Implicit Skinning Implementation

Fabio Turchet

Master of Science,

Computer Animation and Visual Effects

August, 2013

Contents

Abstract

Current character skinning algorithms and techniques like Dual Quaternions or Linear Blend Skinning allow to achieve acceptable deformations in terms of anatomy correctness, but have major problems that have to be constantly addressed in production including loss of volume at joints like elbows, fingers and knees.

Simulation techniques solve many of these issues, but have the drawback of being time consuming and often require a complex pipeline to be developed.

In those situations where good quality skinning is required at real-time speed, a new technique recently developed makes use of implicit surfaces to restore the volume and the correct shape during animation of polygonal meshes, as a post process and efficiently.

Acknowledgements

This work is dedicated first of all to my family: mum, dad and Luca. Without them I couldn't have achieved any of the beautiful things I've done in my life and those that are yet to come. I will never be able to thank them enough. I want also to thank my aunts, uncles and cousins around the world.

Many other persons supported me during this tough year; they are incredibly important to me and I'd like to thank them all: Daniela in primis for all the patience and love you gave me: for sure it wasn't easy to endure an egocentric(but passionate) programmer :) , Federico, Pedro, Mattias, Thomas, Valerio, all the guys I worked with at Axis Animation, and all the beautiful souls who stayed with me during hard times (you know who you are).

Of course I want to say thank you to the Professors at Bournemouth University and in particular Jon Macey, Alexander Pasko, Ian Stephenson and Mathieu Sanchez.
Finally I wish a good luck to my colleagues of the MSc course 2012/2013: may you find the job of your dreams soon !

<< If you can make one heap of all your winnings
And risk it on one turn of pitch-and-toss,
And lose, and start again at your beginnings
And never breathe a word about your loss;
If you can force your heart and nerve and sinew
To serve your turn long after they are gone,
And so hold on when there is nothing in you
Except the Will which says to them: 'Hold on!'>>

from *If* by Rudyard Kipling

<<Oh friends, not these tones!
Rather, let us raise our voices in more pleasing
And more joyful sounds!
Joy! (Joy!)
Joy! (Joy!)
Joy, beautiful spark of the gods
Daughter of Elysium,
We enter, drunk with fire,
Heavenly one, your sanctuary!
Your magic reunites
What custom strictly divided.
All men become brothers,
Where your gentle wing rests>>

from the 9th Symphony of Beethoven, 4^{th} movement (originally in German)

Chapter 1

Introduction

The skinning problem

In the modern Visual Effects and Games industry the characters shown on the screen, seamlessly integrated with live action appear so hyper-realistic not only thanks to physically accurate shaders, motion capture or compositing, but also thanks to good mesh deformations that mimic what occurs anatomically in real life.
The process to achieve this believable shapes during movement is made of two interdependent steps: one that is generally called rigging which gives in output a bone skeleton with sophisticated mechanisms created in order to activate automatic movements of joints or blendshapes, and the second that is commonly referred to as skinning that comprises all those techniques to move the vertices based on the bone position during an animation.
This project explores one of these skinning techniques and in particular a brand new method presented at SIGGRAPH 2013 called "Implicit Skinning: Real-Time Skin Deformation with Contact Modeling" Vaillant et al. (2013).The method is interesting because it shows an innovative way never used before to correct a skinned mesh with linear or dual quaternion skinning as a post process step. So it is not a full skinning method but rather a correction that makes a big difference compared to existing approaches.

Structure

Chapter 2 gives a brief literature review of the skinning problem and current solutions.
Chapter 3 gives a technical background on implicit surfaces and operators, necessary to follow chapter 4 where the core part of the algorithm is explained and detailed.
Chapter 5 shows images and renders of the result of the method applied to three different meshes at different resolutions: finger, arm and knee.
Finally chapter 6 details the implementation and the program design that leads to the final chapter with conclusions 7.

Chapter 2

Related work

When dealing with character skinning the approaches can be divided in simulated and not. Belonging to the second category, in the beginning there was only linear blend skinning (LBS). This pioneering method, introduced by Magnenat-thalmann et al. (1988) produces basic deformations that are not realistic and in particular the candy-wrap effect when for example the wrist or the shoulder twist too much; it doesn't preserve the volume either. For these reasons dual quaternion skinning (DQ) was introduced by Kavan et al. (2008) : it solves the candy-wrap problem and preserves volume better, but it creates unwanted bulges at the joints.
More recently Kavan & Sorkine (2012) proposed a method which not only optimizes skinning weights for standard LBS and DQ, but they introduce a deformer that achieves good quality similar to nonlinear methods being, efficient at the same time.
A common approach used in production is the use blendshapes, meshes sculpted by an anatomy-aware artist for particularly critical poses (typically extreme like for example the arm bent at 130°) that are attached to an existing rig using LBS or DQ and driven by the skeleton.
If the production schedule allows, skin, muscles and fat can instead be fully physically simulated like any other character effect (cloth or hair). Weta Digital Weta Digital, FxguideTV (2013) for example created a system called "Tissue" that realistically achieves anatomically correct deformations at the cost of extra simulation time. An advantage is that blendshapes can be generated out of the simulated mesh and attached to animation rigs to have real time feedback for animators, while the final muscle simulation is done at the end and is driven by the animated skeleton.
At Disney an hybrid solution was adopted as well as showed for example in the movie Tangled McAdams et al. (2011). A muscle-based approach was used instead for the movie Spiderman and developed at Sony Pictures ImageWorks by Miller & Harkins (2007) as a *Maya* framework called "Musculoskeletal pose space shape skinning" Miller, Erick (2012). The approach presented in this work instead is real-time and even though it can not be considered fully ready for videogames it is surely a good post-processing step that improves hugely LBS. The closest work that used implicit surfaces in conjunction with skinning correction is the one by Bloomenthal (2002).

Chapter 3

Technical background

3.1 Implicit surfaces

Implicit surfaces (or Function Representations, FReps) are an alternative way to the ordinary mesh representation using polygons (Boundary Representation, BReps) which have several advantages like the "infinite" resolution, the simplified boolean operations, the volume information as an intrinsic feature. In the Visual Effects and CSG (Constructive Solid Geometry) they are used to generate volumes, for collision detection and modeling.The application presented in this thesis is a good example on how concepts not directly related to animation can be used to help solve the problem of character skinning. An FRep is a multivariate function of the form :

$$f(x_1, x_2, x_3, ..., x_n) \geq 0$$

where, given a point $x = (x_1, x_2, x_3, ..., x_n)$:

$$\text{value of } f(x) = \begin{cases} > 0, & \text{if } x \text{ is inside.} \\ = 0, & \text{if } x \text{ is on the surface.} \\ < 0, & \text{if } x \text{ is outside.} \end{cases} \tag{3.1}$$

By using an algorithm such as marching cubes it is possible to visualize the boundary of the function or any C-isosurface (corresponding to the C-isolevel) providing a definition given in Gourmel et al. (2013) Gourmel (Figure 3.1):

$$S = \{x \in \mathbf{R^3} \mid f(x) = C\}$$

Considering the distance from the boundary a field can be defined and therefore a gradient.

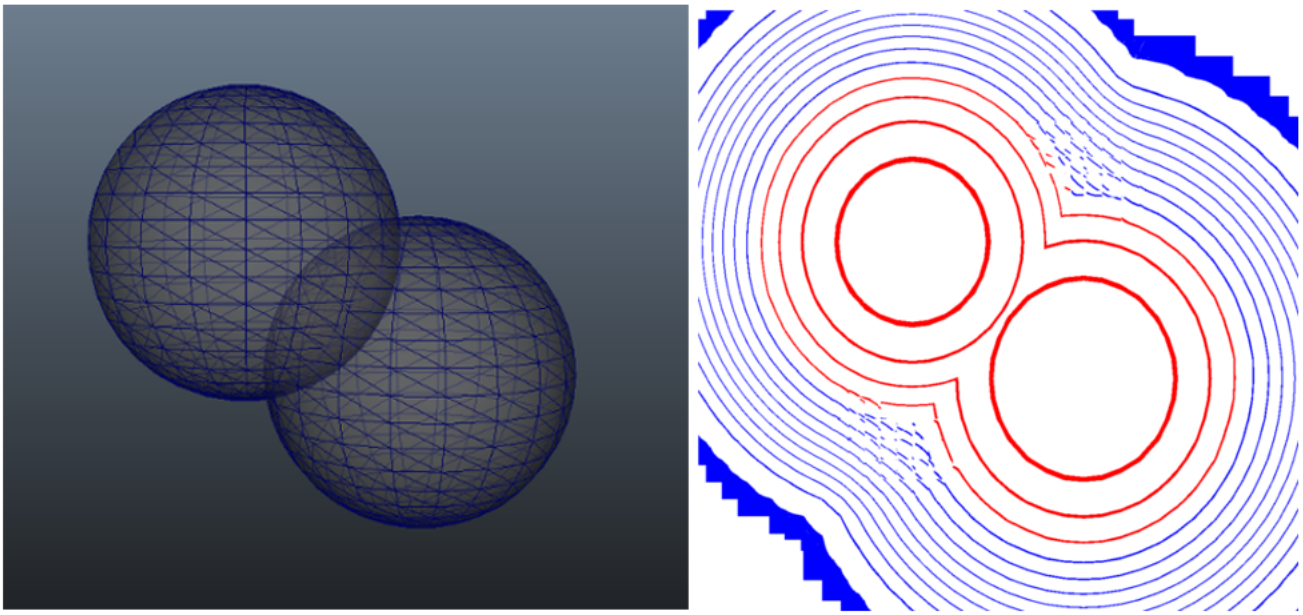


Figure 3.1: *Original mesh (left) and isolines of two spheres in contact(right)*

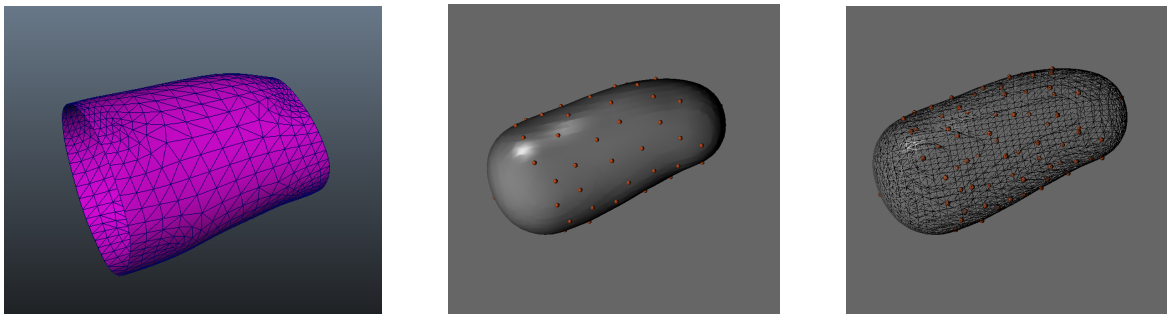
3.1.1 Local and global support

A function representation can have global support, that is its domain of definition is infinite and therefore its boundary is at $f(x) = 0$, or can have local support, that is its domain is limited and after a certain distance r the function is 0. Having defined a maximum distance, the function values can be reparametrized using the formula presented in the original paper and explained by Vaillant Vaillant (2013a) which performs a mapping from $[-r, r]$ to $[0, 1]$:

$$f(x)_{loc\_supp} = \begin{cases} 1, & \text{if } x < r. \\ 0, & \text{if } x > r. \\ \frac{-3}{16}(\frac{x}{r})^5 + \frac{5}{8}(\frac{x}{r})^3 - \frac{15}{16}(\frac{x}{r}) + \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{3.2}$$

therefore the values of the function become:

$$\text{value of } f(x)_{loc\_supp} = \begin{cases} > 0.5, & \text{if } x \text{ is inside.} \\ = 0.5, & \text{if } x \text{ is on the surface.} \\ < 0.5, & \text{if } x \text{ is outside.} \end{cases} \tag{3.3}$$



(a) (b) (c)

Figure 3.2: *(a) original segment; (b)shaded and (c) wireframe of an HRBF reconstructed from the samples (in red)*

3.2 Hermite Radial Basis Function (HRBF)

Hermite Radial Basis Functions are functions used especially in the reconstruction of meshes from point cloud data. In implicit skinning for a mesh M the input is a set of centers :

$$S = \{p \in M | p \text{ is uniformly distributed on M } \} \text{ , where } p = (position, normal)$$

HRBF was used as function to approximate the segmented mesh as it satisfies the following requirements :

1. Smoothness: a noisy field compromises the vertex projection step because the gradients become unstable.
2. Closure of the holes of the segmented meshes.
3. Compact support to allow composition

Given the points and normals and given a smooth function such as $\phi(x) = x^3$ it is possible to calculate a distance field that approximates the samples by solving a linear system of equations. For the mathematical details see Macdo et al. (2011).
Figure 3.2 shows the process of reconstruction for a mesh segment (a finger phalanx).

3.3 Operators

One of the biggest advantages of implicit surfaces is probably their ability to be combined hierarchically to generate new functions. The way to do that is by applying operators.

3.3.1 Classic operators

Union
Union operator is the simplest one and consists in taking at each point the maximum of the two input fields where the function is evaluated Ricci (1973) (Figure 3.3):

$$f = max(f_1, f_2)$$

This operator was improved by Pasko Pasko et al. (1995) and Barthe Barthe et al. (2003) to allow subsequent compositions otherwise not possible using simply max because of its discontinuities when $f_1 = f_2$:

$$f = f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$$

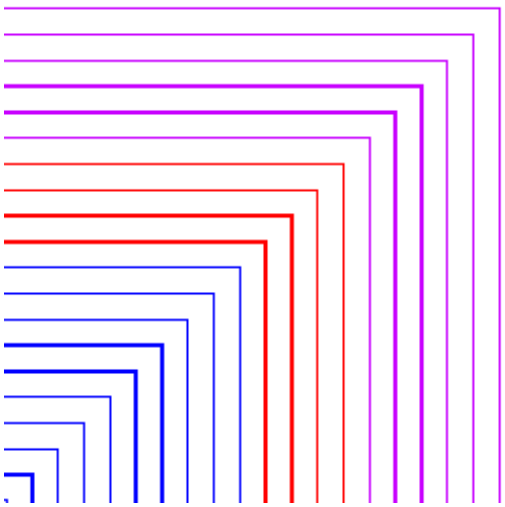


Figure 3.3: *Graph of max operator. Plotted in x is f_1, in y is f_2*

Blend
The simplest blending operator is the following:

$$f = f_1 + f_2$$

but has some limitations, for example it is not able to deal with global support inputs and doesn't provide parameters to control the blend. Therefore other operators were defined like the one by Pasko et al. Pasko et al. (1995) as following (*clean union*):

$$f = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + (f_1/a_1)^2 + (f_2/a_2)^2}$$

where a_0 , a_1 and a_2 can control the blending.

3.3.2 Gradient based operators

Gourmel et al. Gourmel et al. (2013) recently introduced a class of operators starting from the observation that a good blend can be obtained by considering not only the values but also the gradients of the input fields. This allows to solve common problems like bulging where the surfaces are aligned or blending at a distance. This kind of operators have the feature to be able to blend between a clean union and a custom field defined by a silhouette function and have the following components:

- an opening angle $\theta \in [0, \pi/4]$ which defines the amount of custom operator to blend in.
- an angle $\alpha \in [0, \pi]$ which defines the angle between gradients
- a controller $\theta(\alpha)$ which is a function to control for which angle there is a maximum of blending in.
- an opening function $k(\theta)$ which allows to define where the field will be the scaled silhouette curve and where will be the clean union.
- a silhouette curve $\bar{s}(\phi) : [0, \pi/2] \to \mathbb{R}$ which is a curve defined either in polar form or parametrically and determines how the two input functions will combine. It needs to have flat derivatives at the boundaries to produce a C^∞ continuous field. (Figure 3.4)

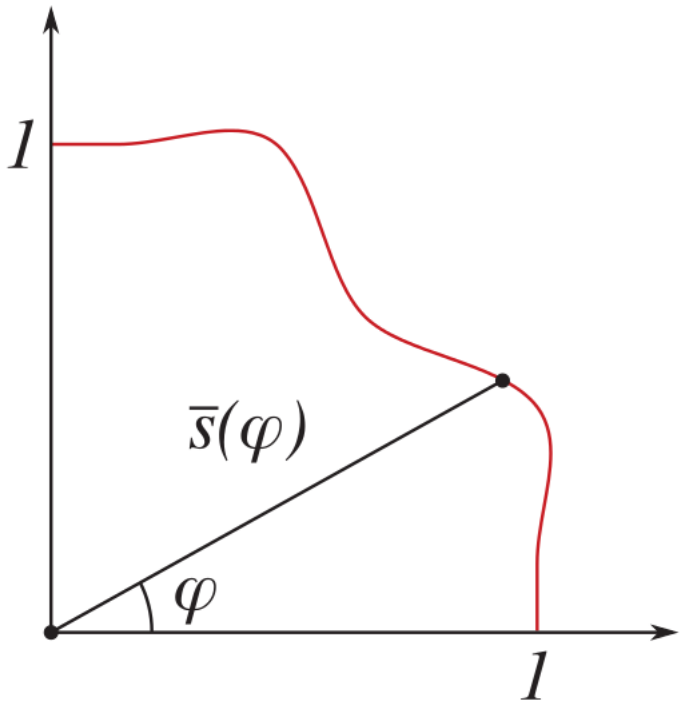


Figure 3.4: *Silhouette curve example Gourmel et al. (2013)*

The operator is defined formally in Gourmel et al. (2013) as :

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2), & \text{if } f_1 \leq k_\theta(f_2) \text{ or } f_2 \leq k_\theta(f_1). \\ \bar{g}(f_1, f_2), & \text{otherwise.} \end{cases} \tag{3.4}$$

where

$$\bar{g}(f_1, f_2) = \{C : \bar{h}_C(f1, f2) = 1\}$$

with

$$\bar{h}_C(f1, f2) = \frac{\sqrt{(f_1 - k_\theta(C))^2 + (f_2 - k_\theta(C))^2}}{\bar{s}(\varphi)(C - k_\theta(C))}$$

and

$$\varphi = \arctan \frac{(f_2 - k_\theta(C))}{(f_1 - k_\theta(C))}$$

Because the equation does not always have an analytical solution, a numerical method is presented in the same paper to fill a 3D texture with this operator.
For more details, plots and graphs of the operators used in the project see section 4

Chapter 4

Implicit Skinning

The implicit skinning technique is based on the concept that if a vertex at rest pose lies on a particular isoline, during animation it should lie on the same one. In order to achieve that a post-process operation on the vertices is applied that makes them march along the gradient defined by the field generated by the mesh and stops when the vertex reaches the original value of the field (or at a contact). To make this possible the individual HRBFs that compose the final field have to be rigidly transformed with the joint animation and this is achieved by applying the inverse bone transformation to the point being queried.

The whole process can be divided in the following steps (the full pipeline is shown in Figure 4.1):

1. Create the skeleton rig for the model
2. Skin the model with linear skinning or dual quaternions
3. Segment the mesh so that each joint has a portion of the original model associated
4. Place samples on the segments and generate HRBF functions for each of them
5. Add extra samples to close the holes
6. For each pair of functions f_1 and f_2 apply an operator to create a third function f_3
7. Compose each f_3 into a final field using max operator
8. During animation compute the field values for each vertex and correct them by applying a projection towards the gradient of the final function
9. Apply tangential relaxation and/or Laplacian smoothing

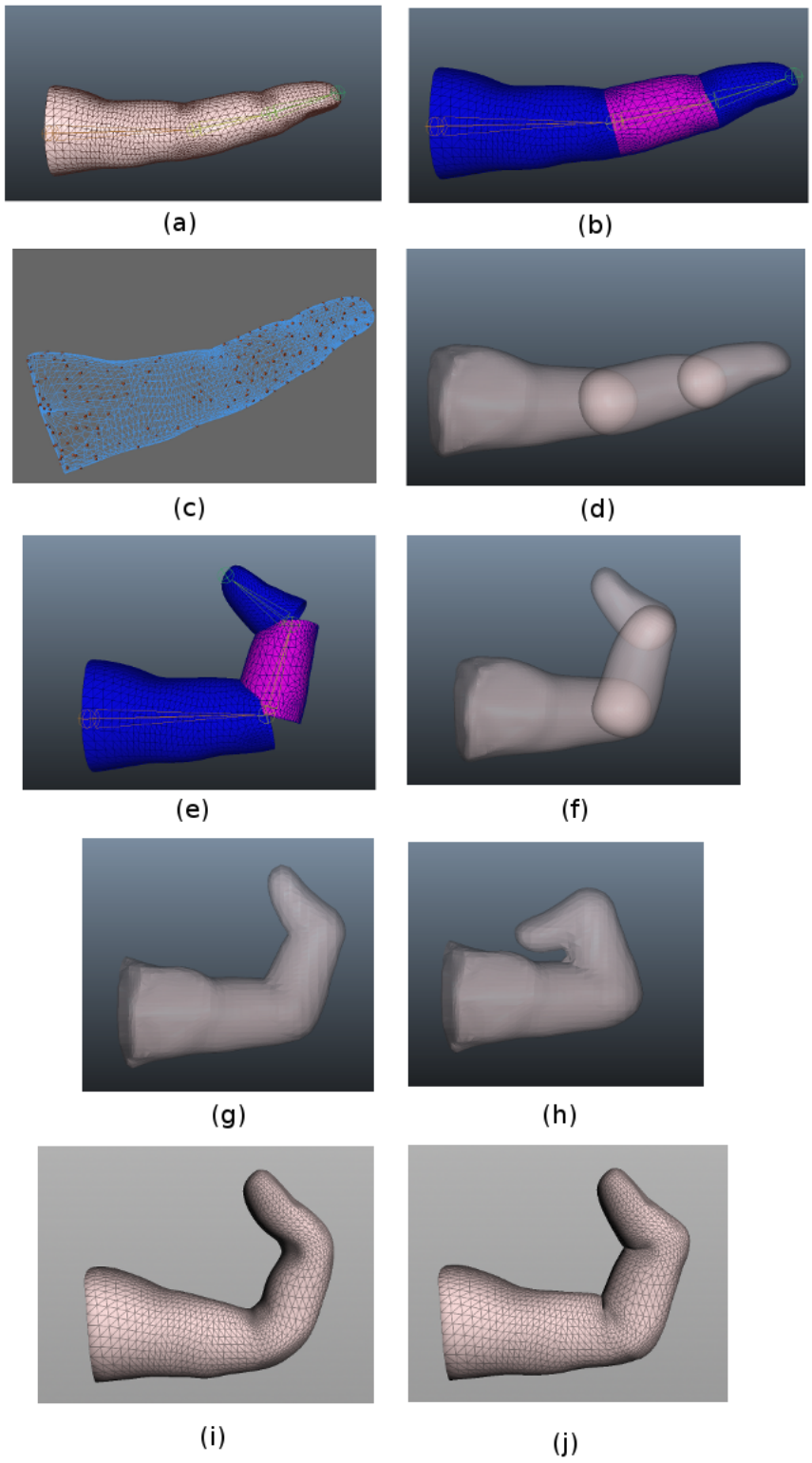


Figure 4.1: *Implicit Skinning Pipeline, (a) input skinned mesh; (b) segments; (c) samples; (d) single HRBFs; (e) segments animation; (f) HRBFs rigidly transformed; (g) composed field; (h) composed field; (i) default skinning; (j) corrected mesh*

4.1 Rigging, Skinning and Segmentation

Rigging, skinning and segmentation can be done in a 3D package like *Autodesk Maya* Autodesk (2013) or *Blender* Blender Foundation (2013). For this project all the examples were created in Maya 2012 and the scene exported in *COLLADA* Collada (2013) format (.dae file extension) using the *FBX_DAE* Maya plug-in and then read back in an standalone program using *AssImp* library Assimp (2013). The examples use classic linear skinning as applied by default in Maya (no heat diffusion nor skin weights painting) in order to produce on purpose bad deformation and to show the technique. Segmentation was done manually but automatic techniques exist, like the one presented in Au et al. (2008).

4.2 HRBF generation

4.2.1 Samples distribution

In order to produce a smooth isosurface that is as close as possible to the original model, it is necessary to sample the mesh by placing the HRBF centers in a way as uniform as possible. To do that Poisson disk distribution is used and in particular the dart throwing approach extended to 3D White et al. (2007) and Cline et al. (2009). This method allows to specify the minimum distance between the samples so to manage the density at will. A number of 50/60 samples per segment is enough in practice.

Even though this step is performed off-line is worth noticing that the technique is faster than normal dart throwing because it considers the regions of the mesh already covered and so not needed to test. It is called "hierarchical" because when a proposed point fails the minimum distance requirement, its related triangle gets split into four sub triangles (called fragments) obtained by connecting the mid points. To accelerate the algorithm a spatial hash was implemented. Shown below is the algorithm in pseudo code.

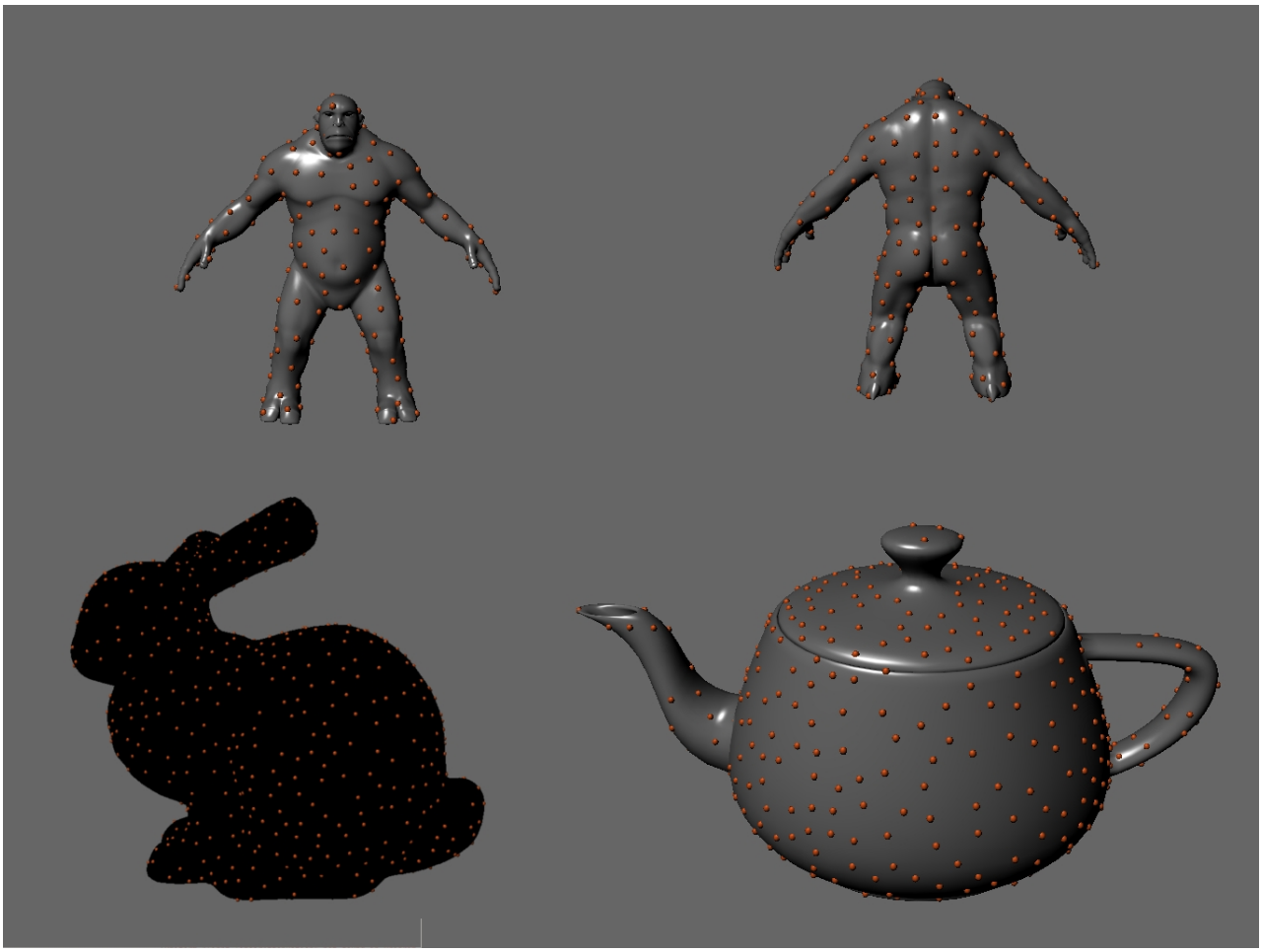


Figure 4.2: *Different models with different resolutions used to test the samples distribution algorithm*

```
FOR EACH triangle in the mesh
    insert it in logarithmic bins according to area at numBin
WHILE (no more fragments)
      keep a list of active fragments
      choose a bin with probability proportional to its total area
      WHILE ( a triangle is not accepted)
            pick a triangle from the bin with probability triProba
            proportional to its area
      generate random barycentric coordinates and get a point
      in the triangle
      IF the sample respects the minimum distance from samples around
         accept it
      IF the fragment is completely covered by other samples around it
         discard it
      ELSE
           split the triangle in 4 and insert in the active list
           the uncovered fragments2
```

where :
A_{max} = `maximum area of triangles in the mesh`
A_t = `area of the current triangle to be inserted`
$numBin = \log_2(A_max/A_t)$
B_{max} = `maximum area in the bin`
$triProba = A_t/B_{max}$

Figure 4.2 shows the application of the algorithm on different meshes.

4.2.2 Adding the extra samples

In order to have a proper field at the joints, two extra samples have to be added at the extremities of a bone. This is necessary to close properly the holes of the segmented meshes and to avoid gaps or non smooth regions as shown in figure 3.2. The two samples are placed along the bone at a distance :

$$s_j = ||joint - v_{closest}||$$

and they point outwards. For the root and the end joints no extra samples are added.

4.3 Binary composition of the HRBFs

Once the segmented meshes are turned into HRBFs, a first level composition is done for each pair of contiguous segments. This gives the freedom to assign different operators to each joint of a character and to integrate them properly. Max, Gourmel or Vaillant's operators can be chosen at this point based on the result desired. But before being able to compose the fields it is necessary to turn the functions from having global to local support. This is done by applying equation 3.2 using :

$$r = min(||v_i - (joint_2 - joint_1)||)$$

where $joint_1$ and $joint_2$ are consecutive joints forming the bone of the segment. In order to get the value of the resulting field three parameters are needed :

1. f_1 : the field value of the first HRBF
2. f_2 : the field value of the second HRBF
3. α : the angle between gradients of f_1 and f_2 which is the input of the controller function returning the angle θ

With these inputs is possible to fetch the value stored in a 3D texture having in u the f_1 sampled values, in v the f_2 values and in w the θ samples.

To get the correct gradient instead there are two possibilities:

1. Using Gourmel's formula for 3D gradients Gourmel et al. (2013) :

$$\nabla g(f_1, f_2) = \nabla f_1 \frac{\partial g}{\partial f_1} + \nabla f_2 \frac{\partial g}{\partial f_2} + \nabla \alpha \frac{\partial \theta}{\partial \alpha} \frac{\partial g}{\partial \theta}$$

2. Computing the gradient per slice as for a normal 2D operator:

$$\nabla g(f_1, f_2) = \nabla f_1 \frac{\partial g}{\partial f_1} + \nabla f_2 \frac{\partial g}{\partial f_2}$$

In the implementation the second version was used.

In the sections below, blend and bulge operators as presented by Gourmel and Vaillant will be described. In the following figures the color convention used is:

$$\text{isoline colour} = \begin{cases} \text{blue}, & \text{if } 0 \leq f < 0.5. \\ \text{red}, & \text{if } 0.5 \leq f < 0.7. \\ \text{purple}, & \text{if } f \geq 0.7. \end{cases} \tag{4.1}$$

For the following isolines graphs an animated finger model was used (Figure 4.3).

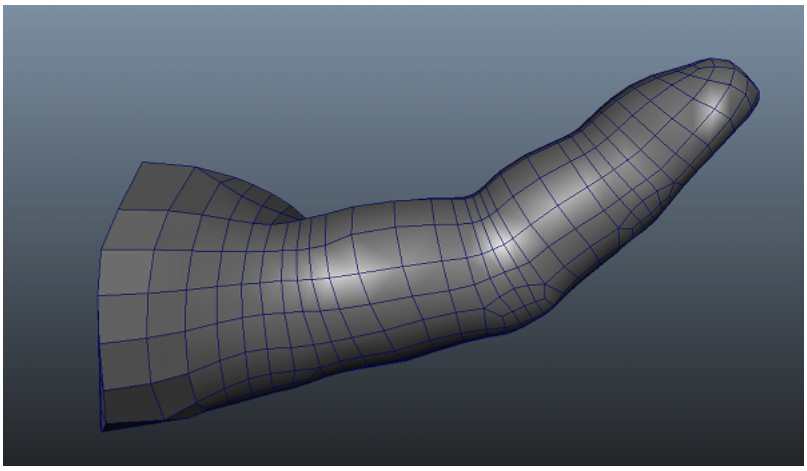


Figure 4.3: *Model used to show isolines of the operators*

4.3.1 Gourmel's blend operator

What makes the difference between an operator and another is the silhouette curve. But the original operator presented in the paper in equation (30) couldn't be fully implemented as the graph doesn't correspond to the one in the paper.

The function should in fact look like in Figure 4.6 a, but when plotting the formula in the paper the result is like in Figure 4.6 b.

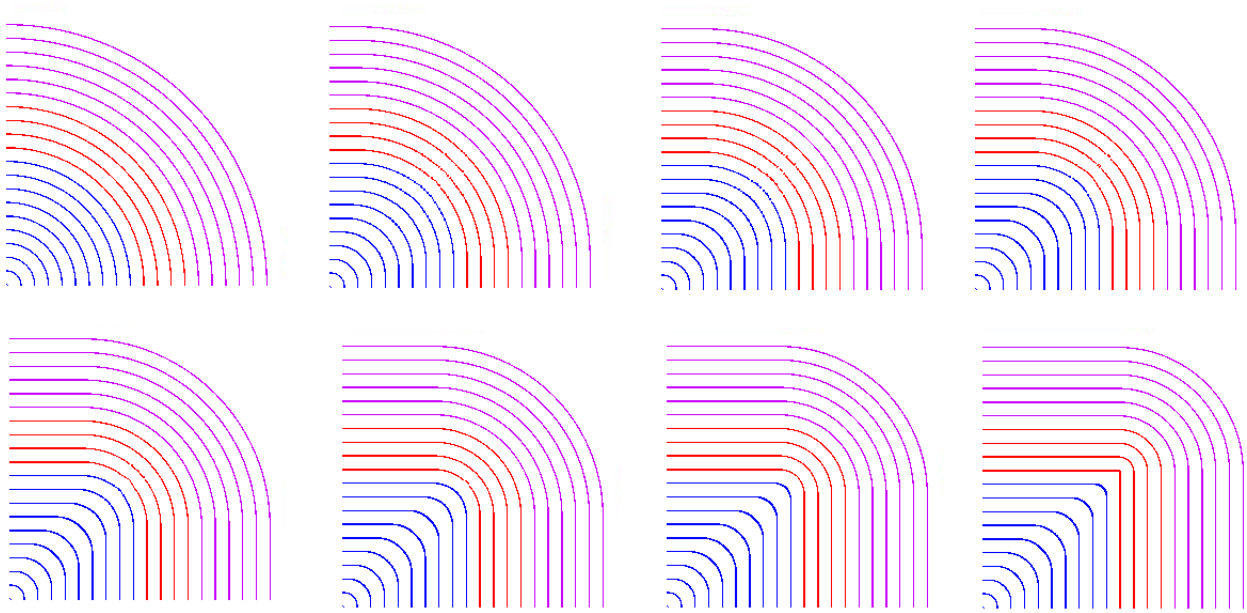


Figure 4.4: *Gourmel's blend operator for increasing value of θ : from 0 (top left) to $\pi/4$ (bottom right)*

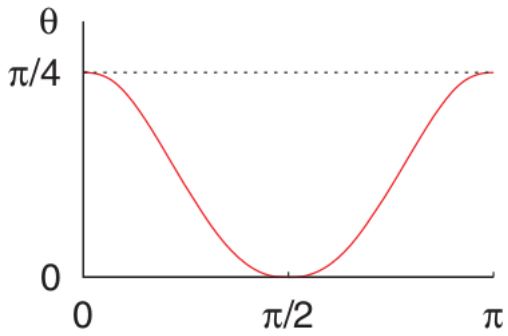
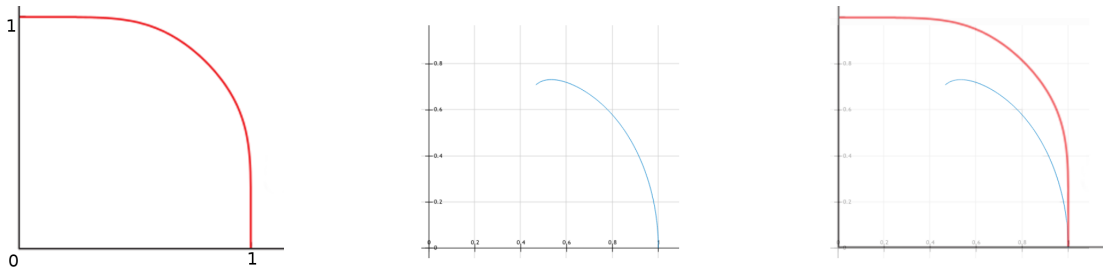


Figure 4.5: *Gourmel's blend operator controller Gourmel et al. (2013)*

For this reason a simple arc of a circle was used (Figure 4.7)

The problem of Gourmel's operators for implicit skinning is that is very noisy in the inside of the field and presents depressions and distortions that can let the vertices march in the wrong directions. As shown in Figure 4.5 the opening function creates a maximum blend in the points where the two surfaces are roughly at 90°, but this is not ideal for bending limbs as shown later.



(a) (b) (c)

Figure 4.6: *Gourmel's blend operator silhouettes: (a) correct Gourmel et al. (2013) , (b) plot of the equation in the paper, (c) comparison*

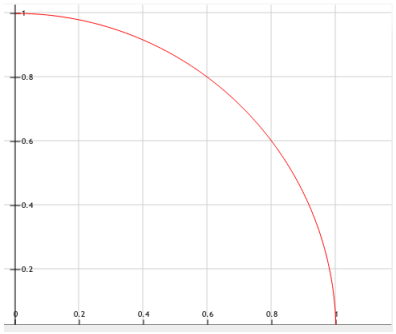


Figure 4.7: *Silhouette function* $\bar{s}(\varphi) = 1$

4.3.2 Gourmel's bulge in contact operator

This operator has a characteristic silhouette curve that mimics the inflation formed when two surfaces collide (the spikes form the typical discontinuities present at contacts). It has maximum bulge behaviour when the gradients form 0°. But in a joint animation another activation of the operator is desired as shown later.

As shown in figure 4.11 the interior part of the field is not suitable for vertex marching.

Figure 4.12shows some early tests involving blend and bulge operators on non animated T shape meshes.

4.3.3 Vaillant's blend operator

For implicit skinning purposes the problems shown above are solved in Vaillant Vaillant et al. (2013) by forcing the interior of the field to be max :

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2), & \text{if } f_1 \leq k_\theta(f_2) \text{ or } f_2 \leq k_\theta(f_1). \\ \max(f_1, f_2), & \text{if } f_1 > 0.7 \text{ or } f_2 > 0.7. \\ \bar{g}(f_1, f_2), & \text{otherwise.} \end{cases} \tag{4.2}$$

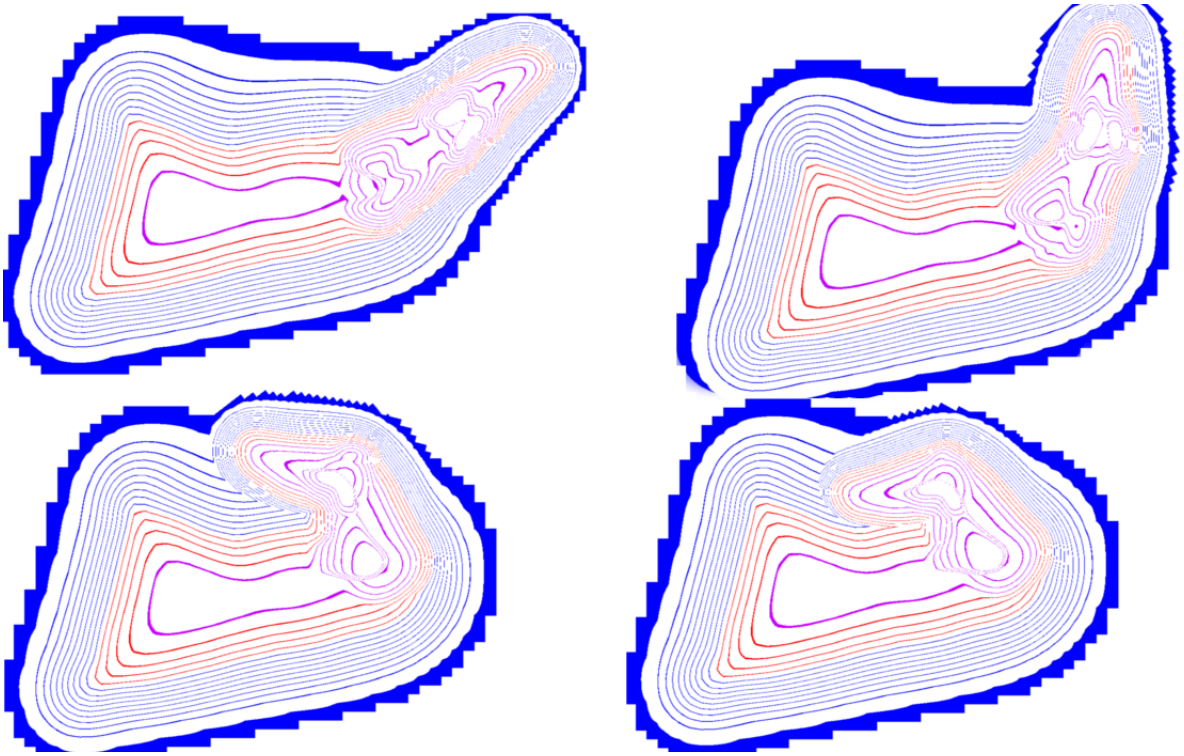


Figure 4.8: *Isolines of a finger animation using Gourmel blend operator*

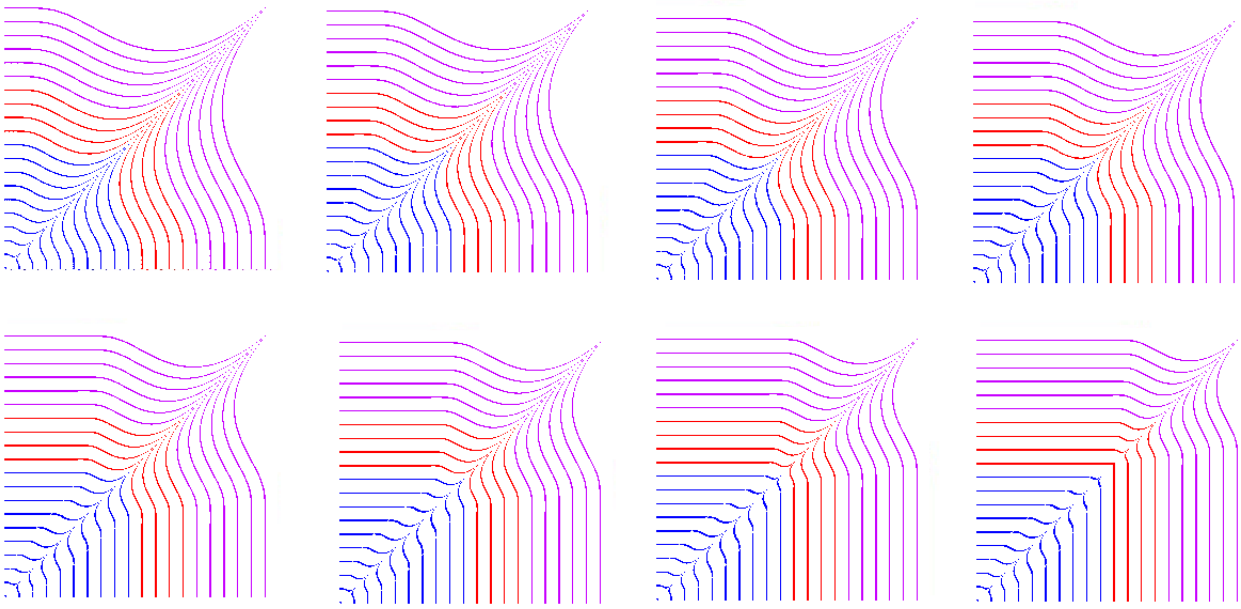


Figure 4.9: *Gourmel's bulge operator for increasing value of θ : from* 0 *(top left) to* $\pi/4$ *(bottom right)*

An alternative is using the operator by Canezin Canezin et al. (2013)

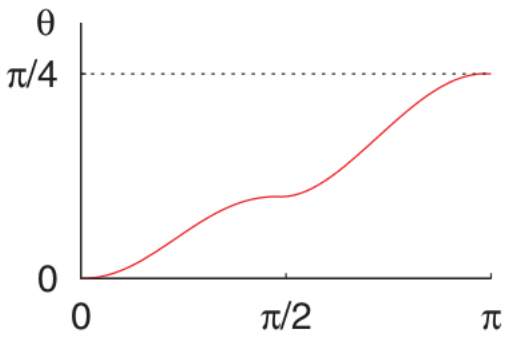


Figure 4.10: *Gourmel's bulge operator controller Gourmel et al. (2013)*

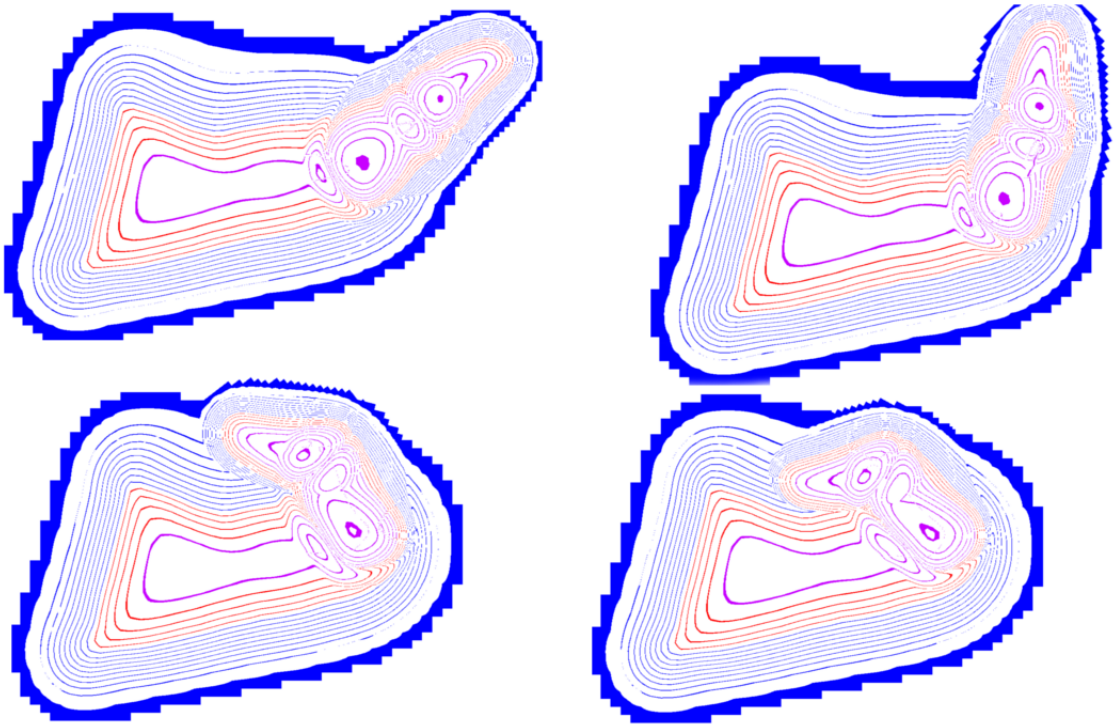


Figure 4.11: *Isolines of a finger animation using Gourmel bulge operator*

As shown in Figure 4.14 (a) the controller has maximum blend for $\pi/4$ (0.78) as it resembles more a real deformation for example of an arm that when slightly bent presents a smooth look. This kind of control can be easily adjusted as there is a connection between the angle formed by the bones and the gradients because the individual HRBFs are rigidly transformed during the animation.

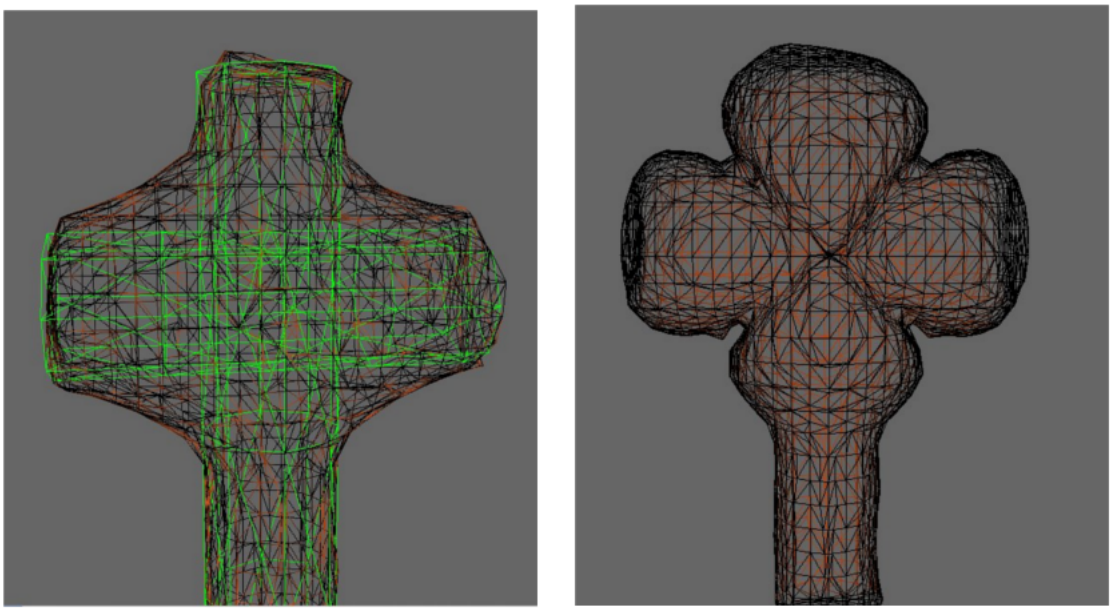


Figure 4.12: *Blend (left) and bulge (right) on a T shape mesh (in green)*

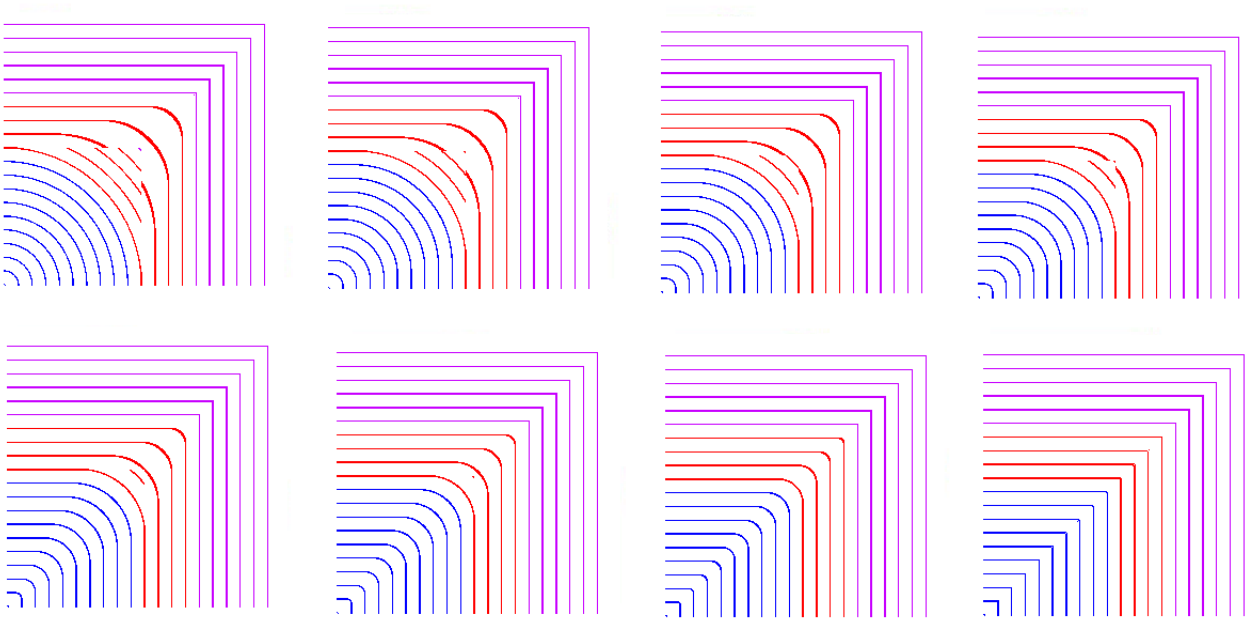
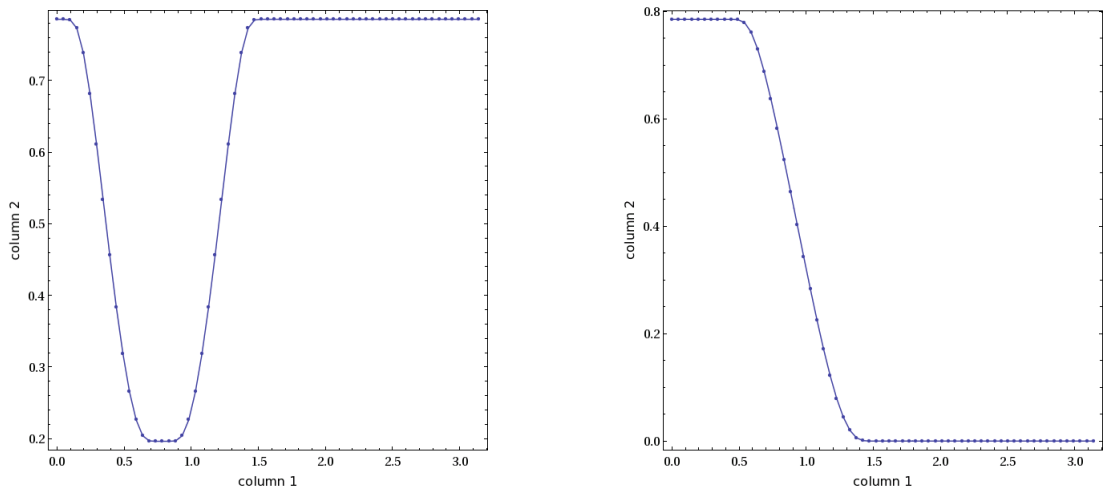


Figure 4.13: *Vaillant's blend operator for increasing value of θ : from 0 (top left) to $\pi/4$ (bottom right)*

4.3.4 Vaillant's bulge in contact operator

Figure 4.15 shows some of the slices of the operator : the diamond shape given by the k_{theta} functions can be noticed. Figure 4.14 (b) shows instead the controller for this operator: it is very different from Gourmel's because the bulge in contact is desired when the joints are



(a) (b)

Figure 4.14: *(a) Controller of Vaillant's blend operator ; (b) Controller of Vaillant's bulge operator*

over 90°.

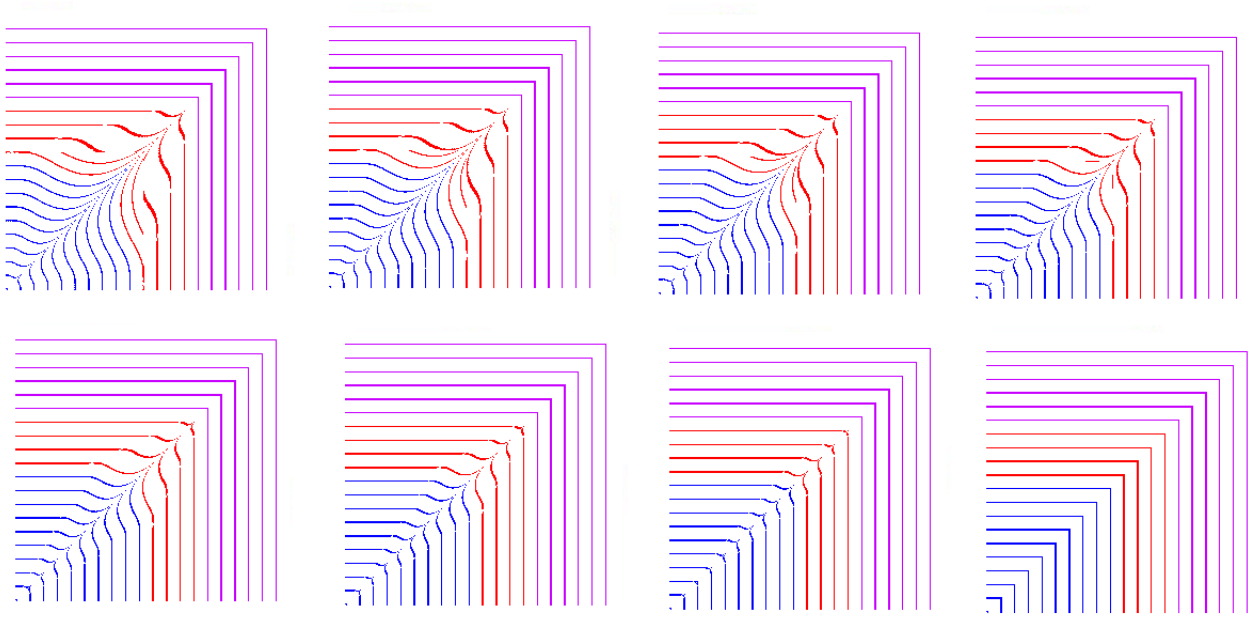


Figure 4.15: *Vaillant's bulge operator for increasing value of θ : from 0 (top left) to $\pi/4$ (bottom right)*

4.3.5 Composition of the final field

Once the pairs of HRBF have been composed using the preferred operator, they have to be blended into a final function. To do so a simple *max* operator is used and the gradient is calculated by taking the gradient associated to the function with the maximum value of the field. More formally:

$$\nabla f(f_1, f_2) = \nabla f_1 \frac{\partial g}{\partial f_1} + \nabla f_2 \frac{\partial g}{\partial f_2}$$

where

$$\frac{\partial g}{\partial f_1} = \begin{cases} 1, & \text{if } f_1 > f_2. \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

4.4 Vertex marching

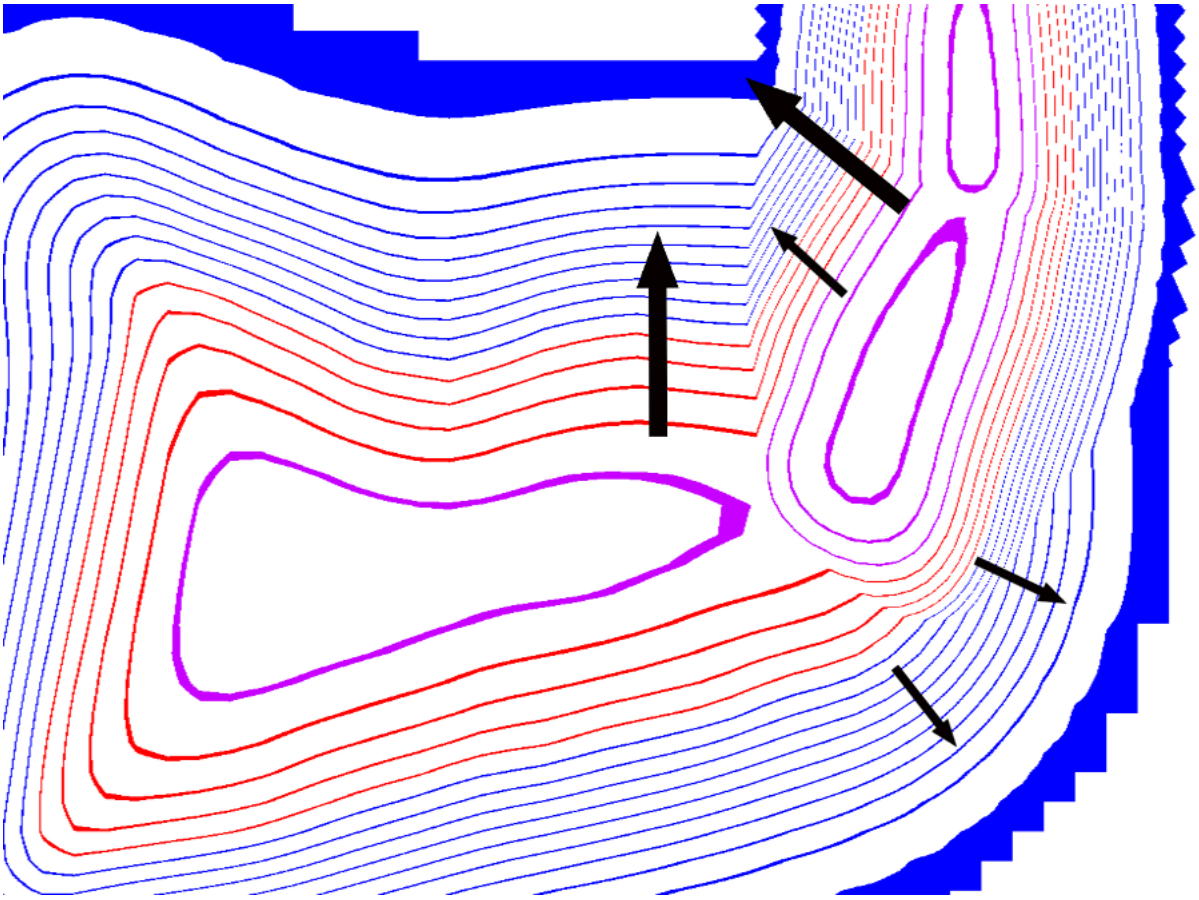


Figure 4.16: *Gradient direction during vertex projection (max operator)*

The core part of the algorithm is the marching of the vertices. In this phase each vertex is projected in the direction of the gradient computed at its current position (see Fig 4.16).

By using the Newton's method at each iteration the difference between the current field value and the original in rest pose is evaluated and if major of a threshold the vertex is moved along the gradient by a value proportional to the difference. In alternative constant steps can also be used. Formally:

$$v_{i+1} = v_i + \sigma(f(v_i) - f(v_{restPose}))\frac{\nabla f(v_i)}{||\nabla f(v_i)||^2}$$

where $\sigma = 0.35$ is a default value that works for most cases. By using the information of the rest pose the algorithm can correct high distortions and undesired effects created by default skinning providing some sort of volume conservation (relative to the original mesh) which makes it an attractive choice in when a fast method to skin many characters (probably secondary) is needed.

4.4.1 Collision detection

Often during an animation parts of a character (for example a finger bending) get in contact and it is really important to detect self intersections to achieve realistic deformations. To do that a simple method used is to measure the angle between gradients in two consecutive iterations and stop the marching if this angle is greater than 55° meaning that the vertex ended up too far and therefore into the field function of the second contacting surface that has gradients pointing in the opposite direction.

4.5 Relaxation and smoothing

Tangential relaxation is used to reduce distortions in the final mesh and it is applied every two steps of projection. The technique works by first projecting the one-ring neighbours to the tangential plane (defined from the vertex point and its gradient) and then calculating the barycentric coordinates in rest pose. During animation each vertex is relaxed using these initial values in order to limit as possible distortions compared to the initial mesh.

Laplacian smoothing is used in those cases, especially at contacts, where some vertices create sharp features. in particular all the vertices which end the iteration because of a contact are marked and a value of $\beta = 1$ is associated to them. Afterwards these values are smoothed out by diffusion and determine the amount of smoothing of the surrounding area (Figure 4.17).

Chapter 5 contains examples with and without smoothing.

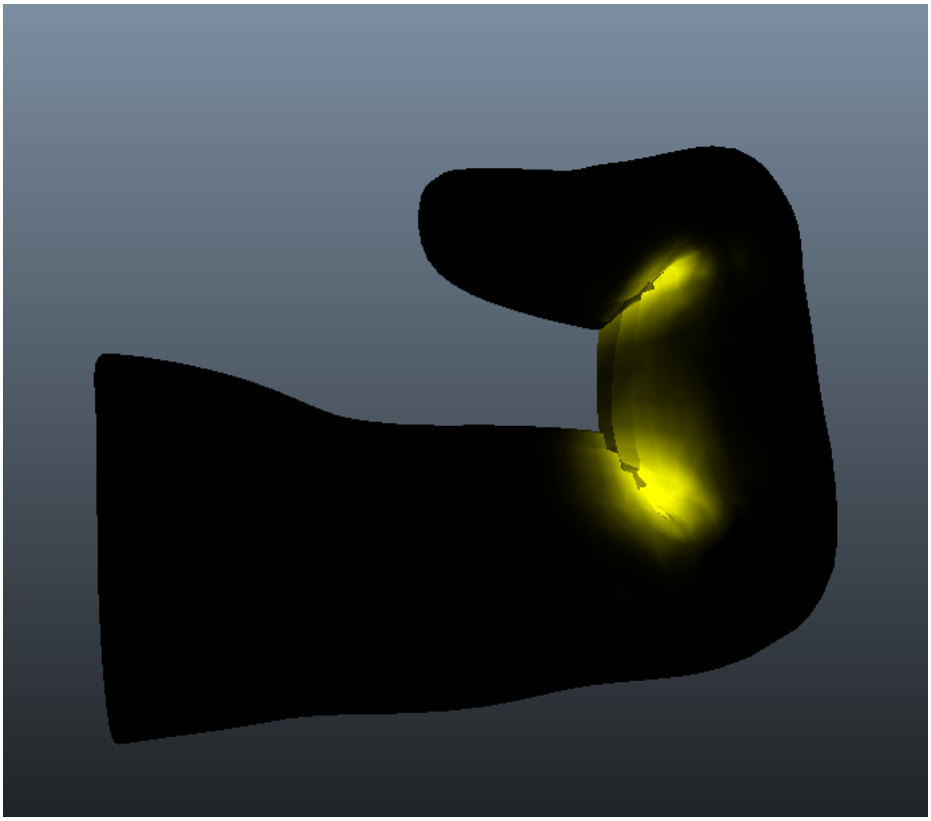


Figure 4.17: *β values diffused as preprocess to laplacian smoothing*

Chapter 5

Applications and results

The test set is composed of three meshes obtained from a realistic human model freely available at TurboSquid.com (2013). In particular an index finger, an arm and a leg were extracted from the original model and prepared to be used. Rigging, skinning and segmentation were done in Maya using the default settings. In this section are shown the results using max operator which behaved best. Both low poly and high poly meshes were used to test speed and robustness. In all cases the texture resolution used to store function values and gradients of the HRBFs is equal to 64.
In general the results achieved are satisfactory and the improvement compared to default linear skinning is evident. For the knee example, like stated in the original paper a correction would be needed to remove the extra sample at the knee to avoid an undesired bulge a bit too pronounced. As long as the max field is smooth and not noisy the vertices march without noticeable poppings. One way to correct undesired effects is by using tangential relaxation and laplacian smoothing, but beforehand it is necessary to have a well defined field otherwise undesired results will occur. For this reason Vaillant's operators are not shown as, probably for an implementation issue, did not behave properly in terms of stability.

Even though the animation was cached to disk for rendering, the program performed in real time. As a side note, the renders below are not aimed to be photo-realistic, but rather to show the comparison with linear skinning.

Table 5.1 contains the number of vertices for each model.

Table 5.2 shows the parameters used, where Δ grad is the increment for numerical gradients, ϵ is the threshold for the difference between two consecutive projection steps and smoothing value is value $[0, 1.0]$ to control the amount of smoothing.

MODEL	Finger low	Finger hi	Arm low	Arm hi	Knee hi
NUM VERTICES	541	2811	314	3362	4535

Table 5.1: *Model set and number of vertices*

Δ grad	ϵ	σ	max iterations	Collision Angle	Smoothing
0.001	0.000001	0.35	50	55°	0.5

Table 5.2: *Parameters used*

MODEL	Finger low	Finger hi	Arm low	Arm hi	Knee hi
projection(ms)	20	95	10	87	106

Table 5.3: *Time in milliseconds to perform one step of vertex marching*

Table 5.3 shows the results, in milliseconds, of the maximum time taken to perform the vertex projection (with multi-threading). The specifications of the machine used are : Intel(R) Core(TM)2 Quad CPU Q9550 at 2.83 GHz, 8 GB of RAM, graphics car NVIDIA QUADRO FX 3800/PCIe/SSE2.

5.0.1 Finger Low Poly

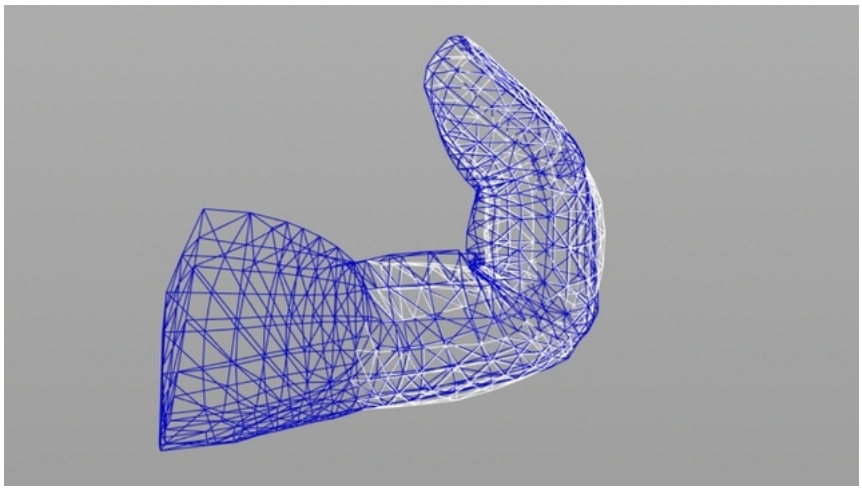


Figure 5.1: *Close-up showing the wireframe comparison between default skinning and implicit skinning*

As shown below (Figure 5.3)and in the videos the finger preserves well the volume even though some vertices are a bit unstable. This is in line with the original paper stating that a decent amount of vertices should be used (here only 541).

5.0.2 Finger High Poly

This model(Figure 5.6) behaves well during the marching showing almost no poppings or instabilities. As shown in Figure the volume is noticeably preserved compared to LBS. Clear is also the benefit of smoothing.

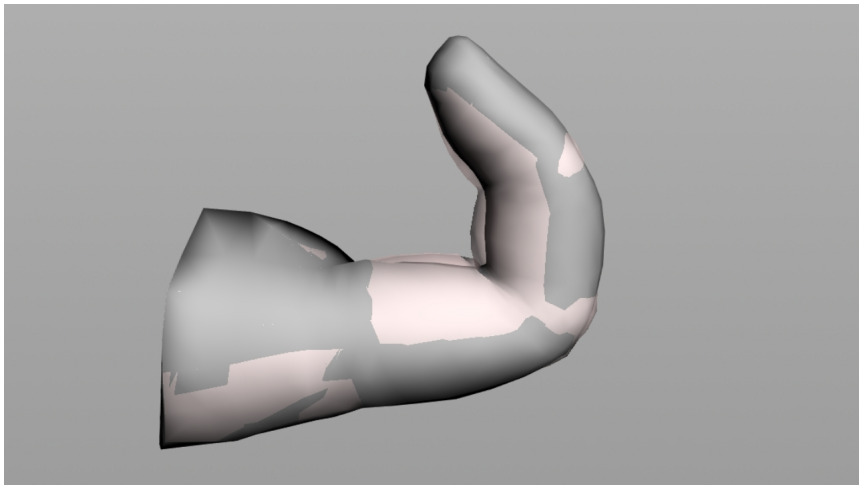


Figure 5.2: *Close-up showing the comparison with transparency between default skinning and implicit skinning*

5.0.3 Arm Low Poly

Given the really low amount of vertices, the result is acceptable (Figure 5.9). The main problem here is the initial model that is not exactly anatomically correct. Having a good model in rest pose is crucial for the technique to work.

5.0.4 Arm High Poly

This model (Figure 5.11) behaves particularly well : the biceps are correctly preserved. Tangential relaxation here would have been beneficial and probably also some amount of bulge in contact.

5.0.5 Knee High Poly

The overall initial volume is definitely preserved, but the knee is maybe a bit too pronounced (Figure 5.14). This could be solved by removing some sample like mentioned in the original paper. The knee is very high poly and this seems to create some problems at the external contact regions. Again, smoothing here helps and also tangential relaxation. The position of the joint and the way the knee mesh is segmented influence the result significantly.

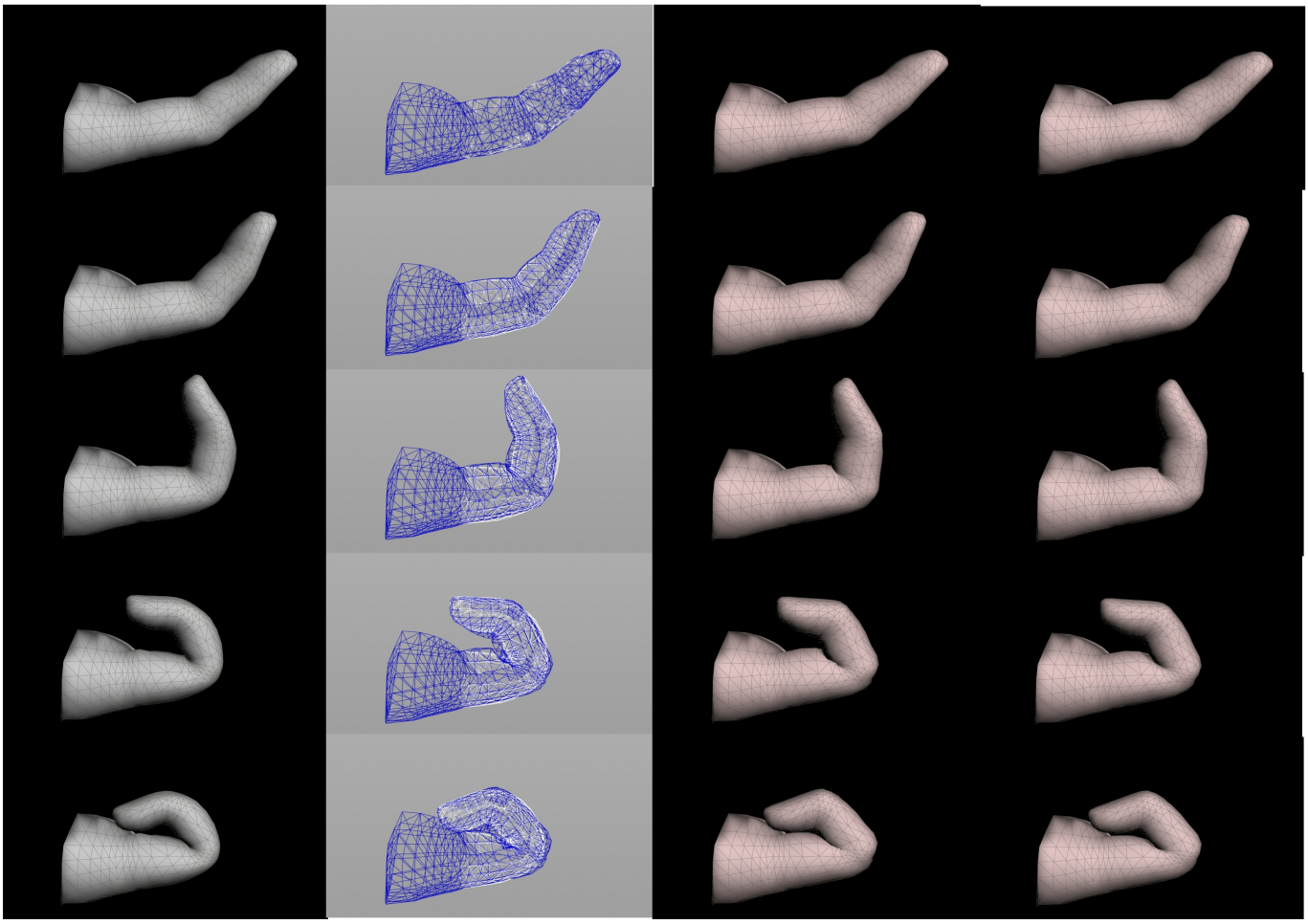


Figure 5.3: *Results comparison for Finger Low Poly. From left to right : default linear skinning, overlapped wireframe with corrected mesh for comparison, implicit skinning with no smoothing, implicit skinning with 0.5 smoothing*

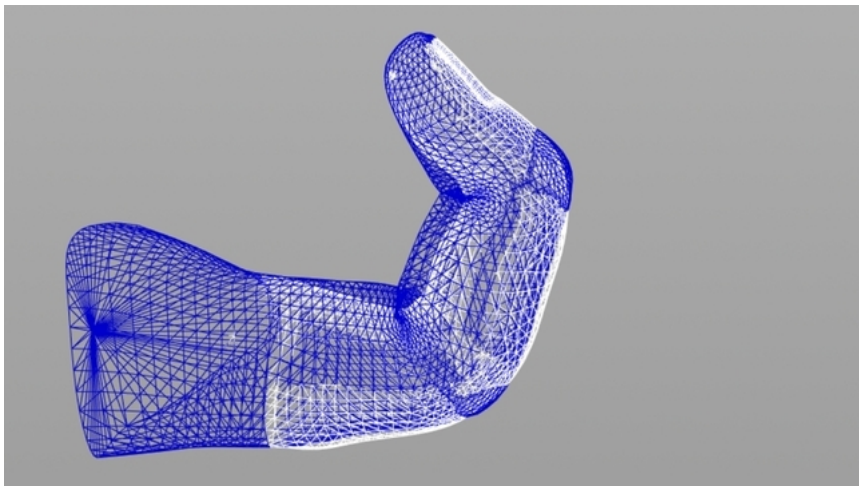


Figure 5.4: *Close-up showing the wireframe comparison between default skinning and implicit skinning*

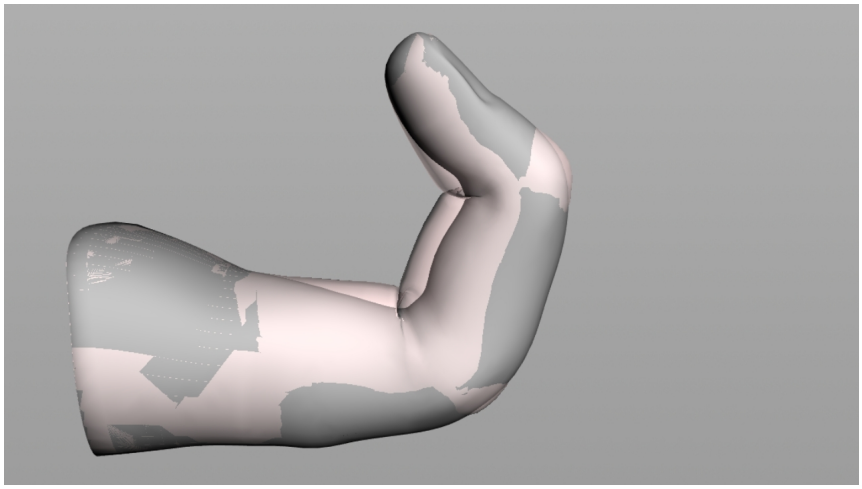


Figure 5.5: *Close-up showing the comparison with transparency between default skinning and implicit skinning*

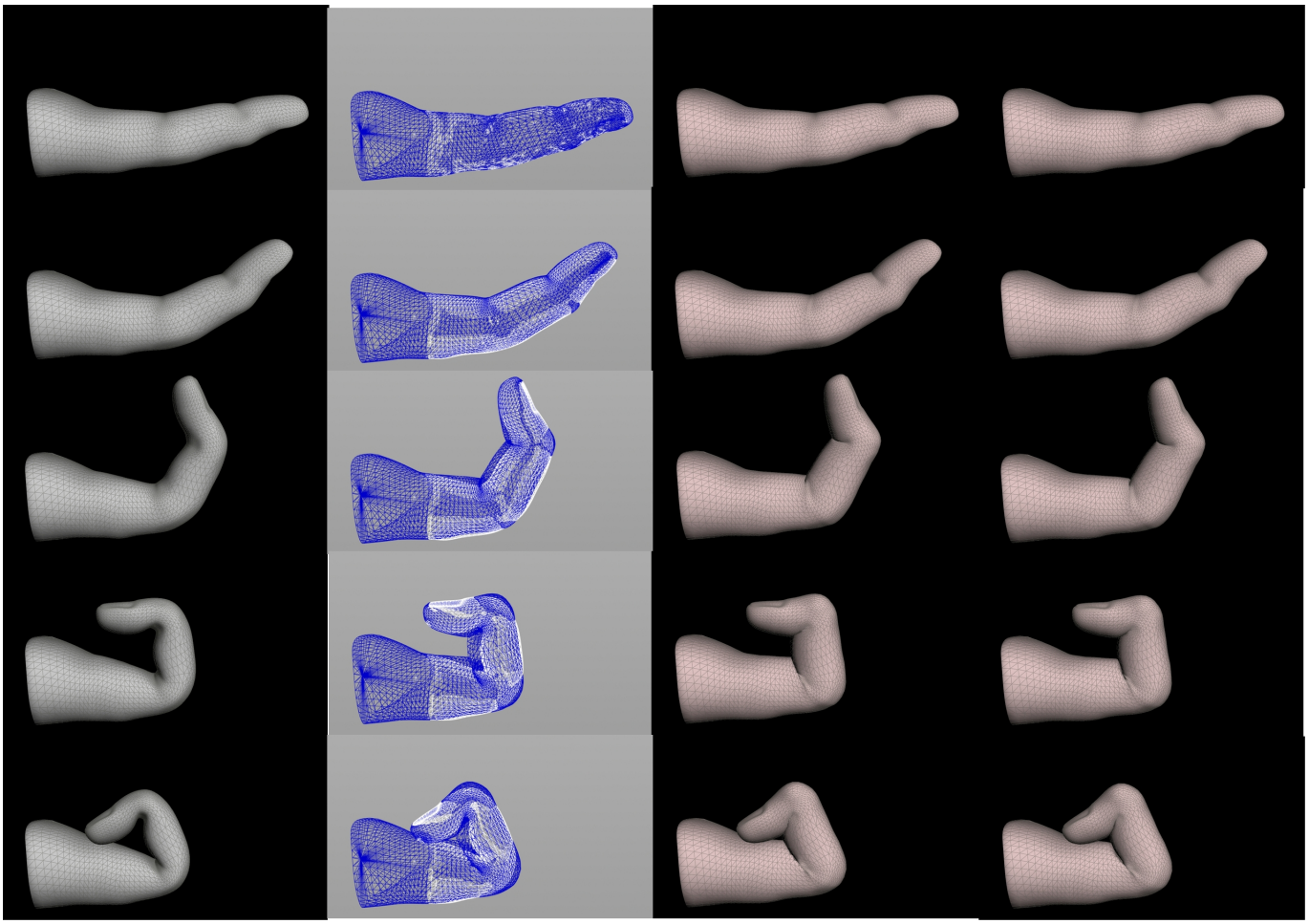


Figure 5.6: *Results comparison for Finger High Poly. From left to right : default linear skinning, overlapped wireframe with corrected mesh for comparison, implicit skinning with no smoothing, implicit skinning with 0.5 smoothing*

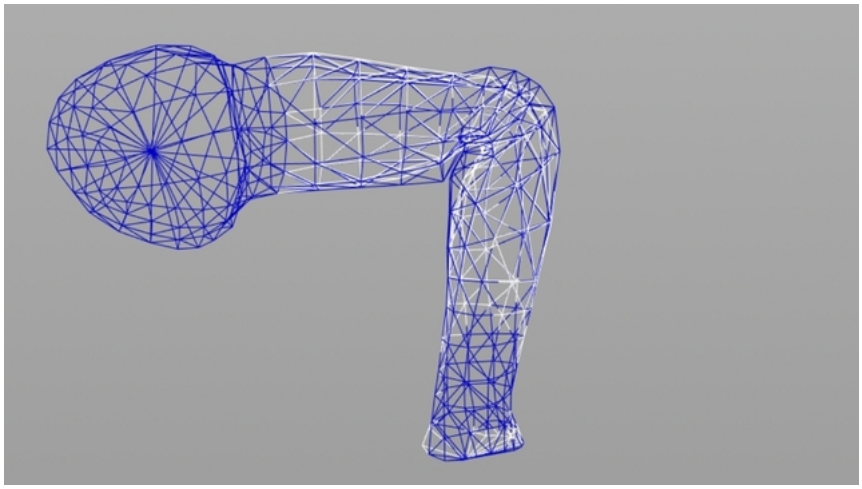


Figure 5.7: *Close-up showing the wireframe comparison between default skinning and implicit skinning*

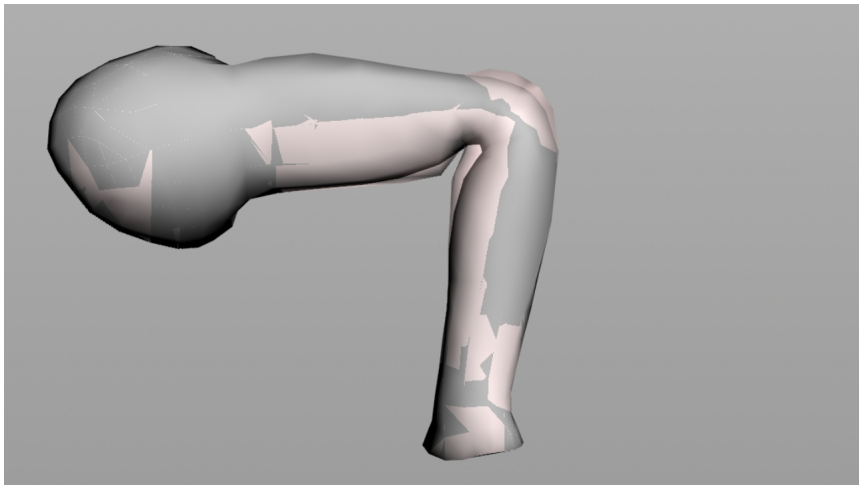


Figure 5.8: *Close-up showing the comparison with transparency between default skinning and implicit skinning*

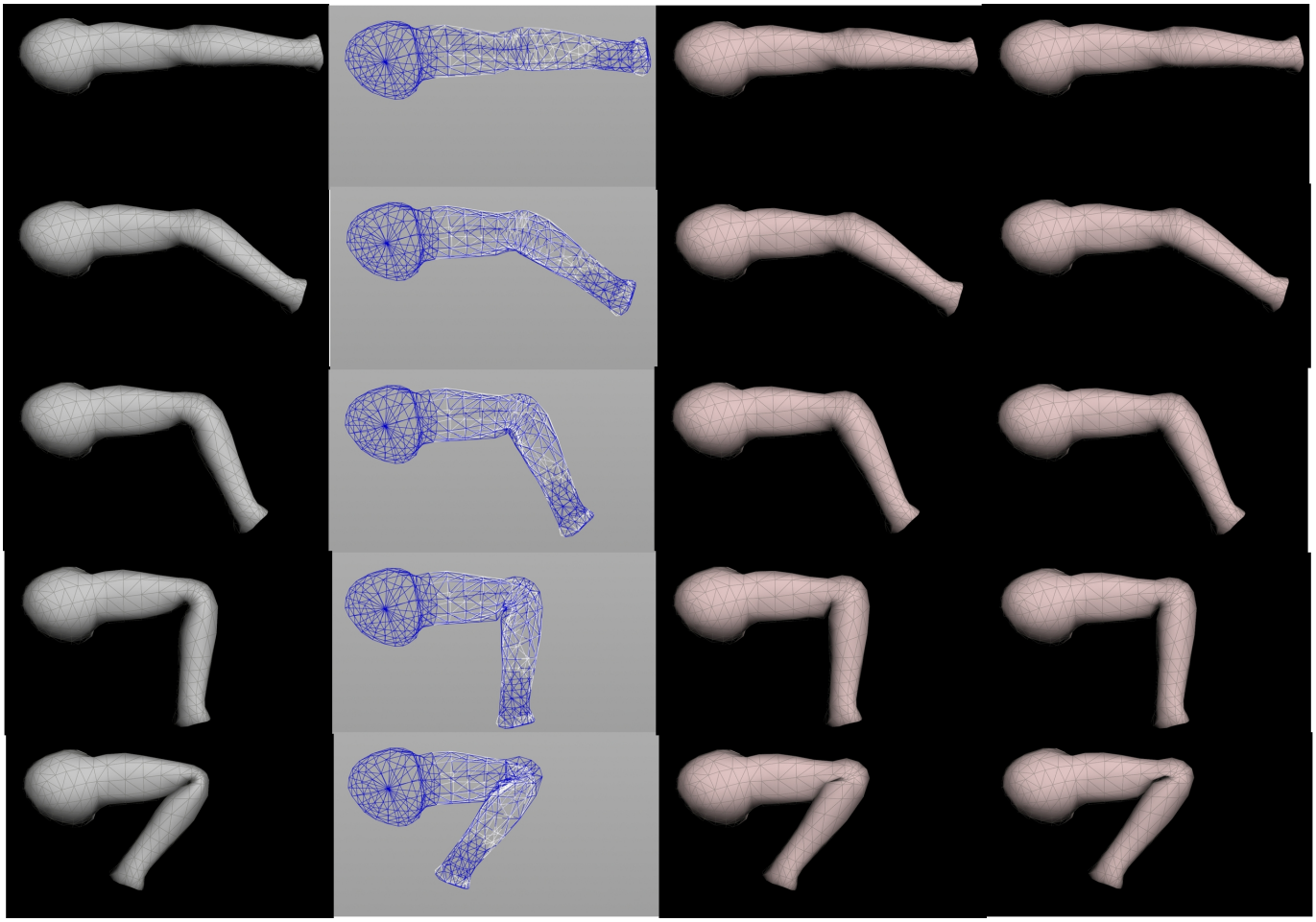


Figure 5.9: *Results comparison for Arm Low Poly. From left to right : default linear skinning, overlapped wireframe with corrected mesh for comparison, implicit skinning with no smoothing, implicit skinning with 0.5 smoothing*

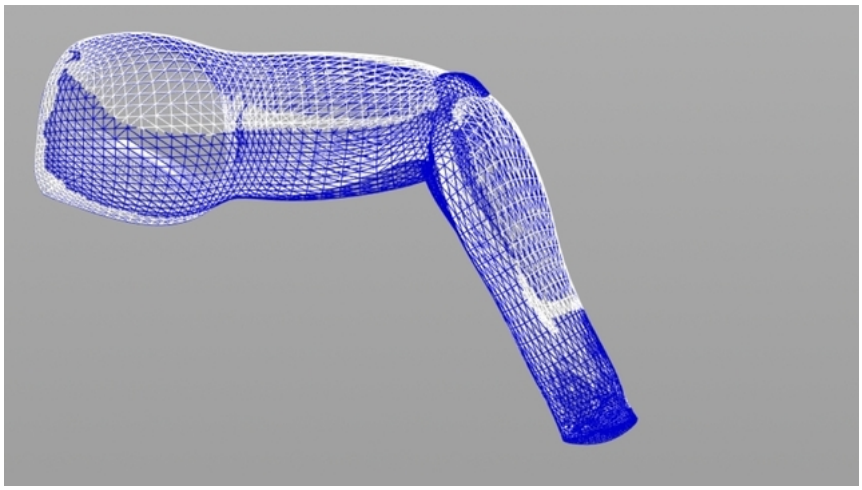


Figure 5.10: *Close-up showing the wireframe comparison between default skinning and implicit skinning*

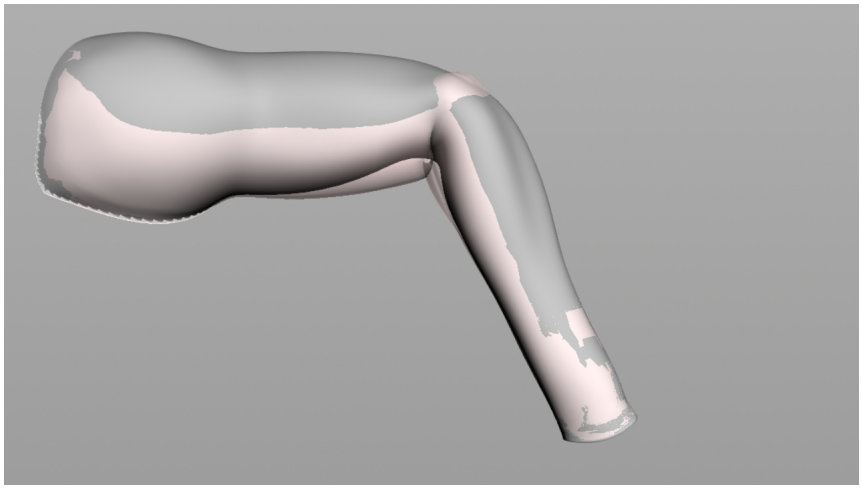


Figure 5.11: *Close-up showing the comparison with transparency between default skinning and implicit skinning*

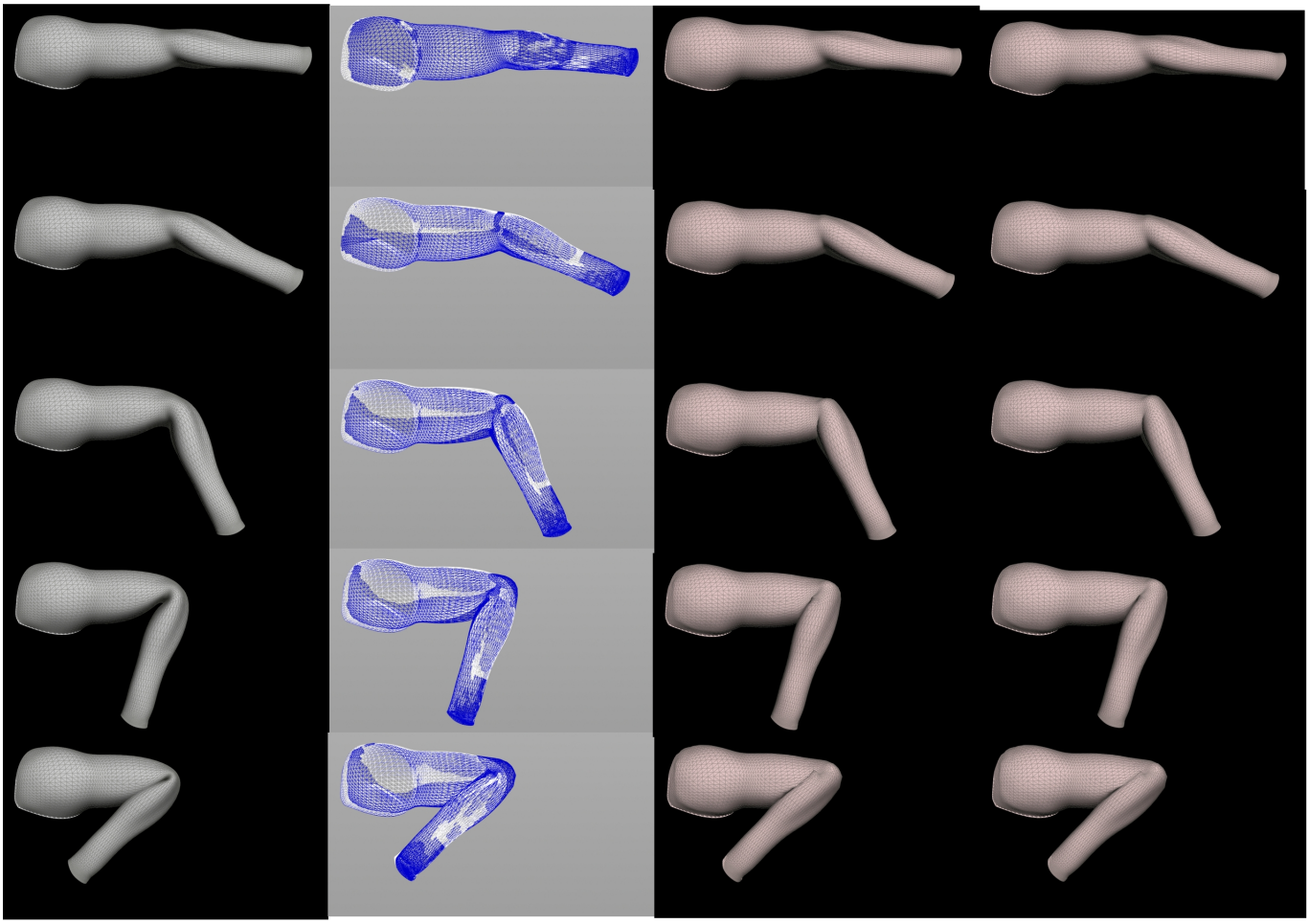


Figure 5.12: *Results comparison for Arm High Poly From left to right : default linear skinning, overlapped wireframe with corrected mesh for comparison, implicit skinning with no smoothing, implicit skinning with 0.5 smoothing*

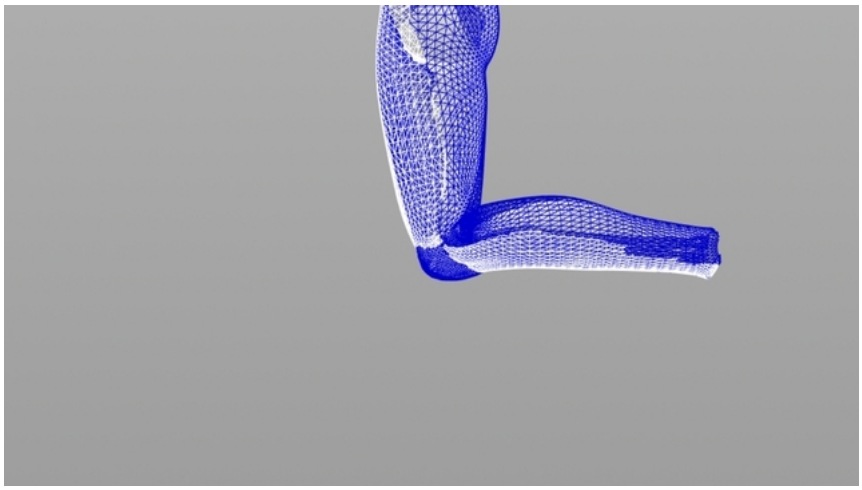


Figure 5.13: *Close-up showing the wireframe comparison between default skinning and implicit skinning*

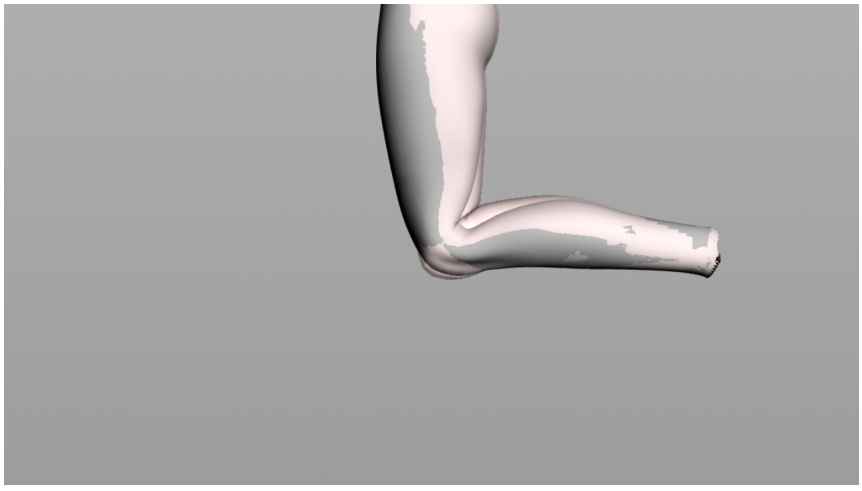


Figure 5.14: *Close-up showing the comparison with transparency between default skinning and implicit skinning*

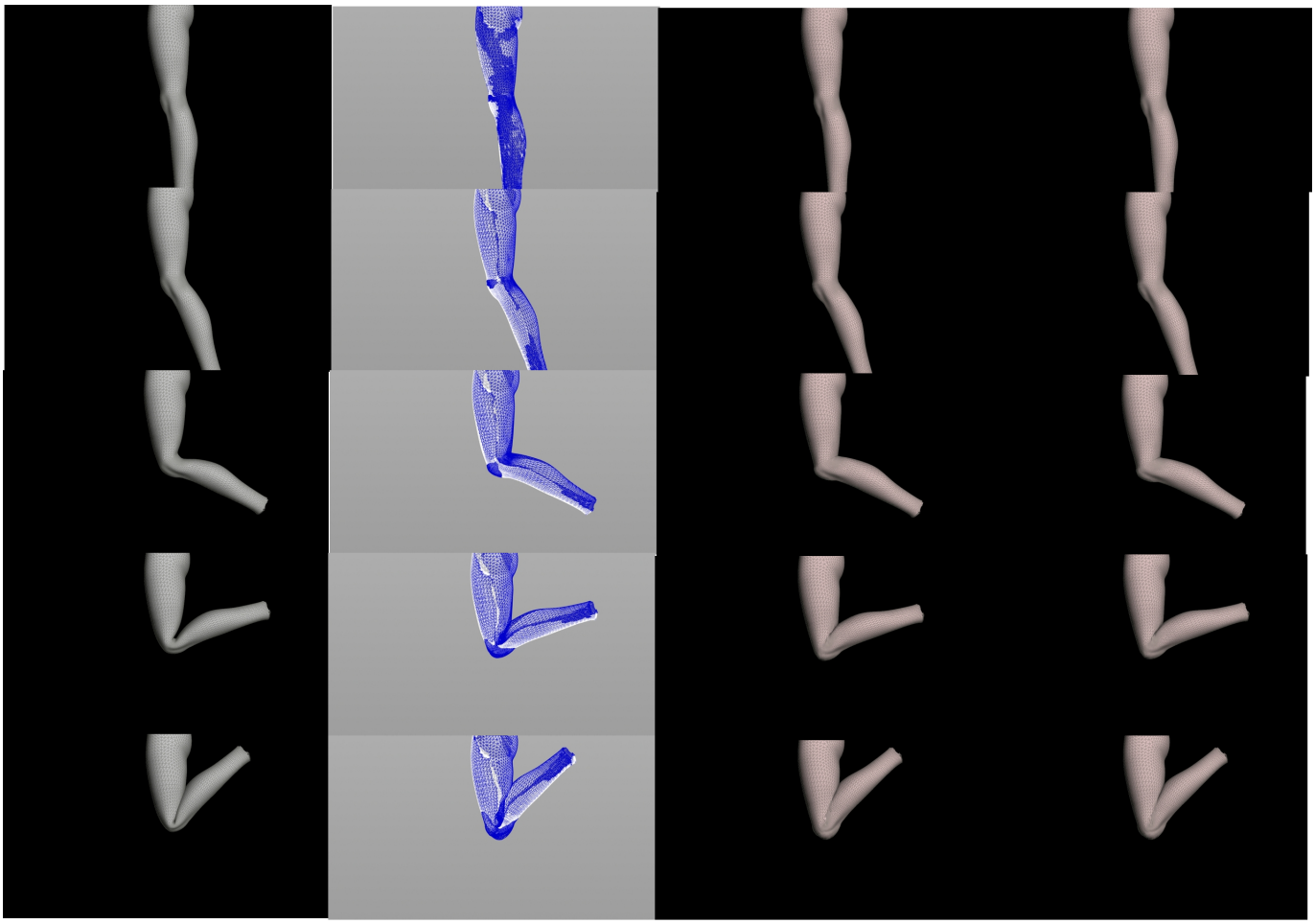


Figure 5.15: *Results comparison for Knee High Poly From left to right : default linear skinning, overlapped wireframe with corrected mesh for comparison, implicit skinning with no smoothing, implicit skinning with 0.5 smoothing*

Chapter 6

Implementation

6.1 Implementation details

6.1.1 Libraries used

For the implementation of the paper a standalone program was created using *C++* and *OpenGL* OpenGL (2013) for the visualization part.

Additional libraries were used as well including :

- AssImp Assimp (2013), to import the collada files from Maya.
- OpenMesh RWTH Graphics (2013), to have an efficient data structure to traverse the mesh, especially querying one-ring neighbours, and to save the objs to disk.
- ISM by Mathieu Sanchez to for the marching cubes algorithm to visualize the isosurfaces.
- hrbf code by Rodolphe Vaillant Vaillant (2013b) to build and query value and gradient of the HRBF composing fields.
- Eigen Eigen (2013) to solve the linear system in the HRBF code.
- NGL Macey (2013), NCCA graphics library that provided the framework to build on.
- Qt Digia (2013), to provide a cross platform environment
- Boost Boost (2013) for utilities and for threads support

6.1.2 Program flow

At first the algorithm was supposed to be implemented on the GPU like in the original paper, but instead a CPU version was created to allow a fast prototyping and have a much

better way of debugging. The flow of the program starts by loading a scene *.dae which will be parsed.

In the first tests a simple skinned cylinder scene was tested but a transformation of 90° was always present after importing. The reason was not *Assimp* fault but rather a problem in the pipeline: when creating a cylinder, *Maya* gives the option of choosing an orientation. If the cylinder is created along x and then a rotation of 90 degrees plus a freeze transformation is performed to make it stand upright, *Maya* stores the matrix and this is passed to *Assimp* even if all the channels are zeroed.

Assimp deals with all the parsing giving access to the skeleton, the mesh, the skinning, the key-frames. In this way it is possible to store at each frame the joint matrices necessary to transform the vertices. These are needed during the linear skinning step which is performed on the CPU, because the algorithm requires access to the transformed mesh and is applied as a post-process per vertex. This solution allows also the default skinned mesh to be saved on disk, something that if done on the GPU would require CUDA or openCL, not a simple shader.

The program automatically finds the segmented mesh obj files in the folder following a naming convention : "joint1" becomes "joint1.obj". The sampler is then called on these meshes and with the centers uniformly distributed the HRBF can be calculated (for the Poisson disk distribution a spatial hash was used to accelerate the neighbours search). These steps are precomputed and include the storing of both the values and gradients (calculated analytically) into 3D textures. This can be done because the functions are locally compact and therefore the bounding box, extended by the individual support r defines the 3D space in which to sample the function.

Depending on the operator chosen, other 3D textures are created for the composed field by following the numerical method presented in Gourmel et al. (2013). This allows to fill each slice of a 3D texture having f_1 , f_2 and θ as u,v,w coordinates and to retrieve values using trilinear interpolation. Also the gradients and the partial derivatives of the opening functions are stored in 1D textures. In the early tests even for low poly models the projection was very slow so after investigating the problem it was found that one of the reason was to test the program in release mode and that the use of textures was the real key to speed. About efficiency the program performs in real time on the test set and this is achieved also by parallelizing the projection algorithm by using multi-threading with the *Boost* libraries. In fact both the marching and the texture filling done as pre-process are multi-threaded.

By having already the textures and having tested the parallelism with *Boost* it shouldn't be difficult to make a GPU implementation.

Regarding the relaxation and smoothing step, because the vertices are processed in parallel, if the algorithms performed in place there would be different results at different executions. To avoid this an additional vertex buffer is necessary and when several iterations are needed it is enough to swap the buffers.

6.1.3 User Interface and debugging

In order to debug properly the algorithm a UI was built (see Fig 6.1) but especially some simple visualization tools that allow to:

- visualize the HRBF samples
- visualize the samples normals
- visualize the isosurfaces (individual and composed)
- visualize at the same moment the default skinned mesh and the corrected one
- visualize the gradient of the composed field
- visualize and cache slices of the composed operator
- visualize and cache the animated isolines for a particular section of a model
- cache the animation

Other debug tools would have been beneficial such as the possibility to visualize the gradient texture, colour the vertices with the βs during tangential relaxation and colouring the vertices during projection distinguishing them in : collided, not modified, and modified. Finally, it would have been beneficial having a system to store the history of the vertex so to trace the path taken while following the gradient.

6.1.4 Design

The design is quite simple and makes use of abstract classes and polymorphisms especially for the hrbf_objects, the hrbf_functions and the operators. Functions and objects in particular use inheritance to specialize themselves into hrbf objects/functions for single functions, binary composed functions and max field functions.

6.1.5 Rendering

The videos shipped with this paper and the images included were rendered in Houdini using the openGL renderer. One of the reasons is that differently from Maya , Houdini supports sequences of objs natively and where Maya crashed importing an obj cached with OpenMesh, Houdini succeeded.
Another reason regarded the possibility to render the meshes easily and fast using the wireframe on shaded style to reveal differences between overlapping models.

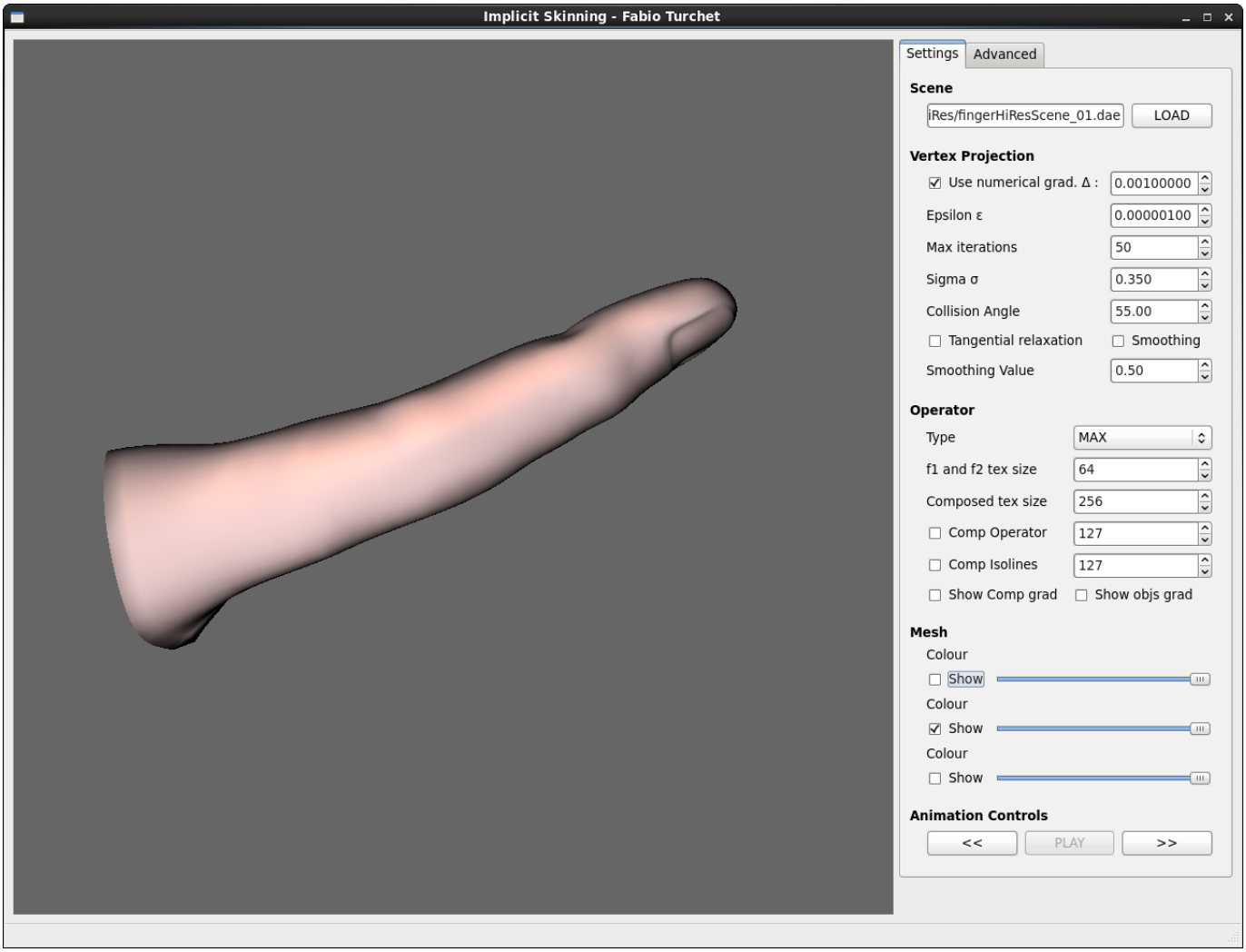


Figure 6.1: *UI of the standalone program created*

Chapter 7

Conclusion

This chapter concludes the thesis and presents known issues and future development paths.

7.1 Summary

In summary, this work achieved the following objectives:

- Implementation and testing of an innovative, brand new technique to correct skinning in character animation.
- Good deformations on models with different resolutions that prove the validity of the technique.
- Verification of the results presented in papers published at SIGGRAPH.
- A framework expandable in the future with new operators.

7.2 Known bugs and issues

Gradient based operators don't work properly, probably due to an implementation issue or because of simple typos in the original papers. Even though the textures are filled in multi-threading, 3D textures for the operators of size 128/256 can take a while to be generated. Tangential relaxation is implemented but not working as expected.
For this project only triangulated meshes attached to simple skeletons where tested, not fully rigged characters. The program should be tested also with dual quaternion skinning, not only linear skinning.
Something to look more deeply into is the reason why sometimes there are little instabilities and poppings during contacts.

7.3 Future work

The results achieved are promising, but a lot of work can still be done especially in terms of optimization. First of all the gradient based operators implementation need to be fixed because this would allow to add for example bulge in contact.
An important development for the future would be to implement a GPU version and a *Maya* / *Houdini* plugin in order to be usable in production. Probably the textures could also be stored on disk to speed up the preprocess step.
More robust and clear debugging tools should be implemented as well especially to diagnose field function problems or to visualize the isosurfaces in real time (marching cubes on the GPU)
Because the results presented make use of the default parameters, probably spending more time tweaking them and finding a good compromise is essential especially in order to produce a tool artist-friendly. Interesting extensions include the addition of implicit surface based physics simulation, the execution of the algorithm on multiple (secondary) characters on the renderfarm and the development of new operators and new silhouette curves.
Finally the technique should be tested on real characters and especially with joint rotations on more than one axis (shoulder), something that probably makes it quite ineffective for facial skinning.

Bibliography

Assimp (2013). 'Assimp - Open Asset Import Library'. Available from: http://assimp.sourceforge.net/ [Accessed 15.08.2013].

O. K.-C. Au, et al. (2008). 'Skeleton Extraction by Mesh Contraction'. *ACM Trans. Graph.* **27**(3).

Autodesk (2013). 'Autodesk®Maya™: 3D Animation, Visual Effects and Compositing Software'. Available from: http://www.autodesk.com/maya [Accessed 15.08.2013].

L. Barthe, et al. (2003). 'Two-dimensional Potential Fields for Advanced Implicit Modeling Operators'. In *Computer Graphics Forum*, vol. 22, pp. 23–33. Wiley Online Library.

Blender Foundation (2013). 'Blender: free and Open 3D creation software'. Available from: http://www.blender.org/ [Accessed 15.08.2013].

J. Bloomenthal (2002). 'Medial-based vertex deformation'. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pp. 147–151, New York, NY, USA. ACM.

Boost (2013). 'Boost C++ libraries'. Available from: http://www.boost.org/ [Accessed 15.08.2013].

F. Canezin, et al. (2013). 'Adequate Inner Bound for Geometric Modeling with Compact Field Functions'. In *SMI '13: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2013*.

D. Cline, et al. (2009). 'Dart throwing on surfaces'. In *Proceedings of the Twentieth Eurographics conference on Rendering*, EGSR'09, pp. 1217–1226, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Collada (2013). 'COLLADA - Digital Asset and FX Exchange Schema'. Available from: https://collada.org/ [Accessed 15.08.2013].

Digia (2013). 'Qt: the modern multi-platform way of writing applications'. Available from: http://qt.digia.com/ [Accessed 15.08.2013].

Eigen (2013). 'Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms'. Available from: http://eigen.tuxfamily.org/ [Accessed 15.08.2013].

O. Gourmel, et al. (2013). 'A gradient-based implicit blend'. *ACM Trans. Graph.* **32**(2):12:1–12:12.

L. Kavan, et al. (2008). 'Geometric skinning with approximate dual quaternion blending'. *ACM Trans. Graph.* **27**(4):105:1–105:23.

L. Kavan & O. Sorkine (2012). 'Elasticity-Inspired Deformers for Character Articulation'. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* **31**(6):196:1–196:8.

J. Macey (2013). 'NCCA graphics library'. Available from: http://nccastaff.bournemouth.ac.uk/jmacey/GraphicsLib/index.html [Accessed 15.08.2013].

I. Macdo, et al. (2011). 'Hermite Radial Basis Functions Implicits'. *Computer Graphics Forum* **30**(1):27–42.

N. Magnenat-thalmann, et al. (1988). 'Joint-Dependent Local Deformations for Hand Animation and Object Grasping'. In *In Proceedings on Graphics interface 88*, pp. 26–33.

A. McAdams, et al. (2011). 'Efficient elasticity for character skinning with contact and collisions'. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 37:1–37:12, New York, NY, USA. ACM.

E. Miller & J. Harkins (2007). 'Musculo-skeletal shape skinning'. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07, New York, NY, USA. ACM.

Miller, Erick (2012). 'Spiderman 3 Muscle and Skin Musculoskeletal Skinning'. Video. Available from: http://www.youtube.com/watch?v=fNCB9wZP8k8 [Accessed 15.08.2013].

OpenGL (2013). 'OpenGL - The Industry Standard for High Performance Graphics'. Available from: http://www.opengl.org/ [Accessed 15.08.2013].

A. Pasko, et al. (1995). 'Function Representation in Geometric Modeling: Concepts, Implementation and Applications'. In *The Visual Computer*, no. 11, pp. 429–446.

A. Ricci (1973). 'A Constructive Geometry for Computer Graphics'. *The Computer Journal* **16**(2):157–160.

RWTH Graphics (2013). 'OpenMesh'. Available from: http://www.openmesh.org/ [Accessed 15.08.2013].

TurboSquid.com (2013). 'Turbosquid, 3D models, textures and plugins'. Available from: https://www.turbosquid.com/index.cfm [Accessed 15.08.2013].

R. Vaillant (2013a). 'Convert implicit surface with global support to compact support'. Available from: http://irit.fr/ Rodolphe.Vaillant/?e=24 [Accessed 15.08.2013].

R. Vaillant (2013b). 'Recipe for implicit surface reconstruction with HRBF'. Available from: irit.fr/ Rodolphe.Vaillant/?e=12 [Accessed 15.08.2013].

R. Vaillant, et al. (2013). 'Implicit skinning: real-time skin deformation with contact modeling'. *ACM Trans. Graph.* **32**(4):125:1–125:12.

Weta Digital, FxguideTV (2013). 'Weta Digital's Tissue System'. Video. Available from: http://www.youtube.com/watch?v=7VlthWa5pu8 [Accessed 15.08.2013].

K. B. White, et al. (2007). 'Poisson Disk Point Sets by Hierarchical Dart Throwing'. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, RT '07, pp. 129–132, Washington, DC, USA. IEEE Computer Society.