Implicit Skinning: Character Skin Deformation Guided by 3D Scalar Fields

by

Rodolphe Vaillant

In Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Implicit Skinning: Character Skin Deformation Guided by 3D Scalar Fields

by

Rodolphe Vaillant

Supervisory Committee

Pr. Loïc Barthe, Supervisor
(Université Paul Sabatier, Toulouse, France)

Pr. Brian Wyvill, Supervisor
(University of Victoria, Canada)

Pr. Mathias Paulin, Additional committee member
(Université Paul Sabatier, Toulouse, France)

A. Pr. Melanie Tory, Additional committee member
(University of Victoria, Canada)

Pr. Pierre Alliez, Outside committee member
(Inria Sophia-Antipolis, France)

A. Pr. Gaël Guennebaud, Outside committee member
(Inria Bordeaux, France)

External examiners
Pr. Mark Pauly,
(École Polytechnique Fédérale de Lausanne, Swiss)

A. Pr. John Lewis,
(Victoria University of Wellington, New Zealand)

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Cotutelle internationale avec "University of Victoria"

Présentée et soutenue par :
Rodolphe Vaillant

le jeudi 24 septembre 2015

Titre :

Implicit Skinning: Character Skin Deformation Guided by 3D Scalar Fields

---

École doctorale et discipline ou spécialité :
ED MITT : Image, Information, Hypermedia

Unité de recherche :
Institut de Recherche en Informatique de Toulouse - UMR 5505

Directeur/trice(s) de Thèse :

Pr. Loïc Barthe et Pr. Brian Wyvill

Jury :
Rapporteurs
Mark Pauly, Professor, École polytechnique fédérale de Lausanne
John Lewis, Associate Professor, Victoria University of Wellington
Examinateurs
Melanie Tory, Associate Professor, University of Victoria
Mathias Paulin, Professeur, Université Toulouse 3 Paul Sabatier
Gaël Guennebaud, Chargé de Recherches, Inria Bordeaux
Pierre Alliez, Directeur de Recherches, Inria Sophia-Antipolis

ABSTRACT

In character animation achieving realistic deformations of the skin is a challenging task. Geometric skinning techniques, such as smooth blending or dual-quaternions, are very popular for their high performance but fail to produce convincing deformations. They look too soft compared to human skin deformation at a rigid bone joint. In addition advanced effects such as skin contacts or bulges are not taken into account. Other methods make use of physical simulation or volume control to better capture the skin behavior, yet they cannot deliver real-time feedback. We developed a novel skinning framework called *implicit skinning*. Our method produces visually plausible deformations in real-time by handling realistic skin contacts and bulges between limbs. Implicit skinning exploits the ability of implicit surfaces to be robustly combined as well as their efficient collision detection. By approximating the mesh by a set of implicit surfaces, we are able to guide the deformation of a mesh character. we can combine the implicit surfaces in real-time, and use the final implicit surface to adjust the position of mesh vertices at each animation step. Since collision detection is very efficient using implicit surfaces we achieve skin contacts between limbs at interactive to real-time frame rates. In this thesis we present the complete implicit skinning framework, that is, the conversion of a mesh character to implicit surfaces, the composition operators and the mesh deformation algorithm on top of the implicit surface. Two deformation algorithms are studied: a fast history dependent algorithm which acts as a post process on top of dual-quaternions skinning and a slower yet more robust history dependent algorithm.

ABSTRACT (French)

Nous présentons une nouvelle approche pour l'animation de personnages de synthèse représentés par des maillages et un squelette. Nous concentrons nos efforts sur la correction des déformations non réalistes des articulations. En effet, les techniques d'animation standard présentent un aspect bien trop mou pour reproduire les déformations des articulations, ce qui rend ses approches peu crédibles. Nombre de systèmes d'animation en temps réel ont tenté de résoudre ce problème connu sans jamais y parvenir totalement, ou requièrent en effort considérable de la part de l'utilisateur. Nous introduisons un nouveau système d'animation appelé *implicit skinning*. Ce dernier est capable de produire en temps réel des déformations à l'aspect plus rigide. De plus, *implicit skinning* introduit des effets plus avancés comme la simulation du contact de la peau entre les membres du personnage, mais aussi des effets de gonflement de la peau et de gonflement musculaire. Notre proposition part de l'observation que les déformations des articulations sont facilement représentables par l'intermédiaire de surfaces implicites, et ce, avec une gestion des déformations rigides et des collisions. Nous tirons donc parti des avantages des deux représentations que sont les maillages et les surfaces implicites, en ajustant l'animation d'un maillage par une technique basée sur squelette sur sa représentation implicite. Nous évitons ainsi les difficultés de visualisation liées aux surfaces implicites tout en gardant des temps d'animation interactifs jusqu'au temps réel. Dans cette thèse nous explorons deux grandes familles d'algorithmes de déformation: un algorithme rapide dit "indépendant de l'historique" qui a pour avantage de s'intégrer facilement en tant que post-traitement de n'importe quel système d'animation géométrique; et enfin un algorithme plus lent mais plus stable dit "dépendant de l'historique".

Contents

ACKNOWLEDGEMENTS

I would like to thank:

Professors Brian Wyvill and Loïc Barthe, this goes without saying, but I would not be here without their support, and for that, I will owe them eternally! During those past few years of research I have experienced and learned far more than ever before. I have been introduced to so many interesting people and got in many stimulating events thanks to them. If it were not for them, I would never have been remotely interested to enrol a Ph.D program and I would definitely have missed this extraordinary opportunity. So, thank you. Thank you for supporting me in those exciting moments of making this research, but also, thank you for supporting me in the far more less thrilling moments – nobody likes paperwork but it has to be done...

Gaël Guennebaud, who basically behaved like a third supervisor and heavily contributed to my research. Your guidance has been tremendously valuable.

Other co-authors, who also significantly helped me in my research, I think in particular to Marie-Paule Cani and Damien Rhomer whose help were priceless.

The Lab, and all the people I have met. Olivier Gourmel for mentoring me, Anthony Pajot for helping me as well, although he really did not have to. Florian Canezin for having the patience to work with me. Mathias Paulin, David vanderhaeghe, Samir Torki, Andra Doran, Frédéric Claux and so many others who assisted me, here and there, any time it was necessary.

The Committee, for accepting to review this thesis on such short notice, and nonetheless were able to give me interesting feedbacks as well as numerous corrections.

My luck, because let's be honest, I never thought I was quick on the uptake, and even if I was we all need a lot of luck in order to succeed.

My family, my big brother, my partner always here for me.

Calvin: I'm being educated against my will! My rights are being trampled!
Hobbes: Is it a right to remain ignorant?
Calvin: I don't know, but I refuse to find out!
Calvin & Hobbes

Chapter 1

Introduction

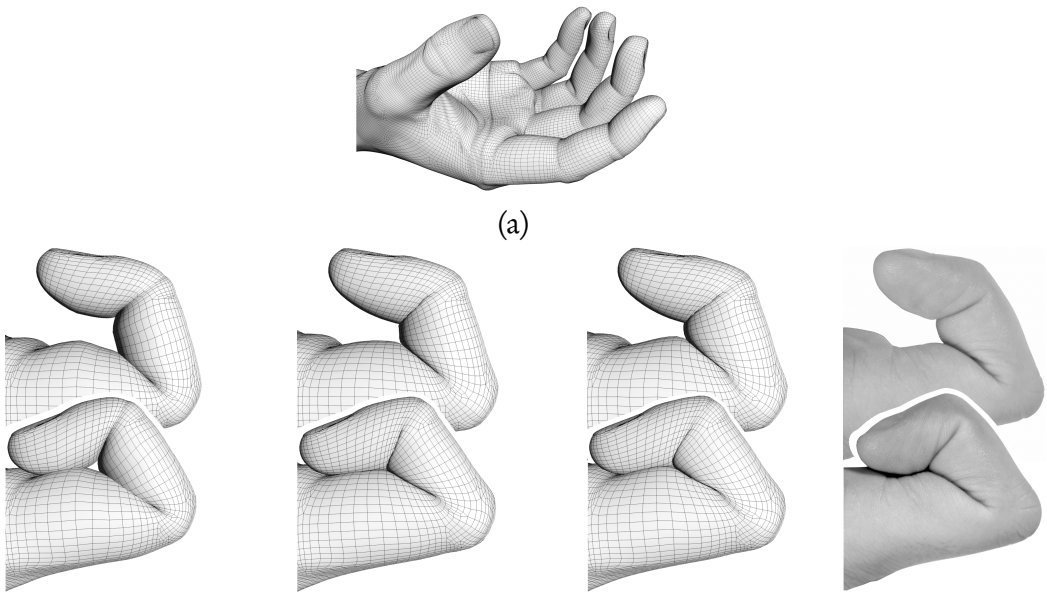
(a) (b) (c) (d)

Figure 1.1: Some examples of popular application fields for character animation such as (a) games, crowd simulation, (b,c) movies (d) dressing of virtual avatars. (Source: Hitman absolution I/O interactive, Frozen Walt Disney, The incredible Hulk Universal Picture, Marvelous designer CLO Virtual Fashion)

1.1 Context

In today's digital world, demands for tri-dimensional content is rapidly growing. The entertainment industry (movie, commercials etc.) need to design virtual worlds in which animated characters and creatures are ubiquitous. In this context, physical accuracy is rarely a requirement and the goal is to achieve visually plausible animations. Instead of a realistic physical simulation, the artist often prefers to have a swift and intuitive control over the animation and deformation of the model. This makes it easier to emphasis certain elements of their work, for instance to better underline a bicep inflation even at the expense of volume preservation. In addition, the user usually has to interact with the virtual world, this

means the simulation must run in real-time. When user interactions is not required, speed remains a concern since it impacts the ability of the artist to produce easily and rapidly character animations. Therefore, real-time feedback and automation of the character's skin deformation is always worthwhile.



(a)

(b) Linear blending (c) Our with contact (d) Our with bulge (e) Real finger

Figure 1.2: Deformations with our method: implicit skinning [VBG+13]. (a) Real-time animation of the index finger of a hand model composed of 31750 vertices. Two poses are shown in each column: (b) standard smooth skinning with linear blending at 95 frames per second (fps), (c) our method which compensates the loss of volume on top of joints and models contact in the fold at 37 fps, (d) implicit skinning with an additional bulge mimicking tissues inflation at 35 fps, and (e) a picture of a real bent finger.

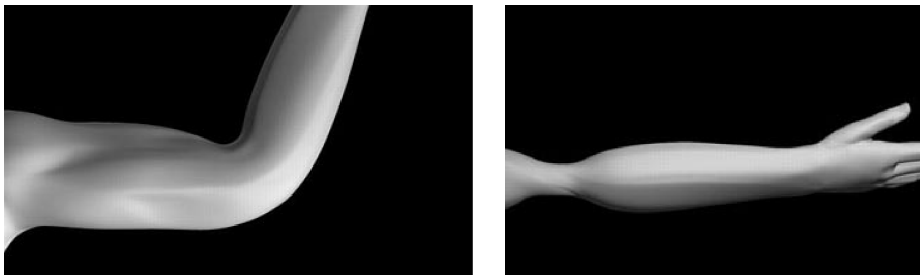
This research focuses on one of the many challenges when animating a virtual character: the production of plausible skin deformations at joints, this process is called *skinning*. We introduce a novel framework for character skinning called *implicit skinning*. The system performs from interactive to real-time frame rates. We enhance the look of traditional skinning algorithm by simulating skin folds and skin contacts, for instance inside the character's joint when a limb is bent (e.g. finger joints Figure 1.2), or between colliding body parts such as an arm against the torso. We also handle more advanced effects such as skin bulges at joints, muscles bulges or even skin elasticity (i.e. the ability for the skin to stretch and squash over the character).

Among the many [MTLT88, HYC+05, FOKgM07a, DSP06, YSZ06, JS11, JBK+12] ap-

proaches to deform mesh-based characters, *linear blend skinning* [MTLT88] (**LBS**) is a prevalent technique important to the research presented here. The technique consist in associating different parts of a mesh to a *skeleton* which drives the motion of the character. A skeleton is defined by a succession of segments (the *bones*) linked together by *joints*. The skeleton can be seen as a graph (usually a tree hierarchy), where bones are the arcs of the graph and joints its nodes. The skeleton is set in motion by rotating bones around their joint. With **LBS** the mesh is deformed according to *skinning weights* which associate each bone with a set of the mesh vertices (Figure 2.1). As detailed later (Chapter 2), skinning weights enable the production of smooth deformations of the mesh at joints. Note the deformation is always computed from an initial position of the skeleton (called the *rest pose* or *bind pose*). This type of algorithm is said to be *history independent*: it guarantees the same result for a given skeleton's pose, regardless of any other previous parameters in time.

Linear blend skinning is the De facto standard in the entertainment industry. Especially for applications such as video games, or stylized animated movies. In these areas the trade-off between visual realism and speed of user interactions, strongly leans towards speed. **LBS** requires less user interactions and is less computationally expensive than other more realistic methods. This explains its popularity in games, where it can animate hundreds of characters in real-time using modern workstations. But the technique suffers from the following well-known drawbacks: the deformation looks mushy, produces loss of volume at joints, and geometry can self-intersects. For instance, the animation of an elbow is more similar to a bent rubber pipe (Figure 1.3 (a)) than a true human joint. Human joints are made of bones and nearly incompressible soft tissues which result in a more rigid looking deformation. Figure 1.3 (b) further illustrates the loss of volume problem, we can see the twisting motion of the forearm collapses every mesh's vertices onto a single point. In addition to these problems, no collision detection is performed. This means self-intersections of the mesh will arise for any large movements of the bones. When geometry self-intersects subsequent simulations steps such as for clothing [BWK03] or anything that requires a clear definition of the inside and outside of the character, are more difficult to compute.

When more realism is needed, example-based methods [MG03] and shape interpolation schemes [LCF00, WSLG07] enable real-time animation, however they greatly increase the amount of necessary user input. For large budget projects, the movie industry often relies on shape interpolation methods since they allow photo-realistic deformations. In this type of framework, the user deforms the character with a standard skinning method, and then re-sculpt the character in each of the poses that are considered troublesome. This process gives a lot of freedom to the artist and the end result strongly reflects the artist's skill and



(a) (b)

Figure 1.3: (a) Linear blending results in a rubbery appearance of the elbow (due to loss of volume). (b) the well known candy wrapper artifact for screw motions. [LCF00]

perseverance. Muscle bulges or details as fine as veins dilatation or skin wrinkles can be reproduced but at the cost of a very long character remodeling.

Lastly, physical simulation can also produce appealing deformations [NTH01, TSIF05], however the amount of computation makes it only suitable for off-line rendering. In addition, muscles and rigid bones of realistic shapes need to be predefined from medical data or by the user. Intermediate methods such as elasticity skinning [MZS+11] don't need such data. This method enables the computation of skin squash and stretch by solving the underlying physical equations of elastic materials. Despite its robustness and visual quality, this type of approach still requires several seconds per animation frame.

1.2 Contributions

Our implicit skinning framework provides the artist with a skinning technique visually better than a traditional smooth skinning while keeping real-time performance and minimizing user input. To improve the visual quality we automatically produce self-penetration free deformations when skin folds, and preserve the aspect of a rigid bone near joints (Figure 1.2(c)). We are also able to generate subtle effects such as the skin bulging at joints (Figure 1.2(d)), muscle bulges and skin elasticity with simple parameters.

Technically, the novelty of implicit skinning lies in the use of implicit surfaces together with an animated mesh to achieve its skinning. Instead of relying on volume discretization and mesh-based collision, we use implicit surfaces to model the volume of the mesh and compute its deformation. The key idea is to approximate the individual mesh parts attached to each skeleton bone using implicit surfaces. By doing this we can exploit their ability to be combined using blending operators (such as union, gradient-based blending or gradient-based bulge-in-contact [GBC+13]). These operators will enable us to model plausible skin

deformations at joints during motion, with timings far superiors to traditional mesh-based approaches.

In this research we will limit our scope to skinning methods. Our state of the art analysis (Chapter 2) will not include other shape deformation methods [Sor06,SA07] which are best used to quickly change the pose of a mesh. Readers interested to delve even further into the realm of character skinning can refer to the corresponding Siggraph 2014 course [JDKL14]. Chapter 3 will lay the technical background on 3D scalar fields and implicit surfaces which is necessary to understand our contributions. We will describe two versions of the implicit skinning framework (Chapter 4): a history independent algorithm lying on top of standard geometric skinning allowing a straightforward implementation within standard animation pipeline, and a more robust history dependent version allowing more advanced effects such as skin elasticity.

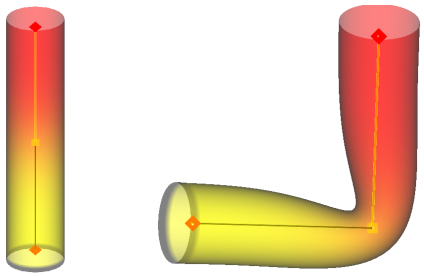
Chapter 2

State of the art

2.1 Geometric skinning

Geometric skinning refers to a class of skinning methods which rely on a coarse representation of the character's skeleton. Most of the time this skeleton is represented with a succession of line segments or in more advanced systems curves. The goal is to deform the mesh by only relying on the skeleton joint rotations. In other words, geometric skinning systems blindly deform the mesh according to a skeleton's pose. These approaches offer visually plausible results but mainly focus on extremely high performance. Geometric skinning is usually seen in games and virtual reality systems where *Linear blend skinning* [MTLT88] is by far the most popular method.

2.1.1 Linear blend skinning



(a) (b)

Figure 2.1: A cylinder colored according to the influence of the first bone (a.k.a skinning weights). Red is the maximal influence and yellow is no influence.

Although it has never been formally introduced in the literature, linear blending was

first demonstrated to compute the deformation of an animated hand [MTLT88]. Figure 2.1 gives an example of linear blending deformation with a cylinder. To achieve such deformations one needs to define the influence of each bone over the mesh (figure 2.1 (a)). When a vertex is influenced by only one bone it will exactly follow the transformation of this bone (this is the case of fully red vertices in figure 2.1). These influences can overlap, for instance vertices in orange are influenced by more than one bone and follow a blended transformation of related bones. Formally each bone is associated with a scalar function $w_i : \mathbb{R}^2 \to \mathbb{R}$ defined over the surface of the mesh. The values of the function are specified for each vertex and form a set of scalar weights for each bone. The sets w_i are called *skinning weights* and may overlap to produce a smooth deformation at a joint. Skinning weights usually range from $[0, 1]$ and are used to interpolate the bones' transformations. The formula to deform the mesh vertices is straightforward:

$$\bar{\mathbf{p}} = \sum_{i=1}^{n} w_i(\mathbf{p}) \mathbf{T}_i \mathbf{p} \tag{2.1}$$

Here $\mathbf{p}$ is a mesh's vertex in rest pose, $\bar{\mathbf{p}}$ a vertex after deformation. n defines the number of bones, w_i is a scalar weight associated to the i^{th} bone for each vertex $\mathbf{p}$ and T_i the transformation matrix of a bone from its rest pose. Usually the weights w_i are normalized to sum to one and matrices T_i represent *rigid transformations* (*i.e.* solely a translation and/or rotation).

While the technique produces smooth deformations at joints for very low computational cost, it has several well-studied drawbacks [MG03, KCvO08, LCF00, BWK03]:

- The loss of volume near a joint that bends. It results in a very rubbery looking skin deformation at this joint.
- Mesh self-intersection for large movements. Since there is no collision detection this cannot be avoided.

The first problem arises from the linear combination of the matrices ($\sum w_i T_i$). Even though the matrices T_i represent rigid transformations, this does not guarantee that their combination results in an another rigid transformation [Ale02]. It is not unusual that the linear combination of matrices returns a matrix with a scaling factor, hence the volume shrinkage.

A simplified example can help to understand this phenomenon. Consider a cylinder mesh and two bones defined along its length (*cf.* figure 2.2 (a)). The vertex $\mathbf{v}$ in the middle has the weights $w_1 = 0.5$ and $w_2 = 0.5$ for each bone. On the figures 2.2 (b) and (c) the

vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ are the results of the transformations T_1 and T_2 applied to $\mathbf{v}$. As defined by the **LBS** equation 2.4, the linear combination of $\mathbf{v}_1$ and $\mathbf{v}_2$ results in $\bar{\mathbf{v}} = 0.5\mathbf{v}_1 + 0.5\mathbf{v}_2$ which is very close to the joint.

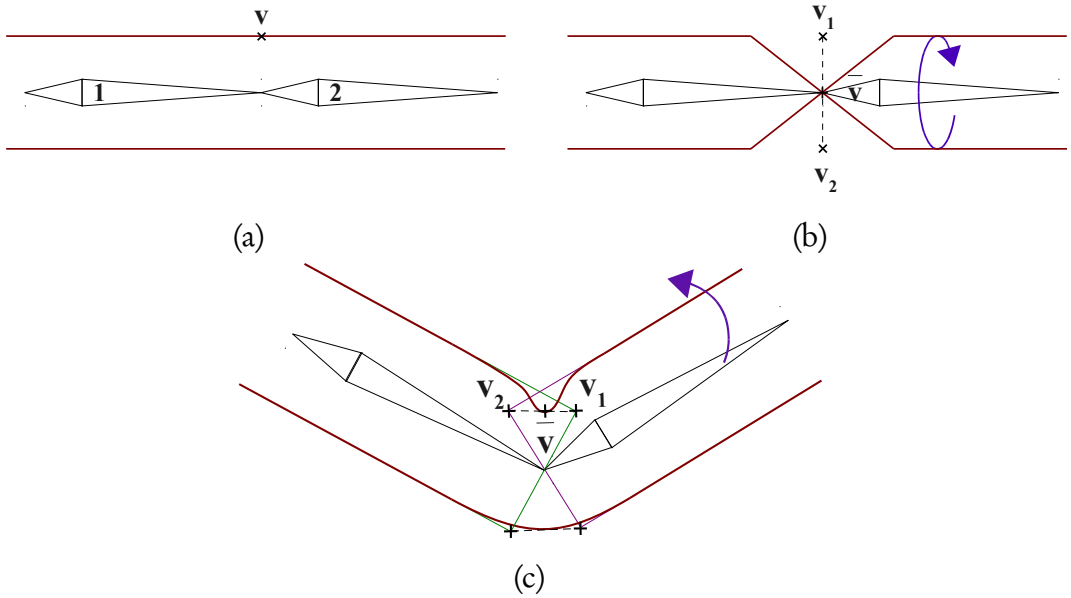


Figure 2.2: (a) We consider both bones numbered 1 and 2 where the bone 2 is set in motion in (b) and (c) while number 1 stays still. We can see the new position $\bar{\mathbf{v}}$ of $\mathbf{v}$ between the two bones. (b) When the bone 2 does a twisting movement its associated vertex $\mathbf{v}_2$ is in the opposite direction of $\mathbf{v}_1$. Their blending will collapse the mesh. (c) When bone 2 rotates towards bone 1 their blending result in a loss of volume, hence the soft aspect of the deformation.

This result is of course not desired when trying to animate characters made of rigid bone joints. Moreover, trying to correct this problem by tweaking the skinning weights can be extremely frustrating for the user. First, skinning weights don't have a clear physical interpretation, it is hard for the user to conjecture what weights will achieve the desired result, unless he is aware of the underlying computation. Secondly, the user cannot avoid the loss of volume by only adjusting the skinning weights w_i. Indeed, only a limited set of poses can be reached by changing the weights. In our example the deformed vertex $\bar{\mathbf{v}}$ can only travel along the line formed by $\mathbf{v}_1$ and $\mathbf{v}_2$. With a limited number of bones the linear system cannot express every position of the ambient space. To increase the pose space the user can resort to adding more joints: a tedious and highly unintuitive process.

The second problem of self-intersection is demonstrated in figure 2.3. Such a defect will make subsequent simulation harder to produce. For instance a clothing algorithm

will have to take this artifact into account which will make it more computationally expensive [BWK03]. No collision detection ultimately means that skin contacts between the different limbs and the resulting bulges are not simulated.

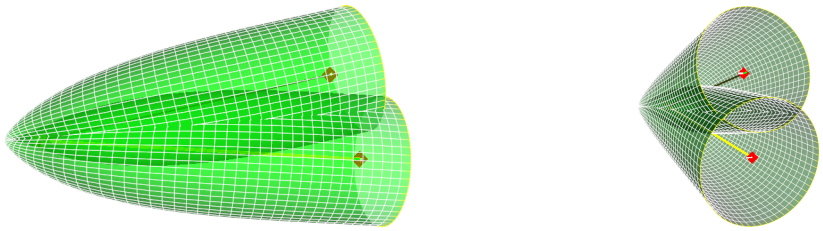


Figure 2.3: Large bending of a cylinder mesh produce a self-intersection.

Despite all these drawbacks linear blend skinning is still very popular. It is one of the approach which requires the least user input, especially with the advent of automatic skinning weights techniques [BP07, JBPS11]. In addition the method is very straightforward to implement which explains its integration into numerous 3D engines and animation packages. Above all it is extremely fast since the operation shown in equation 2.4 can be parallelized per vertex. Implementation on graphics card processors are very common using shader programs. Hundreds of characters can be animated in real-time this way on modern workstations.

2.1.1.1 Lighting

When deforming the geometry, we must also consider how to update its normals. This computation is important as it affects the accuracy of the lighting, and therefore the final aspect of the model. A mathematically straightforward way to do so, is to interpret the **LBS** equation 2.4 as a deformation map $\phi(x,y,z) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. From there, we can transform the normals using the 3×3 Jacobian matrix $\mathbf{J}(\phi(x,y,z))$:

$$\bar{\mathbf{n}} = \mathbf{J}_i^{-T}\mathbf{n} \tag{2.2}$$

The exact formulation for **LBS** is reported by Merry & al [MMG06b]. Unfortunately this approach is computationally expensive and prone to numerical instabilities [TPSH14], therefore a very popular practice is to approximate $\mathbf{J}_i(\phi)$ by $\sum_{i=1}^{n} w_i \mathbf{T}_i$. While inexpensive and straightforward to implement, this approximation makes the assumption that the deformation is locally rigid around the i^{th} vertex. This is rarely the case and can result in lighting artefacts. To mitigate this, one can use a more accurate approximation taking into

account the skinning weights variations and therefore the non-rigid deformation of the surface [TPSH14]

As a side note, if transforming normals is desirable in high performance applications, it can be tedious to formulate a closed form solution for complex skinning algorithms. In addition, pipelines with many deformation algorithms will be harder to maintain and transforming the normals at each stage may not even be the most efficient computation. In this case, we can compute normals directly from the geometry once every deformation is applied, it also has the advantage to work with any deformation.

2.1.2 Alternative blends

Within geometric skinning techniques we can distinguish a class of methods that tries to improve linear blend skinning by seeking alternative formulations to blend the bone transformations (cf. equation 2.4). As explained earlier linear blending can introduce a scaling factor. If the result of the blending were to guarantee the production of rigid transformations, then the loss of volume would be reduced greatly. This is the idea put forward by several researches [KCvO08, Ale02, Kv05]. Instead of doing a linear interpolation between the various positions of the vertices, we can interpolate their angle of rotation.

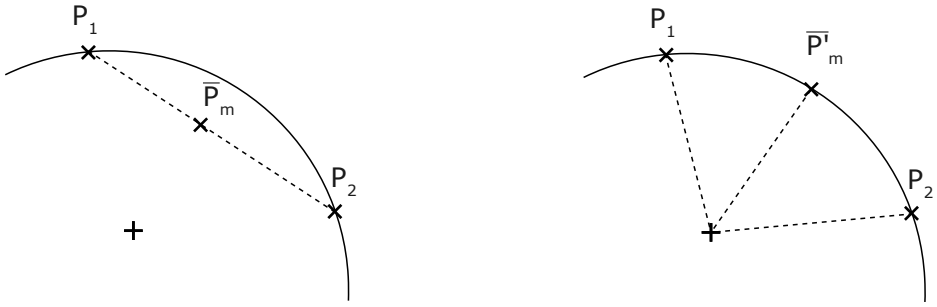


Figure 2.4: Left a linear interpolation produce a vertex $\bar{\mathbf{p}}_m$ not lying on the arc circle which goes through $\mathbf{p_1}$ and $\mathbf{p_2}$. Right the problem is solved by interpolating the angles.

Among alternative blends approache dual quaternion skinning is the fastest method with the lowest amount of artefacts we know to this date. Dual quaternion skinning consists of expressing the bone transformations using *dual quaternions* [KCvO08] instead of matrices. The advantage being that blending of dual quaternions is guaranteed to result in a rigid transformation. We will detail dual quaternions in the next section.

Other methods rely on slightly more complex skinning equations, for instance it is possible to treat the influence of scale, twist and bend motions differently [JS11]. Rigid skele-

ton can also be enhanced using curved bones to produce smoother deformations [FOKgM07b, YSZ06]

2.1.2.1 Dual quaternions skinning

The *dual quaternions skinning* [KCvO08] (**DQS**) speed is in the same order of magnitude as **LBS** and is just slightly slower. In this method, only the way transformations are expressed is changed and matrices become dual quaternions, this means its integration into linear blend skinning pipelines is straightforward. For these reasons dual quaternions skinning tends to become as popular as **LBS**. For instance we can find it in professional packages such as Blender or Maya. **DQS** approach is very similar to **LBS** and also relies on skinning weights, however, the blend $\sum_{i=1}^{n} w_i(\mathbf{p})\,\mathbf{T}_i$ is redefined to work with dual quaternions:

$$\bar{\dot{\mathbf{q}}} = \frac{\sum_{i=1}^{n} w_i(\mathbf{p})\,\dot{\mathbf{q}}_i}{\left\| \sum_{i=1}^{n} w_i(\mathbf{p})\,\dot{\mathbf{q}}_i \right\|} \tag{2.3}$$

Instead of blending matrices we blend dual quaternions. Each dual quaternion $\dot{\mathbf{q}}_i$ represents a bone transformation $\mathbf{T}_i$ without scale and shear. The $\dot{\mathbf{q}}_i$ are averaged using the influence weights w_i, the sum is then normalized. Details to perform dual quaternion additions, multiplications etc. can be found in Kavan & al [KCvO08]. The final result is a dual quaternion $\bar{\dot{\mathbf{q}}}$ expressing the rigid transformation to be applied to the corresponding vertex:

$$\bar{\mathbf{p}} = \text{transform}\left(\bar{\dot{\mathbf{q}}}, \mathbf{p} \right) \tag{2.4}$$

Here transform is a simple procedure extracting rotation and translation from $\bar{\dot{\mathbf{q}}}$ and applying it to $\mathbf{p}$. This procedure is very similar to the one used with standard quaternions except for the translation handling.

To be integrated in a linear blend skinning pipeline each transformation matrix $\mathbf{T}_i$ needs to be converted to its dual quaternion $\dot{\mathbf{q}}_i$ counterpart and the procedure handling blending re-written to use dual quaternions blending instead. This method gives encouraging results, it has a runtime speed close to **LBS** and solves its biggest problem: the loss of volume (*c.f.* figure 2.5).

Since dual quaternions skinning is quite similar to **LBS**, it shares **LBS**'s drawbacks as well. For instance, the untuitive skinning weights editing or the soft deformation which, although better, still looks too flexible to mimic semi-rigid bone joints. Finally no other advanced effects is introduced such as skin contacts or muscle bulge. Dual quaternion skin-

(a) (b) (c) (d)

Figure 2.5: (a) and (c) linear blending. (b) and (d) dual quaternions. Source: [KCvO08]

ning also introduce its own artefacts and limitation:

- The skin flips when a joint's rotation exceeds an angle of 180°.
- The skin exhibit a bulge around joints.
- Handling bone scale is not straightforward

We can observe a flip of the mesh vertices configuration when a rotation exceed an angle of 180°. If a bone goes from 179°to 180°, the position of the vertices will change dramatically. This is because the interpolation between multiple rotations will choose the shortest rotation path and the number of turns are not taken into account. As a result the animation will appear to be discontinuous around these angles. Fortunately, such scenario is pretty unfrequent when animating characters and creatures, a rotation of more than 180°is rarely considered to be appropriate or natural.

Dual quaternion skinning also exhibits a bulge around joints. Young Beom and Jung Hyun [KH14] expose this phenomenon by showing the path vertices follow for different skinning weights:

Figure 2.6 shows the case of only two bones influencing the skin. We observe a vertex is interpolated around a sphere whose center is the joint position and its radius the distance between the vertex and the joint position. In this configuration, the bulge artefact is inevitable as soon as a vertex gets farther from the joint. The user can reduce the bulge artefact by reducing the influence area very close to the joint. The less widespread the skinning weights are over the mesh, the less prominent the bulge will be (Figure 2.7).

It is also possible to completely eliminate the bulge artefact using a simple re-projection as post-process to **DQS** [KH14]. The procedure only relies on geometric considerations between the vertex position and bone or joint positions. For each vertex in the rest pose,

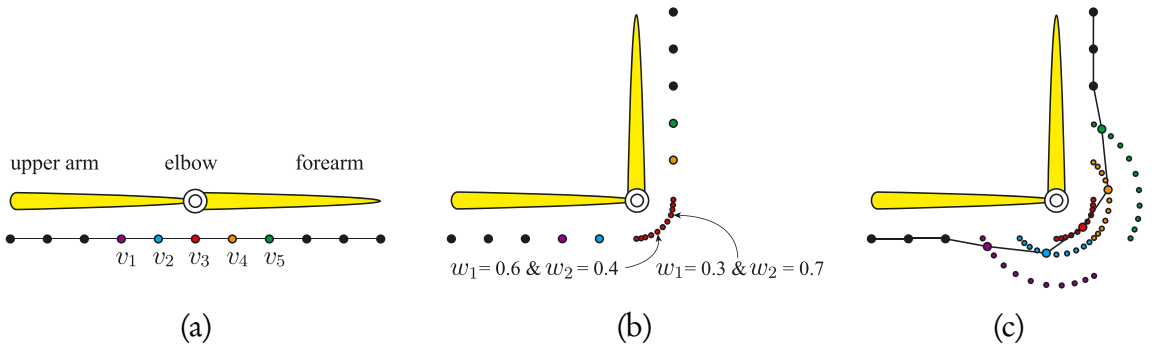


Figure 2.6: (a) rest pose. (b) If we change the skinning weights of v_3, we see its various instances are located on the surface of the sphere centered around the joint. (c) Vertices farther from the joint lies on a sphere with a larger radius. Source: [KH14]

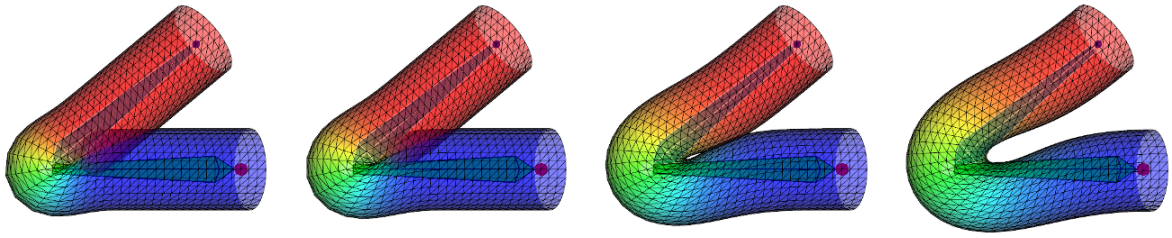


Figure 2.7: Impact of the skinning weights on **DQS** deformation. From left to right the skinning weights overlaps more and more. The bigger the overlapping region is, the more prominent the bulge artefact is.

we can store its distance from the nearest bone or joint. While the mesh is animated, every vertex not located in the fold of the joint is re-projected to its original distance. The runtime procedure is fairly inexpensive as it only involves a few operations per vertex: a vertex transformation, a bone to vertex distance computation and a couple arithmetic operations. Results are shown figure 2.8

Finally, because dual quaternion transformations are restricted to rigid transformations, handling scale is not straightforward. The authors propose to circumvent this problem by decomposing the skinning in two passes: one for the rigid transformations, another for the scale. Unfortunately, Gene et al. showed unwanted deformations may arise with this approach, especially when different scale factors are interacting with each other in the bone hierarchy [LLS+13]. They propose a more robust algorithm by taking into account the relationship between joints, this result in the concatenation of the joint transformations for the whole skeleton hierarchy. This fixes artefacts related to scale, and adds a small computational overhead compared to **LBS**.

To summarize dual quaternion skinning is slightly more complex than **LBS** in terms

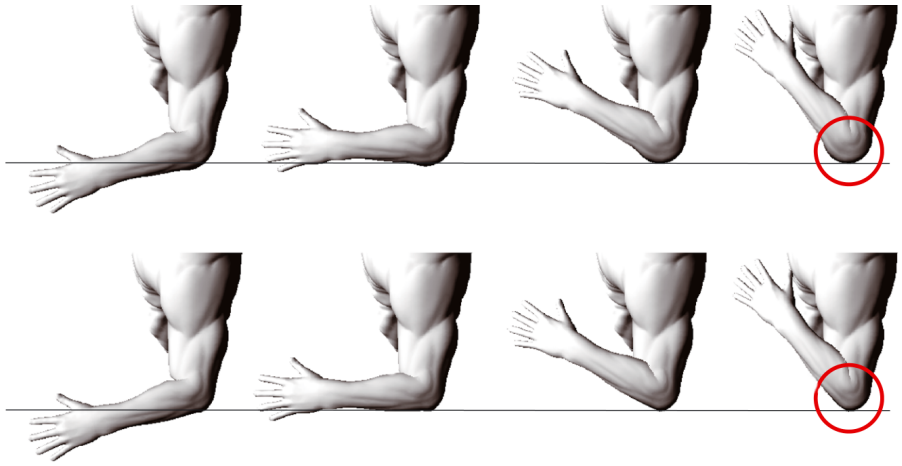


Figure 2.8: (Top row) Dual quaternion skinning exhibits the bulge artefact. (Bottom row) Young Beom and Jung Hyun [KH14] method correct the bulge by re-projecting the prominent vertices. Source: [KH14].

of implementation and produces its own artefacts. Despite those inconveniences, **DQS** is often considered to be a very good alternative over **LBS** since its artefacts are less troublesome or can be avoided using reasonable additional procedures. In addition **DQS** runtime performance is very similar to **LBS**.

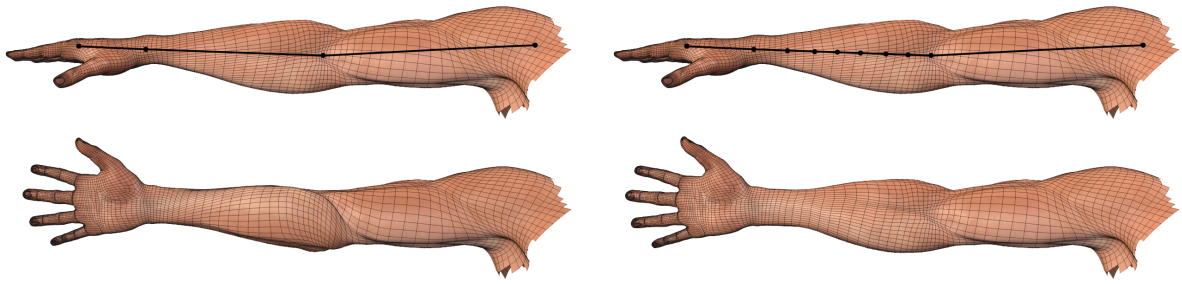
2.1.2.2 Twisting and stretching

Dual quaternion skinning solves the mesh shrinkage when we twist bones, and twisting does not exhibit the bulge artefact which only arises when the joints are bent. But there are other issues when twisting a bone that are not directly addressed by **DQS**.

By design, **DQS** relies on a single set of skinning weights per bone. Usually the sets overlap around joints, this enables one to localize the smooth deformation in between limbs. While it is desirable to have such localized deformation for bending motions, this is not always the case for twisting motions.

Figure 2.9 (a) illustrates the problem, the elbow is twisted with a single joint which results in an unnatural deformation. Indeed, when the elbow twists, one usually expects the forearm to progressively twist along its entire length. A common workaround is to divide the bone into a chain of bones (figure 2.9 (b)). Many set of skinning weights will overlap along the chain, this enables a large twist to be broken down into small increments and to distribute the deformation.

More bones means more sets of skinning weights, which also means more areas where those sets overlap. Ultimately, this will slow down the computation. Numerous vertices



(a) (b)

Figure 2.9: Deformation with dual quaternion skinning, (a) arm twist at a 45°angle using a single joint. (b) the same twist looks more natural if the rotation is distributed along a chain of bones.

will be influenced by more than two bones, this increases the number of dual quaternions blend per vertex. In addition, too many skinning weight sets may be tedious to handle by the user, especially if their editing is required. Finally, more bones makes it harder to animate the skeleton, and more procedures to automate the skeleton's kinematic will be needed to ease this burden.

Despite these issues, chain of bones are popular since this trick does not require any changes of the underlying skinning algorithm. It is not unusual to see softwares provide tools for automatic bone subdivision. Yet, one problem remains unsolved, which is stretching bones without deforming the geometry beyond the end points. Figure 2.10 demonstrates this issue:

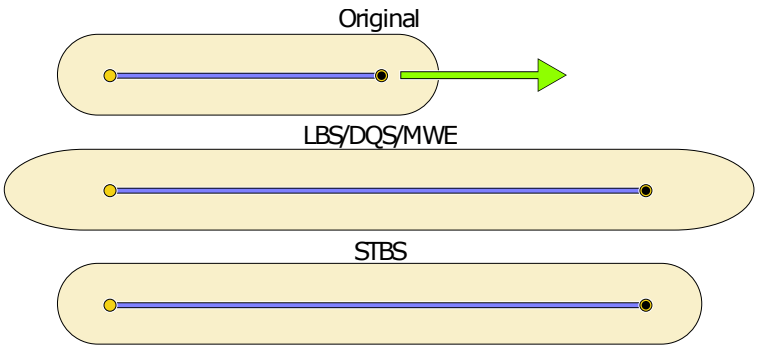


Figure 2.10: A cigar shape deformed with a single bone. With standard geometric skinning such as **LBS** or **DQS** there is a single skinning weight set (here equal to 1 everywhere) which affects the whole shape. As a result the tips of the cigar are deformed past the end points. The stretchable and twist-able skinning approach (**STBS** [JS11]) avoids this by introducing additional skinning weights sets for each end points. Source: [JS11]

Although the figure presents a simple case, the same artefacts are observable for any joint. As stated earlier **LBS** or **DQS** use a single skinning weight set per bone. These weights are usually greater than zero well beyond the end points of the bone, this allows the sets to overlap at joints and produce the smooth deformation. As a result, any scale factor will be applied beyond the end point of the bones.

Limitations related to bone twists and bends are found in every geometric skinning techniques relying on a single set of skinning weights per bone. An approach called stretchable and twist-able bones (**STBS** [JS11]) suggests to use two sets of skinning weights per bone w_i and e_i. The first set w_i is called bone weights. It is the usual skinning weights, mostly constant along the bone and overlapping around joints (Figure 2.1). The second set e_i is called endpoint weights. This time, the skinning weights overlap along the bone and are constant beyond the endpoints (Figure 2.11).

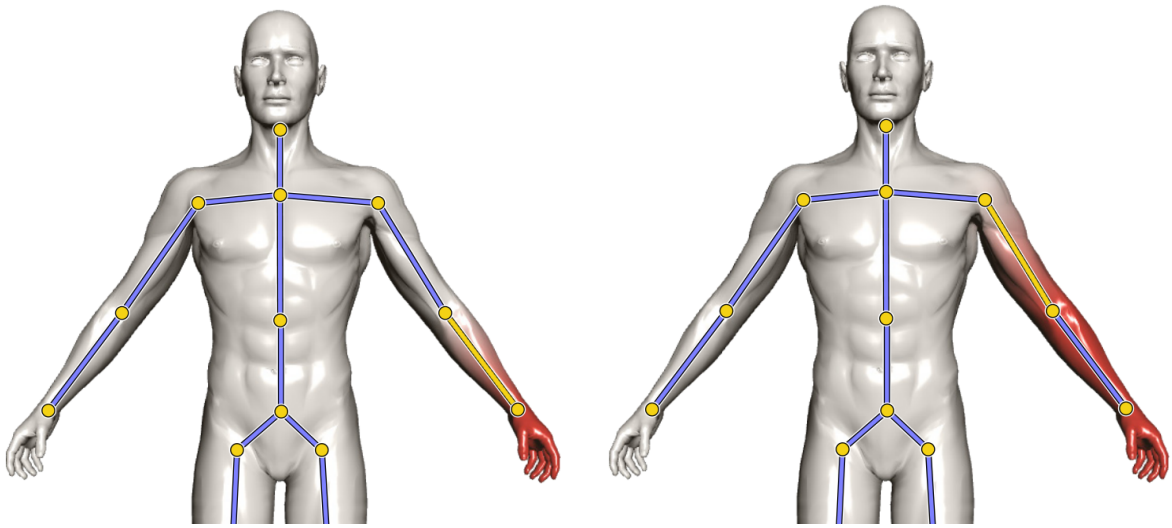


Figure 2.11: Display the endpoints skinning weights e_i used in **STBS**. Red means 1 and white 0. Notice how the weights are constant beyond the endpoints but varies along the bones' length (arm and forearm). Source: [JS11]

One can decompose the skinning equation 2.4 to express stretch and twist separately from the bending transformation, this makes it possible to associate each type of transformation to different skinning weight set. Here we associate e_i to twist and stretch transformations while bending relies on w_i. This enables twist or stretch to be properly distributed along the bone's length. Indeed e_i only varies along the bone's length enabling to easily interpolate user-defined twist and stretch parameters.

Although **STBS** fixes twist and stretch artefacts it come at the price of an additional set

of skinning weights. When automatically generated [BP07, JBPS11] this is not a big issue but if fine tuning is required this makes the skinning interface slightly more complex.

2.1.3 Curved skeletons

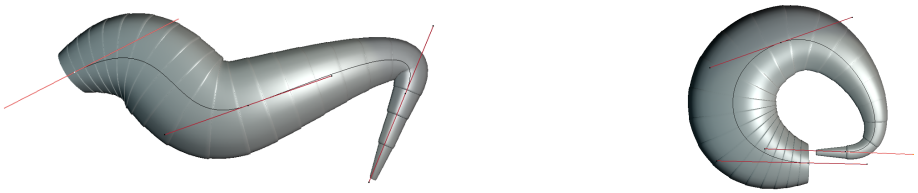


Figure 2.12: Tentacle object deformed using the software Blender. A Bezier curve with only three control points produces interesting deformations impossible to achieve with geometric skinning and the same amount of joints.

Sometimes rigid skeletons are not suitable to animate and deform soft and elongated limbs such as a cat's tail or an octopus' tentacle. Here again, we can make use of chains of bones to obtain smooth deformations. But as stated earlier, this is not ideal: chains of bones can be difficult to animate, the deformation quality depends on the number of bones and more bones increase the computation cost.

A good alternative is to use parametric curves $\mathbf{c}_i : \mathbb{R} \to \mathbb{R}^3$ such as splines [FOKgM07b, YSZ06] or Bézier curves (Figure 2.13), they offer a good control of the object's shape and curvature using a few control points. In addition curve skinning is extremely efficient and is just a bit slower than **LBS** by a few fps, provided that the number of control points are restricted and the curve discretized before its evaluation.

With tentacle like objects the goal is to produce a smooth deformation, therefore smooth and continuous curves seem to be more appropriate than chains of bones. While it may seem unrelated to the deformation of human looking joints, curves are interesting tools for body parts such as the spine, neck or forearms. Besides, a lot of skinning algorithms are also used for creatures where tails and tentacles are not uncommon.

Technically, to deform an object with curves, the first step is to bind the curves $\mathbf{c}_i$ to the mesh. Similar to rigid skeletons, a curve needs to associate to each vertex in rest pose a scalar weight. These weights represent the curve's parameter $t(\mathbf{p}) \in \mathbb{R}$. Given the parameter t one can compute the Frenet frame $\mathbf{F}_i \in \mathbb{R}^{4\times4}$ at the point $\mathbf{c}_i(t)$ and express a vertex position $\mathbf{p}$ relative to $\mathbf{F}_i$ which makes it easy to get the deformed vertex using the animated Frenet frame $\bar{\mathbf{F}}_i$:

$$\bar{\mathbf{p}} = \sum_{i=1}^{n} w_i(\mathbf{p}) \quad \bar{\mathbf{F}}_i(t(\mathbf{p})) \quad \mathbf{F}_i^{-1}(t(\mathbf{p})) \quad \mathbf{p} \tag{2.5}$$

Here we can handle multiple curves by doing a linear blending of the deformed vertex position for each curve. This blending relies on skinning weights $w_i(\mathbf{p})$ similar to the ones used for rigid skeletons. It is also possible to blend the deformed positions of a **LBS** or **DQS** along with vertices deformed by some curves.

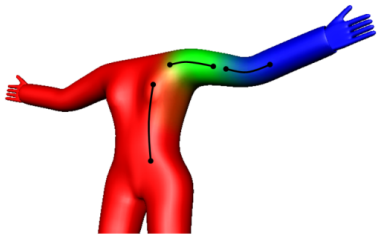
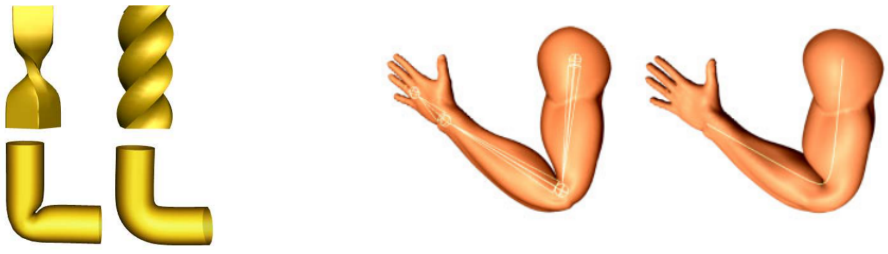


Figure 2.13: Associated weights to blend the deformation of each spline. Each spline is associated to a colour which is averaged according to the vertex skinning weights. Source: [FOKgM07b]

Skinning weights for curves can make use of automatic methods used for rigid skeletons such as the heat diffusion approach [BP07]. A more straightforward way to compute those weights is probably to choose the closest curve point $\mathbf{c}_i(t)$ from the vertex position $\mathbf{p}$ and associate the parameter t to this vertex. The closest distance can be pre-computed numerically and gives satisfying results for objects with cylindrical silhouettes has shown by Forstmann & al [FOKgM07b]. Twist deformations can be driven by the t parameter to interpolate twisting angles defined at the end points of the curve.



(a) (b)

Figure 2.14: (a) Twist and bend using a single spline and three control points, source: [FOKgM07b]. (b) Elbow bend with **LBS** on the left and a three control points spline on the right, source: [YSZ06].

Contrary to a rigid bone with a pivot point aligned with joints, splines are placed along the joint, this allows very smooth deformations (Figure 2.14). While results are interesting it produces unconvincing deformations when a human limb is bent. On the other hand twist deformations can be used for limbs like the forearms.

It is worth noting when only the twist is exploited the approach is identical to the **STBS** method. In this case the **STBS**' weights e_i are equivalent to the curve parameter t. In the end curved skeleton show their true strength when used for spines or organic appendices.

2.2 Skinning weights

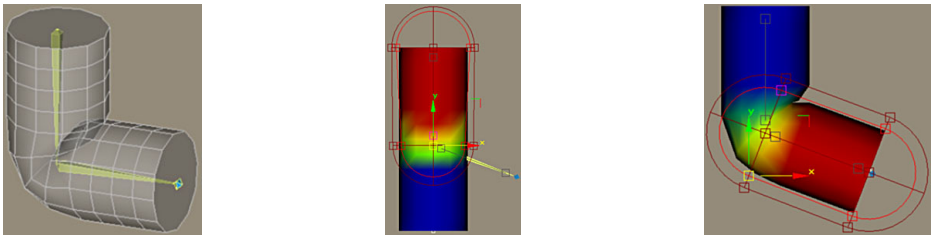


Figure 2.15: Semi-automatic skinning weights using envelopes. Each bone defines its influence according to its envelope which the user manually adjust. At the intersection of several envelopes the influence will be mixed.

The skinning methods previously discussed rely on skinning weights, their properties directly impacts the final deformation, thus, it is crucial to understand skinning weights to obtain good skinning results.

Artist used to manually edit skinning weights with the help of tools such as: envelopes (figure 2.15), weight painting or even editing weight values vertex by vertex!

In addition to being extremely tedious, it is very difficult for the artist to obtain the desired shape. As stated earlier, given a skeleton pose, a geometric skinning cannot produce every possible shape. The artist can lose a considerable amount of time tweaking skinning weights in some hope to model a shape, which is in fact unreachable using the current geometric skinning. To add further confusion, skinning weights have no clear physical meaning. The only insight the user has is that overlapping weights define soft areas.

It is possible to directly edit the shape of the model and infer the skinning weights [MTG03]. Unfortunately while the artist corrects a certain pose the system can break other poses, this makes the approach impractical.

Recent work showed how to compute skinning weights automatically [WSLG07, BP07, JBPS11, DdL14, JWS12]. This considerably speeds up the work of the artist. Automatic skinning weights produce a first draft which the user can tune up afterwards if needed.

In order to produce pleasing deformations automatic skinning weights must respect certain properties. Perhaps the most obvious is the smoothness of the weight functions $w_i(\mathbf{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ since C^0 function will result in C^0 deformation, C^1 in C^1 deformations etc. In addition, the way w_i decays as we go away from the bone also plays an important role. In the previous section 2.1.2.1 and figure 2.7 we show different skinning weights with fast to slow decay (left to right).

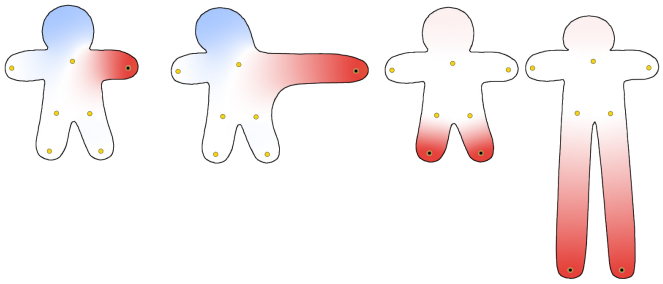


Figure 2.16: **LBS** applied to a 2D character, instead of bones the transformation matrices $\mathbf{T}_i$ are manipulated through point handles. When negative skinning weights are defined (in blue), the head moves in the opposite direction to the prescribed translation that stretches the arm. When legs are stretched the head is compressed since the weights are not null (local maximum). source: [JBPS11].

Previous work highlighted other important properties to produce well behaved skinning deformation [JBPS11, JWS12].

Non-negativity: Negative weights are not intuitive since regions of the mesh with negative weights move in the opposite direction to the prescribed transformation (Fig. 2.16).

Locality and sparsity: Each handle (bones or points) should only influence locally the shape of the model (Figure 2.18).

No local maxima: Weight functions w_i should attain their global maximum (i.e., value of 1) on the handle and should have no other local maxima. In other words, w_i must decay monotonically to ensure no undesired influence happen far away from the handle (Figure 2.16)

Shape-awareness: Informally, shape-awareness means the correspondence between the bones and the influenced areas of the mesh is intuitive. For instance, the arm's bone should influence the arm geometry but not the nearby torso geometry. This means skinning weights w_i should decay according to the geodesic distance inside the volume of the model.

(Figure 2.17)

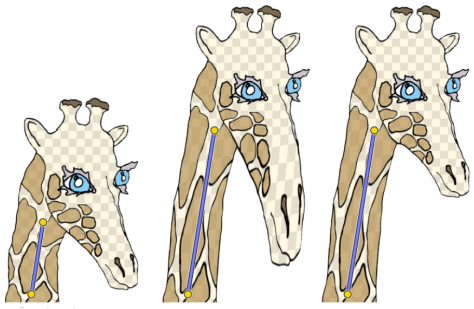
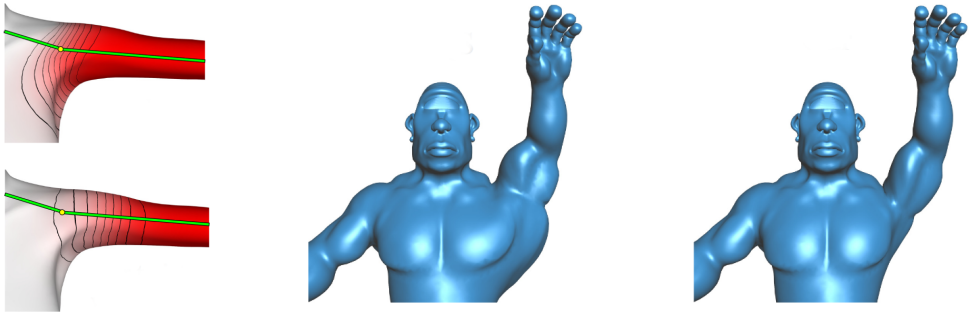


Figure 2.17: Left mesh in rest pose, middle mesh stretched with **LBS** and skinning weights agnostic to the shape (Euclidean distance), finally right mesh deformed wit shape-aware skinning weights. source: [JS11].

While automatic skinning weights produce very nice results it can be difficult to compute them on arbitrary geometries. Non-watertight meshes and self-intersections are not uncommon, handling such cases ease the work of the animator. Olivier Dionne et al. use a sparse voxelisation of the mesh models [DdL14]. This enables computing for each bone an approximate geodesic distance inside the volume of the mesh. The distance can be remapped to produce the desired skinning weights. This gives the opportunity to the user to edit the speed of decay of w_i manually. For instance one can wish to further localize the skinning weights in the shoulder area (Figure 2.17). This area is known to be especially difficult to handle, even with automatic skinning weights respecting the above properties. To give the user the opportunity to reduce the influence of a joint with a single parameter is a great advantage.

Automatic skinning weights now enables geometric skinning algorithms to fully reach their potential. They produce animations free of major artefacts without any user editing. The remaining lack of realism is directly linked to the geometric skinning itself and the number of bones used. When it comes to advanced automatic skinning weights, the major issue that remains is the computation time. For instance, weights generated with sparse voxelisation [DdL14] take a few seconds to compute. This prevents to interactively move the joints, this is detrimental to design real-time user interface allowing skinning weights to be fine tuned. With the increasing complexity of human or creature characters being able to set rigid parts, soft parts and other parameters manually is higly desirable.



(a) (b) (c)

Figure 2.18: How skinning weights locality impacts **DQS** deformations. When the influence of skinning weights is too widespread ((a) top row) it results in an unrealistic bulge (b). One must reduce the skinning weight influence around the shoulder ((a) bottom row) to obtain a more realistic deformation (c). Source: [KS12].

2.3 Example-based approaches

Example-based techniques [LCF00, KJP02, WSLG07, WP02, MMG06a, MG03, KMD+07] are ideal for application that demands:

- A high level of realism.
- A lot of freedom on the user side.
- Interactive performance.

The method aims to adjust the deformation automatically in order to match a set of example poses. While animating, examples are interpolated to produce realistic transitions for poses that are not in the set. The examples can be produced manually by the user, this consists of sculpting the character pose into the desired shape. A long and tedious process, which however enables the user to control the exact shape of the character. Alternatively the examples can be computed by an off-line system automatically (e.g muscle simulation). In this case, heavy animation systems can be used as input of an example-based method enabling to speed up the animation and give an interactive preview of the deformation.

We will distinguish two kind of approaches: those that adjust skinning weights to match the set of examples, and those that work directly on the vertex positions. To summarize, the first approach will often use some variations of linear blending and increase its degrees of freedom while keeping real-time performance. The second will store displacement vectors for each pose and interpolate them while animating.

The biggest advantage of such methods is realism. If examples are produced manually, the user is only limited by his skills and time. Subtle effects like muscle bulges or even tendon tension can be added. Once a sufficient number of poses is added the system can produce deformations even for poses far from the examples. The drawback is it can take weeks to setup a single character.

2.3.0.1 Pose space interpolation

Pose space interpolation [LCF00] is a well known system often used in medium or big animation companies. It is used in realistic scenes as in the short movie "Animatrix - Osiris last flight" with close-ups on arms, hands or shoulders. It is also used for more expressive animations in Walt Disney's "Bolt" to give life to Rhino the hamster.

The technique usually relies on a geometric skinning method. First the user animates the character (using linear blend skinning for instance) and then refine the shape at some

key position by sculpting the mesh. The corrected shape is stored as a set of displacement vectors bound to each pose of the skeleton. Once all the poses are defined, the system can interpolate the displacement vectors given the skeleton pose. This is a scattered data interpolation problem. The pose space deformation introduced by Lewis et al. [LCF00] use *radial basis functions* (RBFs) to solve it.

RBFs are basis functions $\phi : \mathbb{R}^n \to \mathbb{R}$ whose value depends on the distance from the evaluated point to the RBF's node center $\mathbf{c}$. A Gaussian RBF can be written:

$$\phi(||\mathbf{x}-\mathbf{c}||) = \exp \frac{-||\mathbf{x}-\mathbf{c}||^2}{2\sigma^2}$$

For the interpolation problem what we want is a function, which interpolates the displacement vectors between the prescribed poses. For each vertex we need to define a function $\mathbf{d} : \mathbb{R}^n \to \mathbb{R}^3$ which returns the displacement vector $\mathbf{v} \in \mathbb{R}^3$ given the skeleton's pose $\mathbf{x} \in \mathbb{R}^n$. It is done through a linear combination of RBFs:

$$d_x(\mathbf{x}) = \sum_{i=1}^{N} w_{x,i}\phi(||\mathbf{x}-\mathbf{x}_i||) \tag{2.6}$$

Here N is the number of poses $\mathbf{x}_i$ and $w_{x,i}$ scalar weights to be found. The function d_x interpolates the x coordinates of the displacement vector. A pose $\mathbf{x}$ can be defined in many ways. In the context of skinning Lewis et al. suggest using the angles θ_k of the k^{th} joint. A pose is then defined by $\mathbf{x} = [\theta_1, \theta_2 \ldots, \theta_k]$ whose distance to another pose can be computed using a standard Euclidean distance $||\mathbf{x}-\mathbf{x_i}|| = \sqrt{\mathbf{x}^T\mathbf{x}_i}$. Notice that the Gaussian parameter σ_i can be set to the same value for every pose, or chosen for each pose manually. The intuition is that σ defines the radius of influence of a pose.

To compute the interpolating function $\mathbf{d} = (d_x, d_y, d_z)^T$, we need to find the weights $w_{x,i}$, $w_{y,i}$ and $w_{z,i}$ by solving at each vertex three linear systems $\Phi\mathbf{w} = \mathbf{d}$ with:

$$\Phi = \begin{pmatrix} \phi(||\mathbf{x}_1-\mathbf{x}_1||) & \ldots & \phi(||\mathbf{x}_k-\mathbf{x}_1||) \\ \vdots & \ddots & \vdots \\ \phi(||\mathbf{x}_1-\mathbf{x}_k||) & \ldots & \phi(||\mathbf{x}_k-\mathbf{x}_k||) \end{pmatrix}, \mathbf{w} = \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_k \end{pmatrix}, \mathbf{d} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_k \end{pmatrix}$$

Lewis et al. [LCF00] solve it using the least square method:

$$\Phi^T\Phi\mathbf{w} = \Phi^T\mathbf{d}$$

Pose space deformation is a very generic approach not limited to skinning. It corrects a

set of parameters in the pose space which can be defined in various ways. For instance we can use the method for facial animation. A pose could be the mood of the character (joy, sadness etc.) and modified with a slider going from happy, through neutral, to sad. Moreover, it is not mandatory to use vectors as corrective parameters, for instance we could use the distance between the eyes or the eyebrow's height. All we need is to define how a pose is represented (joint angles, buttons, sliders etc.), the distance between poses and the corrective parameters.

In practice the technique enables the user to refine the deformations incrementally. Each time the shape is not satisfactory, an example can be added at a specific pose. The challenge for the user is to visualize a large pose space. When the number of examples increases, it can be overwhelming to understand what role plays each pose to the final deformation. The parameter σ which defines the distance between poses can be also unintuitive to setup.

Pose space deformation will enable interactive frame rates, however this runtime performance is directly linked to the number of examples. The evaluation of the interpolating function is $0(n)$ where n is the number of poses and it is computed for each vertex. This extra cost will be directly added to the underlying animation pipeline. Furthermore, pose space deformation is not especially efficient at saving memory since it has to store every vertex in the mesh for each pose in addition to the weights $\mathbf{w}_i$. Some work has been done which significantly reduces the memory footprint using principal component analysis [KJP02]. Finally, when the extrapolation of the data set is not satisfactory one may rely on a more elaborate method based on shape interpolation for better results [WSLG07].

2.3.0.2 Multiple weight enveloping

The number of achievable shapes with linear blending is very limited. But if we increase the number of weights [WP02] the degrees of freedom will be augmented. If we go back to equation 2.4 and show the entries of the transformation matrix $\mathbf{T}_i$:

$$\bar{\mathbf{p}} = \sum_{i=1}^{N} w_i \mathbf{T}_i \mathbf{p} = \sum_{i=1}^{N} w_i \begin{pmatrix} t_{i,11} & t_{i,12} & t_{i,13} & t_{i,14} \\ t_{i,21} & t_{i,22} & t_{i,23} & t_{i,24} \\ t_{i,31} & t_{i,32} & t_{i,33} & t_{i,34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{p}$$

Instead of a single scalar weight associated to the bone i^{th} we can assign a whole matrix corresponding to every entry of T:

$$\bar{\mathbf{p}} = \sum_{i=1}^{N} \begin{pmatrix} w_{i,11}\, t_{i,11} & w_{i,12}\, t_{i,12} & w_{i,13}\, t_{i,13} & w_{i,14}\, t_{i,14} \\ w_{i,21}\, t_{i,21} & w_{i,22}\, t_{i,22} & w_{i,23}\, t_{i,23} & w_{i,24}\, t_{i,24} \\ w_{i,31}\, t_{i,31} & w_{i,32}\, t_{i,32} & w_{i,33}\, t_{i,33} & w_{i,34}\, t_{i,34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{p} \tag{2.7}$$

Adjusting theses weights to match a set of examples can be done using least square fitting. The main drawback lies in the memory consumption. More weights means more memory and more computation when deforming the mesh. This can prevent efficient implementation on a graphic card processor due to bandwidth restrictions and the limited number of attributes per vertex. However there is a more compact solution [MMG06a] similar to this method using fewer skinning weights.

2.3.0.3 Additional joints

Increasing the degrees of freedom of the linear blending can also be achieved by adding more joints in strategic areas. Their positions will depend on the set of examples to reproduce. These joints are often called *virtual joints* since they are not intended to be directly manipulated by the end user. They usually don't have an obvious anatomical meaning.

The first technique introducing this idea [MG03] relies on linear blending and tries to capture the usual deformations such as muscle bulges while avoiding a loss of volume. Only a predefined number of joints are added. For the bicep bulge, joints are added in the middle of the arm's bone and orthogonal to it. These new joints will scale proportionally to the arm's bending angle. Loss of volume can be avoided similarly by adding joints between bones.

Once the joints are added given these heuristics, each vertex is assigned to the k most influential bones. To this end each vertex is expressed in the local frame of every bone. Then we observe the vertex' local position according to the set of examples. The larger the amplitude of the vertex' position the weaker the bone influence is. When the k most influential joints are defined, one can compute the skinning weights that adjust linear blending deformation to the set of examples. This is done by a least square fit:

$$\| \sum_{i=1}^{n} \bar{\mathbf{p}}_i - \mathbf{p}_{ex,i} \|^2$$

Where $\mathbf{p}_{ex,i}$ is the vertex position for the i^{th} example and $\bar{\mathbf{p}}_i = \sum_{i=1}^{m} w_i T_i \mathbf{p}_i$ the vertex deformed by linear blending. This is a bilinear problem in the skinning weights w_i and vertices $\mathbf{p}_i$. Its resolution is performed by an iterative process but its solution is relatively

quick. For instance for 6000 vertices with 5 joints per vertex and 50 examples, computation lasts 5 minutes. Moreover the authors only use a CPU implementation while the operation could be parallelized easily.

To conclude, this method is the least memory intensive among the example-based approaches. It relies on an unmodified linear blending with only a few additional joints. This makes it a good candidate for gaming applications. It is also very robust when the pose is far from the set of examples.

2.4 Volume preservation

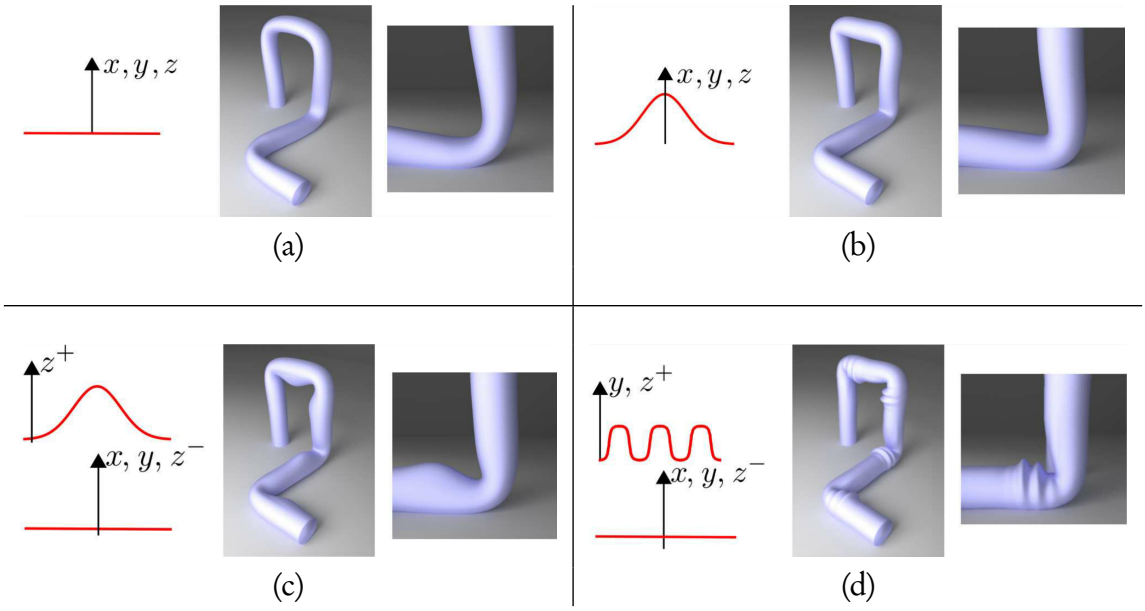


Figure 2.19: (a) Deformation with linear blending (b) Isotropic volume correction (c) Muscle bulge created by increasing the z axis influence (d) A wrinkle effect. [RHC09]

Another approach for solving linear blending loss of volume artifacts, is to minimize this loss of volume directly [FTS08, RHC08, RHC09]. First the mesh volume must be computed at rest pose $V(\mathbf{p}_r)$, this is done by a simple operation looking up every triangle T of the mesh:

$$V(\mathbf{p}) = \frac{1}{6} \sum_{(i,j,k)\in T} \mathbf{p}_i \cdot (\mathbf{p}_j \times \mathbf{p}_k) \tag{2.8}$$

While animating, corrective vectors $\mathbf{u}$ are computed in order to displace vertices, Wolfram Von Funck & al [FTS08] show only a cubic equation needs to be solved in order to modulate the corrective vectors' lengths and restore the rest pose volume. Damien Rhomer & al [RHC09] demonstrate a more flexible process where the corrective vectors u are minimized under the constraint $V(\mathbf{p}+\mathbf{u}) = V(\mathbf{p}_r)$.

$$\begin{cases} \min & ||\mathbf{u}||^2 \\ \text{constraint} & V(\mathbf{p}+\mathbf{u}) = V(\mathbf{p}_r) \end{cases} \tag{2.9}$$

The minimization is done iteratively for each of the three axis (x,y,z) of $\mathbf{u}$ in the local frame of the joint. This enables the user to specifies the amount of correction for each axis.

The system even allows the definition of 1D functions to control the amount of volume correction along the bones for each axis. Muscles bulges are then possible or even wrinkles (cf. figure 2.19).

The method offers a good control over deformation and solves the loss of volume problem, but it does not address the rigidity of the deformation: the result still looks very soft, skin contacts between limbs are not handled either. Moreover computing the volume requires a water-tight mesh which adds an additional constraint for the user.

2.5 Summary

In this chapter we presented an overview of the main skeleton based real-time skinning methods. We can distinguish two different groups: automatic methods, example-based methods.

Automatic methods such as dual quaternions (used with automatic skinning weights) are extremely fast but produce soft or rubbery results compared to a true bone joint deformation. In addition, the user have limited control over the final deformation. Volume preservation is more flexible but operates at low frame-rates and the joints still look rather soft.

Example-based methods can be very realistic and offer a total control of the deformation. But this is only possible with the help of the user time and skills. Pose space deformation will often require dozens of different poses. Each pose usually needs an entire re-sculpt of the character's limb in its new position. Fine details can be added such as wrinkles or even dilated veins, again, it mainly depends on the user's experience and perseverance.

As an extra catagory, we saw how to handle the deformation of limbs without joints such as tentacles spines etc. Such specialized deformations require additional techniques, such as curve deformation which are used in conjuction to a more traditionnal skinning system.

Lastly we notice that none of these methods provide automatic skin contact out-of-the-box. Methods including skin contacts are usually too slow to be used in real-time [MZS+11] or only guarantee the mesh to be self-intersection free [vFTS06, vFTS07, AS07]. Self-intersection free does not mean true skin contact is produced, because large gaps can remain in between skin parts of the folding area of the limbs.

Chapter 3

Technical background on 3D scalar fields and implicit surfaces

3.1 Introduction

Our work heavily relies on implicit surface modeling techniques. For readers unfamiliar with the topic this section provides a technical background necessary to fully grasp our contributions.

First we introduce the necessary definitions and vocabulary related to implicit surfaces. Secondly we describe a technique for implicit surface reconstruction from a point cloud (*Hermite Radial Basis Function* **HRBF** [Wen05, MGV11]). Finally, we describe popular implicit surface composition operators and in particular gradient-based composition operators [GLC+11, CGB13].

3.2 Implicit surfaces

3.2.1 Examples

Implicit surfaces are defined with *potential functions* $f : \mathbb{R}^3 \to \mathbb{R}$. For instance, the Cartesian equation of a sphere defines an implicit surface:

$$||\mathbf{p}-\mathbf{c}||^2 - R^2 = 0 \tag{3.1}$$

$$\Leftrightarrow f(\mathbf{p}) = 0 \tag{3.2}$$

When $f(\mathbf{p}) = 0$ the point $\mathbf{p}$ belongs to the sphere of center $\mathbf{c} \in \mathbb{R}^3$ and radius $R \in \mathbb{R}$. The surface is said to be defined implicitly since the equation does not compute explicitly points on the surface. On the contrary, a parametric equation $h(u,v) : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ explicitly defines a surface as we can directly compute points on the surface for each parameter (u, v). Indeed, $h(u,v)$ directly returns a point on the surface while we need to look up f to find points associated to the value 0. Interestingly, f can be seen as defining a volume explicitly since we can directly test if a point is inside $f(\mathbf{p}) < 0$ or outside $f(\mathbf{p}) >= 0$ the volume. From this perspective h can define an implicit volume.

3.2.2 Definitions

The function f maps every 3D point of the ambient space $\mathbf{p}$ to a scalar value, also called *potential* of f. Therefore f is often called a *scalar field* or a *potential field*. One way to interpret f is to think of it as a temperature field where each point in space is associated to a temperature. In this case, we could draw a surface going through the points having the same temperature. This set of points would define an implicit surface.

Other interpretations of f are possible. For instance f can be seen as a *distance field* since f defines the distance from points $\mathbf{p}$ to the implicit surface $f(\mathbf{p}) = 0$. As we go away from the surface $f(\mathbf{p}) = 0$, points associated to the same values $f(\mathbf{p}) = c$ form an *iso-surface* associated to the *iso-value* c. Consider the sphere equation 3.2 for which $c = 0$, it defines an iso-surface whose shape is a sphere of radius R. Similarly $c = 1$ defines a sphere of larger radius $\sqrt{R^2+1} > R$.

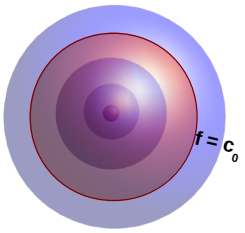


Figure 3.1: A scalar field f can be seen as an infinity of implicit surfaces, each surface is defined as a set of point of equal iso-value $f(\mathbf{p}) = c$. In the case of the sphere equation it defines an infinity of nested spheres

Formally an implicit surface $\mathscr{S}$ is defined as the set of points for which f has equal iso-value c:

$$\mathscr{S} = \{\mathbf{p} \in \mathbb{R}^3 | f(\mathbf{p}) = c\}$$

Visualizing the potential field defined by f in 3D can be tedious, usually one re-writes f as a 2D planar cut of the 3D potential field, for instance with $f_c(x,y) = f(x,y,0)$ as depicted in Figure 3.2. Here, instead of *iso-surfaces*, we visualize *iso-curves* of the 2D potential field $f_c(x,y)$. Two points located on the same level curve have the same potential, reading the figure is the same as reading a geographical map with elevation curves. It is worth noting the gradient ∇f is always orthogonal to the iso-curves.

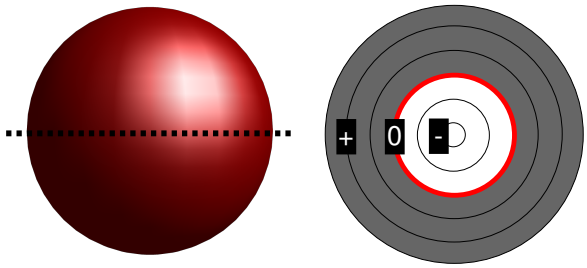


Figure 3.2: Left: cut along (x,y) plane of the scalar field of a sphere (equation 3.2) and right the iso-curve visualization.

One can easily compute the normal to an iso-surface at any point by computing the gradient of f:

$$\nabla f(x,y,z) = \begin{pmatrix} \frac{\partial f(x,y,z)}{\partial x} \\ \frac{\partial f(x,y,z)}{\partial y} \\ \frac{\partial f(x,y,z)}{\partial z} \end{pmatrix}$$

The normal at the (x,y,z) coordinates:

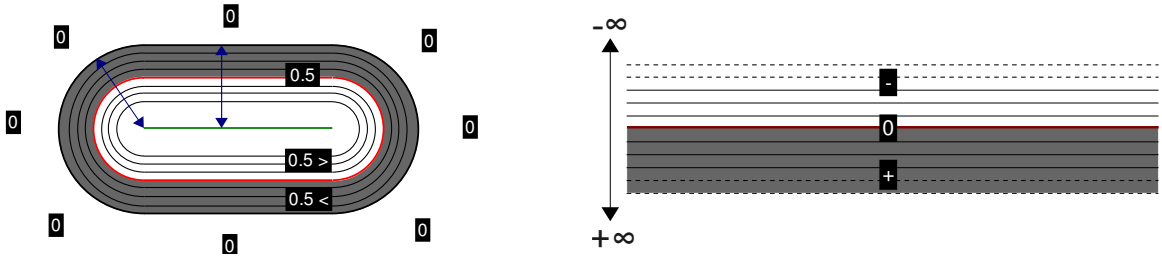
$$\nabla \mathbf{n}(x,y,z) = -\frac{\nabla f(x,y,z)}{||\nabla f(x,y,z)||}$$

3.2.3 Implicit surface support

The scalar field of an implicit surface can have various behavior, here we distinguish two types of support for implicit surface primitives: *global support* and *compact support*. A primitive with global support will see its potential vary anywhere in the ambient space $\mathbb{R}^3$. On the other hand, compactly supported primitives will only vary in a finite interval and potential outside a certain range will be constant. Formally:

- for a globally supported primitive we say *there are no balls such as the field-function is constant outside.*
- For a compactly supported primitive, *there is a ball such as the field-function is constant outside.*

Figure 3.3 depicts global and compact support.



(a) (b)

Figure 3.3: (a) compactly supported primitive, the distance-field of this cylinder is computed as the distance from a point to its median segment, the blue arrow represents the primitive radius beyond which the potential is null. (b) has global support, it is the plane equation, the potential approaches infinity as we get farther from the plane.

3.2.3.1 Global support

The equation 3.2 has global support, the sphere's potential indefinitely increases as we go away from its center. This is also the case of some Cartesian equations defining simple primitives such as cylinders, planes etc. When modeling 3D objects we adopt certain convention to represent globally supported primitives:

- the iso-surface $f(p) = 0$ is the surface of interest which the user manipulates and visualize.
- $f(p) < 0$ inside the object
- $f(p) > 0$ outside the object

3.2.3.2 Compact support

Primitives with compact support are often defined by the inverse distance from a point, segments, polygons etc. In this case, the distance equals one at the center of the primitive and smoothly decreases until zero is reached. The distance stays constant outside a specified radius. We adopt the following conventions for compactly supported primitives:

- $f(p) = 0.5$ is the surface of interest.
- $f(p) > 0.5$ inside the object
- $f(p) < 0.5$ outside the object

3.2.3.3 Global versus compact support

Compact support has appealing features over global support. Because the field's range is confined this enables scalar-field to be stored into a 3D discrete grid, this greatly accelerates the evaluation of the field-functions (needed for instance in ray-casting [GPP+10]). Generally speaking, compact support is easier to apprehend and work with. For instance, when blending several objects, one only focus on local interaction between primitives. We can take advantages of this locality to further optimize the blending algorithm computation. In addition, interesting composition operators were recently developed for compactly supported primitives such as gradient based composition operators [GLC+11, CGB13] (described later in this chapter). For all the above reasons our work only relies on primitives with compact support.

3.3 HRBF surface reconstruction

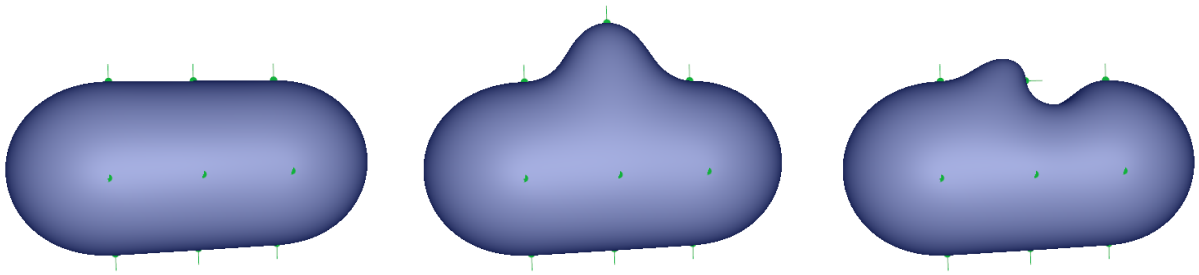
To model complex shapes, one usually makes use of the natural ability of the implicit surfaces to blend. It is possible to combine simple primitives (spheres, cylinders etc.) in various ways, for instance union, blend, intersection etc. While this can be useful to model new models, it can be difficult to approximate existing shapes this way. In our work we rely on a surface reconstruction method called Hermite Radial Basis Function (**HRBF**) [Wen05, MGV11] which allows us to compute implicit surface primitives approximating an existing mesh model or subpart of the model. We summarize the **HRBF** method in this section.

3.3.1 Method overview

The reconstruction problem consists in finding a field-function $f : \mathbb{R}^3 \to \mathbb{R}$ satisfying the constraints $f(\mathbf{p}_i) = c$ with $i \in [1; N]$. In other words, we seek an implicit surface f which zero iso-surface fits our input data, i.e a point cloud defined by the positions $\mathbf{p}_i$. Therefore f interpolates the iso-value c in the ambient space $\mathbb{R}^3$. Unfortunately, we need more constraints since solving for $f(\mathbf{p}_i) = c$ leads to the trivial solution $f(\mathbf{x}) = c$ (in other words f is constant everywhere in the ambient space). One way to alleviate this, is to add more

constraints with offset points inside and outside the desired implicit surface (i.e adding points with potential equal to $c \pm \epsilon$). Offset points are not ideal since they can produce self-intersection of the implicit surface in concavities, this can make the reconstruction unstable. This approach is commonly known as the **RBF** fitting method. A more in depth study of the **RBF** approach can be found in Reuter's thesis [Reu03].

In our work we use a more flexible variant of the **RBF** technique called **HRBF** [SOS04] for Hermite **RBF**. In this version, in addition to the iso-value constraints $f(\mathbf{p}_i) = c$ we use gradient constraints $\nabla f(\mathbf{p}_i) = \mathbf{n}_i$. The interpolated data are Hermite data, i.e. the pair of points and normals $(\mathbf{p}_i, \mathbf{n}_i)$. Adding the gradient constraints avoid us the hassle of adding more value constraints with offset points. This results in a simpler and more robust reconstruction technique (see figure 3.4).



(a) (b) (c)

Figure 3.4: Reconstruction of Hermite data (green dots and lines) with various configurations. (a) Capsule shape reconstruction then (b) one sample's position is changed and (c) only a single sample's normal is rotated.

Within the **HRBF** framework f is expressed as follows:

$$f(\mathbf{x}) = \sum_i^N \alpha_i \phi(||\mathbf{x}-\mathbf{p}_i||) + \beta_i^T \nabla\phi(||\mathbf{x}-\mathbf{p}_i||)$$

$$f(\mathbf{x}) = \sum_i^N \alpha_i \phi(||\mathbf{x}-\mathbf{p}_i||) + \beta_i^T \left(\phi'(||\mathbf{x}-\mathbf{p}_i||)\frac{\mathbf{x}-\mathbf{p}_i}{||\mathbf{x}-\mathbf{p}_i||}\right)$$

With N the number of samples, $\alpha_i \in \mathbb{R}$ a scalar and $\beta_i \in \mathbb{R}^3$ a vector. ϕ is a radial basis function for which we recommend the polyharmonic spline $\phi(\mathbf{x}) = \mathbf{x}^3$. Empirical experiments using our skinning technique showed us $\mathbf{x}^3$ was more robust than other **RBFs** such as thin plate splines, Gaussian or other polynomial.

The expression of the gradient of f will be necessary:

$$\nabla f(\mathbf{x}) = \sum_i^N \alpha_i \phi'(||\mathbf{x}-\mathbf{p}_i||)\frac{\mathbf{x}-\mathbf{p}_i}{||\mathbf{x}-\mathbf{p}_i||} + \underbrace{\nabla\nabla\phi(||\mathbf{x}-\mathbf{p}_i||)}_{\text{Hessian of } \phi(||\mathbf{x}-\mathbf{p}_i||)}\beta_i$$

In order to interpolate the surface going through the input point cloud, we must find the weights α_i and β_i solving the following linear system:

$$\begin{pmatrix} f(\mathbf{p}_i) \\ \nabla f(\mathbf{p}_i) \end{pmatrix} = \begin{pmatrix} c \\ \mathbf{n}_i \end{pmatrix}$$

the system contains $n+3n$ equations which expands as follows:

$$\begin{pmatrix} f(\mathbf{p}_1) \\ \vdots \\ f(\mathbf{p}_N) \\ f_x(\mathbf{p}_1) \\ f_y(\mathbf{p}_1) \\ f_z(\mathbf{p}_1) \\ f_x(\mathbf{p}_2) \\ \vdots \\ f_z(\mathbf{p}_N) \end{pmatrix} = \begin{pmatrix} c \\ \vdots \\ c \\ \mathbf{n}_{1,x} \\ \mathbf{n}_{1,y} \\ \mathbf{n}_{1,z} \\ \mathbf{n}_{2,x} \\ \vdots \\ \mathbf{n}_{N,z} \end{pmatrix}$$

This system is dense and not symmetrical and will be robustly solved with a standard **LU** decomposition. Fortunately, once the coefficients α_i and β_i are computed they can be stored once and for all, thus it does not impact the evaluation of the implicit surface f.

3.3.2 HRBF support

Due to our choice to use the **RBF** $\phi(\mathbf{x}) = x^3$, the generated field-function f has global support (note that c is set to zero to respect our conventions). This prevent us to use composition operators designed for compactly supported primitives. We could have used a compact **RBF** but as stated earlier, experiments showed us this impacts the stability of the reconstruction. Instead we can map our globally supported primitive to a compactly supported primitive. For this we use a map function $t_r : \mathbb{R} \to \mathbb{R}$ which takes into input a globally supported primitive and outputs a bounded scalar-field. The new scalar-field $t_r(f(\mathbf{x}))$ has a compact support between $[0;1]$ and is null outside the radius r (c.f. figure 4.5). The

mapping is detailed section 4.4.

3.4 Field function composition

The main strength of implicit surfaces comes from their simplicity to be combined together into a new shape. Indeed, expressing the union, intersection or difference between implicit surfaces is extremely straightforward. Implementing a boolean modeling interface (figure 3.5) becomes easy with this representation.

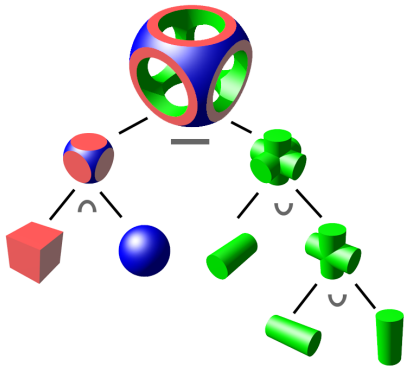


Figure 3.5: Boolean modeling, union ∪, intersection ∩ and difference — are performed among several objects resulting in a more complex shape. The order composition operations are applied form a **CSG** tree (Constructive Solid Geometry).

On the other hand, it is known the combination of parametric surfaces (or meshes) is substantially harder to implement due to the numerous special cases to handle and numerical instabilities. In this section, we describe the composition mechanism between field functions. As stated earlier, we will only discuss the composition of compactly supported primitives.

3.4.1 2D composition operators

Let us define the combination between two surfaces. Most composition operators are defined as a 2D function $g : \mathbb{R}^2 \to \mathbb{R}$. The operator takes in input two field functions, f_1 and f_2, then returns a new scalar field $g(f_1, f_2)$ representing the combination of the two input implicit surfaces.

One of the simplest composition operator was introduced by Ricci and is expressed with the function maximum [Ric73]. It is possible to perform the union of two primitives (figure 3.6) by only computing the maximum between the two scalar-field $max(f_1, f_2)$.

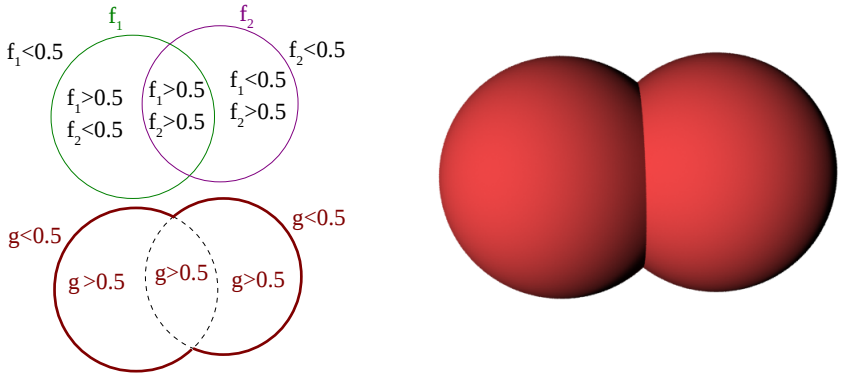


Figure 3.6: Upper left, illustration of two spheres and the sign of the potential according to the different regions (recall a compact support implies $f < 0.5$ outside and $f > 0.5$ inside). On the right, resulting surface from the composition $g(f_1, f_2) = max(f_1, f_2)$.

Interestingly one can infer the intersection and difference operator from the definition of the union:

$$\begin{aligned} \mathcal{S}_1 \cap \mathcal{S}_2 &= \neg(\neg\mathcal{S}_1 \neg\mathcal{S}_2) \\ \mathcal{S}_1 - \mathcal{S}_2 &= \mathcal{S}_1 \cap \neg\mathcal{S}_2 \end{aligned}$$

Therefore, from the max operator the intersection is $min(f_1, f_2)$ and the difference $min(f_1, 1 - f_2)$.

Other operators allow more complex effects such as a smooth blending of the objects or even the production of wrinkles or bulges around the colliding areas. A popular operator producing a smooth blend is the sum operator $g(f_1, f_2) = f_1 + f_2$. This operator is often used with metaballs [BW97] (see figure 3.7 left). It also possible to control the amount of blend surrounding the colliding parts with more advanced operators [BWd04].

In our work, we rely on a special type of composition operator called gradient-based operator. Those operators behave differently according to the angle between the gradients (∇f_1 and ∇f_2) of two distinct primitives [GLC+11]. This can be used to localize a blending effect, such as bulge, within a particular area. In addition, gradient-based operators solve multiple problems found in less advanced operators (e.g. the sum operator).

An operator driven by the gradient angle of two primitives is expressed as follows: $g_d(f_1(\mathbf{p}), f_2(\mathbf{p}))$ where f_1 and f_2 are the composed primitives at point $\mathbf{p}$ and α is an interpolation parameter between two types of operators. For instance, g can interpolate between a union and a blending operator. The parameter is controlled by a function $d(\alpha) : \mathbb{R} \to \mathbb{R}$

called controller (figure 3.8). The controller takes into input the angle $\alpha = \nabla f_1 \measuredangle \nabla f_2$ between the two primitives and maps it to the desired composition operator. With very simple geometric observation one is able to apply an operator in specific areas by customizing the controller function d.

Let's observe a composition operator which interpolates between a smooth blending when $\alpha = \frac{\pi}{2}$ and an union $\alpha = 0$ (i.e. collinear gradients). The result is depicted in Figure 3.7. Three cylinders are composed together in order to form the letter "A". On the left smooth blending is applied everywhere, the final object bulges at the intersection of the cylinders and there is no hole in the middle, contrary to what we would expect. On the right smooth blending is localized appropriately and a standard union applied everywhere else, therefore no bulge occurs and the hole in the middle of the letter preserved. Smooth blending localization is driven by the angle between the gradients of the cylinders (or equivalently the normals). Areas where gradients are collinear or almost collinear will be composed with a standard union, areas where the angles is closer to $\frac{\pi}{2}$ will be blended.

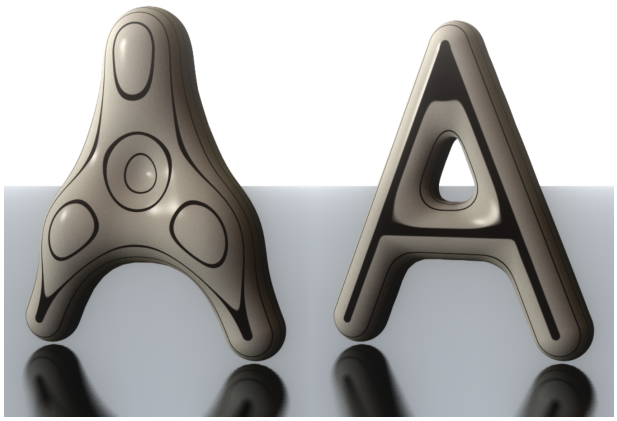


Figure 3.7: Left, three cylinders form the shape "A" using a standard smooth blending operator. Right, gradient based operator the blending is localized at the intersection of the cylinders and the hole preserved. Source: [GLC+11]

This type of control is especially useful in the character animation context. For instance, when animating a finger joint or elbow joint, we can localize bulge in the fold of the finger joint or a smooth blend in the interior of the elbow.

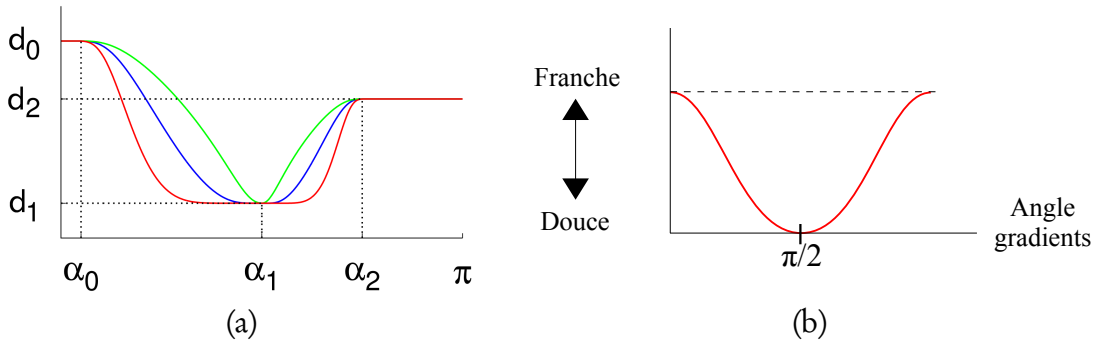


Figure 3.8: (a) Controller function as described by Gourmel & al [GLC+11]. $d(\alpha)$ can be parametrized by three control points $\mathbf{p}_0(d_0, \alpha_0)$, $\mathbf{p}_1(d_1, \alpha_1)$ et $\mathbf{p}_2(d_2, \alpha_2)$. It is also possible to customize the slope of the curve between the points $\mathbf{p}_0$ and $\mathbf{p}_1$, or also between $\mathbf{p}_1$ and $\mathbf{p}_2$ by setting scalar values w_0 and w_1. We plot different values of w_0 and w_1 with different colors. (b) Plots the controller function for figure 3.7 right.

Chapter 4

Implicit skinning

4.1 The bigger picture

This research focus on skinning, a quite specific topic within computer animation, but we further motivate this work by broadening our scope.

An underlying goal of this research, is to explore the use of implicit surfaces and blending operators used in conjunction with meshes. Meshes and implicit surfaces can be considered as dual representations, meaning the strength of one is the weakness of the other and vice versa. For instance implicit surfaces are more efficient for collision detection. Indeed they are explicitly defined by the notion of inside and outside of a volume. Closed meshes define volume implicitly, therefore they need intermediate steps to distinguish interior from exterior. In other words, any operations that need an explicit volume will be more straightforward using implicit surfaces. For instance boolean modeling is known to be more robust with implicit surfaces and more generic to implement (i.e with less special cases to handle). The corollary is that explicit operations on surfaces need an explicit surface representation. Rendering meshes is several order of magnitude faster than implicit surfaces and they are easier to parameterize.

One of the innovation of this research is to further the exploration of a hybrid implicit-mesh technique [LAG01]. A practical application such as character skinning, demonstrates the advantage of the duality presented above. Issues such as collision detection are to be considered to simulate skin contacts between limbs. Correct parametrization is important as well to produce plausible skin deformation (e.g skin sliding and elasticity). Finally the system must be fast enough to enable animators to work with the character.

4.2 Framework overview

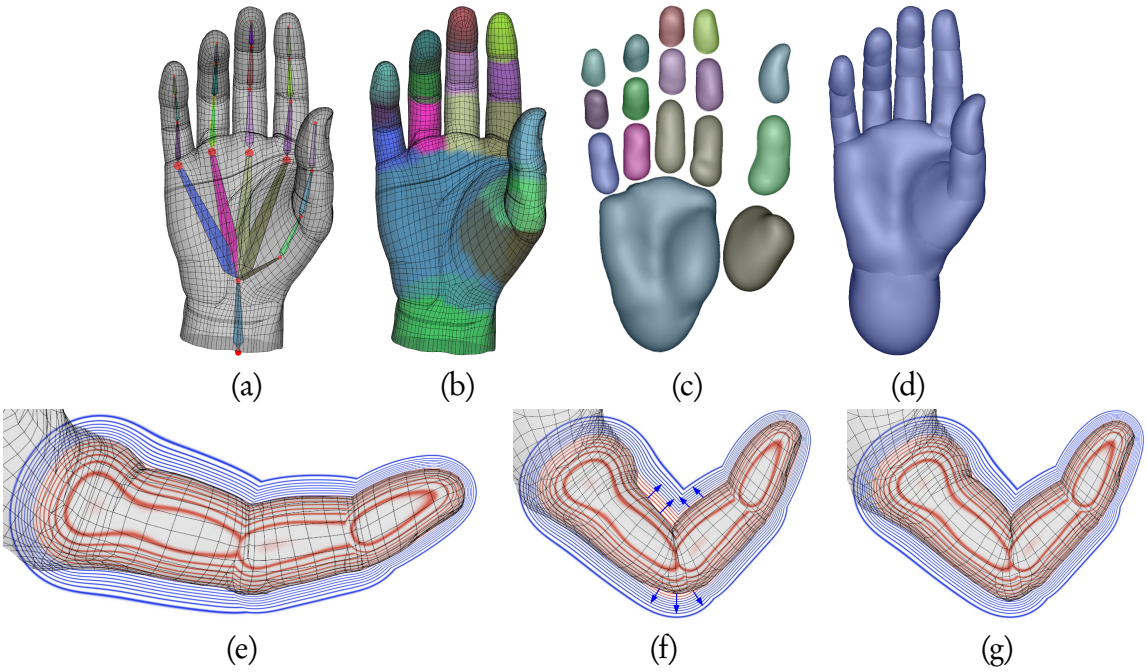


Figure 4.1: **Overview.** (a) An input mesh with its animation skeleton and (b) mesh segmentation. (c) Implicit surfaces computed as 0.5-isosurfaces of HRBFs approximating each part of the mesh. (d) Composition with a union operator, and resulting shape. (e) Each vertex is assigned to a field value in the rest pose. (f) During animation, field functions are transformed rigidly and the mesh deformed with a computationally inexpensive method (dual quaternion skinning or rigid skinning), then mesh vertices can march along the field gradient until they reach their individual iso-value. (g) This produces the final skinned mesh. In (e), (f) and (g), the isosurfaces of the field function are plotted in a vertical plane centered on the skeleton. In blue, the outside values ($f < 0.5$) and in red the inside values ($f > 0.5$).

The steps of the framework we set out to devise are illustrated in Figure 4.1. As for standard geometric skinning [MTLT88, KCvO08], we will start from a mesh equipped with an animation skeleton defined as a hierarchy of bones (Figure 4.1(a)). In addition, the mesh will already be partitioned with respect to skeleton bones (Figure 4.1(b)). Since partitioning is out of the scope of our research, this input can either be provided by artists or automatically generated. For instance, we can deduce the partitioning from skinning weights, by simply assigning vertices to bones with the largest influence.

From these initial settings, we can use Hermite Radial Basis Functions (**HRBF**) [Wen05, MGV11] to approximate each part of the mesh with the 0.5-isosurface of a smooth scalar field $f_i : \mathbb{R}^3 \to \mathbb{R}$ (Figure 4.1(c), Section 4.4). We can then define, a single field function f

from the combination of the f_i using either the union [Ric73], or gradient-controlled bulge operators [GBC+13] depending in the desired result (Figures 4.1(d)). In addition to these state of the art operators, we define a new family of gradient-based composition operators specifially designed for the implicit skinning. These operators are able to capture contact surfaces between different parts of the implicit skin and to generate a smooth field around it (Section 4.5). They allow more control over the shape of the deformation and further stabilize our isosurface tracking algorithm.

After these steps we have the whole character approximated by a single field function f. The last pre-computation needed is that, each vertex $\mathbf{v}$ of the mesh needs to store its current field value $f(\mathbf{v})$. This step will ensure the details of the mesh are encoded (similarly to a displacement map each vertex saves its distance from the 0.5 isosurface).

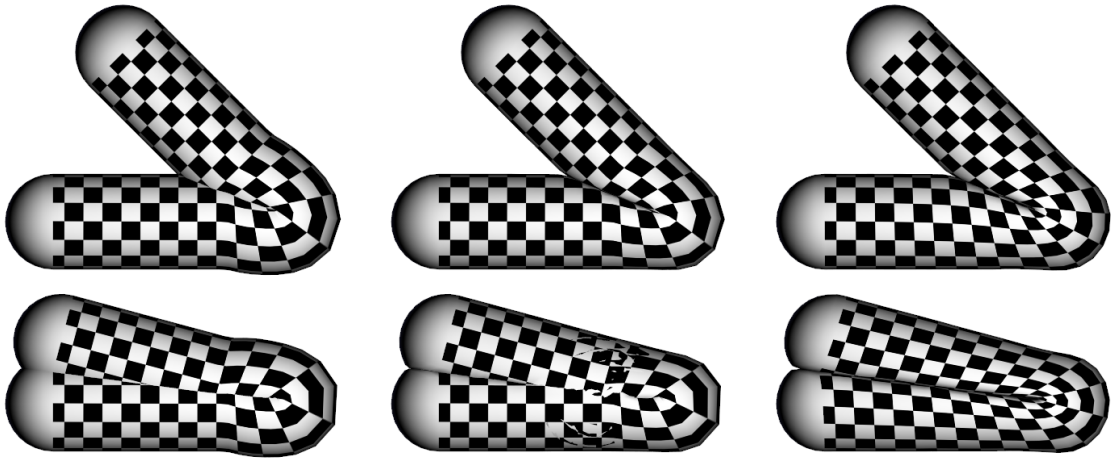
From these settings we need to deform the character while animated. Deforming the implicit surface f is done easily by moving the different implicit parts f_i as a rigid body. The blending operators will ensure the time coherency and well-behaved shape of f. Tracking the implicit surface f as it changes with the mesh is the key operations for successful skinning. Two types of algorithms can be designed: history independent algorithms [VBG+13] or history dependent algorithms [VGB+14] (both presented Section 4.6.3.1). We dub the history independent version the *implicit skinning* method [VBG+13] and the history dependent version the *elastic implicit skinning* [VGB+14].

4.3 Contributions

To the best of our knowledge, the idea to use implicit surfaces to drive the deformations of a pre-existing mesh-based character while preserving its details has not been previously explored. This allowed us to develop the first method simulating, in real time, skin contacts and other effects such as skin bulge, skin elasticity or even muscle bulging.

In addition to a novel skinning approach, our main technical contributions are:

- **A specific reconstruction method** for mesh parts, based on Hermite Radial Basis Functions (HRBF) [Wen05].
- **New composition operators** enabling us to approximate the shape of the character's skin during animation with an *implicit skin* that includes contact surfaces with a controllable depth, as depicted in Figure 4.20(c) (Section 4.6.3.2).
- **Two isosurface tracking algorithms.** First, a fast, history independent algorithm easy to integrate in standard animation pipeline (Part of our implicit skinning method [VBG+13]).



(a) - Dual Quat. (b) - Imp. Skinning (c) - Elastic Imp Skinning

Figure 4.2: Illustration of different skinning solutions on a cylinder deformed with a single joint. (a) Dual quaternions produce bulge and self-intersection artifacts. (b-top) The implicit skinning corrects these artifacts, (b-bottom) but the projection stage fails for too large bending angles. (c) Elastic implicit skinning avoids self-intersections, produces a contact surface and minimizes mesh distortions, even for very large bending angles.

It consists in a fast marching algorithm for mesh vertices in a scalar field, used in conjunction with a specific mesh relaxation method avoiding the introduction of mesh distortions. Finaly, a slower, yet more robust history dependent tracking method based on a linear relaxation energy (inspired from the as-rigid-as-possible energy [SA07]) to update the skin mesh with a plausible tangential distribution of vertices, as shown in Figures 4.2(c) or 4.16(c) Section 4.6.3.1. The history dependent tracking was presented along with our work introducing elastic implicit skinning [VGB+14].

- **Easy extensions to advanced effects** such as muscle inflation (Figure 4.23) and skin sliding, thanks to the clear decoupling between volumetric deformations handled by the implicit skin and stretching effects handled by the skin mesh (Section 4.6.3.2).

The resulting techniques generates visually plausible skin deformations in real-time. Our methods automatically generates contact surfaces between skin parts, without requiring any collision detection step.

4.4 Reconstruction (f_i)

Problem setting. Given a bone i and its associated sub-mesh $\mathscr{M}_i$, our goal is to reconstruct a smooth scalar field f_i that tightly approximates $\mathscr{M}_i$. In contrast to the standard surface reconstruction problem, our f_i needs to satisfy several specific constraints.

Firstly, high field smoothness is essential to avoid unexpected behaviors of the gradient-controlled operators (Section 4.5) and to stabilize gradient-descent projection (Section 4.6.2.1 and 4.6.3.1) during animation. Smooth fields also require fewer parameters, which reduces their memory footprint and speeds up the fitting and evaluation steps. In order to reach this smoothness while preventing loss of details of the input shape, our insight is to embed the mesh in the resulting scalar field, rather than having it exactly coincide with the 0.5-isosurface: each vertex $\mathbf{v}_j$ stores its local field value $f(\mathbf{v}_j)$ at rest, enabling it to be projected onto its own isovalue during deformation.

Secondly, the field function f_i should appropriately close the large holes left by the mesh partitioning near joints as depicted by the blue curve in Figure 4.3(a).

Lastly, to allow for local compositions of the f_i, as well as to speed-up the evaluation of f, the field functions should be compactly supported. Following our conventions from the technical background, we require f_i to range in $[0,1]$, with 0.5 being the reference isovalue, and use the convention: $f_i(\mathbf{x}) > 0.5$ if $\mathbf{x}$ is inside, $f_i(\mathbf{x}) < 0.5$ if $\mathbf{x}$ is outside.

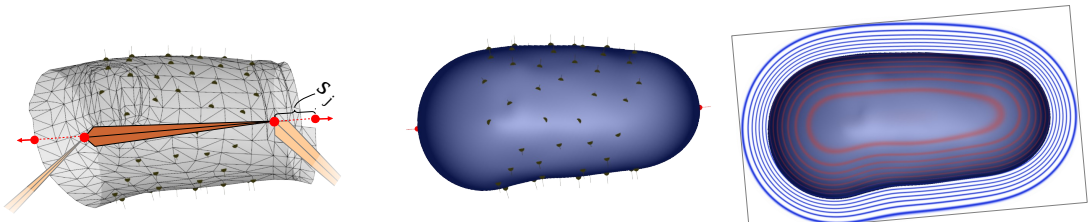


Figure 4.3: Reconstruction of a mesh part (a phalanx of the hand shown in Figure 4.1). (a) HRBF nodes are uniformly spread on the mesh surface, and two additional nodes in red are aligned with the bone to close the holes. (b) The resulting implicit surface, and (c) its associated compactly supported scalar field. [VBG+13]

Hermite RBF. Given all these constraints, we propose the use of Hermite RBFs [Wen05, MGV11] to reconstruct a global signed distance field d_i (the mesh being approximated by the zero isosurface). This distance field is then re-parameterized to yield the compactly supported field function f_i.

The HRBF enables interpolating Hermite data (set of positions and normals) seamlessly without having to populate the whole space with RBFs, nor relying on offset constraint points which can produce reconstruction artifacts [SOS04]. In addition, it is scale independent and it robustly reconstructs concavities. Recall, we seek a distance field d_i of the form:

$$d_i(\mathbf{x}) = \sum_{k=1}^{m} \left(\lambda_k \phi(||\mathbf{x} - \mathbf{v}_k||) + \boldsymbol{\beta}_k^T \nabla \phi(||\mathbf{x} - \mathbf{v}_k||) \right) , \tag{4.1}$$

where the vertices $\mathbf{v}_k$ are the HRBF centers, the scalars λ_k and vectors $\boldsymbol{\beta}_k$ are the unknown coefficients, and ϕ is a smooth function for which we recommend the polyharmonic spline $\phi(\mathbf{x}) = \mathbf{x}^3$. Given a set of m points $\mathbf{v}_k$ with prescribed normals $\mathbf{n}_k$, the $4m$ unknown coefficients are easily found by solving for the system of $4m$ equations: $d_i(\mathbf{v}_k) = 0$ and $\nabla d_i(\mathbf{v}_k) = \mathbf{n}_k$. At this stage, d_i has global support and will be re-parametrized to a compact support with $d_i = 0.5$ fitting our Hermite data.

There still remains the delicate choice of the HRBF centers. Since we only want to approximate the input sub-mesh $\mathcal{M}_i$ while leaving out the details, a natural choice is to sample the mesh surface with a few samples. To handle arbitrary meshes robustly, we employ a Poisson disk sampling strategy [WCE07] (Figure 4.3(a,b)). In all our tests, targeting around 50 to 100 samples has always been sufficient.

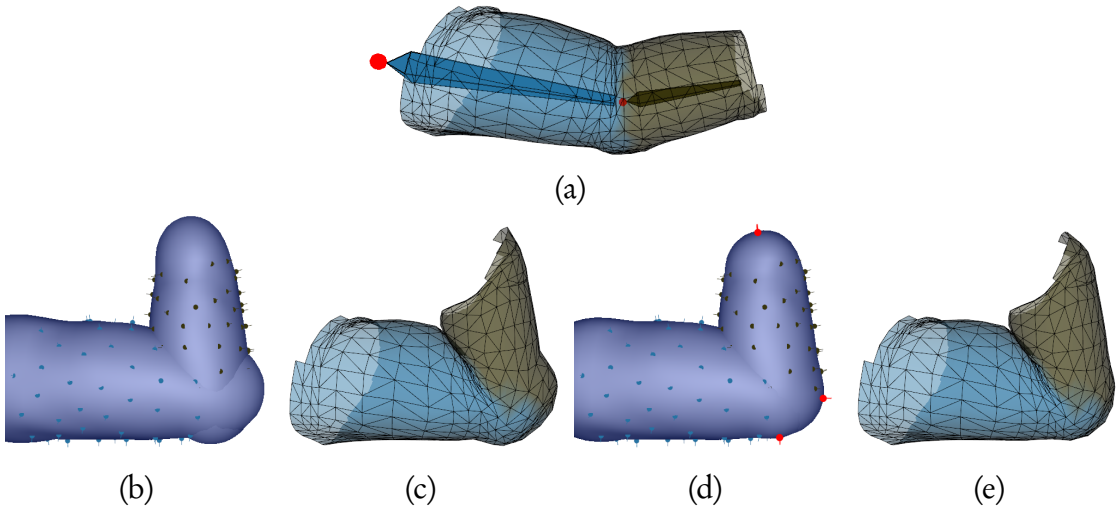
Even though degree three polyharmonic RBFs naturally fill the large holes left at the bone extremities (Figures 4.4(a,b)), they have to be closed in such a way that the common extremities of two adjacent fields slide over each other without introducing gaps, or creating outgrowths as is the case in Figure 4.4(b). Indeed, this would result in a poor skin deformation at joints, as illustrated in Figure 4.4(c). Ideally, the extremities of two fields at a given joint should be filled with spherical components of the same radius and centered at the joint location, which is impossible in general. Following this observation, we propose to adjust the closure of the distance field as in Figure 4.4(d) by adding one HRBF center at each extremity along the line supporting the bone i and at a distance s_j from the respective joint j (Figure 4.3). Their normal constraints are aligned with the bone and point outward. The distance s_j is set as the distance from the joint to the nearest vertex. This allows us to generate a plausible bone joint deformation automatically (Figure 4.4(e)).

Finally, in order to let the HRBFs produce a very smooth surface, samples which are too close to a joint are automatically removed. To this end we employ culling planes orthogonal to the bone and placed at a distance h of the extremities. Formally, let $\mathbf{b}_i^0$ and $\mathbf{b}_i^1$ the two extremities of the bone i, all vertices $\mathbf{v}_k$ that do not satisfy the following criterion

are removed:

$$h < \frac{(\mathbf{v}_k - \mathbf{b}_i^0)^T(\mathbf{b}_i^1 - \mathbf{b}_i^0)}{||\mathbf{b}_i^1 - \mathbf{b}_i^0||^2} < 1 - h\,. \tag{4.2}$$

We found $h = 0.05$ to be an effective value. Since HRBFs are computed in a few milliseconds, vertices could be easily and interactively added or removed by the user if the automatically reconstructed surface was not fully satisfactory.



(a) (b) (c) (d) (e)

Figure 4.4: Two implicit surfaces reconstructed from two phalanxes of the index finger (Figure 1.2(a)) with a joint in its center shown in (a). (b) When reconstructed without our additional closure constraints (extra points along the bone axis), the junctions overlap, resulting in unexpected bumps. (c) Undesired visual result obtained after projecting the mesh vertices. (d) The additional closure constraints used to solve the problem are shown in red. (e) Resulting mesh after vertex projection: bumps are prevented and the mesh adequately models the joint.

Re-parameterization. Finally, the compactly supported field functions f_i we seek are computed using the following remapping: $f_i(\mathbf{x}) = t_r(d_i(\mathbf{x}))$, t_r being defined as:

$$t_r(x) = \begin{cases} 1 & \text{if} \quad x < -r \\ 0 & \text{if} \quad x > r \\ \frac{-3}{16}(\frac{x}{r})^5 + \frac{5}{8}(\frac{x}{r})^3 - \frac{15}{16}(\frac{x}{r}) + \frac{1}{2} & \text{otherwise}\,, \end{cases}$$

where r is a value which sets the size of f_is compact support. We plot t_r figure 4.5. The purpose is to convert the infinite support of the HRBF d_i into a finite support field function

f_i, while ensuring smoothness at the border of its support. More precisely, t_r is defined as to map the HRBF values of d_i which are in $[-r, r]$ smoothly onto the interval $[0, 1]$, with $t_r(-r) = 1$ and $t'_r(-r) = 0$ (inside the shape), $t_r(0) = 0.5$ (on the surface) and $t_r(r) = t'_r(r) = 0$ (outside). In order to avoid getting constant values $f_i = 1$ and thus null gradients inside the shape, we set r to the distance between the bone and the farthest sampling point used for reconstruction. Then, $f_i = 1$ is only reached on the skeleton and the support size nicely scales with the size of the reconstructed shape. Note that this piecewise function t_r is C^2 at its junctions. This guarantees the continuity of the gradient.

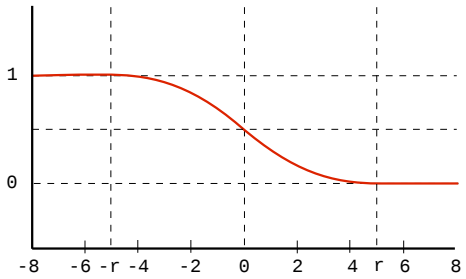


Figure 4.5: Plot of the field-function mapping t_r from global support to compact support, the radius r is set to $r = 5$. Note that $t'_r(-5) = t'_r(5) = t''_r(-5) = t''_r(5) = 0$, $t_r(0) = 0.5$, $t_r(-r) = 1$ and $t_r(r) = 0$.

The gradient ∇f_i must be re-parameterized as well. Our mapping t_r composed with the scalar-field f_i tells us: $\nabla(t_r \circ f)(\mathbf{x}) = t'_r(f(\mathbf{x})).\nabla f(\mathbf{x})$. Therefore we need the first derivative of t_r:

$$t'_r(x) = \begin{cases} 0 & \text{if} \quad -r < x < r \\ \frac{-15}{16r}.(\frac{x}{r})^4 + \frac{15}{8r}.(\frac{x}{r})^2 - \frac{15}{16r} & \text{otherwise} \end{cases}$$

4.5 Implicit skin definition

Once the individual compactly supported field functions f_i are computed, we combine them to define a global field function f fitting the whole mesh to be skinned. In our early work [VBG+13] we mainly relied on Ricci's max operator [Ric73] $f = \max_k(\tilde{f_k})$ to compose the scalar fields f_i. The max operator introduced a gradient discontinuity which we used at our advantage to encode contact surfaces. The gradient discontinuities were conveniently placed in the fold of limbs colliding against each other, our tracking algorithm would detect the discontinuity and stop vertices in the fold. Unfortunately, while this gave satisfactory results, we noticed later the gradient discontinuity was responsible for flickering artifacts because we use those direction to project vertices.

In our latest work [VGB+14] we solve the problem of enabling skin volumes to blend while sharing local contact surfaces using two new binary operators, organized in a composition tree [WGG99]. The leaves of the tree are the field functions f_i each transformed in the same way as its corresponding bone and the composition operators are the interior nodes. A binary composition operator is a function $g : \mathbb{R}^2 \to \mathbb{R}$ combining two scalar fields f_1 and f_2 in a new 3D scalar field $f_k(\mathbf{p}) = g(f_1(\mathbf{p}), f_2(\mathbf{p}))$. Thus, each node of the tree defines a 3D scalar field and the root is the scalar field f including the implicit skin. As introduced earlier in the technical background, our application use a special type of composition operators called gradient-based operators.

Recall gradient-based operators $g_d : \mathbb{R}^2 \to \mathbb{R}$ are parameterized by a function $d : \mathbb{R} \to \mathbb{R}$ whose argument is the angle α between ∇f_1 and ∇f_2. The parameter d controls the interpolation between two types of compositions, allows the discrimination of regions that blend according to the gradient angle and the application of different types of compositions. In the case of skinning, we use it to localize blending or bulge-in-contact effects in the fold only.

In our context, the effect of these operators is simpler to understand by noticing that the angle α between gradients in the folding regions is often close to the joint rotation angle. In the rest pose, no rotation is applied and thus, no specific skinning effect is to be performed. In that case, we need the 0.5 iso-surface of the scalar fields f_i to be combined with a union while the rest of the iso-surfaces are smoothly blended around the implicit skin. This type of operator is called a clean-union operator [PASS95, BBCW10]. When a joint such as the elbow bends between $\alpha = 0$ and $\alpha = \pi/2$, the skin does not crease and no contact iso-surface need be generated. The use of a clean-union operator produces a smooth field around the implicit skin. For larger bending angles, $\alpha > \pi/2$ at an elbow, the contact iso-surface has to

be created using our contact operator. In this case the larger the bending angle, the deeper the contact surface goes towards the joint. Finally, when the gradients point in opposite directions ($\alpha = \pi$), it means two disjoint skin parts have come into contact, for instance an arm against the torso, and a full contact operator should be applied.

Our first operator g_{d_c} is parameterized by the controller d_c for contact handling. It smoothly interpolates between a clean-union (no contact) when $d_c(\alpha) = 0$ and a full contact (contact of maximal length) when $d_c(\alpha) = 1$. Therefore g_{d_c} can control the depth of the contact. The behavior presented above leads us to the plot of $d_c(\alpha)$ depicted in Figure 4.6(a).

Figures 4.7(a,b) illustrate the relation between (a) the plot of a contact composition operator $g_{d_c}(f_1, f_2)$ and (b) its effect on the composition of two scalar fields f_1 and f_2 for an intermediate value of d_c. In Figure 4.7(a), the abscissa (resp. ordinates) are values of f_1 (resp. f_2) and vertical (resp. horizontal) lines represents its iso-surfaces. The operator defines the way the iso-surfaces of f_1 and f_2 are combined to produce the resulting scalar field f_k. Values of g_{d_c} are those of f_k when $f_k(\mathbf{p}) = g_{d_c}(f_1(\mathbf{p}), f_2(\mathbf{p}))$. Here, $g_{d_c} = 0.5$ on the 0.5 iso-surface of f_1 (in yellow) up to its intersection with the 0.5 iso-surface of f_2 (in purple), and on the 0.5 iso-surface of f_2 up to its intersection with the 0.5 iso-surface of f_2. The result is the union of these two surfaces. In addition, $g_{d_c} = 0.5$ on the green line which starts at the intersection of the combined 0.5 iso-surfaces and goes inside the fields f_1 and f_2 following $f_1 = f_2$. This part of $g_{d_c} = 0.5$ is the contact surface that is shown with the same color on f_k in Figure 4.7(b). This line always starts at the intersection of the combined primitives and its length is controlled by d_c. Other iso-lines of g_{d_c} smoothly link values of f_1 to values of f_2 by following the shape of $g_d = 0.5$, resulting in a smooth distance field around $g_{d_c} = 0.5$ in Figure 4.7(a) and thus around $f_k = 0.5$ in Figure 4.7(b).

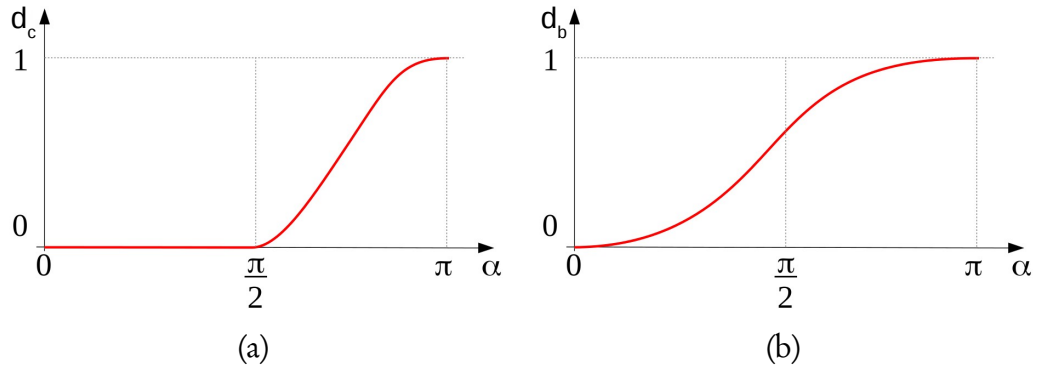


Figure 4.6: (a) Plot of $d_c(\alpha)$ to get contact at joints, and (b) $d_b(\alpha)$ to mimic the inflation of tissues in contact situation.

By designing the same operator, but without contact (i.e., when $d_c = 0$), we obtain the

clean-union operator presented in Figure 4.7(c) whose effect on scalar fields is depicted in Figure 4.1(c). On the other hand, with a contact line of maximal length (i.e., when $d_c = 1$), we get the full contact operator shown in Figure 4.7(d).

Our second operator g_{d_b} is built similarly. It interpolates both the depth of the contact and the inflation of the bulge-in-contact between a clean union operator when $d_b(\alpha) = 0$ (no contact and no bulge, Figure 4.8(a)) and a full contact operator with maximal bulge when $d_b(\alpha) = 1$ (Figure 4.8(c)). Figure 4.8(b) illustrates this operator with intermediate contact length and bulge-in-contact ($d_b(\alpha) = \frac{1}{2}$). The bulge-in-contact is performed by the inflation of the 0.5 iso-lines of f_1 and f_2 in Figures 4.8(b,c). During the animation of a finger, the parameter $d_b(\alpha)$ is set as illustrated in Figure 4.6(b). As we can see, in that case, the bulge starts at low bending angles. Results produced by this operator are illustrated in Figure 4.24.

In practice, the composition operator g_{d_c} is applied by default at each skeleton joint, but the user can select g_{d_b} if desired. The binary composition tree (where implicit primitives are the leaves and composition operators are the nodes) is then built as follows: We start by creating nodes that link pairs of primitives with bulge-in-contact composition g_{d_b}. This forms the bottom of the tree. Then all the other nodes are added on top using gradient-based contact g_{d_c}, until all primitives are combined within a single tree. This specific composition enables us to achieve both the required bulge surfaces at fingers and the contacts we are looking for.

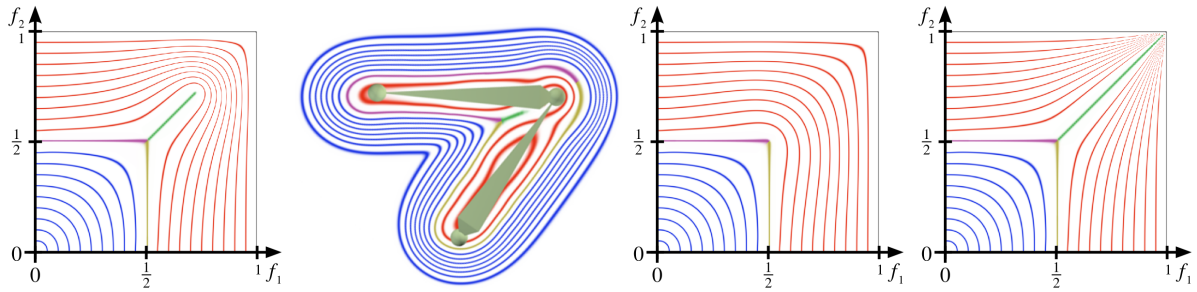


Figure 4.7: (a) Our contact operator $g_{d_c}(f_1, f_2)$ for an intermediate value of d_c and (b) its application to the composition of two field functions articulated by a joint. In all figures, the contact surface is in green. (c) Clean-union operator ($d_c = 0$, no contact) and (d) full contact operator ($d_c = 1$).

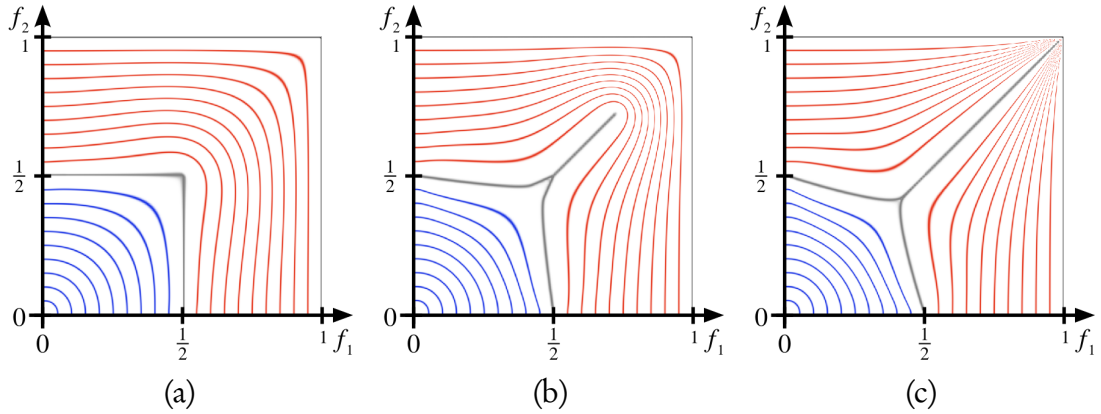


Figure 4.8: Our gradient-based bulge-in-contact operator g_{d_b}. It interpolates between (a) a clean union ($d_b(\alpha) = 0$ no contact and no bulge) and (c) a full contact with maximal bulge ($d_b(\alpha) = 1$). In (b), this operator for an intermediate value of d_b.

4.5.1 Bi-harmonic evaluation

Since our scalar fields f_i have compact support, they are bounded in space and thus, can be stored in 3D grids enabling fast evaluation. It is the same for composition operators: we pre-compute and store them in 3D grids for the three entries (f_1, f_2, d).

Most advanced composition operators, such as those based on gradient blending, have very complex closed form equations, so some of them must be evaluated numerically, for instance, using binary search.

Building new composition operators this way is a long and tedious task: one must derive intricate equations for each new operator and find the appropriate numerical evaluation method. Fortunately, in the case of skinning, only C^1 continuity is required for composition operators. This enables us to propose a simpler and more general way of constructing C^1 operators g directly into a discrete grid.

Our solution is to use bi-harmonic interpolation of some constraints: that is, the function g must satisfy the fourth-order bi-Laplacian equation $\Delta^2 g = 0$, where Δ is the Laplacian differential operator, while matching constraints at specific input values, as those listed in the previous section. This choice ensures a minimal continuity of order C^1. We directly solve this partial differential equation for each 2D slice (f_1, f_2) of the grid using a standard finite difference discretization. We use Dirichlet constraints [Olv13] on the system boundaries, i.e. we preset values of g at the boundaries. The grid boundaries constraints are set according to the specific boundary properties of composition operators expressed by Canezin et al. [CGB13] (Equation 4.3) and inner grid constraints are placed along the pro-

file curve defining the 0.5 iso-curve of $g(f_1, f_2)$ (Equation 4.4). Formally, we set:

$$g(f_1, 0) = f_1,\ g(0, f_2) = f_2,\ g(f_1, 1) = g(1, f_2) = 1, \tag{4.3}$$

and

$$g(f_1, f_2) = 0.5 \quad \forall (f_1, f_2) \in \text{profile curve}. \tag{4.4}$$

In addition to these Dirichlet boundary conditions, the normal derivatives have to be constrained to zero on the $g(f_1, 0)$ and $g(0, f_2)$ axes to ensure a C^1 continuity at the boundaries of g.

Even though bi-harmonic interpolation does not guarantee the monotonicity of the function variation between constraints, we found the practical results satisfactory in our case without having to resort to more involved quadratic programming solvers [JBPS11, JWS12].

Our different operators are thus generated by constraining $g = 0.5$ along a profile curve. For the contact operator g_{d_c}, the profile curve is composed of three line segments (in purple, yellow and green in Figure 4.7). The purple and yellow lines defined as the linear interpolation of the points $(0, 0.5), (0.5, 0.5)$ and $(0.5, 0.5), (0.5, 0)$ are fixed in all slices $(f_1, f_2)\ \forall d \in [0, 1]$. The green line is the linear interpolation of the points $(0.5, 0.5)$ and $(0.5 + \frac{d}{\sqrt{2}}, 0.5 + \frac{d}{\sqrt{2}})$.

Aside from the operator g_{d_c}, the profile curve is in general not aligned with grid vertices (e.g., bulge operator). We constrain the grid vertices of the intersected grid cells with the shortest signed distance to the profile curve.

The bulge-in-contact operator g_{d_b} is constructed following exactly the same procedure (see Figure 4.8). The only difference are the purple and yellow profile curves that are defined with cubic B-splines whose control points are interpolated with respect to d_b, between an aligned position when $d_b = 0$ and the maximal bulge when $d_b = 1$. The number and positions of control points can be customized by the user to obtain the desired bulge shape. To expose less parameters to the user, we fixed the control points: $(0, 0.5)$, $(0.15, 0.45)$, $(0.39, 0.40)$ $(0.44, 0.44)$ and $(0.5, 0.5)$ which showed satisfactory results on the finger joints.

Using this evaluation procedure, our composition operators are pre-computed once for all in 3D grids and at run time, a simple texture fetch is required for each evaluation. More generally, this approach enables the design of free-form composition operators: one only needs to specify a profile curve to generate a new operator.

4.6 Iso-value tracking

4.6.1 Overview

The tracking algorithm aims at driving the character deformations given the previously built field-function f. The algorithm projects vertices onto the field f. As the implicit surface may model skin contacts in the folds of the limbs or other effects such as bulges, the projection step will ensure the deformation captures all those effects. In addition, we must ensure parametric distorsion introduced by the projection step are minimized.

We developed two types of tracking algorithms a history independent and history dependent algorithm. Both algorithms posses their own strength and weakness which we summarize below.

History independent algorithms are easier to parallelize per frame and fit better to standard animation pipelines, however, they are prone to flickering since time coherency is harder to guarantee without knowing the previous and next frame. Such algorithm will need to deform the mesh only based on its rest pose. In this version [VBG+13] we use a computationally inexpensive geometric skinning: the dual quaternion skinning [KCvO08]. Once the mesh is deformed with **DQS**, the field functions f_i are rigidly transformed and the mesh vertices march following the gradient of f (Figure 4.1(f)) until they reach their original field value, thus producing the final deformation (Figure 4.1(g)).

History dependent algorithms are difficult to parallelize per frame, therefore they are less straightforward to implement in standard animation pipelines. Nevertheless, such algorithms are usually less prone to flickering and more robust to extreme deformations. Time dependency leaves more room for realistic effects since the physic of time dependent phenomena can be simulated. Because we rely on the previous frame information, this version [VGB+14] deforms the character without the help of a geometric skinning and avoids the tedious skinning weights that come with it. Re-projecting the vertices on the implicit surface f can be achieved the same way we do with the history independent approach. To guarantee the mesh always comes back to its original shape when the skeleton returns to its rest pose, we developed a relaxation scheme which *globally* minimize the parametric distortions of the mesh while deformed.

4.6.2 History independent tracking

The history independent version is part of our framework called *implicit skinning* [VBG+13]. We give a first glimpse of this approach Chapter 1 in Figure 1.2 of our introduction. Note that additional materials such as a 5 minute video including *results* and a *brief summary* can be found on-line[1].

4.6.2.1 Details

During animation, the implicit primitives f_i associated with each bone are rigidly transformed and combined as explained in the previous section, resulting in a time-dependent global field f. Directly deforming the mesh by tracking the 0.5-isosurface using the field function derivatives [SS11] would result in a smooth distorted mesh after several transformations. Our approach is rather to use the whole region where f is non-zero as an embedding volume for the mesh, used to control its deformations from a standard geometric skinning pose. This enables us to have the mesh track the implicit deformations at low cost, while not losing any surface detail.

Dual quaternions [KCvO08] are used to compute an initial position of the mesh vertices at each animation step. The challenge then consists in designing a fast projection operator to correct vertex positions which preserves the details of the mesh at rest, while tracking the current implicit deformation modeled by f. Our solution makes use of projection in gradient field directions, tangential relaxation, and local Laplacian smoothing, as detailed below.

Vertex projection. During deformation, we project each vertex $\mathbf{v}_i$ back onto its original iso-value iso_i (with $iso_i = f(\mathbf{v}_i)$ in the rest pose), enabling us to account for both the detailed mesh shape and for the current deformation. This is done using Newton's method, which is both faster and more robust than the use of a constant step size for the gradient descent:

$$\mathbf{v_i} \leftarrow \mathbf{v_i} - \sigma\,(f(\mathbf{v_i}) - iso_i)\,\frac{-\nabla f(\mathbf{v_i})}{||\nabla f(\mathbf{v_i})||^2} \quad . \tag{4.5}$$

Using $\sigma = 0.35$ is an effective compromise between speed of convergence and accuracy.

In order to prevent any self-intersection of the final surface (Figure 4.9(a)), as well as to preserve the regularity of the mesh, the projection has to be stopped at the contact surface between different skin parts (Figure 4.9(b)). Although f models contact surfaces as the 0.5-

[1] http://rodolphe-vaillant.fr/permalinks/implicit_skinning_project.php

isosurface, this was not the case of our earliest work with implicit skinning [VBG+13]. In this early work, contact surfaces were modeled in the scalar with gradient discontinuities since we used the maximum operator to blend the primitives f_i. To detect collisions using the gradient discontinuity we used the following conservative heuristic: during projection, we keep track of the angle between the gradient at two consecutive iterations, and stop when it is greater than 55°, thus considering we are on the contact surface (see Figures 4.1(g) and 4.9(b)).

(a) (b)

Figure 4.9: Section of the Armadillo's knee of Figure 4.10 showing the inside part of the mesh. The blue area is the interior side of the mesh and the clear gray its outside. The white lines are the mesh boundaries produced by the cut. As we can see in (a), LBS generates self-intersections in the fold that are converted in (b) into a contact region using our projection.

In our more recent work [VGB+14] we show it is more robust to model contact surfaces with the 0.5-isosurface as detailed section 4.5. In this case, detecting the gradient discontinuity performs as well and give similar results. Alternatively, when the contact iso surface is properly modeled, we found we can stop vertives when we detect a change of monotonicity in the scalar-field while marching.

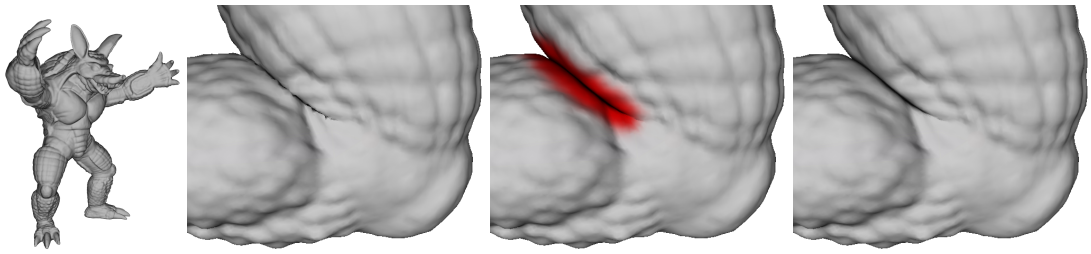
Tangential relaxation. Since the initial vertex positions provided by geometric skinning might be far away from their final position, simply using the previous method could introduce high distortions of mesh faces, and at the extreme cause self-intersections. In order to both minimize them and improve the march of the vertices in the gradient field, we interleave projection steps with tangential relaxation steps, which move each vertex towards the weighted centroid of its neighbours. More precisely, given a vertex $\mathbf{v}_i$, let $\mathbf{q}_{i,j}$ be its one-ring neighbors projected onto its tangent plane. As a preprocess, we compute the barycentric coordinates $\Phi_{i,j}$ such that $\mathbf{v}_i = \sum_j \Phi_{i,j}\, \mathbf{q}_{i,j}$, using the mean value coordinate technique [HF06]. Each relaxation step moves the vertices tangentially to the local surface

using:

$$\mathbf{v}_i \leftarrow (1-\mu)\mathbf{v}_i + \mu \sum_j \Phi_{i,j}\, \mathbf{q}_{i,j}\,, \tag{4.6}$$

where $\mu \in [0,1]$ controls the amount of relaxation. While by construction, this has no effect on the rest pose, it improves the distribution of the vertices after deformation. A relaxation step is applied after each pair of vertex projection steps with an adjusted value of μ, such that the displacement of the vertices decreases with the distance to their target isosurface. This is done by setting:

$$\mu = \max\left(0, 1-(|f(\mathbf{v}_i)-iso_i|-1)^4\right)\,. \tag{4.7}$$



(a) (b) (c) (d)

Figure 4.10: Animation of the Armadillo's knee (a). (b) The result of the projection without smoothing. (c) In red, the smoothed area where $\beta_i > 0$ and (d) the result after a few steps of Laplacian smoothing.

Laplacian smoothing. Even though we assume smooth inputs, a deformation performed by the max operator (union) or our gradient-based contact operator (Figure 4.7) will introduce sharp features at the boundaries of contact regions (Figure 4.10(b)). The latter should be smoothed out in order to mimic realistic skin deformations. We remove these high frequencies by locally applying Laplacian smoothing to the final mesh, at and around contact regions:

$$\mathbf{v}_i \leftarrow (1-\beta_i)\mathbf{v}_i + \beta_i \tilde{\mathbf{v}}_i\,, \tag{4.8}$$

where $\tilde{\mathbf{v}}_i$ is the centroid of the one-ring neighborhood of $\mathbf{v}_i$, and β_i controls the amount of smoothing. This is only done for mesh vertices marked as in contact regions, so that surface details are preserved elsewhere. To this end, we set β_i to 1 for vertices stopped at a gradient

discontinuity, and to 0 for the others, and smooth the β_i values over the mesh by diffusion, prior to applying Laplacian smoothing (Figure 4.10(c)). As illustrated in Figure 4.10(d), this effectively smoothes out the mesh oscillations at sharp features, resulting in organic-looking shapes. Note that this smoothing step has to be performed after the projection step. Therefore it cannot be combined with the relaxation that is interleaved with projection during the march of the mesh vertices.

4.6.2.2 Implementation and results

	vertices	bones	memory	1 joint	animation	
					DQS	Implicit Skinning
Hand	31,750	21	10.5	35	95	15
Armadillo	172,974	23	11.5	36	87	3
Juna	32,056	55	22.5	86	340	15
Dana	4,164	67	32.5	270	> 800	95
Carl	6,866	67	32.5	300	> 800	68

Table 4.1: Implicit skinning benchmarks [VBG+13]. For each model, from left to right: the number of vertices, the number of skeleton bones, the memory consumption for the storage of HRBFs in MB, the frame rates in fps when animating a single joint using our technique, and the frame rates in fps when moving simultaneously most bones in a real animation (poses are shown in Figure 4.11) with dual quaternions and our technique.

Implementation. All our results were generated on a Intel Core i7 3770*K* at 3.5GHz, with 16GB of memory and a Geforce 680. Our CUDA implementation makes extensive use of GPU parallelism. In particular dual quaternion skinning, projection, tangential relaxation and localized Laplacian smoothing are all parallelized over the vertices. The individual field functions (Section 4.4) and composition operators (Section 4.5) being both compactly supported, are respectively sampled into 3D textures of resolution 32^3 (computed from the HRBF equation in 15 ms) and 128^3 with trilinear interpolation, leading to very fast evaluations of f and g_k on the GPU. Our models are composed of 20 to 70 bones, requiring 10Mb to 35Mb of memory for storing the field functions (Table 4.1).

The performances of the implicit skinning method [VBG+13] for animating the different models are summarized in Table 4.1. The frame rates only stand for the deformation time. As we can see, the performance of our technique mainly depends on the number of deformed mesh vertices that are to be projected on the implicit surface. However, the

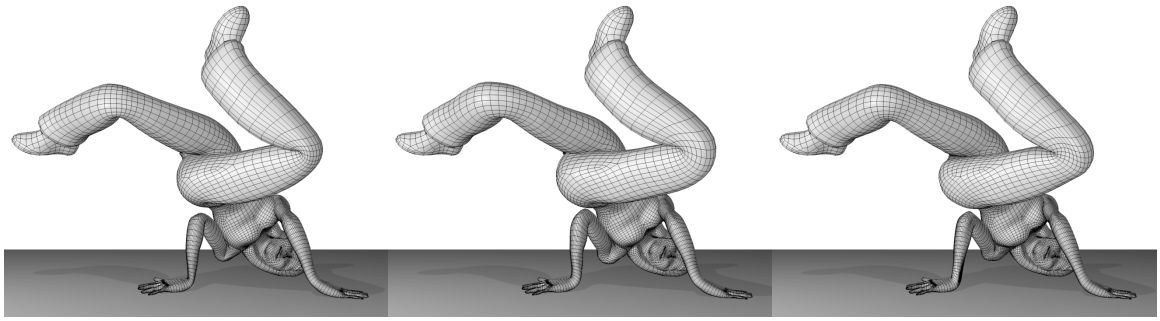
parallel implementation performed on recent GPUs allows us to animate simultaneously most bones of models composed of thousands of vertices at more than 60 fps and models composed of tens of thousands of vertices at more than 12 fps (examples of animation poses are shown in Figures 4.12 and 4.11). Even highly detailed models composed of more than 170,000 vertices are still animated at several fps.

Figure 4.11: Different models skinned with implicit skinning [VBG+13]. Number of bones and vertices, memory and frame rates are given in Table 4.1.

Deformations. We illustrate the use of implicit skinning [VBG+13] with different composition operators, and on several models and poses (Figure 4.11). The union operator is used for the fingers joints in Figure 1.2(c) and the Dana model in Figure 4.12. It allows us to correct the "elbow collapse" effects of the LBS as well as the smooth outgrowth produced by dual quaternions. It also nicely captures the aspect of a solid bone at joints with, inside the bent region, a contact between skin parts as illustrated in Figures 4.12. At bone joints, this makes our approach closer to real skin deformation than other real-time skinning techniques.

Finally, the bulge-in-contact operator can be used to mimic the tissues/fat bulging when fingers bend. This is illustrated in Figure 1.2(d) on phalanx deformations during the hand animation. In all these animations, smooth deformations and contact are adequately generated while the mesh details are preserved, as illustrated on the Armadillo in Figure 4.11, its thigh and calf in Figure 4.10 and on Juna's shoulder in Figure 4.13(c).

Our technique also addresses the candy wrapper effect when bones twist. This is illustrated on a twisted cylinder in Figure 4.14 where we can see that the deformation is adequately corrected with our operators, even when the bones bend. We illustrate our result in Figure 4.14(c).



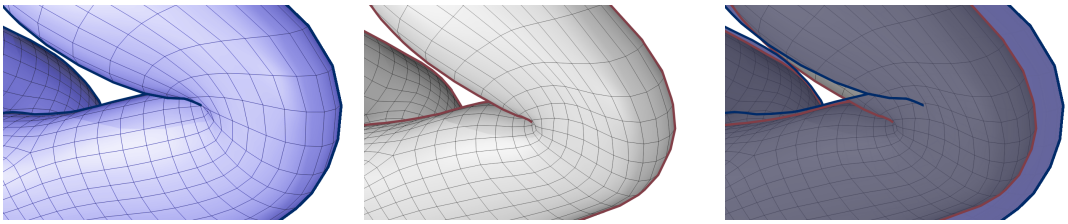


Figure 4.12: Dana model in a break-dance pose. (a) From left to right, the model is deformed with LBS, dual quaternions and our technique. Note the very smooth deformation with visible loss of volume produced by the LBS (left). (b) Close-up on Dana's knee deformed with dual quaternions and (c) after correction with our method using the union operator. (d) The difference is shown by superposing the two surfaces in (b) and (c). [VBG+13]

Parameters and shape control. The parameter settings given for implicit skinning allow the automatic production of results. The parameters used by our method are: the num-

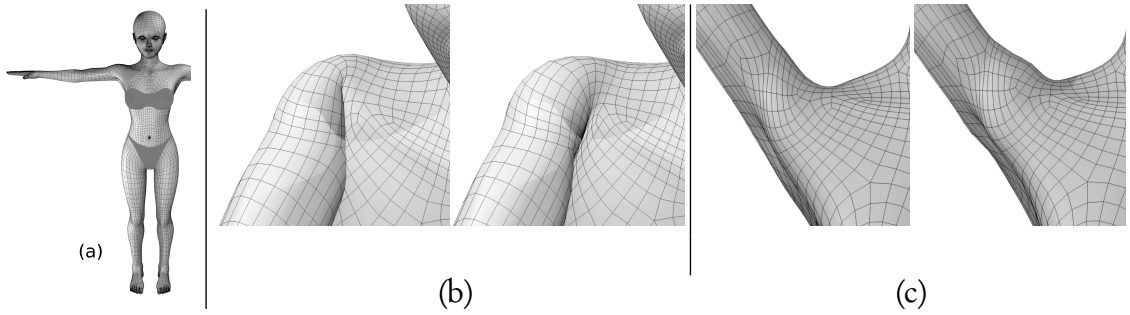


Figure 4.13: Skinning of the Juna model shown in (a). Dual quaternion skinning (left) vs our skinning with gradient-based blending (right) on (b,c) two poses of the shoulder. (b) While geometric skinning produces self intersections, implicit skinning [VBG+13] generates the contact and keep it farther from the joint. (c) Shoulders details are smoothed by geometric skinning and well preserved by our method.

ber of sub-mesh vertices used as HRBF centers for fitting implicit surfaces; the location of closure points (Section 4.4); the choices of the composition operators; the shape of the associated function $d(\alpha)$ when gradient-based composition operators are used (Section 4.5); and finally σ and μ that respectively set the step size of the vertices displacements in the gradient direction and the strength of the tangential relaxation during projection (Section 4.6.2.1).

If not considering the composition operator that is left as a free parameter able to perform a union or a bulge-in-contact, the deformation is controlled by preset parameters. In our technique, the resulting shape is not very sensitive to a slight variation of these parameter values and modifications are predictable. Thanks to the interactive feedback provided by our method, we could easily set these parameter values experimentally in interactive sessions, as illustrated in the accompanying video of implicit skinning [VBG+13] with the adjustment of the bulge size for a finger. Also, when reconstructing the mesh with implicit surfaces, we are seeking for a smooth approximation of mesh parts, independently from the part size and geometry. Fifty to one hundred samples were sufficient for all our models, which are of different resolution and include different levels of detail. Automatically modulating this number according to some given approximation error may however be a useful extension.

In our tests, complex joints such as shoulders may require the user to add or remove a few sample points used as HRBF centers at the vicinity of the joint and slightly adjust the closure point location. Here, the number of points to be added or removed mainly depends on the coherency of the input partitioning solution (Figure 4.1(c)). In the implicit skinning

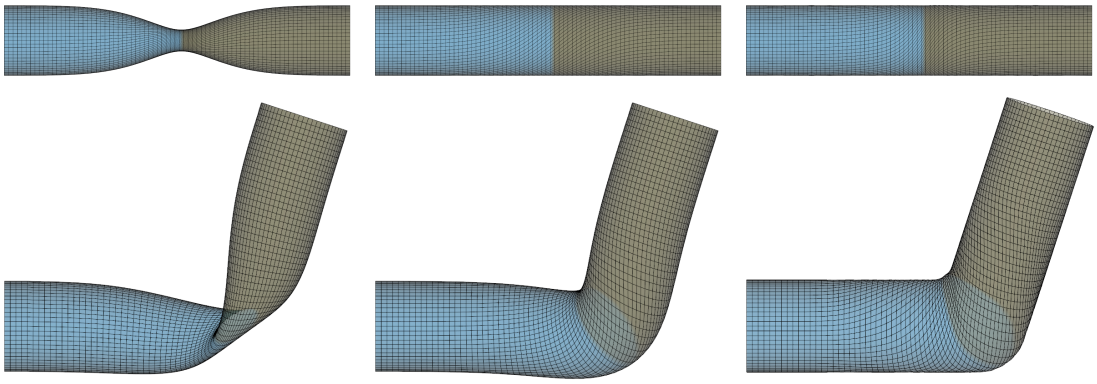


Figure 4.14: Illustration of implicit skinning for modeling smooth skin surfaces at joints (new gradient-based contact operator), when the bones respectively twist (top row), or twist and bend (bottom row). (a) LBS, (b) dual quaternions, (c) our technique with gradient-based composition.

framework, this action is again performed with real-time feedback.

By default, the composition operator is set to the union for all the model bone joints, as was done for the Dana model (Figure 4.12). The result is already a more plausible deformation than the one provided by LBS, dual-quaternions or the more recent elasticity-inspired deformer [KS12] thanks to the rigid bone aspect generated outside the fold and to the contact produced inside. The deformation can then benefit from the gradient-based contact operator at elbows, shoulders and knees, or from the bulge-in-contact operator at fingers. These pre-set operators are provided to the user: he only has to select among a list of deformations for each joint, for instance, a sharp deformation with the union operator or a bulging deformation with the bulges-in-contact operator.

4.6.2.3 Discussion and limitations

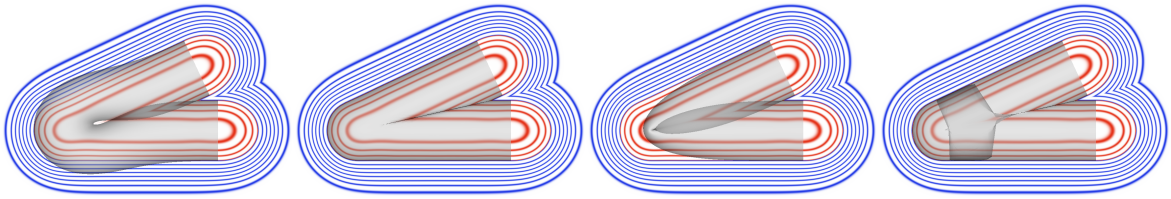


Figure 4.15: The field of a bent cylinder, reconstructed with a union operator. In red, the part inside the 0.5-isosurface, and in blue the outside. The mesh is shown in transparency. (a) The mesh deformed using dual quaternions is (b) adequately corrected by implicit skinning [VBG+13]. However, (c) the mesh deformed with LBS, has sets of vertices crossing each other, and located beyond the bones. (d) It leads to many projections in the wrong direction where our method fails.

Influence of the initial geometric skinning solution. At each animation step, the mesh is first deformed using a standard geometric skinning, before being corrected through the projection of its vertices to the adequate isosurfaces. Therefore, the quality of our results is affected by the choice of the initial skinning solution. In particular, our correction gives better results if the diffusion of the bone influences is large enough to produce a smooth initial deformation. The geometric skinning used should also avoid the apparition of deep self-intersections which would send some mesh vertices beyond the bones. In that case, these vertices would be projected on the opposite surface, as illustrated with the union operator in Figure 4.15. We can observe smooth dual quaternion skinning deformations, enable an adequate correction (Figure 4.15 (a) and (b)), while the input **LBS** deformations

generate deep self-intersections (Figure 4.15 (c) and (d)), causing a failure of our method. Also, in extreme poses, even though contact is handled, the deformation may be inaccurately corrected: some mesh vertices may cross a contact and be stopped, while they were supposed to be projected on the visible skin part, outside the fold.

Smooth skin parts. The neck of a giraffe, the elephant's trunk or the arms of an octopus smoothly deform when they are animated. This effect is not handled by our method which focuses on more rigid bone-like joints. In this case, the smoothing property of LBS near a joint, usually seen as an artifact, may produce a result closer to the expected shape. Alternatively a curved-based skining method may be more appropriate.

Time independence. At each animation step, the initial mesh vertex positions are computed from the rest pose after a geometric skinning deformation. Therefore, the final vertex positions at a given animation step do not depend on their positions at the previous step. The consequences are two-fold. On the one hand, the solution might be subject to flickering. In practice, as soon as the mesh resolution is dense enough to capture the local shape at joints, the continuity in time of the initial solution computed with geometric skinning coupled with the smoothness of the field functions prevent this effect. On the other hand, this eases the integration of our approach in standard animation pipelines where frames are computed in parallel.

Mesh resolution. As mentioned in the previous paragraph, the mesh resolution should be dense enough to capture features like contact, blend or bulge. When the mesh has a too coarse resolution, a very few vertices lie on the different features introduced by our skinning technique at joints. Thus, they are not appropriately represented and the relaxation followed by the local smoothing (at the end of the projection) are likely to degrade the vertices distribution, especially if the mesh is very irregular. This can make the mesh deformation unrealistic and discontinuous in time.

4.6.2.4 Summary

We presented implicit skinning [VBG+13] an approach for skinning virtual characters, which operates in real-time. Based on the embedding of the skin mesh into a deformable implicit volume, made of parts rigidly attached to each bone, it does not suffer from the loss of volume inherent to standard geometric skinning methods. Choosing the way the implicit parts are combined enables us to generate contact surfaces between neighboring

skin parts, and to drive organic bulging effects. As a result, our approach achieves visually plausible deformations of the different joints in a character's body, even for large bending angles. Since no optimization or collision processing steps are required, our approach is robust and efficient. Lastly, computations being independent from one frame to the next, the method perfectly fits into standard animation pipelines.

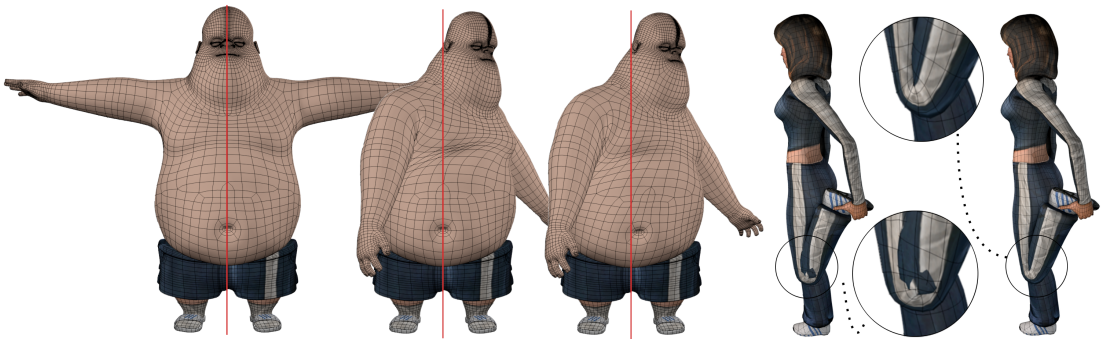


Figure 4.16: (a) The Jeff model in rest pose. Its shoulders are rotated and skinned with (b) implicit skinning [VBG+13] which locally deforms the mesh according to some preset weights and (c) elastic implicit skinning [VGB+14] which automatically produces a plausible skin elasticity (notice how the belly button stretches). On the right, the Dana's knee is bent with an extreme rotation angle. (d) Implicit Skinning fails to handle deep self-intersections while (e) elastic implicit skinning allows large bending angles and the self-intersection at the knee is handled correctly.

4.6.3 History dependent tracking

We published elastic implicit skinning [VGB+14] a history dependent version which is a more robust alternative to the implicit skinning [VBG+13]. We give a first peek of this approach Figure 4.16. This version introduced more advanced and more flexible blending operators (presented in Section 4). In addition, a new tracking algorithm which avoids the need for a geometric skinning was presented. Note that additional materials such as a 5 minute video including *results* and a *brief summary* can be found on-line[2].

The main goal of elastic implicit skinning was to get rid of the shortcomings present in the previous implicit skinning algorithm, most of the problems came from the need of an initial solution (i.e. dual quaternion skinning) for the vertex position. This meant, we had to rely on skinning weights and their tedious definition. For large bending angles, implicit skinning could not properly project the vertices of the initial solution (Figure 4.16 (c)). In addition, mesh distortions introduced by the initial solution were not sufficiently minimized (Figure 4.16 (b)).

By designing a history dependent tracking algorithm we were able to avoid the need for an intial solution. Our tracking method is based on a linear relaxation energy (inspired from the as-rigid-as-possible energy) and updates the skin mesh with a plausible tangential distribution of vertices, as shown in Figures 4.16(c) and 4.2(c) This provides more automa-

[2]http://rodolphe-vaillant.fr/permalinks/elastic_implicit_skinning_project.php

tion and robustness than implicit skinning while still being applicable at interactive frame-rates. The solution presented in this section captures skin contact and elasticity, as well as muscle inflation and skin sliding effects.

4.6.3.1 Details

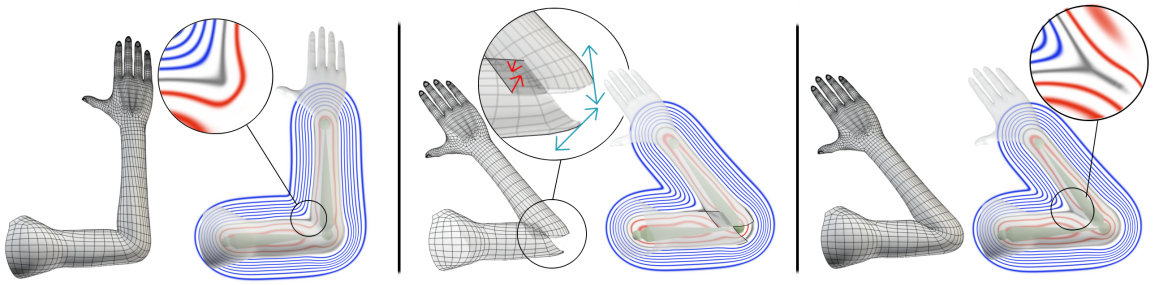


Figure 4.17: (a) (b) (c) We illustrate one step of our animation algorithm During animation, (a) the deformed mesh parts at an initial step n (b) are rigidly transformed at the step $n+1$. Then, vertices are projected onto their own iso-value $f(\mathbf{v})$ (red arrows) and tangentially relaxed (blue arrows). (c) The final result with the contact surface illustrated on the mesh and in grey in the implicit skin.

One of the main challenge of our general implicit-skinning approach is to ensure that the initial mesh properly follows the iso-surface defined by the time-varying global scalar field f. Even though f is usually obtained through the composition of implicit primitives f_i which are rigidly transformed, the underlying implicit skin exhibits severe non-linear deformations around the joints. User defined non-linear deformations might also be applied to the f_i to mimic some complex effects (see section 4.6.3.2). Moreover, in our context the mesh embeds many spatially varying data such as relief details and texture layers. Therefore, the topology and density of the mesh cannot be arbitrarily changed, and each vertex must carefully track its position on the implicit surface to avoid unnatural distortions.

To this end, we propose to track the iso-surface incrementally from one animation step to the next using a three stage procedure:

1. Each vertex is moved by applying the transformation difference of its nearest bone, as defined by the initial mesh partitioning (Figure 4.17 (b)).

2. Vertices are projected onto their respective iso-value (derived from the rest pose) along the gradient direction using Newton iterations.

3. This yields a good initial guess on which we can apply a tangential relaxation scheme to account for the stretch of the implicit skin while reducing distortions.

Since stages one and two are already described in the implicit skinning section 4.6.2, the rest of this section focuses on the last stage.

Tangential relaxation energy The relaxation scheme must reduce the distortions while maintaining the mesh on the underlying surface. Because an accurate physical simulation of the skin would be prohibitively expensive, we seek the simplest and fastest relaxation scheme able to track the implicit skin robustly. Among our various attempts, our first test was to make use of the relaxation scheme from the implicit skinning method (i.e. based on mean value coordinate). Experiments showed when applying implicit skinning incrementally, this relaxation scheme would not preserve triangle areas. As a result, the tracking would fail due to large stretching of the mesh even for moderate bending angles. We thus derived a novel relaxation scheme inspired by Sorkine et al. [SA07] as-rigid-as-possible (ARAP) energy:

$$E(\{\hat{\mathbf{p}}_i\}) = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 , \tag{4.9}$$

where $\mathbf{p}_i$ are the vertex positions in rest pose, $\hat{\mathbf{p}}_i$ are the unknown final positions, $\mathbf{R}_i$ is a 3×3 rotation matrix, and $\mathcal{N}(i)$ denotes the one-ring neighborhood of vertex i. The weights w_{ij} are classically defined from the cotangent formula [MDSB02], i.e., $w_{ij} = \frac{1}{2}(\cot(\alpha_{ij}) + \cot(\beta_{ij}))$, where α_{ij}, β_{ij} are the angles opposite to the edge (i, j).

As motivated and detailed below, in our relaxation scheme the rotations $\mathbf{R}_i$ are prescribed a priori taking advantage of the specificity of skinning deformations. Indeed, the energy E is by construction invariant to translation while the rotation $\mathbf{R}_i$ locally cancels rigid transformations. In most applications of this kind of energy, including iso-surface tracking [BN07], $\mathbf{R}_i$ is computed from the polar decomposition of the Jacobian of the unknown deformation. In addition to being an expensive process, this strategy exhibits several drawbacks. Firstly, this inserts a non-linear component in the energy, E which thus requires additional iterations to be minimized while introducing local minima. In addition, as illustrated in Figure 4.18(a,b), there is no smoothness constraints on the target rotations $\mathbf{R}_i$ and the result is only C^0 at the constraint boundaries. Moreover, we observed during animation that vertices appear to slide over the implicit skin instead of behaving as if the skin had elasticity, resulting in an unrealistic animation of the skin.

Finally, as depicted in Figure 4.18(d), this strategy is well known to be subject to triangle inversions which are very tedious to avoid, even on a planar domain [SKPSH13].

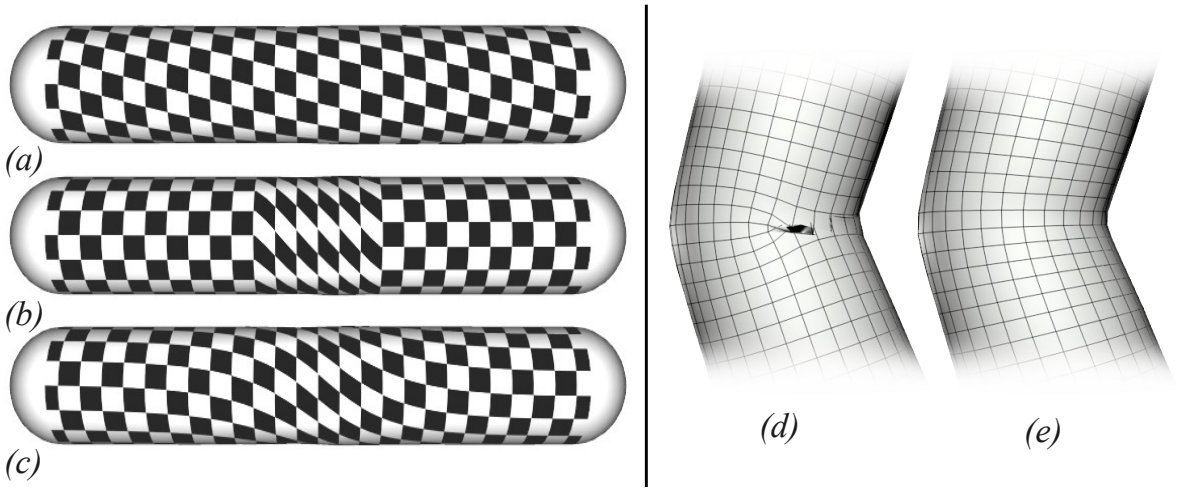


Figure 4.18: Standard ARAP energy: (a), (b) and (d) are compared to our dual-quaternion based rotation targets (c) and (e). In (a) only the extremities of the tube are fixed while in (b) a larger part is fixed. Both cases lead to C^0 deformations on the constraints with an unplausible distribution of vertices inside the domain. In (d) triangle inversions occurred. In (c) and (e) these problems are avoided and our technique produces an adequate result.

We address all these drawbacks by fixing *a priori* the target rotations $\mathbf{R}_i$. In order to leverage a coherent behavior over time and to recover the rest-pose, these rotations must not depend on the positions of the mesh vertices of the current animation step but only on the rest pose and affine transformations of the bones.

Following the first stage of our tracking procedure, a first choice consists of using the rotation associated with the nearest bone of vertex i for $\mathbf{R}_i$. As illustrated in the first row of Figure 4.21, this strategy yields promising results despite the discontinuous nature of the underlying rotation field. An effective solution thus consists of performing a blend of the bone transformations as in standard geometric skinning methods. In this research we use the dual-quaternion interpolation scheme [KCvO08] with default harmonic weights [BP07]. As discussed in more details in Section 4.6.3.2 and illustrated in Figure 4.21, we emphasize that both the choice of the blending technique and weights have only subtle effects on the final distribution of the vertices, and they do not affect the final shape or robustness of the tracking. Figures 4.18(c,e) show the superiority of our approach that better localize continuous tangential deformations and prevents triangle inversions.

On-surface constraints The energy E has to be minimized while maintaining the vertices to their own field value. This is a non-linear constraint which implies an iterative scheme in which $\hat{\mathbf{p}}_i$ is obtained through a sequence $\hat{\mathbf{p}}_i^0, \hat{\mathbf{p}}_i^1, \ldots$, where $\hat{\mathbf{p}}_i^0$ corresponds to the initial projection onto its respective iso-value, as obtained through the aforementioned second stage of the tracking system. At each iteration of the minimization, we first locally linearize this constraint by enforcing each vertex to lie on its respective current tangent plane. This can be accomplished by writing the unknown position $\hat{\mathbf{p}}_i^{k+1}$ as:

$$\hat{\mathbf{p}}_i^{k+1} = \hat{\mathbf{p}}_i^k + \begin{bmatrix} \mathbf{u}_i^k & \mathbf{v}_i^k \end{bmatrix} \begin{pmatrix} u_i^{k+1} \\ v_i^{k+1} \end{pmatrix}, \tag{4.10}$$

where $\mathbf{u}_i^k$, $\mathbf{v}_i^k$ are two arbitrary tangent vectors of the current iso-surface at $\hat{\mathbf{p}}_i^k$, and u_i^{k+1}, v_i^{k+1} are the 2D coordinates of $\hat{\mathbf{p}}_i^{k+1}$ in this local frame. Minimizing E under this constraint is still a linear and very sparse problem. Since there is no need to completely solve this problem before projecting the solution on the surface and iterating, we interleave one Jacobi iteration minimizing E with one Newton iteration projecting $\hat{\mathbf{p}}_i^{k+1}$ onto its respective field value. At each iteration, an approximation of the local tangent frame is obtained for free from the gradient computed during the Newton iteration.

The choice for Jacobi iterations is motivated by its simplicity to be parallelized on the GPU. Higher performance may be achieved using the conjugate-gradient method which can also be efficiently implemented on GPUs [WBS+13]. In order to guarantee the availability of a very good initial guess $\hat{\mathbf{p}}_i^0$ and to avoid projection issues, the skeleton animation between two successive frames is decomposed into several virtual frames such that the maximal rotation angle of a bone is smaller than a third of the current bending angle. The intermediate skeleton configurations are linearly interpolated. Such a decomposition has little impact on the performance because a smaller animation step leads to a faster convergence.

4.6.3.2 Implementation and results

Similar to our earliest version implicit skinning [VBG+13], the deformation part of our elastic implicit skinning system [VGB+14] is entirely executed on the GPU using CUDA. Parallelization is performed on a per-vertex basis with one kernel responsible for the computation of the initial guesses (rigid transformations and Newton projections) and a second kernel performing one iteration of the minimization (one Jacobi iteration, plus one Newton projection). This second kernel is applied multiple times per animation step until convergence. As previously done, we accelerate the evaluation of the scalar field f, the scalar

	#vertices	#bones	time step (rad)	#Jacobi iterations	full animation	Imp skinning
Hand	31.7k	21	0.5	600	650ms	83ms
Hand	31.7k	21	0.05	330	220ms	83ms
Armadillo	16.4k	43	0.05	170	70ms	23ms
Jeff	13.3k	45	0.5	300	100ms	30ms
Jeff	13.3k	45	0.05	100	50ms	30ms
Jeff	13.3k	45	0.005	10	24ms	20ms

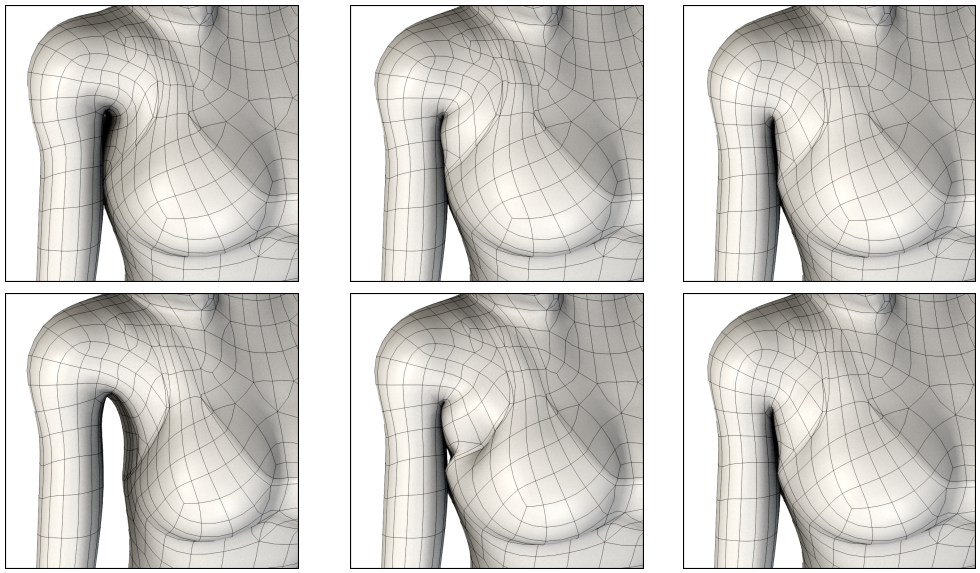
Table 4.2: Statistics and performance of elastic implicit skinning [VGB+14] and implicit skinning [VBG+13] for various models and time steps when all joints are animated simultaneously.

fields f_i by indexing them in a 32^3 grid. Both our composition operators g_k and scalar fields f_i are stored in 3D textures and evaluated using hardware trilinear interpolation.

Results were produced on an Intel Core i5-3570K at 3.4GHz, with 8GB of memory and a Geforce 670 GTX. The memory consumption is exactly the same as for the implicit skinning method: for the models presented in this section, from 10 to 30MB of extra texture memory are required to store the operators and field functions. During an interactive session, joints are usually adjusted one at a time. In this case, deformations occur mostly locally: only a small fraction of vertices have to be projected and relaxed, hence the density of the mesh has a small impact on performance. For a single joint, we thus observed a rate between 30 to 300 frames per second. More detailed statistics and performance results are given in Table 4.2 for complete animations and different time steps. As expected, the number of minimization iterations and thus the computation time highly depends on the magnitude of the skeleton transformation between two animation steps.

Provided a mesh with an animation skeleton, our system automatically produces a default solution: partitioning and field-functions f_i are computed automatically as described by Vaillant et al. [VBG+13], and skin elasticity and self-contacts are automatically and globally resolved during the animation.

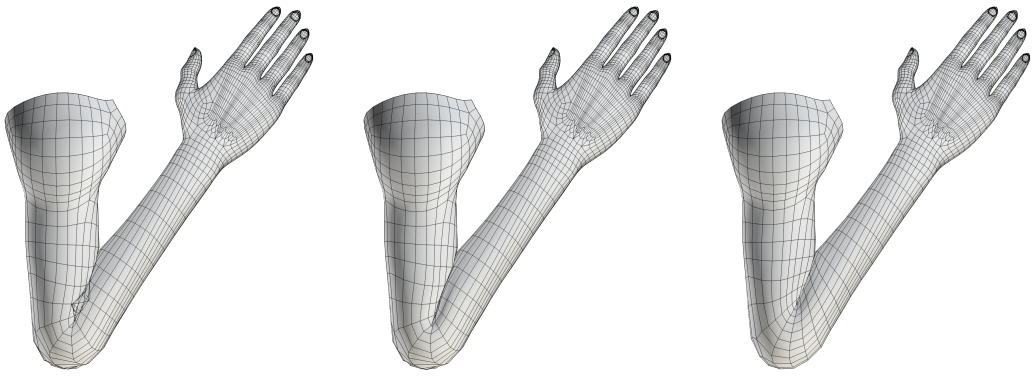
As demonstrated in Figure 4.19, elastic implicit skinning produces satisfactory behavior without requiring any tedious tuning of skinning weights, while previous work still exhibit distortions even after an artist edits the weights manually. The sensitivity of our method with respect to skinning weights is further evaluated and compared on a simpler configuration in Figure 4.21. Recall that in our system such weights are only used to blend the target rotation in the minimization of our tangential energy. Therefore, even large changes in the weights leads only to small differences in the distribution of the vertices, and the shape of



Dual Quaternion Skinning | Implicit Skinning | Elastic Implicit Skinning

Figure 4.19: Comparison of different skinning method using manually edited skinning weights (top), and automatic weights (bottom).

the implicit skin remains unchanged. In contrast, the projection method of the implicit skinning technique fails if the weights are not smooth enough, hence, further editing of the skinning weights is often needed.



(a) Implicit Skinning using a standard union (max) | (b) Implicit Skinning using union with contact | (c) Elastic implicit skinning

Figure 4.20: (a) Without proper modeling of the surface contact, vertices tend to be projected towards the exterior of the contact region thus leading to incorrect projections. Note that this is especially true for skinning weights produce not smooth enough deformations. (b) Our operators address this precise issue, unfortunately, tracking with implicit skinning arbitrarily extends the contact region in an uncontrollable manner. (c) Elastic implicit skinning adequately tracks the skin fold.

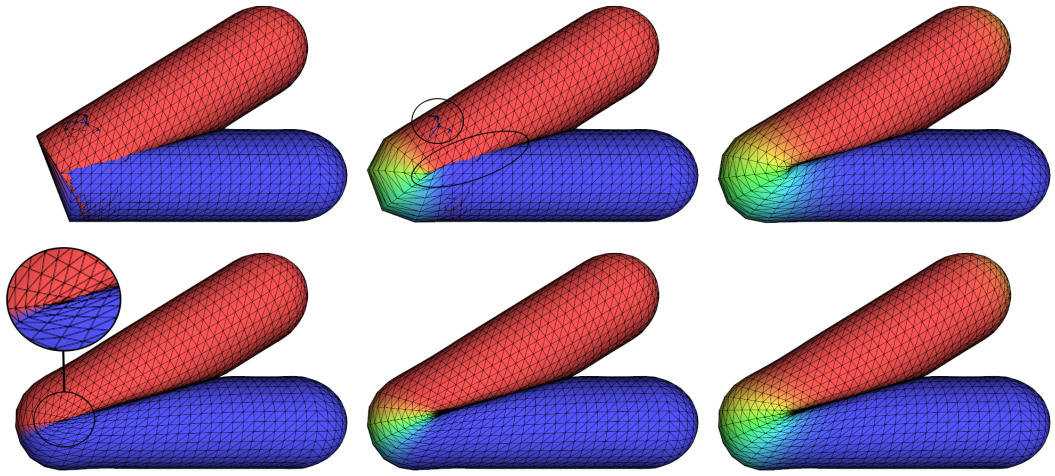
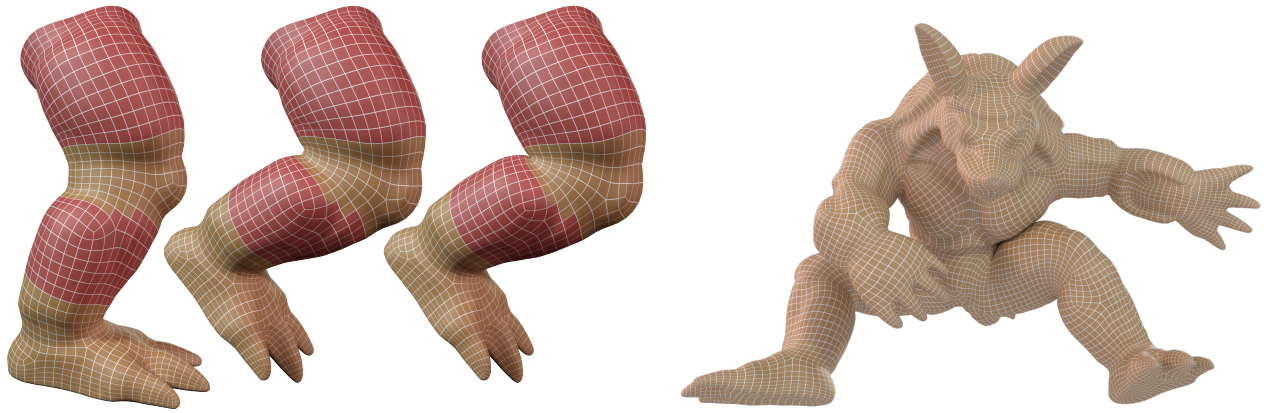


Figure 4.21: Comparison of the sensitivity of the vertex distribution regarding variations of skinning weights (shown in color code) for the implicit skinning (top row) and elastic implicit skinning (bottom row). From left to right, skinning weights are rigid, smooth and smoother respectively.

The global nature of the skin sliding effect produced by our tangential relaxation is especially visible on the character's belly of Figure 4.16-left. In contrast to geometric or implicit skinning, with elastic implicit skinning the belly button is stretched as the torso twists. As shown in Figure 4.22-top, this skin elasticity can be easily controlled by painting regions where the tangential relaxation is not wanted. In such regions, vertices are only rigidly transformed according to their nearest bone.



(a) (b) (c) (d)

Figure 4.22: The Armadillo's knee (a) is bent using (b) our default solution with tangential relaxation applied everywhere, and (c) by disabling the relaxation for red areas. (d) Global self-contact between the belly and the leg.

Global contacts can be observed in Figures 4.22(d) and 4.25. The shape of the deformation can also be adjusted by choosing between different types of composition operator to enable, for instance, subtle bulge in contact where this is desired as on the finger of Figure 4.24. More advanced deformations can also be performed by designing and controlling additional field functions. For instance, in Figure 4.23 the user added an extra implicit primitive with its associated bone to mimic the contraction of a bicep by applying a scaling proportional to the arm bending angle. Such a manipulation would not be possible without a proper tracking system taking care of the skin elasticity and contacts.

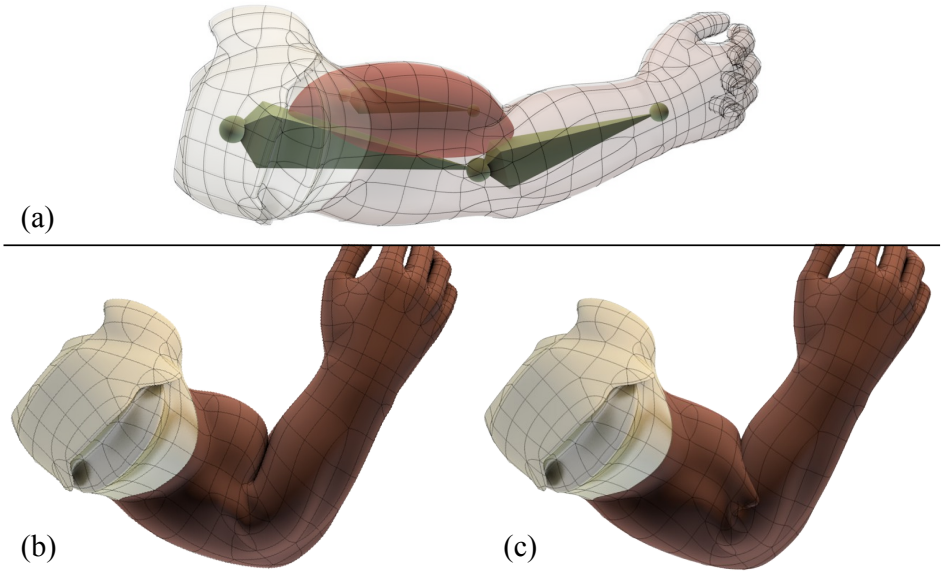
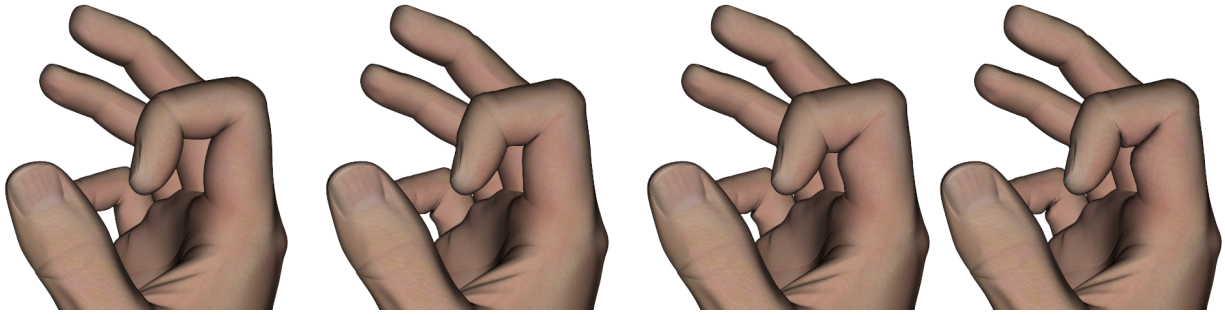


Figure 4.23: Reproduction of the inflation of a bicep by the addition of an extra primitive (a) and its animation using our system (b). Owing to the prominent shape of the muscles, implicit skinning yields projection artifacts (c).

In addition to all these novel possibilities, our system enables us to reach extreme bending angle without introducing artifacts. This can be seen in Figure 4.16(e) for a bending angle around 150 degrees. This high robustness is due to the combination of both a proper modeling of surface contacts and the incremental nature of our novel surface tracking mechanism. Figure 4.20(a) shows that with a standard union operator, implicit skinning exhibits artifacts if it is implemented with skinning weights that are not smooth enough. In Figure 4.20(b) our novel contact operator, used alone in implicit skinning (i.e., without elastic implicit skinning tracking system), improves the projection artifacts but does not reduce mesh distortions. Finally, our novel tracking algorithm (Figure 4.20(c)) enables even larger bending angles, a proper tracking of the fold depth and a reduction of mesh distortions.



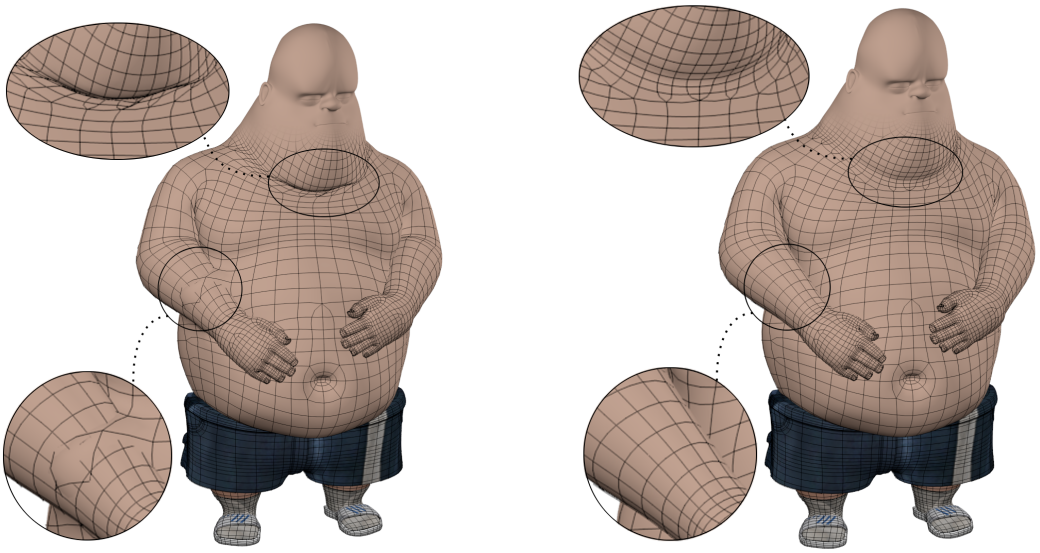
DQS | Elastic implicit skinning with default contact operator | Elastic implicit skinning with bulge and contact | Implicit skinning with bulge

Figure 4.24: Comparison of: dual quaternion skinning, elastic implicit skinning (without bulge and with bulge) and implicit skinning with bulge.

4.6.3.3 Limitations

Owing to the incremental nature of our tracking technique, our approach gets closer to a physical simulation of the skin than the original implicit skinning method. Even though our approach is considerably simpler and faster than physical simulations, it still has some of their drawbacks. Firstly, the result for a given animation step is not entirely determined by the skeleton pose. Because of the non-linearity introduced by the projections on the implicit skin the result might depend on the complete history of the animation. For a similar reason, the result might also depend on the chosen time step. As with any simulation, choosing the right time step is a trade-off between the computation cost and the numerical stability: a very large time step will lead to very poor initial guesses with similar problems as implicit skinning, while a very small time step will be prohibitively expensive.

In practice, as long as no incorrect projection occurs during animation, our solution is almost insensitive to a particular time step. The automatic solution proposed in Section 4.6.3.1 to choose an appropriate time step strives to make sure that the initial guess obtained through rigid transformation of the previous animation-step does not lead to a projection in the wrong direction. The proposed solution is rather conservative as, for instance, we observed that bending an arm from a fully open to a $\pi/2$ angle in a single time-step is correctly handled by our tracking system. Higher performance could thus be achieved by relaxing the proposed heuristic. An implementation could also use large time steps, and dynamically subdivide the animation when incorrect projections or triangle inversions are detected.



Implicit Skinning Elastic Implicit Skinning

Figure 4.25: Comparison of our system to implicit skinning on the model shown in rest pose in Figure 4.16-(a). The arm collides too deeply with the belly to be properly recovered by implicit skinning while the chin is unrealistically creased. These issues are properly addressed by elastic implicit skinning.

4.6.3.4 Summary

In this section, we have presented the elastic implicit skinning [VGB+14] a technique that takes advantage of the best features of implicit skinning, and makes the method more robust and suitable for production pipelines. We showed the importance of properly modeling and controlling contact surfaces using advanced implicit composition operators. We also showed how to track a deformable implicit surface by deriving a linearized relaxation energy from Sorkine et al.'s as-rigid-as-possible energy with some *a priori* knowledge on the expected deformation that we can extract from the animation skeleton. As a result, our skinning system is robust, handles contact even when surfaces are deeply inter-penetrating, and maintains a plausible solution where implicit skinning fails. The method thus reduces the amount of necessary user interaction, while maintaining interactive performance, and enables new features such as skin elasticity.

Chapter 5

Conclusion

In this disertation we explored the joint use of mesh and implicit surface representation in the context of character skinning. We described a novel skinning framework which core idea is to embed a mesh character into a deformable volume, this volume is represented with several scalar fields composed together. We were able to harness the strength of implicit surfaces to produce enhanced deformations of the mesh in real-time. Our deformation includes: skin contact at joints and colliding limbs, skin bulge (e.g. around a bent joint), muscle bulge and skin elasticity.

We divided our framework into three stages: reconstruction, composition and tracking. The reconstruction is responsible for the correct embedding of the mesh into the deformable volume. Composition enables us to introduce advanced deformations such as bulges and contact areas. As for the tracking, it is here to make sure the integrity and details of the mesh are presevered while the deformations of the implicit skin are correctly captured by the mesh.

This work open the door to many different avenues. In the future, we would like to investigate more advanced field combination schemes to model a wider set of deformations. For instance, a hand pushing on the belly should produce a hollow larger than the hand while the hand itself would barely deform. It means we have to define the softness of individual implicit surfaces, this may requires to extend the binary gradient-based composition operators to n-ary compositions. Inspired by the work of Rohmer et al. [RHC09] and Barthe et al. [BGC01], implicit skinning could be directly tuned from profile curves sketched by the user, and depicting the desired skin shape. These profiles curves, possibly including wrinkles such as those that appear when a wrist articulates, would drive the generation of specific gradient-based compositions.

It would be very interesting to use the implicit surfaces to drive the skeleton animation

of the character in addition to its skin deformation. For instance the field values along contact surfaces could be used, either to output contact forces to be applied to skeleton bones, or simply to detect that an angular limit is reached at the joint.

Other secondary effects can also use the extra information provided by the implicit surfaces. For example, cloth simulation could benefit from a dedicated tracking algorithm preventing intersection with the body. Finally, the more general use of our technique on deformable dynamic 3D objects is still to be explored.

Bibliography

[Ale02] Marc Alexa. Linear combination of transformations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 380–387, New York, NY, USA, 2002. ACM.

[AS07] Alexis Angelidis and Karan Singh. Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 129–140, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[BBCW10] Adrien Bernhardt, Loïc Barthe, Marie-Paule Cani, and Brian Wyvill. Implicit blending revisited. *Comput. Graph. Forum*, 29(2):367–375, mai 2010.

[BGC01] L. Barthe, V. Gaildrat, and R. Caubet. Extrusion of 1D implicit profiles: Theory and first application. *International Journal of Shape Modeling*, 7:179–199, 2001.

[BN07] Antoine Bouthors and Matthieu Nesme. Twinned meshes for dynamic triangulation of implicit surfaces. In *Graphics Interface*, Montréal, may 2007.

[BP07] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[BW97] Jules Bloomenthal and Brian Wyvill, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[BWd04] Loïc Barthe, Brian Wyvill, and Erwin de Groot. Controllable binary CSG operators for "soft objects". *International Journal of Shape Modeling*, 10(2):135–154, 2004.

[BWK03] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 862–870, New York, NY, USA, 2003. ACM.

[CGB13] Florian Canezin, Gaël Guennebaud, and Loïc Barthe. Adequate Inner Bound for Geometric Modeling with Compact Field Function. *Computer & Graphics (proceedings of SMI 2013)*, 37(6):565–573, July 2013.

[DdL14] Olivier Dionne and Martin de Lasa. Geodesic binding for degenerate character geometry using sparse voxelization. *IEEE Trans. Vis. Comput. Graph.*, 20(10):1367–1378, 2014.

[DSP06] Kevin G. Der, Robert W. Sumner, and Jovan Popović. Inverse kinematics for reduced deformable models. *ACM Trans. Graph.*, 25(3):1174–1179, July 2006.

[FOKgM07a] Sven Forstmann, Jun Ohya, Artus Krohn-grimberghe, and Ryan Mcdougall. Deformation styles for spline-based skeletal animation. In *In Symp. on Computer Animation*, pages 141–150, 2007.

[FOKgM07b] Sven Forstmann, Jun Ohya, Artus Krohn-grimberghe, and Ryan Mcdougall. Deformation styles for spline-based skeletal animation. In *In Symp. on Computer Animation*, pages 141–150, 2007.

[FTS08] Wolfram Von Funck, Holger Theisel, and Hans Peter Seidel. Volume-preserving mesh skinning. *Workshop on Vision, Modeling and Visualization (VMV)*, 2008.

[GBC+13] Olivier Gourmel, Loïc Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. A gradient-based implicit blend. *ACM Transactions on Graphics*, 32(2), 2013.

[GLC+11] Olivier Gourmel, Barthe Loïc, Marie-Paule Cani, Wyvill Brian, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. Gradient-based implicit modeling. To be published in ACM TOG journal, January 2011.

[GPP+10] Olivier Gourmel, Anthony Pajot, Mathias Paulin, Loïc Barthe, and Pierre Poulin. Fitted BVH for Fast Raytracing of Metaballs. *Computer Graphics Forum, Eurographics 2010 Proceedings*, 29(2):281–288, mai 2010.

[HF06] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transaction on Graphics (TOG)*, 25(4), 2006.

[HYC+05] Dae-Eun Hyun, Seung-Hyun Yoon, Jung-Woo Chang, Joon-Kyung Seong, Myung-Soo Kim, and Bert Jüttler. Sweep-based human deformation. *The Visual Computer*, 21(8-10):542–550, 2005.

[JBK+12] Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 31(4):77:1–77:10, 2012.

[JBPS11] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 30(4):to appear, 2011.

[JDKL14] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, 2014.

[JS11] Alec Jacobson and Olga Sorkine. Stretchable and twistable bones for skeletal shape deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 30(6):165:1–165:8, 2011.

[JWS12] Alec Jacobson, Tino Weinkauf, and Olga Sorkine. Smooth shape-aware functions with controlled extrema. *Computer Graphics Forum (proceedings of EUROGRAPHICSACM SIGGRAPH Symposium on Geometry Processing)*, 31(5):1577–1586, 2012.

[KCvO08] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27:105:1–105:23, November 2008.

[KH14] YoungBeom Kim and JungHyun Han. Bulging-free dual quaternion skinning. *J-COMPUT-ANIM-VIRTUAL-WORLDS*, 25(3–4):323–331, May 2014.

[KJP02] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 153–159, New York, NY, USA, 2002. ACM.

[KMD+07] Ladislav Kavan, Rachel McDonnell, Simon Dobbyn, Jiří Žára, and Carol O'Sullivan. Skinning arbitrary deformations. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, pages 53–60, New York, NY, USA, 2007. ACM.

[KS12] Ladislav Kavan and Olga Sorkine. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 31(6):to appear, 2012.

[Kv05] Ladislav Kavan and Jiří Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 9–16, New York, NY, USA, 2005. ACM.

[LAG01] Antoine Leclercq, S. Akkouche, and E. Galin. Mixing triangle meshes and implicit surfaces in character animation. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 37–47, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. 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*, SIGGRAPH '00, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[LLS+13] Gene S. Lee, Andy Lin, Matt Schiller, Scott Peters, Mark McLaughlin, and Frank Hanner. Enhanced dual quaternion skinning for production use. In *ACM SIGGRAPH 2013 Talks*, SIGGRAPH '13, pages 9:1–9:1, New York, NY, USA, 2013. ACM.

[MDSB02] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proc. VisMath*, pages 35–57, 2002.

[MG03] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 562–568, New York, NY, USA, 2003. ACM.

[MGV11] I. Macêdo, J. P. Gois, and L. Velho. Hermite radial basis functions implicits. *Computer Graphics Forum*, 30(1):27–42, 2011.

[MMG06a] Bruce Merry, Patrick Marais, and James Gain. Animation space: A truly linear framework for character animation. *ACM Trans. Graph.*, 25:1400–1423, October 2006.

[MMG06b] Bruce Merry, Patrick Marais, and James Gain. Normal transformations for articulated models. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[MTG03] Alex Mohr, Luke Tokheim, and Michael Gleicher. Direct manipulation of interactive character skins. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, I3D '03, pages 27–30, New York, NY, USA, 2003. ACM.

[MTLT88] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface '88*, pages 26–33, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[MZS+11] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 37:1–37:12, New York, NY, USA, 2011. ACM.

[NTH01] Victor Ng-Thow-Hing. *Anatomically-based models for physical and geometric reconstruction of humans and other animals*. PhD thesis, Toronto, Ont., Canada, Canada, 2001. AAINQ58941.

[Olv13] P. Olver. *Introduction to Partial Differential Equations*. Undergraduate Texts in Mathematics. Springer International Publishing, 2013.

[PASS95] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[Reu03] Patrick Reuter. *Reconstruction and Rendering of Implicit Surfaces from Large Unorganized Point Sets*. PhD thesis, Université Bordeaux 1, France, 2003.

[RHC08] Damien Rohmer, Stefanie Hahmann, and Marie-Paule Cani. Local volume preservation for skinned characters. *Computer Graphics Forum*, 27(7):1919–1927, October 2008. Special Isuue: Proceedings of Pacific Graphics 2008, Tokyo Japan.

[RHC09] Damien Rohmer, Stefanie Hahmann, and Marie-Paule Cani. Exact volume preserving skinning with shape control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 83–92, New York, NY, USA, 2009. ACM.

[Ric73] A. Ricci. Constructive Geometry for Computer Graphics. *computer journal*, 16(2):157–160, May 1973.

[SA07] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 109–116, 2007.

[SKPSH13] Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)*, 32(5):125–135, 2013.

[Sor06] Olga Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

[SOS04] Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph.*, 23(3):896–904, 2004.

[SS11] Jos Stam and Ryan Schmidt. On the velocity of an implicit surface. *ACM Trans. Graph.*, 30(3):21:1–21:7, May 2011.

[TPSH14] Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Accurate and efficient lighting for skinned models. *Computer Graphics Forum (proceedings of EUROGRAPHICS)*, 33(2), 2014.

[TSIF05] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In K. Anjyo and P. Faloutsos, editors, *ACM/Eurographics Symposium on Computer Animation (SCA)*, pages 181–190, 2005.

[VBG+13] Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. Implicit skinning: real-time skin deformation with contact modeling. *ACM Trans. Graph.*, 32(4):125:1–125:12, July 2013.

[vFTS06] W. von Funck, H. Theisel, and H.-P. Seidel. Vector field based shape deformations. *Transactions on Graphics (Proc. ACM SIGGRAPH)*, 25(3):1118–1125, July 2006 2006.

[vFTS07] W. von Funck, H. Theisel, and H.-P. Seidel. Explicit control of vector field based shape deformations. In *Proc. Pacific Graphics 2007*, 2007.

[VGB+14] Rodolphe Vaillant, Gäel Guennebaud, Loïc Barthe, Brian Wyvill, and Marie-Paule Cani. Robust iso-surface tracking for interactive character skinning. *ACM Trans. Graph.*, 33(6):189:1–189:11, November 2014.

[WBS+13] Daniel Weber, Jan Bender, Markus Schnoes, André Stork, and Dieter Fellner. Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum*, 32(1):16–26, 2013.

[WCE07] Kenric B. White, David Cline, and Parris K. Egbert. Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, RT '07, pages 129–132, Washington, DC, USA, 2007. IEEE Computer Society.

[Wen05] Holger Wendland. *Scattered Data Approximation*, chapter 16.2 - Hermite–Birkhoff interpolation. Cambridge University Press, 2005.

[WGG99] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree - warping, blending and boolean operations in an implicit surface modeling system. *Comput. Graph. Forum*, 18(2):149–158, 1999.

[WP02] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 129–138, New York, NY, USA, 2002. ACM.

[WSLG07] Ofir Weber, Olga Sorkine, Yaron Lipman, and Craig Gotsman. Context-aware skeletal shape deformationskinning arbitrary deformations. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3), 2007.

[YSZ06] Xiaosong Yang, Arun Somasekharan, and Jian J. Zhang. Curve skeleton skinning for human and creature characters. *Computer Animation and Virtual Worlds*, 17(3-4):281–292, 2006.