Spring Rigs for Skinning

Nicholas Toothman
University of California, Davis
California State University, Bakersfield
njtoothman@ucdavis.edu

Michael Neff
University of California, Davis
mpneff@ucdavis.edu

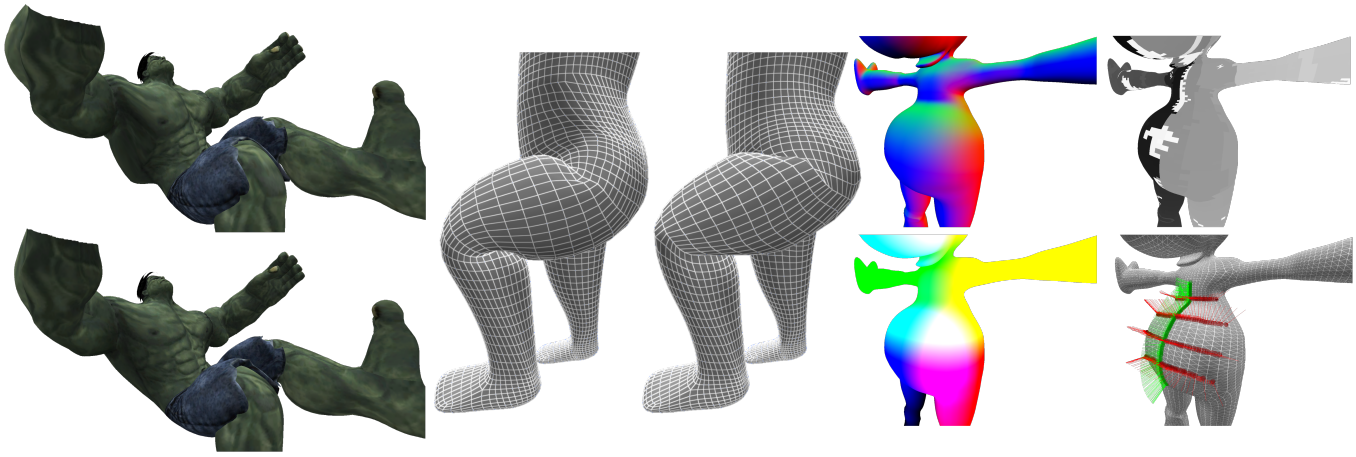

Figure 1: Left: LBS artifacts from skin weights (top) resolved with spring forces (bottom). pose without surface edits (top) and with (bottom). Middle: hip and knee bend with LBS (left) and spring forces (right). Right deferred rendering output for surface normals (top left), positions (bottom left), and primitive IDs (top right), used to draw accurate strokes directly on the mesh surface (bottom right).

ABSTRACT

Animation tools have benefited greatly from advances in skinning and surface deformation techniques, yet it still remains difficult to author articulated character animations that display the free and highly expressive shape change that characterize hand-drawn animation. We present a new skinning representation that allows skeletal deformation and more flexible shape control to be combined in a single framework, along with an intuitive, sketch-based interface. Our approach offers the convenience of skeletal control and smooth skinning with the functionality to embed surface deformation and animation as a core component of the skinning technique. The approach binds vertices to attachment points on the skeleton, defining a vector from bone to surface. Three types of springs are defined: intervertex springs help maintain surface relationships, springs from vertices to the attachment point help maintain appropriate bone offsets, and torsion springs around these attachment vectors help with deformation control as bones rotate. Edits to the mesh surface can also be represented by varying the radial length and direction of these vectors, enabling a new range of expressive power. Use of sketch-based interfaces and graphics hardware make both skeletal and mesh deformation simple to control and fast enough for interactive use.

MIG '19, October 28–30, 2019, Newcastle upon Tyne, United Kingdom

ACM ISBN 978-1-4503-6994-7/19/10…$15.00
https://doi.org/10.1145/3359566.3360074

CCS CONCEPTS

• **Computing methodologies** → **Animation**; **Shape modeling**; *Mesh geometry models.*

KEYWORDS

skinning, binding, surface deformation, animation

ACM Reference Format:
Nicholas Toothman and Michael Neff. 2019. Spring Rigs for Skinning. In *Motion, Interaction and Games (MIG '19), October 28–30, 2019, Newcastle upon Tyne, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3359566.3360074

1 INTRODUCTION

One of the strengths of hand-drawn character animation is the free shape change that it allows. Consider how classic characters ranging from Snow White to Aladdin freely change form to maximize the clarity of their communication. This expressive shape change empowers their impact. Creating fluid shape change for 3D digital characters is possible, but difficult due to the nature of the medium; a 3D character mesh may contain thousands of vertices,

and manually adjusting them over a series of frames to achieve a desired shape change is infeasible. Instead, shape is largely controlled with abstract handles, such as lattice deformers for free-form deformation and skinning for anatomical deformation. Skinning is the process of discretely associating a mesh's vertices with the skeleton's bones and defining the algorithm that translates the skeleton's motion to the mesh. While powerful, these techniques do not provide the easy specification of arbitrary shape achieved by pencil on paper.

Skeleton rigs have long been a useful abstraction for posing characters. However, skeleton-based representations lead to rigid 3D characters with limited deformation that is concentrated around the joints, and adding more deformers requires significant rigging work. In this paper, we present an extension of skeleton-based techniques designed to support easy shape change while still maintaining the advantages of skeletal abstraction. In particular, we wish to support shape change that may or may not be physically correct, as the artist desires, but enhances the artist's expressive power. Our approach consists of: a new skinning representation that addresses classic skinning artifacts while also allowing shape changes to be encoded; a set of tools for controlling shape change; and a sketch-based interface to make it intuitive to author skeleton and surface deformations in a single framework. Employing GPU acceleration, the approach is suitable for real-time use.

The basic idea of our skinning approach is to extend traditional techniques to allow radial and other forms of scale that are not aligned with the direction of the bones. Recent work extended traditional skinning to allow bone stretching [16], but not off-axis scale. We achieve this by making explicit the connection between each mesh vertex and an attachment point on a bone. The vector between a surface vertex and its attachment point defines a scale vector that can be used to deform the mesh. See Figures 3 and 2 for a glimpse at attachment points and scale directions.

As the skeleton moves, our skinning method updates vertex locations using a combination of two techniques: first, a rigid transformation that tracks the motion of the bone, followed by an iterative solver that removes discontinuities between adjacent vertices through the use of torsion, linear, and prismatic springs. In addition to eliminating common skinning artifacts, this approach enables configurable shape control and surface-level deformations that compose nicely with anatomical deformations from the skeleton.

Introducing scale vectors as a free parameter in the skinning formulation affords two main benefits: the shape of the mesh can be better controlled around a bending joint and a large range of off-axis deformations can be achieved to support free shape change. Parameterized controls adjust the performance of joint deformations. We present tools for authoring significant surface deformations and combining them into animations. A sketch-based approach is used to unify the skeleton and surface deformations into a single, coherent and intuitive workflow.

2 RELATED WORK

2.1 Skinning

Skinning allows the movement of a high-resolution character mesh to be controlled by a comparatively low degree of freedom skeleton, with joint rotations providing an intuitive control abstraction. Linear blend skinning (LBS), also known as skeletal subspace deformation (SSD) and smooth skinning, allows vertices to be associated with multiple bones by specifying weight values indicating their influence [31]. The skinned position of each vertex is the weighted average of the transformation associated with each bone. Despite the artifacts caused by linearly blending transformation matrices, including the "candy wrapper" collapse from twist rotations and increasing volume loss for large bends, LBS remains a dominant technique because of its speed and simplicity. Dual Quaternion skinning (DQS) represents joints with screw transformations, which prevents the artifacts encountered in LBS, but can introduce undesired spherical bulge around large joint bends and tightly-packed twist rotations around joints [20].

Extensive work has been done to alleviate skinning artifacts. Rigging additional support joints can mitigate volume loss for large bends, at the cost of recomputing skinning weights when the skeleton topology changes. For DQS, bulge artifacts can be corrected with a post-processing stage in the animation pipeline to restore original distances between vertices and bones [24]. Computing skin weights of a higher quality is a popular approach. Accounting for surface or volume when calculating skin weights improves the distribution of joint influence across the mesh [4, 9, 15]. Employing finite element models to compute skin weights can also result in more elastic behavior [21]. Leveraging example mesh poses to compute skin weights yields results that accurately reproduce the poses, but requires iterations of pose modeling and weight computation until the desired fit is achieved [32].

Besides changing skin weights, another strategy is to change the skinning algorithm. Le and Hodgins precompute optimized centers of rotation for clusters of similar skin weights, then uses a decomposed LBS algorithm to enforce more consistent transformations for each cluster [27]. These centers of rotation are similar to our attachment points, but they are not interchangeable: the centers are optimized for blend-based skinning, not for even distribution along the skeleton, and will not produce satisfactory deformations if used as attachment points.

Stretchable, twistable bones (STB) addresses artifacts caused by changing bone lengths and allows twist to be spread along a bone [16]. An extension of LBS, STB decomposes joints into separate rotation, translation, and scale transforms, then skins the mesh using traditional skinning weights as well as *endpoint weights*, which map a vertex to a parameterized position along the length of a bone. Our method uses similar endpoint weights to compute attachment points for each vertex. It is worth mentioning the use of mesh contraction for computing attachment points, where attractive and repulsive surface forces "deflate" a mesh until it approximates a skeleton [3]. However, this technique is best suited for meshes without any skeleton, and our contribution makes use of rigs with skeletons already built.

Traditionally, skinning alone does not support mesh-level or secondary edits. Volumetric or cage-based methods aim to support character posing and a degree of elastic deformation. Harmonic coordinates rig a mesh to a bounded cage volume for deformation using generalized barycentric coordinates, but mesh editing is limited by the cage resolution, and achieving a particular mesh pose can require unwieldy cage transformations [17]. Related approaches

use mean-value coordinates [18] or higher order barycentric coordinates [26]. Another approach is to estimate a pose space of mesh deformations from examples [29, 43] that can then be interpolated. Implicit skinning achieves real-time elastic deformation by transforming and compositing isosurfaces made from mesh partitions, then using them to adjust vertex positions and produce natural features, such as contact bulges on joint bends. [46, 47]. Implicit surfaces have also been for surface authoring, such as sketch-based editing [2] and making plausible volumetric deformations alongside skinning [36].

Other work has proposed methods to provide skeletal control over characters while supporting elastic deformations. Capell et al. achieve skeletal control and secondary motion from physical forces by use of a volumetric mesh with bones confined to edges in the control lattice [7]. This method aims to strike a balance between skeleton-driven and secondary animation. This method depends on physical simulation, although the use of a skeleton helps to localize and reduce the computation cost. Other approaches calculate a deformation energy over the mesh that controls how much it can bend (e.g. [5, 6, 39, 49]). While powerful, physics-based approaches do not accommodate the full range of unrealistic, free shape deformations often used in more cartoony animations, as favored by our method. However, our strategy of using position-based dynamics to correct artifacts from other skinning methods has been previously applied to meshes using stretch and position constraints on tetrahedra [1, 35, 37].

The idea of radial offset has been used with spline-based skeletal animation with impressive results [11]. Here, the authors apply spline-aligned deformation to circumvent LBS artifacts [40] and attach radial FFD grids to each joint, which allows for frontal, lateral, and radial scaling to be applied as deformation styles. Skeletal animation with deformation styles is possible, but the deformation style itself is static. Our method enables both animated surface and joint deformations to occur.

The technique most similar to our method is projective skinning, which also employs dynamics (projective rather than position-based), avoids skin weights, and connects the surface and skeleton [25]. Here, the authors volumize the skeleton and resolve contacts with the surface, while we project the surface onto the skeleton. Their approach to deformation involves least-squares energy minimization to manage elastic strain, skin stretch, and collision, while ours is to simulate springs of varying stiffness between the mesh and skeleton until a desired shape is achieved.

2.2 Deformation

In contrast to skeleton-based deformation, mesh deformation schemes offer superior control over arbitrary mesh regions. Free-form deformation (FFD) lattices are well established, simple to create, and powerful tools for deforming space [30, 38]. However, such techniques lack the high-level, anatomic structure of skeletons. Performing edits like adding muscle bulges or creases becomes easier, but bending an elbow or posing fingers becomes more difficult. While it is possible to hierarchically assign FFD lattices to bones for simultaneous mesh deformation and skeletal animation, they fail to adhere to skinning constraints. Keyframing lattice control points can generate animation, but using even small lattices in 3D ($4 \times 4 \times 4$) quickly raises the number of controls the animator must manage. Lattices also lack the ability to preserve local geometry over a mesh, so features like wrinkles may get lost if the deformation is extreme. In comparison, Laplacian surface editing (LSE) is capable of deforming the mesh while preserving such features by minimizing changes to a point's world and differential coordinates [42]. This technique depends on the user specifying a region of interest (ROI) to deform. Recent work has explored methods to automatically specify ROIs with minimal effort by the user [50]. As with lattices, LSE-based approaches lack the articulated order of rigged models. As our chosen method for surface deformation, we supply LSE with ROIs created by drawing directly on the mesh surface.

2.3 Sketch

Sketch-based methods for posing and animation have matured considerably in recent years, as part of a growing interest in sketch as a flexible input modality ([12, 14, 19, 28, 33, 45]). Using sketch interfaces is appealing for a number of reasons. Aside from providing a natural interface for user input, sketch systems excel in situations where coarse input and speed are favored. Kho and Garland [22] controlled free-form mesh deformations by sketching over a region and then sketching a new position for the region. Our approach also uses sketch for region definition and deformation, but operates on existing skeletons and enables explicit region selection for precise control.

The line of action concept from traditional hand-drawn animation works particularly well for defining a character's overall pose and has been explored in recent work [13, 34]. When used for 3D applications, inferring depth values for 2D input can be ambiguous. Avoiding this ambiguity has proven to be a challenging problem with approaches varying depending on the problem domain, from assisted methods that predict and suggest options as the user sketches [44, 45], to computational methods that rely on constraints to find a suitable answer [8]. In our case, strokes defining regions for deformation are placed precisely on the surface with the use of deferred rendering and framebuffer objects, which we will cover in the Implementation section.

We employ a sketch interface for both skeletal posing and mesh deformation. This provides a unified workflow across the full range of operations supported in our system. Sketch allows rapid exploration of ideas with minimal effort, allowing users to remain in "flow", rather than having to resort to options and menus for every action. It provides control that is both accessible for novices and sufficiently precise for experts.

3 SKINNING

To enable expressive deformation, we present a new skinning formulation (Sec. 3.2) that is designed to easily accommodate free shape change (Sec. 4) using sketch-based interfaces (Sec. 5).

3.1 Attachment binding

Our skinning method operates by directly connecting each vertex to a bone, resulting in a scale vector between each mesh vertex and its attachment point on the skeleton. Formally, for a vertex v_i, the attachment point $a_i = p + t_i(c - p)$ is a position along the bone

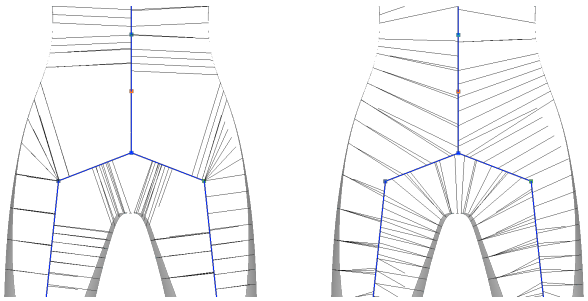

Figure 2: Orthographic cross-section of character mesh showing attachment binding on edge vertices. Left: direct projection produces gaps that causes discontinuity when joints rotate. Right: 6 iterations of smoothing and reprojection resolve most of the gaps.

between parent and child joint positions p and c parameterized by $t_i \in [0, 1]$. The scale vector $s_i = v_i - a_i$ defines the vertex relative to its attachment point. For the best skinning results, attachment points should be chosen to minimize the length of s_i, and adjacent vertices should have similarly adjacent attachment points. A direct projection test onto each bone may produce results that satisfy the first requirement, but in many cases, can also create large gaps on the skeleton between points (see Figure 2-left). If the mesh already has skinning weights, these can be used to speed up attachment point computation: v_i is projected onto each of the bones immediately attached to the joint with the greatest skin weight, and the closest projection is chosen for a_i. While these results are sufficient for skinning, they can be improved to reduce deformation time. Applying Laplacian smoothing ([10]) on the attachment points sets a_i to the average of its neighbors, $a_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} a_j$; following this step, re-projecting a_i onto the nearest bone closes gaps between points and maintains adjacency (see Figure 2-right). This can be repeated a number of times, but is best used sparingly, as excessive smoothing can draw attachment points too close together on long bone chains. In practice, we use 6 iterations of projection and smoothing.

3.2 Skinning Formulation

Skinning is calculated in two stages. First, the vertex v_i's attachment point a_i is rigidly transformed by its bone. Computing the skinned attachment point $a_i' = p' + t_i(c' - p')$ permits changes in bone position and length. Next, the rigid skinned vertex position v_i' is found as: $v_i' = a_i' + R(p)(s_i) = a_i' + s_i'$, where $R(p)$ is the rotation component of p's transform. This form of rigid skinning allows the vertices to track the bone movements, but introduces rotation discontinuities around joint bends (see Figure 3-left). These are resolved in the second stage through the use of spring forces to minimize differences between vertices and remove the discontinuities. The second stage iterates until convergence, resulting in a smooth deformation (Figure 3-right).

To resolve discontinuities and smooth out shapes, we use up to 4 types of spring-driven forces. Spring forces are computed according to Hooke's law, $F = kx$, where x is the displacement from rest and k is a stiffness coefficient. For two of the forces, the springs used to

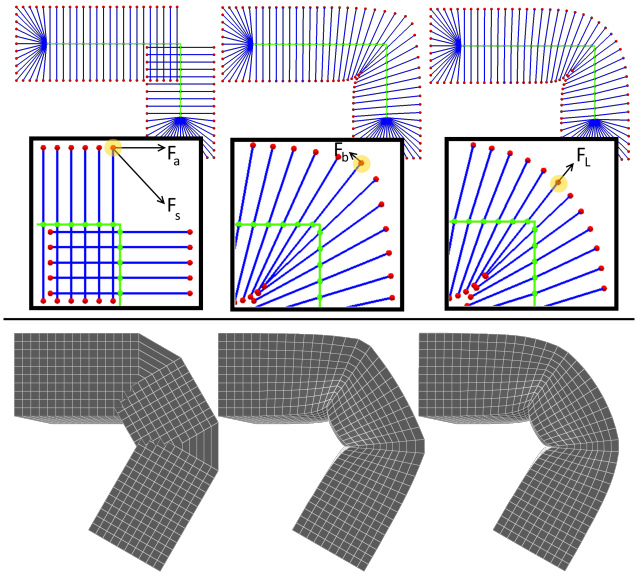

Figure 3: Top: Applying spring forces to 2D cross-section. Surface vertices are red, bones and attachment points are green, and scale vectors are blue. Spring forces are illustrated for the vertex highlighted in yellow. Left column: bone torque F_b has no effect on rigidly-skinned meshes, but attachment torque F_a and surface edge F_s are computed to close the gap between a vertex and its neighbors. Middle: F_b, F_a, and F_s resolve gaps, but leave a pointed outer bend. Right: increasing F_s and decreasing F_l help the vertex reach equilibrium without collapsing too close to its attachment point on the skeleton.

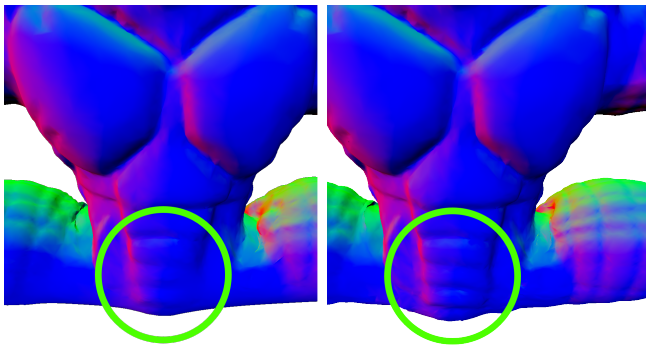

Figure 4: Controlling shape change during skinning. Left: using only F_s diminishes muscle features. Right: including F_l with F_s helps preserve features.

relax the mesh act between neighboring vertices, so the net force computed for v_i is a sum of forces between vertex i and its one-ring neighbors $j \in N(i)$. Thus, the displacement x is the difference between a measure for the current and bind values of v_i and v_j. The first of these neighborhood forces is called the attachment torque force, $F_a = k_a x_\theta$, where x_θ is the change in angle between s_i and

s_j from their angle at bind. By itself, F_a rotates a vertex around its attachment point to minimize angles with its neighbors, which can introduce skewing across the mesh when the skeleton bends. To counter this, a bone torque force F_b attempts to restore the bind angle between s_i and its attaching bone. If the mesh has been rigidly skinned and F_b is the only acting force, $|F_b| = 0$, but when joined with F_a, the two forces act to drive anatomical deformation. However, because these two forces are constructed to not change $|s_i|$, they may produce artifacts on joint rotations where the outer bend region develops a pointed crease (Figure 3-middle).

Removing this artifact requires two additional forces. First, a linear spring force F_s is added to restore relative edge lengths between adjacent vertices, resolving overstretch and compression (Figure 3-right). In this case, x is the difference of the current and bind edge length between v_i and v_j. Because there is no relation between the mesh surface and the skeleton in F_s, by itself it is likely to reduce mesh volume on large deformations, which can also lead to loss of surface details (see Figure 4). This is addressed with a final linear force F_l that attempts to restore s_i to its bind length.

To compute the net force on a vertex, F_b and F_a are converted from angular forces to linear. The magnitude of these forces is computed as $kx|s_i|$, where x is either the displacement angle between neighbors for F_a or between s_i and its bone for F_b. The force directions are assumed to be orthogonal to s_i and use the cross product between s_i and s_j for F_a, or between s_i and the bone direction for F_b, as the axis of rotation. With the forces all in linear form, the net force on v_i is:

$$F_{\text{net}}(i) = F_a + F_b + F_s + F_l \tag{1}$$

On each solver iteration, the vertex position is updated as:

$$v_i'' = v_i' + dt \times F_{\text{net}}(i), \tag{2}$$

where dt is a time step parameter. Equation 2 behaves in the same manner as used by Wilhelms and Van Gelder [48], in that the force value computed is not used for full physical simulation, but instead to help vertex positions converge to a rest state. In implementation, however, it is possible to use semi-implicit Euler integration with the forces instead, and adding a damping term for velocity can generate effects such as force propagation along the surface. For stability and memory requirements, the approach in Equation 2 is preferred.

To some degree, the system's timestep and epsilon variables can influence the solver's behavior and final shape. We find it most appropriate to keep the timestep fixed at $\frac{1}{30}$ seconds. Increasing this may cause larger discontinuities to resolve in fewer iterations, but it can also introduce oscillations before convergence. The solver's iteration and convergence behavior is flexible. Our system is configured to repeat execution until a maximum count of 50 has been reached, or when the largest vertex position change between solves is below a threshold initially set to $AABB_{min} \times 1e{-}4$, where $AABB_{min}$ is the smallest dimension of the model's axis-aligned bounding box. In practice, the timestep and threshold variables may require adjusting at initialization, but then may be left intact. While the timestep and threshold do impact the final mesh deformation, the primary choices for shape control are further discussed in Section 4.

3.3 Skinning Features

Applications requiring fast, simple skinning have generally relied on linear blend (LBS) or dual quaternion (DQS) skinning, or combinations and variations thereof. Each of these techniques computes a weighted average of multiple transformations in order to determine the final coordinates of each mesh vertex. The artifacts caused by this blending are well-known. The "candy-wrapper artifact" is a collapse of volume at the center of rotation that occurs with LBS for twist rotations. It peaks at 180 degrees when rotating a vertex in the child frame transforms it to be on the opposite side of the joint to the same vertex in the parent frame. Blending these two locations leads to collapse of the mesh. DQS avoids the collapse, but the spacing of twist around the joint depends primarily on skin weights, so further shape control requires weight editing. Our approach avoids both issues, allowing arbitrarily large axial rotations without collapse (Figure 5). Both DQS and LBS reset for large rotations, whereas our method will continue to accumulate the rotation.

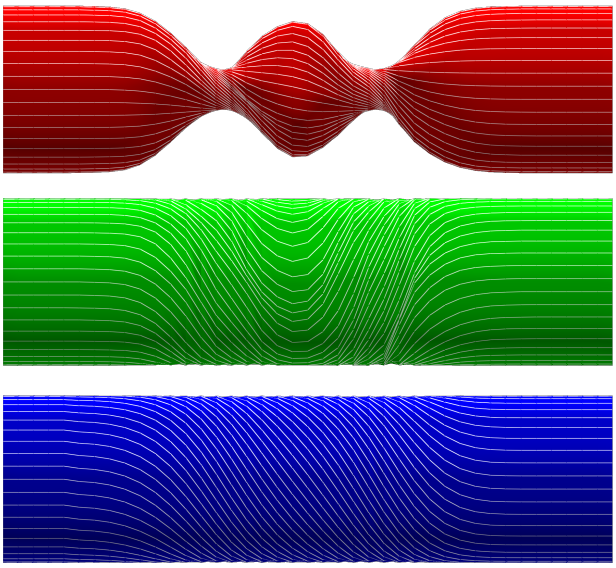

Figure 5: Axial rotation on a cylinder with LBS (red), DQS (green) and springs (blue). Where LBS begins to collapse on large twists, DQS better preserves vertex-skeleton distance, but can change twist direction as the twist increases. Spring forces avoid collapse and diffuse the twist across the mesh length.

Blended transform skinning also creates errors for large swing rotations. LBS will create a narrowing of the mesh, known colloquially as "macaroni elbow", while DQS creates a bulge. These artifacts are noticeable in Figure 6. Particularly noteworthy, the artifacts depend significantly on the skin weight distribution. Our approach avoids these artifacts and offers parameters to control the deformation behavior around joint bends.

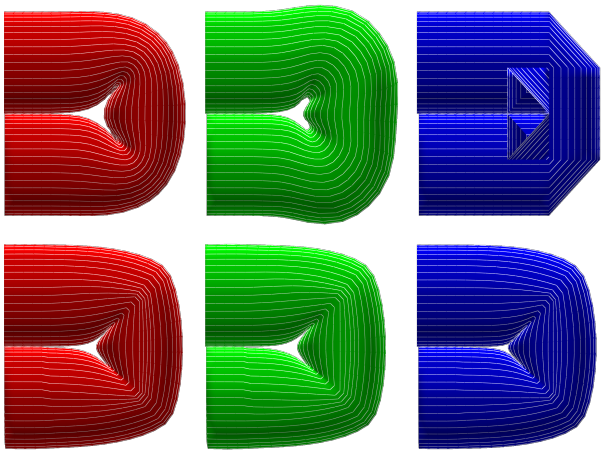

Figure 6: A 90 deg. rotation of a cylinder is shown. The top row shows LBS (red), DQS (green), and rigid skinning (blue). The bottom row shows the results of applying spring forces on the skinned mesh above it. The spring force results are relatively consistent regardless of initial skinning choice.

4 SHAPE CONTROL

Our skinning formulation supports a variety of techniques for , including skeleton and surface-based deformations. For blended-based skinning, shape control is mainly found through the act of creating and editing the skeleton and skin weights, but doing so to create subtle details in animation can be limited or tedious. Instead, we support a range of handles to determine the deformation behavior and final shape.

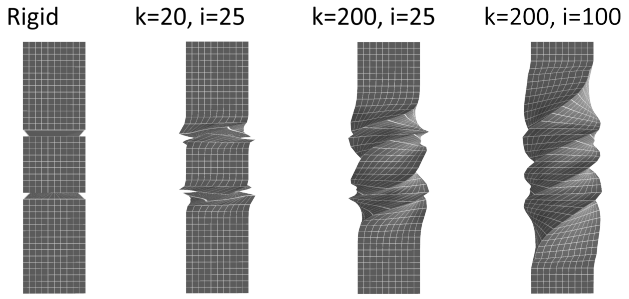

Figure 7: Altering stiffness coefficients (k) for F_a and iteration counts (i) to produce different shapes.

Stiffness coefficients are essential for computing the spring forces in Equation 1. Coefficients can be distinct for each vertex pair ($k(i,j) = k(j,i)$) as well as for each force type. For example, increasing the coefficient for F_a can lower the iteration count necessary to reach a desired shape, and lowering the coefficient for F_l can allow the surface to contract over large bends, allowing a range of shapes beyond classical skinning as shown in Figure 7. The choice of stiffness coefficients for each force provides some dimensions of shape control for the deformation. Our method supports individual coefficients for each adjacent vertex pair. For F_s, we use $k = \frac{a_i + a_j}{|v_i - v_j|^2}$, where a_i, a_j are the areas of the two triangles sharing the edge between i and j. This is shown to provide more accurate elastic behavior than uniform stiffness [48]. By default, the same stiffness coefficient is used for F_a. For F_b, we set $k = |\frac{\theta}{90}|$, where θ is the bind angle of s_i and its bone, to enforce greater stiffness for vertices whose s_i is orthogonal to the bone run. For F_l, $k = 1$. Finally, for overall shape control, each force type has a global coefficient scale that can be used to change the dominating forces.

Conditional coefficients can further add control over the final shape. Wilhelm and Van Gelder scale coefficients differently when edge lengths are lower than their bind values [48]. For F_s, this allows vertices near a joint's outer bend to continue relaxing while the inner bend quickly converges. Similarly, we can scale $k(i,j)$ for F_a or F_s when vertices i and j are attached to different bones.

Scale vectors are implicitly defined to run between vertices and their attachment points. This representation allows us to author surface-level deformations by altering scale vector magnitudes and directions between poses. With this system, we can designate specific vertices as kinematic (where their pose and animation is decided by the user), and then configure the solver to more strongly maintain their scale vector lengths and/or directions. Then the kinematic vertices will not get deformed by the solver as much as the non-kinematic neighbors, but will still drive relaxation across the surface. Over a number of iterations, the solver will effectively diffuse the kinematic edits to the nearby regions until it drops off. This behavior is further controlled by designating boundary and in-between vertices as well as kinematic vertices using regions of interest (ROIs), a popular technique commonly associated with surface editing techniques [41, 42].

4.1 Surface deformations

To produce surface deformations compatible with the spring force system, we select a sparse set of vertices to serve as kinematic handles by drawing a baseline stroke directly on the mesh surface. The region of interest is automatically computed around the handles using the mesh's adjacency data and a maximum neighbor distance set by the user. For example, if the distance is 4, then all vertices up to 4 edges away from the handles are included in the ROI. Drawing a baseline stroke also generates an offset stroke, which defines how the handles should deform. For each handle, we compute the change in scale vector length necessary to move it onto the offset stroke. The solver then applies F_l to the handles for their new lengths and runs normally for the rest of the ROI. Allowing a small fraction (10%) of F_s to the handles helps to soften the edit and help it blend with its neighbors. Through this technique, we can support surface deformation automatically in the skinning process as an extension of the spring forces. Each surface deformation is thus represented as a set of alternate scale vectors for the chosen region. Given multiple edits on the same mesh, we can interpolate between the sets of scale vectors to produce transitional surface animations. By updating the mesh's s_i values with these edits on each frame, we can drive mesh-based animations alongside the usual skeleton animations (Figure 8).

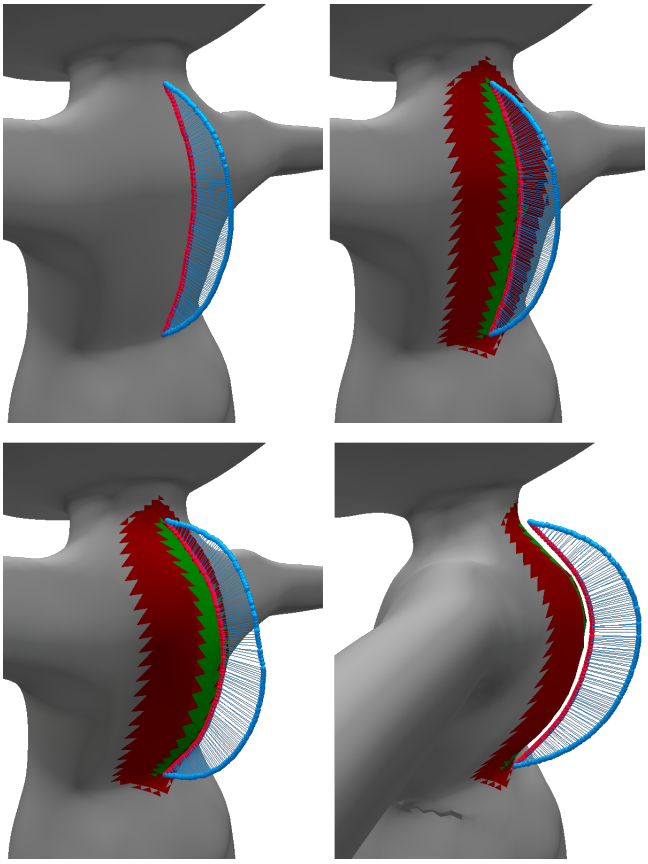

Figure 8: Surface editing with strokes. Top left: the baseline (red) is drawn down the spine and automatically produces an offset (blue). Top right: the region of interest defines kinematic vertices (green) which are deformed by the stroke, and passive vertices (red) which are deformed by springs. Bottom left: after applying the edit. Bottom right: after applying joint rotations in the spine.

5 SKETCH-BASED INTERFACE

Achieving a desired character shape may require numerous iterations and fine-tuning before the result is satisfactory. To encourage fast and easy exploration of ideas, we offer sketch-based controls for both skeleton posing and surface deformation.

5.1 Skeleton

Posing involves drawing a stroke across the skeleton to select nearby joints and establish a baseline reference, then drawing an offset curve to approximate the target shape. This input style resembles the line-of-action stroke mapping presented by Kho and Garland [23] and seen in differential blending [34], but with more explicit user control over the mapping. Each 2D coordinate of the stroke is projected from screen-space back into the 3D scene using a plane with its normal set opposite to the camera's view direction and its center at the camera's look-at position. The user can edit the plane position and normal to fit their needs, and after projection, the baseline can be adjusted to revise placement. Because user input is numerous and somewhat noisy, the screen-space stroke coordinates are reduced using the Ramer-Douglas-Peucker algorithm, and the remaining points serve as input for computing a composite Bézier curve.

The baseline selects skeleton joints by their screen-space distance from the stroke, then it compute a scalar t-value $\in [0, 1]$ for each joint to represent its closest point on the stroke. It also tracks the offsets between the joints and their closest points. When the user draws another stroke, it is also converted into a composite Bézier curve, and then used to calculate transforms between the two curves at each joints' t-values. Then, we sequentially transform each joint to produce the best fit along the new stroke while preserving the joints' offsets from the baseline. The user may continue drawing strokes until achieving the desired shape, or they can optionally transform control knots on the strokes for fine-tuning existing poses. Figure 9 depicts the process.

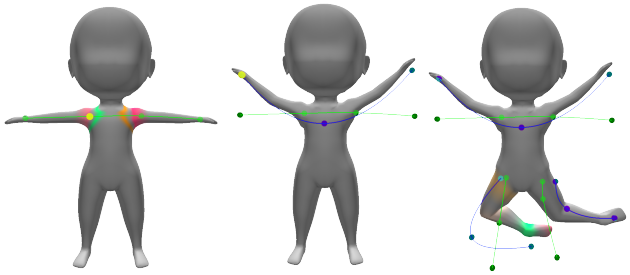

Figure 9: Posing the character with input strokes. The baselines (green) select joints and serve as references for the offset curves (blue). The process can be repeated and layered as needed.

5.2 Surface

Surface deformation requires the user to specify both a region of influence and the desired transformations of a subset of points within the region. We have experimented with several techniques for interacting with the 3D surface and developed a method based on direct mesh sketching. A previous sketch-based approach automates detecting and deforming silhouette vertices in screen-space, then using LSE to deform the surrounding area [50]. This is a powerful technique, but is limited to vertices that can be detected on clear edges. In our system, we permit the user to interactively create ROIs directly on the mesh surface. This required a new method for vertex selection, which we accomplish with a deferred rendering pipeline, described in Section 6.

6 IMPLEMENTATION

The skinning technique can be made very fast by appropriately leveraging graphics hardware. Because the spring forces run on iterations, using the standard vertex shader pipeline stage for this would require transform feedbacks to store and access the results of each pass. In addition, there is a fixed limit to the attributes per vertex accessible in the vertex shader. Instead, we implement our

skinning technique using compute shaders for OpenGL. Compute shaders can access as much data as the GPU can store through shader storage buffer objects (SSBOs). Mesh vertices are stored in SSBOs as structs containing vectors for position, attachment, and force. To avoid redundant computation, a preliminary compute shader runs before each iteration to compute and store the distance and angle between every unique edge pair. Then, the spring force solver looks up these edge values when computing forces for vertex neighbors. Because the solver runs in iterations, we use a double buffer to separately read and write results on each iteration, then the output buffer for each pass becomes the input buffer for the next pass. This design permits rapid iteration execution, as only the input and output bindings have to change between each pass. We also designed the solver to optionally restart if any parameters change, allowing the user to interactively adjust them until satisfied with the final shape.

For blend-based skinning, vertex normals are transformed by the vertex's weighted transform in the vertex shader stage. However, our use of a non-linear spring-based solver for mesh and surface deformations makes this approach unsuitable. To resolve this, we run a compute shader to update face normals, then a second compute shader to average these for vertices based on a dihedral angle threshold, which allows the mesh to portray both smooth surfaces and sharp edges. This GPU-accelerated approach for computing normals is sufficiently fast and negligible in cost compared to the deformation compute shader.

Because mesh deformation is decoupled from rendering, the actual rendering time is fast regardless of the deformation size, on the same order as rendering a static mesh with the same geometry count. After skinning with compute shaders and recomputing normals, we bind the contents of the SSBO as vertex attributes and render using typical shader stages. We also exploit a deferred rendering pipeline to query surface data needed for drawing strokes directly on arbitrary surfaces. In the fragment shader, we output surface color, vertex position, normal, depth, primitive IDs, and barycentric coordinates (generated in a geometry shader) to separate texture channels attached to a framebuffer. This leverages the rendering pipeline's rasterizing, interpolating, and z-buffering to save visible surface data for any camera and pose without additional effort or computation. This also makes it fast and simple to access these channel textures on the CPU, so creating a surface stroke then becomes a matter of querying texture values under the input device's screen coordinates. The result is a set of 3D positions, normals, and directions used to create Frenet frames and define a smooth stroke along the surface, which is used to create regions of interest and transformation handles. The deferred rendering output and examples of surface strokes are shown in Figures 1 and 8.

7 RESULTS

The deformation and rendering systems have been designed to support real-time deformations. With the default parameters, the spring force solver is capable of reaching convergence within 50 iterations while maintaining at least 24 frames per second. Considering the normal re-computation and deferred rendering that are also occurring in each frame, we find this acceptable for real-time use. Unlike LBS or DQS systems, our solver must access and compute more data than is possible in a typical vertex shader environment. Due to the iterations and overhead, spring skinning can take one or two orders of magnitude longer to compute than than typical GPU skinning; however, it still achieves acceptable rates for real-time animation for meshes of various sizes.

Table 1: Compute times (in milliseconds) for different iteration counts across input meshes. For reference, LBS performed in a compute shader completes in < 0.01 ms for all meshes.

Mesh	Bones	Vertices	Tris	I=25	I=50	I=100
Chibi	36	16314	32624	0.53	0.91	1.78
Box	4	1802	3600	0.28	0.54	1.10
Plus	17	5634	11264	0.34	0.59	1.27
Human	85	10774	21204	0.42	0.84	1.50
Armadillo	19	90000	180000	0.77	1.59	3.04

Table 1 presents a comparison of deformation compute times across various mesh sizes. The spring force solver executes until 25, 50, and 100 iterations have been reached. The software ran on a machine with an Intel i7-6700K (4.0 GHz) processor, 32 GB DDR4, and an nVidia Geforce GTX 1070 (8GB) graphics card. Although our method has much higher compute times than LBS, its implementation on graphics hardware enables it to achieve interactive framerates. Adding user-authored surface deformations introduces overhead on each frame due to buffer writes from the CPU to the GPU, but this tends to be a comparatively small set of vertex data to update. Making static, one-time mesh edits likewise has negligible effect on performance. As expected, the time needed for computing these edits is on the order of milliseconds and directly dependent on the region of interest size; larger regions and meshes take longer to compute than smaller ones.

8 DISCUSSION AND CONCLUSION

This approach to skinning offers anatomical and surface-level controls for mesh deformation in a single framework in real-time. The spring force solver's parameters enable a variety of final mesh shapes ranging from naturalistic to exaggerated cartoon styles, with performance comparable to established methods. It performs quite well with or without bone stretch, twist, length adjustment, or surface edits, and it can readily use rigs with existing skeletons and skin weights. The examples shown in Figures 1 demonstrate its value as a skinning techniques as well as its flexibility. This method behaves well for meshes of various thickness and skeleton configurations, and it is highly customizable for the expression of a wide range of visual effects, although it may require some tuning from the user to receive the best quality. The deformation quality is comparable to LSE and ARAP, but for sufficiently large surface deformations, these solvers do not account for a skeleton rig, and the resulting shape may not conform to the skeleton's pose. Our approach supports believable surface accommodation for both natural and exaggerated character posing.

Issues to be aware of include the reliance on vertex attachment points. This is similar to how LBS depends primarily on skin weight quality. The compensation for this issue is finer shape control

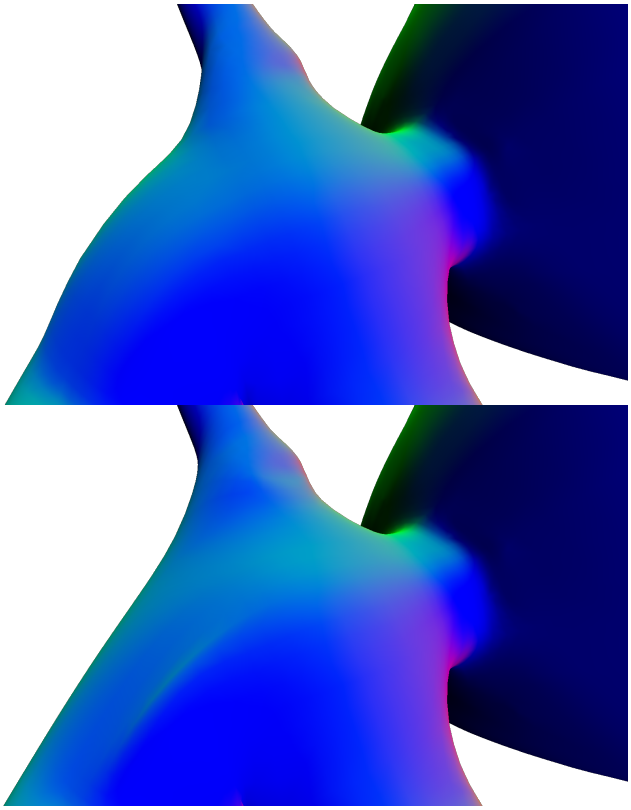

Figure 10: LBS without spring forces (top) and with (bottom). Note the development of features along the spine and in the shoulder.

around joints, which is reconfigurable during execution, unlike traditional skin weights. Despite the dependency on attachment quality, spring force skinning performs well in a variety of mesh configurations. However, manual culling of helper bones may be necessary for more complex rigs before computing attachment points. For rigs composed of separate meshes for body, clothes, hair, etc., it may be preferred to selectively use spring forces on a chosen subset of geometry.

Spring forces grants more control than traditional skinning, producing high-quality results and flexible shape control, provided the system parameters are suitable for the geometry. The default values for our timestep, convergence threshold, and spring coefficients help prevent the mesh from diverging or otherwise deforming in an undesirable manner. Further work on this technique may explore automatic parameter adjustment during animation. For example, the iteration count might be reduced for small deformations, or increased to accommodate larger anatomical changes.

There are opportunities to improve performance for the spring force solver. Presently, the system performs rigid skinning followed by the solver execution on every frame. This has the potential of introducing popping artifacts between frames, but the default parameters we provide make popping artifacts largely avoidable. We have experimented with structuring the deformer pipeline to use one frame's solution as the next frame's initial guess, a technique used with elastic implicit skinning for the same concern [47]. This can significantly reduce convergence time and improve performance, but at the cost of frame independence, which has consequences for batch rendering. Using a variation of STB instead of rigid skinning as our first stage may also improve convergence by supplying a smoother initial guess before the solver runs, but as with LBS and other smooth skinning methods, this can limit the solver's ability to control how the final shape appears. Further work is necessary to explore this option.

Using spring forces and scale vectors to drive mesh shape is the most appealing features of our technique. These enable shape control for skinning and surface deformations in the same framework, and the implementation on graphics hardware makes it interactive for the artist and appropriate for controlling animation in real-time. There are many opportunities to further exploit this, such as enabling inertia and momentum effects on the surface, accelerated collision detection between opposing surfaces, and interactions with external physical forces. In its present form, our method partially resolves colliding regions and enables some degree of secondary motion. In future work, we plan to develop more implicit behaviors between the solver's forces to create more deliberate secondary effects, emphasizing bulge and contraction on bone stretches and impact propagation for collision detection.

REFERENCES

[1] Nadine Abu Rumman and Marco Fratarcangeli. 2015. Position-based skinning for soft articulated characters. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 240–250.

[2] Baptiste Angles, Marco Tarini, Brian Wyvill, Loïc Barthe, and Andrea Tagliasacchi. 2017. Sketch-based implicit blending. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 181.

[3] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. 2008. Skeleton Extraction by Mesh Contraction. In *ACM SIGGRAPH 2008 Papers (SIGGRAPH '08)*. ACM, New York, NY, USA, Article 44, 10 pages. https://doi.org/10.1145/1399504.1360643

[4] Ilya Baran and Jovan Popović. 2007. Automatic Rigging and Animation of 3D Characters. *ACM Trans. Graph.* 26, 3, Article 72 (July 2007). https://doi.org/10.1145/1276377.1276467

[5] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. 2006. PriMo: Coupled Prisms for Intuitive Surface Modeling . In *Symposium on Geometry Processing*. 11–20.

[6] Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. 2007. Adaptive space deformations based on rigid cells. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 339–347.

[7] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. Interactive Skeleton-driven Dynamic Deformations. *ACM Trans. Graph.* 21, 3 (July 2002), 586–593. https://doi.org/10.1145/566654.566622

[8] James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popović, and David Salesin. 2003. A Sketching Interface for Articulated Figure Animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. 320–328. http://dl.acm.org/citation.cfm?id=846276.846322

[9] Olivier Dionne and Martin de Lasa. 2013. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 173–180.

[10] David A Field. 1988. Laplacian smoothing and Delaunay triangulations. *Communications in applied numerical methods* 4, 6 (1988), 709–712.

[11] Sven Forstmann, Jun Ohya, Artus Krohn-Grimberghe, and Ryan McDougall. 2007. Deformation Styles for Spline-based Skeletal Animation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '07)*. 141–150. http://dl.acm.org/citation.cfm?id=1272690.1272710

[12] C. Grimm and P. Joshi. 2012. Just DrawIt: a 3D sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. Eurographics Association, 121–130.

[13] Martin Guay, Marie-Paule Cani, and Rémi Ronfard. 2013. The Line of Action: An Intuitive Interface for Expressive Character Posing. *ACM Trans. Graph.* 32, 6, Article 205 (Nov. 2013), 8 pages. https://doi.org/10.1145/2508363.2508397

[14] Takeo Igarashi and John F. Hughes. 2001. A Suggestive Interface for 3D Drawing. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and*

Technology (UIST '01). 173–181. https://doi.org/10.1145/502348.502379
[15] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-time Deformation. *ACM Trans. Graph.* 30, 4, Article 78 (July 2011), 8 pages. https://doi.org/10.1145/2010324.1964973
[16] Alec Jacobson and Olga Sorkine. 2011. Stretchable and Twistable Bones for Skeletal Shape Deformation. *ACM Trans. Graph.* 30, 6, Article 165 (Dec. 2011), 8 pages. https://doi.org/10.1145/2070781.2024199
[17] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic Coordinates for Character Articulation. *ACM Trans. Graph.* 26, 3, Article 71 (July 2007). https://doi.org/10.1145/1276377.1276466
[18] Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics* 24, 3 (July 2005), 561–566.
[19] O.A. Karpenko and J.F. Hughes. 2006. SmoothSketch: 3D free-form shapes from complex sketches. In *ACM Transactions on Graphics (TOG)*, Vol. 25. ACM, 589–598.
[20] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with Dual Quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. 39–46. https://doi.org/10.1145/1230100.1230107
[21] Ladislav Kavan and Olga Sorkine. 2012. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 196.
[22] Y. Kho and M. Garland. 2005. Sketching mesh deformations. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. ACM, 147–154.
[23] Youngihn Kho and Michael Garland. 2005. Sketching mesh deformations. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. ACM, 147–154.
[24] YoungBeom Kim and JungHyun Han. 2014. Bulging-free Dual Quaternion Skinning. *Comput. Animat. Virtual Worlds* 25, 3-4 (May 2014), 323–331. https://doi.org/10.1002/cav.1604
[25] Martin Komaritzan and Mario Botsch. 2018. Projective skinning. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 12.
[26] T. Langer and H.P. Seidel. 2008. Higher order barycentric coordinates. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 459–466.
[27] Binh Huy Le and Jessica K. Hodgins. 2016. Real-time Skeletal Skinning with Optimized Centers of Rotation. *ACM Trans. Graph.* 35, 4, Article 37 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925959
[28] Y.J. Lee, C.L. Zitnick, and M.F. Cohen. 2011. ShadowDraw: real-time user guidance for freehand drawing. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 27.
[29] J.P. Lewis, M. Cordner, and N. Fong. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 165–172.
[30] Ron MacCracken and Kenneth I. Joy. 1996. Free-form Deformations with Lattices of Arbitrary Topology. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. 181–188. https://doi.org/10.1145/237170.237247
[31] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface '88*.
[32] Alex Mohr and Michael Gleicher. 2003. Building efficient, accurate character skins from examples. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 562–568.
[33] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (Feb. 2009), 85–103.
[34] A. Cengiz Öztireli, Ilya Baran, Tiberiu Popa, Boris Dalstein, Robert W. Sumner, and Markus Gross. 2013. Differential Blending for Expressive Sketch-based Posing. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. 155–164. https://doi.org/10.1145/2485895.2485916
[35] Junjun Pan, Lijuan Chen, Yuhan Yang, and Hong Qin. 2018. Automatic skinning and weight retargeting of articulated characters using extended position-based dynamics. *The Visual Computer* 34, 10 (2018), 1285–1297.
[36] Valentin Roussellet, Nadine Abu Rumman, Florian Canezin, Nicolas Mellado, Ladislav Kavan, and Loïc Barthe. 2018. Dynamic implicit muscles for character skinning. *Computers & Graphics* 77 (2018), 227–239.
[37] Nadine Abu Rumman and Marco Fratarcangeli. 2016. State of the art in skinning techniques for articulated deformable characters. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications: Volume 1: GRAPP*. SCITEPRESS-Science and Technology Publications, Lda, 200–212.
[38] Thomas W. Sederberg and Scott R. Parry. 1986. Free-form Deformation of Solid Geometric Models. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 151–160. https://doi.org/10.1145/15886.15903
[39] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. In *ACM Transactions on Graphics (TOG)*, Vol. 26. ACM, 81.
[40] Karan Singh and Eugene Fiume. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 405–414.
[41] Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing (SGP '07)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 109–116. http://dl.acm.org/citation.cfm?id=1281991.1282006
[42] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '04)*. 175–184. https://doi.org/10.1145/1057432.1057456
[43] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. 2005. Mesh-based inverse kinematics. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH '05)*. 488–495.
[44] Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion Doodles: An Interface for Sketching Character Motion. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 424–431. https://doi.org/10.1145/1015706.1015740
[45] Steve Tsang, Ravin Balakrishnan, Karan Singh, and Abhishek Ranjan. 2004. A Suggestive Interface for Image Guided 3D Sketching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. 591–598. https://doi.org/10.1145/985692.985767
[46] Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. 2013. Implicit skinning: real-time skin deformation with contact modeling. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 125.
[47] Rodolphe Vaillant, Gäel Guennebaud, Loïc Barthe, Brian Wyvill, and Marie-Paule Cani. 2014. Robust iso-surface tracking for interactive character skinning. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 189.
[48] Jane Wilhelms and Allen Van Gelder. 1997. Anatomically based modeling. In *SIGGRAPH*, Vol. 97. Citeseer, 173–180.
[49] Weiwei Xu, Jun Wang, KangKang Yin, Kun Zhou, Michiel Van De Panne, Falai Chen, and Baining Guo. 2009. Joint-aware manipulation of deformable models. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 35.
[50] Johannes Zimmermann, Andrew Nealen, and Marc Alexa. 2007. SilSketch: Automated Sketch-based Editing of Surface Meshes. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling (SBIM '07)*. 23–30. https://doi.org/10.1145/1384429.1384438