important consequence is that the Q_1 manifold is "less non-linear" than $SO(3)$. This is illustrated in Figure 4. In the left part of the figure, $\mathbf{R}_1$ and $\mathbf{R}_2 \in SO(3)$ differ by a 180 degrees rotation ($\mathbf{I} \in SO(3)$ is the identity). A linear average of $\mathbf{R}_1$ and $\mathbf{R}_2$ results in a singular matrix. In the right part of the figure, the same situation is depicted on the Q_1 manifold. In particular, $\mathbf{q}_1, \mathbf{q}_2$ are unit quaternions corresponding to $\mathbf{R}_1, \mathbf{R}_2$, and $\mathbf{1} \in Q$ is one, interpreted as a quaternion. Note that $\mathbf{q}_1$ and $\mathbf{q}_2$ are much closer than $\mathbf{R}_1$ and $\mathbf{R}_2$. This is because one full loop corresponds to a 720

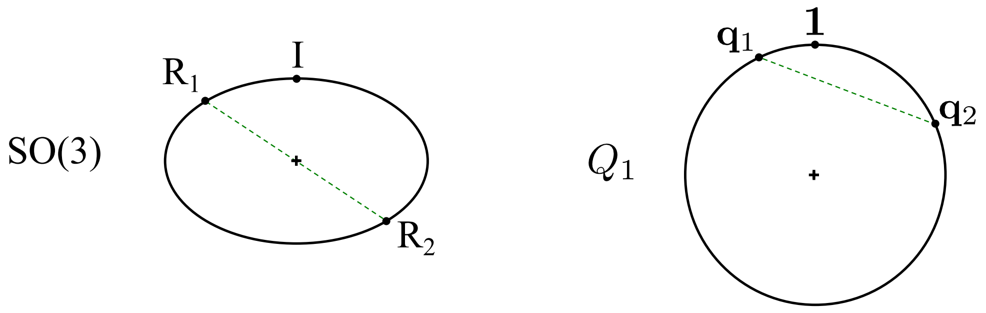

Figure 4: *Illustration of the geometry of rotations ($SO(3)$, left), compared to unit quaternions (Q_1, right).*

degrees revolution in Q_1, but only to 360 degrees in $SO(3)$. In other words, distances between rotations in Q_1 are twice as small as the corresponding distances in $SO(3)$. As a result, linear averaging of $\mathbf{q}_1$ and $\mathbf{q}_2$ is now much more accurate, as is obvious from Figure 4. The line connecting $\mathbf{q}_1$ and $\mathbf{q}_2$ is much closer to the manifold and does not pass through a singularity. This benefit does not come for free, however, because we need to be careful about picking the signs during matrix to quaternion conversion: wrong signs can lead to a typically undesired long-arc interpolation. Note that projection of a non-unit quaternion on the Q_1 manifold is extremely simple: $\mathbf{q}/\|\mathbf{q}\|$.

The SLERP algorithm has been generalized to more than 2 rotations by Buss and Fillmore [2001], however, resulting in an iterative procedure. Because fully accurate blending of rotations is not necessary in skinning, it is possible to approximate the correct manifold-intrinsic averages by linear combination of unit quaternions, followed by normalization (projection on Q_1). This has been utilized in early quaternion-based skinning methods such as [Hejl 2004] and [Kavan and Žára 2005]. The practical impact of these techniques is limited due to their handling of the translational component of the skinning transformations.

Skinning transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m \in \mathbb{R}^{3\times4}$ typically contain a non-trivial translation vector. Assuming that the left 3×3 submatrices of $\mathbf{T}_1, \ldots, \mathbf{T}_m$ are rotations, a straightforward solution is to blend the rotations and translations independently. Linear blending of translations is perfectly justified, because translation vectors form a linear space, not a curved manifold. If we implement this approach we find out that, even with perfectly intrinsic rotation blending [Buss and Fillmore 2001], the results are not acceptable – often much worse than with linear blend skinning, see Figure 5. The reasons for this behavior are discussed in detail in [Kavan et al. 2008]. In short, by splitting a rigid transformation into a rotation and translation pair, we are committing to a specific pivot point (center of rotation), around which the rotations will be interpolated. By default, this center of rotation corresponds to the origin of material-space coordinates, which is typically located near the object's center of mass – this explains the unusual result in Figure 5(left).

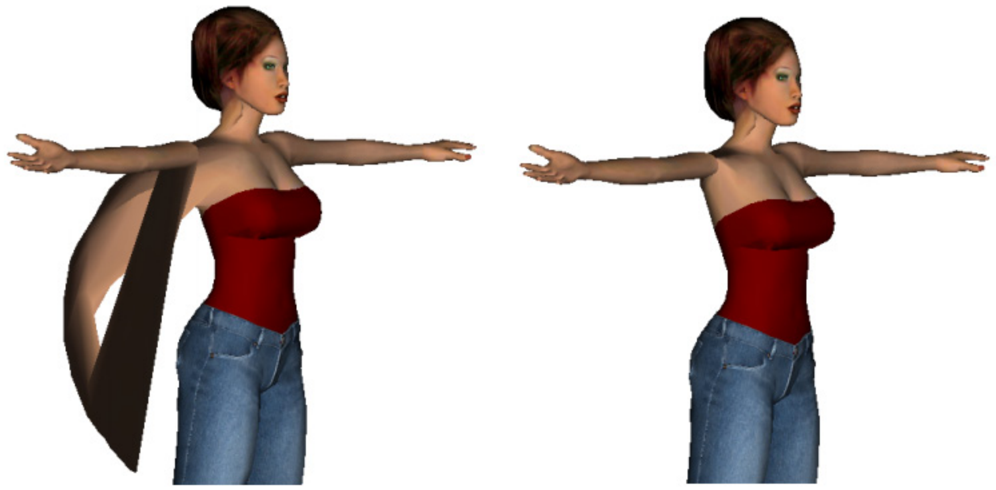

Figure 5: *Separate blending of rotation and translation components leads to unacceptable results (left), worse than standard linear blend skinning (right).*

It is of course possible to design a more suitable center of rotation, e.g., coinciding with the joint [Hejl 2004], or computed using least squares optimization [Kavan and Žára 2005]. However, it is possible to find situations where each of these strategies results in artifacts [Kavan et al. 2008]. The latter paper shows that the complications with the choice of the center of rotation disappear if we use unit dual quaternions to represent rigid body transformations. The reason is that unit dual quaternions represent rigid transformations using their intrinsic parameters, thus avoiding an explicit choice of a center of rotation. Dual quaternions are closely related to screw motions studied in theoretical kinematics [McCarthy 1990].

While the underlying mathematics may not be trivial, an actual implementation of dual quaternion skinning is quite straightforward. First, the transformation matrices $\mathbf{T}_1, \ldots, \mathbf{T}_m \in \mathbb{R}^{3\times4}$ are converted to unit dual quaternions. These unit dual quaternions are blended linearly, similarly to linear blend skinning (Equation (1)). Because linear combination of unit dual quaternions does not in general produce a unit dual quaternion, a normalization (projection) operation is performed. The resulting unit dual quaternion can be converted to a matrix which transforms a rest-pose vertex $\mathbf{v}_i$. The double cover property of regular quaternions occurs also in dual quaternions; specifically, unit dual quaternions from a double cover of $SE(3)$. It is therefore imperative to carefully choose signs of the dual quaternions obtained by converting matrices $\mathbf{T}_j$. Note that all steps in this algorithm are simple closed-form operations. While linearity is lost (due to the projection on unit dual quaternions), the resulting algorithm does not require any iterations and can be implemented very efficiently.

Dual quaternion skinning successfully eliminates the candy-wrapper artifacts (see Figure 3), but has a number of limitations, which we discuss in the following. A relatively benign issue is that linear blending of unit dual quaternions followed by normalization (Dual-quaternion Linear Blending, DLB) is not perfect manifold-intrinsic averaging, similarly to the $SO(3)$ case illustrated in Figure 4. A completely intrinsic blending can be achieved by applying Lie-algebraic averaging [Govindu 2004], which is a generalization of spherical averages [Buss and Fillmore 2001]. A dual quaternion version of Lie-algebraic averaging has been discussed in [Kavan et al. 2008] (Algorithm DIB). For shortest path interpolations, the differences between DLB and DIB are insignificant; in applications such as skinning the differences between DLB and DIB are barely noticeable.

A more serious concern especially in a production environment are non-rigid transformations. Other than uniform scale, dual quaternions are unable to represent non-rigid transformations, such as non-uniform scale and shear. These effects are especially important with stylized and cartoon characters. One possibility to side-step this limitation is to break the skinning process in two steps: in the first step, the rest-pose (including the skeleton) is rescaled. In the second step, dual quaternion skinning is applied to produce the desired pose (articulation) [Kavan et al. 2008]. While a similar approach has been applied successfully in a production setting (Disney's *Frozen* [Lee et al. 2013]), the question of optimal blending of general affine transformations remains open. An early investigation of this problem has been carried out by Alexa [2002], who proposes linear blending of matrix logarithms, followed by matrix exponential:

$$\mathbf{T}_{\texttt{blend}} = \exp\left(\sum_{j=1}^{m} \log(\mathbf{T}_j)\right) \tag{14}$$

From the viewpoint of Lie-algebraic averaging, the problem of this method is that the logarithms correspond to using the Lie algebra at the identity, even if the input transformations $\mathbf{T}_j$ are all far away from the identity. This leads to non-shortest-path interpolations, which has been criticized in a rather harsh way by game developers [Bloom et al. 2004]. Alexa's method has been applied in skinning [Cordier and Magnenat-Thalmann 2005], but its non-shortest-path nature can lead to artifacts, as shown in [Kavan et al. 2008]. An improved method to blend affine transformations is presented by [Rossignac and Vinacua 2011], however, the problem that $\log(\mathbf{T}_j)$ may not exist in the real domain persists. Even if we restrict ourselves only to matrices with a positive determinant, the logarithm can still be ill-defined if the matrix has two (real) negative eigenvalues of different magnitude. One possibility to avoid this problem is by introducing an intermediary transformation, essentially subdividing the motion into two smaller ones [Rossignac and Vinacua 2011]. To our knowledge, this method is yet to be tested in skinning.

While the shortest-interpolation-path property makes sense when interpolating the elements of $SO(3)$, real elastic materials can be twisted multiple times. In this case, both rotations matrices and quaternions fall short in representing such deformations (quaternions do remember one full revolution, but two full revolutions return us back to the starting point, i.e., are "forgotten", see Figure 4). This limitation of both linear and dual-quaternion-based methods inspired the work of Oztireli et al. [2013], called *Differential Blending*. Differential blending assumes a connected, rooted skeleton. Skinning transformations $\mathbf{q}_j$ are broken into smaller pieces $\mathbf{q}_{j,k}$ along the path from the root to the target bone: $\mathbf{q}_j = \mathbf{q}_{j,1} \ldots \mathbf{q}_{j,l}$. The main idea is that the blending of these smaller pieces can be done with previous methods, and the blended pieces are subsequently multiplied (composed) together:

$$\texttt{DiffBlend}(w_j; \mathbf{q}_j) = \prod_{k=1}^{l} \texttt{Blend}(w_j, \mathbf{q}_{j,k}) \tag{15}$$

In this formula, `Blend` is a standard blending operator (linear or dual quaternion) which blends input transformations using the provided weights w_j. The resulting `DiffBlend` takes correctly into account multiple revolutions, i.e., longer interpolation paths, see Figure 6.

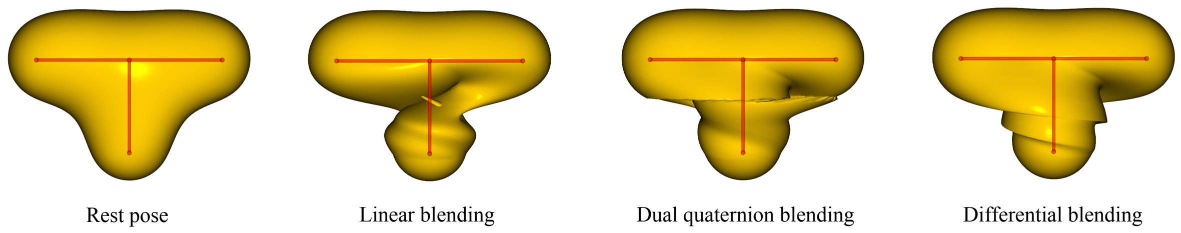

Figure 6: *Creating a corkscrew-type deformation fails with linear and dual quaternion blending. Multiple revolutions are correctly handled by differential blending.*

Another problem with dual quaternion skinning is known as a "bulging artifact" [Kavan and Sorkine 2012; Kim and Han 2014]. The problem is best explained on a simple cylinder skinned using two bones, connected with one joint (crude approximation of the human arm, with the joint corresponding to the elbow). In this case, dual quaternion skinning reduces to spherical blending around the joint. Spherical blending can be intuitively understood as a "point on a stick" approach: each vertex is attached to an imaginary rigid stick linking the vertex to $\mathbf{p}$. In other words, each vertex is constrained to a fixed sphere centered at $\mathbf{p}$; the radius of this sphere is $\|\mathbf{v}_i - \mathbf{p}\|$. This works very well when we are twisting the cylinder, but bending produces somewhat unnatural bulging effects, see Figure 7. Note that this bulging effect is not always undesired: for example, when modeling knuckles of the

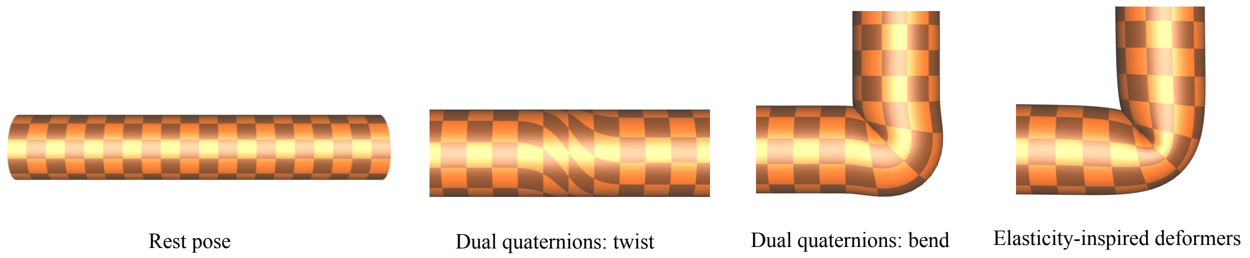

Figure 7: *Demonstration of dual quaternion bulging effects while bending a cylinder. Elasticity-inspired deformers [Kavan and Sorkine 2012] avoid this problem.*

hand. In some cases, however, the bulging is unwanted, and several strategies to eliminate it have been proposed. Observing that linear blend skinning does not produce bulging while bending, Autodesk Maya allows users to blend the result of linear and dual quaternion skinning. This requires an additional blending weight. The problem with this approach is that even a small amount of linear blend skinning re-introduces the candy-wrapper artifacts, so a compromise must be sought. The *swing-twist deformer* presented in [Kavan and Sorkine 2012] is based on the same observation (linear blending works well while bending), but combines linear and dual quaternion blending in a non-linear way. Specifically, the rotation of a joint is decomposed into a swing and twist components; the twist is blended spherically (using SLERP) and the swing is blended using linear matrix interpolation, composing the result using matrix multiplication. This eliminates the bulging artifacts without re-introducing the candy-wrapper artifacts while twisting. The swing-twist deformer leads us to the idea of generalized deformation primitives, discussed in the next section.

5 Deformation primitives

By deformation primitives we mean the elementary building blocks of a deformation. In linear and dual quaternion skinning, the deformation primitives are the individual skinning transformations $\mathbf{T}_1, \ldots, \mathbf{T}_m \in \mathbb{R}^{3\times4}$. Intuitively, these transformations specify how parts of the input object will deform. This is very obvious with binary skinning weights, which simply partition the input object into several parts. Smooth skinning weights then mean that there will be smooth blending between the individual parts. In character animation, these parts typically correspond to bones, e.g., the forearm, upper arm, etc.

While affine transformations are certainly the most common deformation primitives, there are other possibilities which present certain benefits, pioneered by works such as [Singh and Fiume 1998; Kalra et al. 1998; Hyun et al. 2005]. For brevity, we refer to deformation primitives as *deformers*. The work of Forstmann and colleagues [Forstmann and Ohya 2006; Forstmann et al. 2007] introduces spline-based deformers, see Figure 8. The idea is to bind rest-pose vertices $\mathbf{v}_i$ to a spline curve. Specifically, for vertex $\mathbf{v}_i$ we find the closest point on the spline, and bind $\mathbf{v}_i$ to the Frenet frame at this point. At run-time, we evaluate the deformed spline including its Frenet frames, which determine the transformation of $\mathbf{v}_i$. Each spline corresponds to one deformer. Multiple spline-based deformers are blended together linearly, similarly to linear blend skinning.

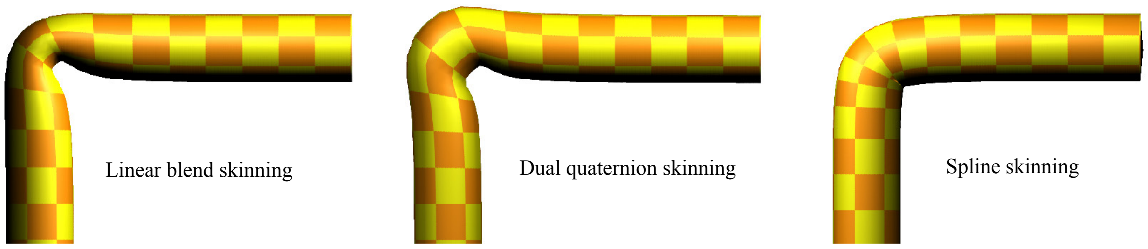

Figure 8: *Spline skinning offers more accurate control of deformations then linear or dual quaternion skinning.*

[Gregory and Weston 2008] present an advanced version of the previous idea, called *Offset Curve Deformations* (OCD). OCD supports spline and other smooth curves; for concreteness, we will assume B-spline curves. Similarly to [Forstmann et al. 2007], OCD connects a "master" B-spline to the bones of our object, smoothing out their piecewise-linear nature. The main idea of OCD is to associate each vertex $\mathbf{v}_i$ with an individual B-spline, which is an offset curve of the "master" B-spline. In contrast to spline-based skinning [Forstmann et al. 2007], this allows us to better control the distribution of the deformations, especially on the inside and outside of a bend. Because B-splines are linear, the resulting skinning technique is also linear. It turns out that OCD is equivalent to Animation Space, discussed in Section 3. A great benefit of the OCD technique is that its parameters can be adjusted in a user-friendly way.

Jacobson and Sorkine [**?**] observed that linear or dual quaternion skinning is unable to control deformations along the bone; the blending is constrained to a typically rather small region near the joint. Around the bone, the weights are typically binary (1 for the closest bone and 0 for the others), which does not allow us to control deformations of the bone itself. Stretching and twisting of bones, highly desirable e.g. with stylized or cartoon characters, can be supported by using an additional weight function, called *endpoint weight* [Jacobson and Sorkine 2011]. The endpoint weight varies from 0 to 1 as we move from one bone endpoint to the other; there is one endpoint weight for each bone. The endpoint weight provides us with the information where a vertex $\mathbf{v}_i$ is located with respect to the bone, which allows us to implement effects such as stretching and twisting of the bones, see Figure 9.

Focusing on deformations in the vicinity of a joint, Kavan and Sorkine [2012] propose a general concept of *joint-based deformers*. A joint-based deformer is a function $\Gamma : SO(3) \times \mathbb{R}^3 \to \mathbb{R}^3$ which takes a joint rotation and a rest-pose vertex position as input and produces a deformed vertex position as output. The swing-twist deformer, discussed in Section 4, is a specific example of a joint-based deformer. Individual deformers are blended linearly, similarly to linear blend skinning. The idea is that artifacts such as candy-wrappers will be avoided by the deformers themselves; the blending serves only to ease the transitions between the individual deformers, assuming there are no large discrepancies between the blended deformers.

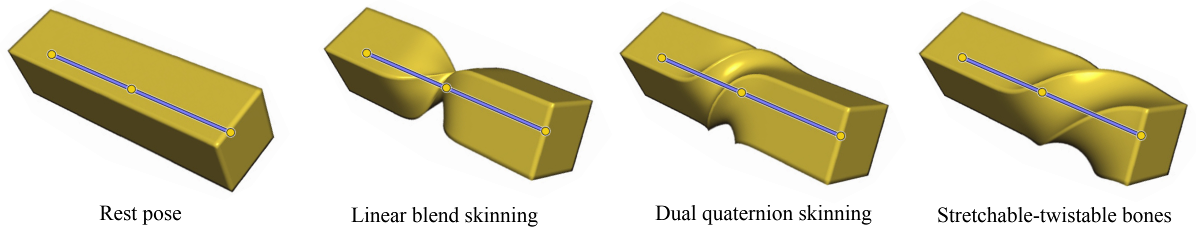

Figure 9: *Unlike linear and dual quaternion skinning, stretchable and twistable bones [Jacobson and Sorkine 2011] allow us to spread deformations along the length of a bone.*

6 Computing normals of skinned surfaces

In the previous sections, we discussed deformations of the actual shape, represented using a polygon mesh. For rendering, it is necessary to calculate not only deformed vertex positions $\mathbf{v}'_i \in \mathbb{R}^3$, but also their corresponding normals $\mathbf{n}'_i \in \mathbb{R}^3$. Of course, once the deformed vertex positions $\mathbf{v}'_i$ have been computed, the corresponding normals can be estimated by averaging normals of adjacent triangles with appropriate weights [Botsch et al. 2010]. While easy to implement, this approach is not well suited for parallel processing, because the normals computation step would have to wait until all $\mathbf{v}'_1, \dots, \mathbf{v}'_n$ have been computed. Especially in GPU implementations of direct skinning methods, it is advantageous to calculate the deformed normals $\mathbf{n}'_i$ along with the vertex positions $\mathbf{v}'_i$.

If our model is deformed by a global linear transformation $\mathbf{M} \in \mathbb{R}^{3\times 3}$ (translations obviously do not affect the normals), the normals transform by its inverse transpose: $\mathbf{M}^{-\mathsf{T}}$. With linear blend skinning, we can use this technique with $\mathbf{M}_i = \sum_{j=1}^{m} w_{i,j}\mathbf{T}_j$. For dual quaternion skinning we can calculate the matrix $\mathbf{M}_i$ similarly, by blending unit dual quaternions. In either case, the normals are computed as:

$$\mathbf{n}'_i = \mathbf{M}_i^{-\mathsf{T}}\mathbf{n}_i \tag{16}$$

While this method is straightforward and very common, the normals computed this way are not always a good approximation of the true normals which we would obtain by averaging normals of adjacent triangles. This is because for skinned models (either linear or dual-quat), the transformation matrix $\mathbf{M}_i$ changes from one vertex to another; Equation (16) is correct only in parts of the mesh where the skinning weights are constant, i.e., $\mathbf{M}_i$ is constant. In areas where skinning weights have a non-trivial gradient, Equation (16) leads to biased normals, illustrated in Figure 10. In this figure, the shape is transformed by two skinning transformations $\mathbf{T}_1$ and $\mathbf{T}_2$. The problem is that the entire deformation has been achieved only by translating $\mathbf{T}_2$; the linear parts of both $\mathbf{T}_1$ and $\mathbf{T}_2$ are identities.

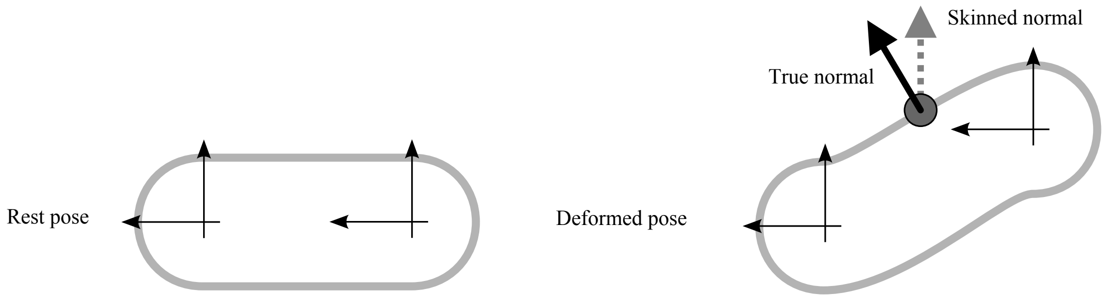

Figure 10: *A 2D capsule object demonstrating that skinned normals can be a poor approximation of true, geometric normals.*

The challenge of calculating more accurate normals of skinned surfaces has been opened by Merry et al. [2006b] and further refined by Tarini et al. [2014]. The key idea is to assume that skinning weights are continuous functions, which allows us to define the gradient of a weight in a particular vertex: $\nabla w_{i,j} \in \mathbb{R}^3$. Weight gradients allow us to obtain a more accurate approximation of the Jacobian of the skinning transformation:

$$\mathbf{J}_i = \sum_{j=1}^{m} w_{i,j}\mathbf{T}_j + \sum_{j=1}^{m} \mathbf{T}_j\mathbf{v}_i(\nabla w_{i,j})^{\mathsf{T}} \tag{17}$$

If we compute our normals as $\mathbf{n}'_i = \mathbf{J}_i^{-\mathsf{T}}\mathbf{n}_i$, we obtain skinned normals that correctly account for effects such as those shown in Figure 10. [Tarini et al. 2014] discusses many practical improvements of this idea, such as how the inversion of the Jacobian can be avoided, and presents a detailed implementation recipe.

7 Upcoming trends and open problems

In the previous sections, we attempted to cover the main trends in direct skinning methods. It is important to note that this list is not exhaustive and almost certainly will change in the future. An excellent example of a recently hatched technique is Implicit Skinning [Rodolphe et al. 2013].

8 Acknowledgements

Many thanks to Marc Alexa, Daniele Panozzo, Sven Forstman, Cengiz Oztireli, Jarek Rossignac, Ilya Baran, and Bruce Merry for sharing their results and expertise.

References

ALEXA, M., AND MÜLLER, W. 2000. Representing animations by principal components. In *Computer Graphics Forum*, vol. 19, Wiley Online Library, 411–418.

ALEXA, M. 2002. Linear combination of transformations. In *ACM Transactions on Graphics (TOG)*, vol. 21, ACM, 380–387.

BADLER, N. I., AND MORRIS, M. 1982. Modelling flexible articulated objects. In *Proc. Computer Graphics' 82, Online Conf*, 305–314.

BLOOM, C., BLOW, J., AND MURATORI, C., 2004. Errors and omissions in Marc Alexa's "Linear combination of transformations". `http://www.cbloom.com/3d/techdocs/lcot_errors.pdf`.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LÉVY, B. 2010. *Polygon Mesh Processing*. AK Peters.

BUSS, S. R., AND FILLMORE, J. P. 2001. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics (TOG) 20*, 2, 95–126.

CORDIER, F., AND MAGNENAT-THALMANN, N. 2005. A data-driven approach for real-time clothes simulation. In *Computer Graphics Forum*, vol. 24, Wiley Online Library, 173–183.

FAURE, F., GILLES, B., BOUSQUET, G., AND PAI, D. K. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30* (August), 73:1–73:10.

FORSTMANN, S., AND OHYA, J. 2006. Fast skeletal animation by skinned arc-spline based deformation. In *Proc. Eurographics, short papers volume*.

FORSTMANN, S., OHYA, J., KROHN-GRIMBERGHE, A., AND MCDOUGALL, R. 2007. Deformation styles for spline-based skeletal animation. In *Proc. SCA*, 141–150.

GOVINDU, V. M. 2004. Lie-algebraic averaging for globally consistent motion estimation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, IEEE, I–684.

GREGORY, A., AND WESTON, D. 2008. Offset curve deformation from skeletal animation. In *ACM SIGGRAPH 2008 talks*, ACM, 57.

HANSON, A. J. 2005. Visualizing quaternions. In *ACM SIGGRAPH 2005 Courses*, ACM.

HEJL, J. 2004. Hardware skinning with quaternions. *Game Programming Gems 4*, 487–495.

HYUN, D.-E., YOON, S.-H., CHANG, J.-W., SEONG, J.-K., KIM, M.-S., AND JÜTTLER, B. 2005. Sweep-based human deformation. *The Visual Computer 21*, 8-10, 542–550.

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph. 30*, 6, 165:1–165:8.

KALRA, P., MAGNENAT-THALMANN, N., MOCCOZET, L., SANNIER, G., AUBEL, A., AND THALMANN, D. 1998. Real-time animation of realistic virtual humans. *Computer Graphics and Applications, IEEE 18*, 5, 42–56.

KAVAN, L., AND SORKINE, O. 2012. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) 31*, 6, 196:1–196:8.

KAVAN, L., AND ŽÁRA, J. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM, 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.

KAVAN, L., COLLINS, S., AND O'SULLIVAN, C. 2009. Automatic linearization of nonlinear skinning. In *Proc. I3D*, 49–56.

KIM, Y., AND HAN, J. 2014. Bulging-free dual quaternion skinning. *Computer Animation and Virtual Worlds 25*, 3-4, 323–331.

LEE, G. S., LIN, A., SCHILLER, M., PETERS, S., MCLAUGHLIN, M., AND HANNER, F. 2013. Enhanced dual quaternion skinning for production use. In *ACM SIGGRAPH 2013 Talks*, ACM, 9.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH*, 165–172.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Graphics Interface*, 26–33.

MCCARTHY, J. M. 1990. *Introduction to theoretical kinematics*. MIT press.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4, 1400–1423.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Normal transformations for articulated models. In *ACM SIGGRAPH 2006 Sketches*, ACM, 134.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3 (July), 562–568.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 471–478.

NEUMANN, T., VARANASI, K., WENGER, S., WACKER, M., MAGNOR, M., AND THEOBALT, C. 2013. Sparse localized deformation components. *ACM Transactions on Graphics (TOG) 32*, 6, 179.

ÖZTIRELI, A. C., BARAN, I., POPA, T., DALSTEIN, B., SUMNER, R. W., AND GROSS, M. 2013. Differential blending for expressive sketch-based posing. In *Proc. SCA*.

PARK, S. I., AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. In *ACM Transactions on Graphics (TOG)*, vol. 25, ACM, 881–889.

RODOLPHE, V., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Transaction on Graphics (TOG). Proceedings of ACM SIGGRAPH*.

ROSSIGNAC, J., AND VINACUA, Á. 2011. Steady affine motions and morphs. *ACM Transactions on Graphics (TOG) 30*, 5, 116.

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2005. Simple and efficient compression of animation sequences. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 209–217.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, vol. 19, ACM, 245–254.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 405–414.

TARINI, M., PANOZZO, D., AND SORKINE-HORNUNG, O. 2014. Accurate and efficient lighting for skinned models. In *Computer Graphics Forum*, vol. 33, Wiley Online Library, 421–428.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proc. SCA*, 129–138.

WEBER, J. 2000. Run-time skin deformation. In *Proceedings of game developers conference*.