Bournemouth University

Faculty of Science and Technology

Implicit skinning

CPU implementation

by

Federico Pietro Leone

MSc Computer Animation and Visual Effects

August 2017

Contents

1 Introduction

1.1 Abstract

In this report a study of the implicit skinning method is presented. After covering the necessary mathematical background, a CPU based implementation is presented. The implementation is not free from flaws, but the reasons behind these limitations are investigated in depth. The work concludes with an analysis of the problems arose during the development and how those problems are addessed. Finally, two new possible applications of the implicit skinning method are briefly described.

1.2 Motivation

Traditional skinning methodologies like LBS Magnenat-thalmann et al. (1988) and dual quaternion L. Kavan, et al. (2008) introduce artefacts like loss of volume or unwanted bulges. These problems can be solved in different ways. The loss of volume, for example can be fixed by redistributing the rotation along multiple joints properly weighted. Other problems can be solved by using corrective blend shapes as a last layer of mesh deformation. An interesting aspect of implicit skinning is that this algorithm tackle all of this problems at once and gives the ability to the animator to add more deformations within a single algorithm. Vaillant, et al. (2013)

Recently MPC F.Turchet et al. (2015). published a paper to improve implicit skinning with wrinkles.
The idea was considered by Vaillant, et al. (2013), who investigated several methods for blending scalar fields. O. Gourmel, et al. (2013). also explores and present a variety of different blending methods in his paper. The easy applicability of any blending method to implicit skinning shows the versatility of this technique. Vaillant, et al. (2013)

2 Related work

Implicit skinning is a corrective algorithm for skinning algorithms like dual quaternion L. Kavan, et al. (2008) and LBS Magnenat-thalmann et al. (1988). The goal of the method is to improve the geometric deformation in order to produce a plausible skin look R. Vaillant, et al. (2013). The goal is accomplished by partitioning the mesh into sub meshes, which are rigidly transformed during the animation. These sub-meshes are also approximated by a scalar field which represents their volume. The single volumes are blended together into a global scalar field. The idea of using meshes in conjunction with implicit representation was already considered in several papers, but the one that is more similar to the implicit skinning approach is the work by Bloomenthal (2002). That approach is able to address the candy wrapper effect and the loss of volumes at the articulations. The main difference is that is not able to detect self-collisions and, more important it manages the mesh differently. The mesh is in facts encoded through its medial axis, while in implicit skinning by look R. Vaillant, et al. (2013) the mesh is immersed into this global scalar field. To each mesh vertex corresponds a iso-value. This iso-value is stored alongside the relative vertex. During animation, the vertices happen to travel far away from their original iso-values, a surface tracking algorithm ensures that the vertices are then moved back to the position of their original iso-values. This means that the mesh is not substituted, but the shape is only corrected. The scalar field can be blended in different ways; the way the scalar field are composed determines the final look of the original mesh. To this end the work by O. Gourmel, et al. (2013) and L. Barthe, et al. (2004) are very important. R. Vaillant, et al. (2013) work is strictly related to those techniques in particular the blending functions based on gradients are explicitly derived from the work of O. Gourmel, et al. (2013). The basic idea of merging scalar field is described by A. Ricci (1973). That approach is also used by R. Vaillant, et al. (2013).

3 Implicit skinning method

Implicit skinning is a corrective methodology used to improve traditional mesh deformation algorithms like LBS and dual quaternion. Vaillant, et al. (2013) These two methods are based on the principle of determine the vertices' location based on a weighted sum of influence objects (joints). The orientation of the joints affects the shape of the mesh producing the effect of animated skin. Both methods are able to apply a major deformation to the skin and successfully pose a character, but the deformations produced are not free from unwanted artefacts. The implicit skinning method not only aims to tackle defects, but is also able to add important details to the animated mesh in order to generate a visually plausible effect of skin deformation.
The improvements to the standard approaches are: self-penetration free deformations at locations where the skin folds, bulges emulating tissues inflation and volume preservation. Unlike physical simulation, algorithms applied are not time dependant and can provide real-time feedback to the artist. Vaillant, et al. (2013).

A key aspect of the implicit skinning method is that it makes use of implicit surfaces to preserve and control the shape of the original geometry. In particular, the method does not substitute the mesh, nor modify its topology. Each vertex of the mesh is assigned to an iso-level. The algorithm preserves this condition throughout the animation by automatically projecting the vertex to appropriate location. Vaillant, et al. (2013)

3.1 Pipeline

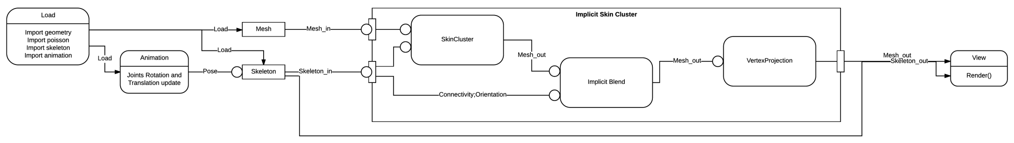

Figure 1: Pipeline diagram.

The general idea is to take an input mesh, creating a signed distance field to approximate each component (relative to a joint), blending the different fields together and, based on the values produced by the global Signed Distance Function, adjusting the original mesh's vertices location.

In the next section it will be explained in detail the implicit skinning method starting from the input required. The most important aspects and factors that characterise this approach will be investigated, and the related techniques described, any time it is necessary.

3.1.1 INPUT

The inputs required by the algorithm are a mesh bound to an animation skeleton (defined as a hierarchy of bones), and skinning weights associated to every vertex. This input can be either provided by the artist or automatically generated. It is also required that the mesh is partitioned into sub-meshes. One for each bone. Of this sub-meshes is not required the whole geometry, but only few sample points. Vaillant prescribes fifty points evenly distributed on the mesh. Vaillant, et al. (2013). For the distribution of the points on the geometry a natural choice is to use the Poisson disk sampling algorithm. K. B. White, et al. (2007). Each of those points is required to have normal's information associated.

3.1.2 Approximation using HRBF

Each sub-mesh provided corresponds to a bone. The sub-meshes are represented by a small set of points evenly distributed on the mesh's surface. The goal is to find a smooth scalar field which goes through the set of input points and whose normals match the points' normal. Moreover, the function must close large holes in the reconstructed meshes and it should also be compactly supported. The need of a compact support will be investigated later into the local vs global support section. Briefly, the compact support is required to allow local composition of the field functions . Each field function approximates a sub-mesh and their composition reconstructs the original input mesh M. In other words, the constitute a partition of the mesh M and the associated to the are composed in a global function which gives an implicit representation of the original mesh M. Vaillant, et al. (2013).

Another advantage of Macedo's approach, I. Macedo, et al. (2011). , is that it allows a simple implementation. The matrix based algorithm which allows the computation of the HRBF derives from functional analysis concepts, and scattered data approximation theory. These results are valid for any dimension of the ambient space. I. Macedo, et al. (2011).

Among the advantages which are more relevant to the implementation of the implicit skinning algorithm, there is HRBF's scale independency and also, its robustness to coarse data implies that it can robustly reconstruct concavities. Vaillant, et al. (2013).

3.1.3 Building the scalar field and interpretation

Following the HRBF formula as in R. Vaillant (2013a).

$$f(x) = \sum_{i}^{N} \alpha_i \varphi(||x - p_i||) + \beta_i * \bigtriangledown[\varphi(||x - p_i||)]$$

Where N is the number of sample points p_i , φ is the radial basis function. $\alpha \epsilon R$ and $\beta \epsilon R^3$ are weights to be found in order to interpolate the input data: pairs of points and normals. $x \epsilon R^3$ is the evaluated position of the ambient space
For the implicit skinning method Valliant choses $\varphi = x^3$.
To notice that β and the $\bigtriangledown$ are two vectors, therefore the product between them is the dot product.The scalar field defined must satisfy certain constraints. In particular:

- $f(p_i = 0$ for each sample point p
- $\bigtriangledown(f(p_i) = n_i)$
- $f(x) < 0$ for each $x \epsilon R^3$
- $f(x) > 0$ for each position $x \epsilon R^3$ valuated outside the object.

This means that on the surface, the scalar field value is 0 ($f(p_i) = 0$) , and the gradient ∇(f(x)) represent the normal to the implicit surface for $\forall$ x | $f(x) = 0$. In particular, if $x = p_i$ then ∇ $(f(p_i)) = n_i$, in other words the surface normal matches the point's normal.
Note: the gradient of a function always points in the direction of the maximum increment of the function. It's easy to see that the function increment is towards the outside of the implicit object, for this reason it's the natural choice to compute the surface's normal.
R. Vaillant (2013a).

Stated the constraints, the unknowns and α and β are found by solving the following linear system with a LU decomposition:

$$\begin{pmatrix} f(p_i) \\ \bigtriangledown f(p_i \end{pmatrix} = \begin{pmatrix} 0 \\ n_i \end{pmatrix}$$

Expanding the notation to the N components:

$$\begin{pmatrix} f(p_0) \\ \vdots \\ f(p_N) \\ \bigtriangledown f(p_0) \\ \vdots \\ \bigtriangledown f(p_N) \end{pmatrix} = \begin{pmatrix} c \\ \vdots \\ c \\ n_0 \\ \vdots \\ n_N \end{pmatrix}$$

Since it is a linear system it is possible to represent it in form of matrix multiplication:

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \\ \beta_{1,x} \\ \beta_{1,y} \\ \beta_{1,z} \\ \vdots \\ \beta_{N,z} \end{pmatrix} = \begin{pmatrix} c \\ \vdots \\ c \\ n_{1,x} \\ n_{1,y} \\ n_{1,z} \\ \vdots \\ n_{N,z} \end{pmatrix}$$

Where A is:

$$\begin{pmatrix}
\Phi(l_1(x_1)) & \dots & \Phi(l_N(x_1)) & e_{1,x}(x_1) & e_{1,y}(x_1) & e_{1,z}(x_1) & \dots & e_{N,z}(x_1) & \\
\Phi(l_1(x_2)) & \dots & \Phi(l_N(x_2)) & e_{1,x}(x_2) & e_{1,y}(x_2) & e_{1,z}(x_2) & \dots & e_{N,z}(x_2) & \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \\
\Phi(l_1(x_N)) & \dots & \Phi(l_N(x_N)) & e_{1,x}(x_N) & e_{1,y}(x_N) & e_{1,z}(x_N) & \dots & e_{N,z}(x_N) & \\
a_1(x_1) & \dots & a_N(x_1) & (b_1 + d_{1,x}d_{1,x} \cdot c_1) & d_{1,y}d_{1,x} \cdot c_1 & d_{1,z}d_{1,x} \cdot c_1 & \dots & (b_N + d_{N,z}d_{1,x} \cdot c_N) & \\
a_1(x_1) & \dots & a_N(x_1) & d_{1,x}d_{1,y} \cdot c_1 & (b_1 + d_{1,y}d_{1,y} \cdot c_1) & d_{1,z}d_{1,y} \cdot c_1 & \dots & (b_N + d_{N,z}d_{1,y} \cdot c_N) & \\
a_1(x_1) & \dots & a_N(x_1) & d_{1,x}d_{1,z} \cdot c_1 & d_{1,y}d_{1,z} \cdot c_1 & (b_1 + d_{1,z}d_{1,z} \cdot c_1) & \dots & (b_N + d_{N,z}d_{1,z} \cdot c_N) & \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_1(x_N) & \dots & a_N(x_N) & d_{1,x}d_{1,z} \cdot c_1 & d_{1,y}d_{1,z} \cdot c_1 & (b_1 + d_{1,z}d_{1,z} \cdot c_1) & \dots & (b_N + d_{N,z}d_{1,z} \cdot c_N) &
\end{pmatrix}$$

Notice that the parameter x_i is implicit. The foloowing expressions are renamed as:

$$b_i(x) = b_i$$

$$d_i(x) = \begin{pmatrix} d_{i,x}(x) \\ d_{i,y}(x) \\ d_{i,z}(x) \end{pmatrix} = x - p_i$$

$$e_i(x) = \begin{pmatrix} e_{i,x}(x) \\ e_{i,y}(x) \\ e_{i,z}(x) \end{pmatrix} = \varphi'(||x - p_i||)\frac{x - p_i}{||x - p_i||} = \varphi'(l_i(x)) \cdot \frac{d_i(x)}{l_i(x)}$$

To see how the matrix A is developed refer to R. Vaillant (2013a).. Gradient and matrix development, instead I will try to give an interpretation of the HRBF formula copied here for convenience:

$$f(x) = \sum_i^N \alpha_i \varphi(||x - p_i||) + \beta_i * \bigtriangledown[\varphi(||x - p_i||)]$$

As previously stated, the scalar field resulting from the summation of the N terms in the above expression produces a smooth surface, which reconstruct the original mesh. But if we consider a single input, the aspect of HRBF is a blob similar to a metaball. The difference between a metaball and the blob produced by the HRBF is that the function of the metaball is bound by its own surface; the function decreases from the centre to the metaball's radius, then stop variating outside the object (locally supported). On the other hand, the HRBF has global support, which means that the function is defined at any point of the space, globally. Another difference between the two functions is that, unlike metaballs', HRBF's centre lays on the surface of the blob and the normal to the point controls the blob's orientation. Finally, the sum of the N blobs reconstructs the original mesh.R. Vaillant (2013a).

In the implicit skinning method, in order to produce a very smooth result, all the samples which are too close to joints are removed from the set of inputs. R. Vaillant, et al. (2013).

$$h < \frac{(v_k - b_i^0)^T (b_i^1 - b_i^0)}{||b_i^1 - b_i^0||^2} < 1 - h$$

3.2 Re-parametrisation: local vs global support skinning and composition into a global field function

So far the K sub-meshes m_i have been sampled by a set of N pairs (point,normal) distributed as in Poisson-disk distribution . K. B. White, et al. (2007). The pairs have been used as an input for the HRBF function whose output is a scalar field which reconstruct the original sub-mesh. So, now there are K non connected HRBFs which approximate the original K sub-meshes. The next step is composing the K HRBFs into a single global field, so that as the union of the K sub-meshes m_i produces the original mesh M, the composition of the K HRBFs produces an approximation to the original mesh M . In order to do that it's necessary to convert the global field functions that we have to compactly supported field functions. The following expression gives the functions re-mapping:

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

Where r sets the size of f_i compact support and it is equal to the maximum distance between the bone and the farthest sampling point used for reconstruction.R. Vaillant, et al. (2013).

3.3 Blending

All the scalar fields generated from the sub-meshes samples are then combined into a single global scalar field. The operation is performed in two steps. First are combined fields relative to adjacent bones, two by two. Then all the resulting fields are combined in a single global field using a union operator like Ricci's max A. Ricci (1973). . In the first step R. Vaillant, et al. (2013) uses a gradient-based blending. The idea was first developed by O. Gourmel, et al. (2013). , and then modified and adapted by R. Vaillant, et al. (2013) for the implicit skinning method. The idea is to use two different blending functions (union and bulge, for example) and calculating an interpolated value between the two of them. The interpolation is driven by the angle formed between the two related joints.

3.4 Surface tracking

The goal now is to deform the character skin to apply the improvements the implicit skinning method is created for. As already said, this method does not substitute the mesh, nor alter its topology. Instead, an iso value is associated to each vertex. The iso value correspond to the iso-level measured at vertex position, when the character's mesh is in default pose. During animation the artefacts due to Dual Quaternion or LBS, pull the vertices away from their original iso-level. The vertex projection method pushes the vertices back onto their original iso-value, forcing them to travel along the field's gradient. R. Vaillant, et al. (2013)

Vertex Projection During the animation the vertices are projected back to their iso-values according to the following formula:

$$v_i \leftarrow v_i + \sigma(f(v_i) - iso_i)\frac{\bigtriangledown f(v_i)}{|| \bigtriangledown f(v_i))||^2}$$

The σ value is a scalar controlling the speed of convergence and the accuracy. Vertex projection is stopped at gradient discontinuities of the scalar field, because those correspond to contact regions. This will prevent the algorithm to produce self-penetrating geometries. R. Vaillant, et al. (2013)

Tangential relaxation Every two steps of vertex projection are interleaved with a tangential relaxation step. This is done because the original and target position of the vertex could be distant from each other. The tangential relaxation step moves the vertex along the tangential plane towards the centroid of its one-ring neighbours

$$v_i \leftarrow (1-\mu)v_i + \mu \sum_j \varphi_{i,j} q_{i,j}$$

$q_{i,j}$ is the one-ring neighbours of v_i projected to its tangential plane. $\varphi_{i,j}$ is the baricentric coordinates that satisfies: $v_i = \sum_j \varphi_{i,j} q_{i,j}$ The value of mu controls the amount of relaxation. It is updated before each iteration and decreases with the distance of the vertices to their target iso-surface. Following the formula as given in R. Vaillant, et al. (2013):

$$\mu = max(0, 1 - (|f(v_i) - iso_i| - 1)^4)$$

Laplacian smoothing A smoothing step is performed after the vertex projection and tangential relaxation steps, when the vertex has reached its destination iso-value. The union operator produces sharp features at contact regions, so in order to better mimic realistic skin these areas around contact regions have to be smoothed. The operation is performed accordingly with the following formula:

$$v_i \leftarrow (1-\beta_i)v_i + \beta_i v_i$$

v_i is the centroid of the one-ring neighbourhood of v_i. The value of beta variates between 0 and 1 and controls the amount of smoothness applied. It is equal to 1 at contact regions, 0 otherwise. These values of beta are then diffused to prevent the introduction of sharp features.

This completes the description of the Implicit skinning method. In the next section are presented additional details about the input and why the standard geometric deformation algorithms (DQs and LBS) need correction R. Vaillant, et al. (2013).

3.5 More about the input

3.5.1 What dual quaternions and linear blending are and why they need correction.

The above picture shows the difference between LBS and dual quaternions. The left picture represent the LBS induced transformation, the right one is the same transformation with dual quaternions. LBS consists in a linear interpolation between the points P_1 and P_2. The output would be a position P_m along this direction.
The right image shows how dual quaternions performs a spherical interpolation rather than a linear one. In this case the output from the algorithm would be a position lying on the arc circle which connects the points P_1 and P_2. The trajectory that the point transformed via dual quaternions would follow, involves two main implications. The first one, dual quaternion addresses the problem of volume loss which affects LBS. The second implication is that, whilst dual quaternions prevents the candy wrapper effect, it also introduces the risk of unwanted bulges when two joints bend. From a performance point of view, the two algorithms perform almost the same, but they differ in how the motions of the joints is represented.
Technically the difference between LBS and dual quaternions is that the first one uses matrices to represent the motion of the joints, the second one uses dual quaternions. Quaternions allow to express easily rotations in three dimensions. Dual quaternions allow to express a rotation and a translation like an affine matrix. Note that a rotation and a translation are exactly what is required to represent a joint. More precisely, dual quaternion, unlike matrices, is guaranteed to represent only a rotation and a translation. For this reason it cannot introduce any scale

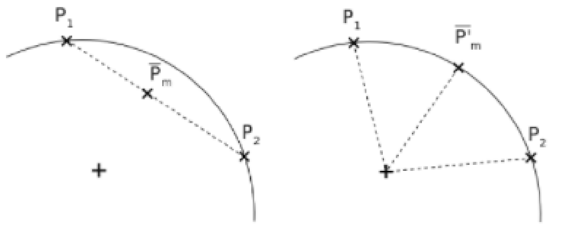

Figure 2: Linear blending and Dual quaternions. The above picture is from L. Kavan, et al. (2008).
.

factor shrinking the mesh joints. R. Vaillant (2013a).

The following formula gives the transformed position of a point using dual quaternions:

$$\dot{q} = \frac{\sum_{i=1}^{n} w_i \dot{q}_i}{|| \sum_{i=1}^{n} w_i \dot{q}_i ||}$$

The idea is to blend dual quaternions, instead of matrices. $\dot{q}$ is the dual quaternion to be used to transform a vertex. $\dot{q}_i$ is the i-th bone (represented by the dual quaternion), which action is weighted by the scalar w_i. The sum is then normalized with $|| \sum_{i=1}^{n} w_i \dot{q}_i ||$

This brief explanation about LBS and dual quaternion shows how the artefacts introduced by the two methods are deeply connected with the methods' nature itself. Implicit skinning is a valid way to overcome these problems. Recently other methods to improve the deformation quality with dual quaternions have been introduced B. Huy Le, et al.(2016).

4 Implementation

The implementation consists in a C++ application, supported by Eigen for matrix calculations and by libigl library, mostly for the visualisation part and some algorithms not directly related to implicit skinning. The input for the software was prepared using Autodesk Maya 2016 Autodesk (2016).

- Eigen: Data storage, matrix calculations Eigen (2017).
- Libigl: Scene visualisation, and algorithms like automatic weights calculation and marching cubes. Libigl(2017)
- Hrbf code by Rodolphe Vaillant. R. Vaillant (2013a).
- Autodesk Maya: modelling and input setup.Autodesk (2016)
- MeshLab: samples based on poisson disk distribution MeshLab(2017)

The application is almost entirely based on matrices for storing information and implement algorithms.

4.1 The implicit skinning pipeline

The application is almost entirely based on matrices for storing information and implement algorithms. Before the data flow starts, information is loaded into data structures (mainly sets of matrices). The implicit skinning algorithm is a corrective method for mesh deformation algorithms, for this reason the input set is not trivial.

The input consists of:

- Main geometry: deformed mesh to be altered.
- Skeleton: represented by a joint hierarchy.
- Sub-meshes: a partition of the main mesh based on the skeleton. One bone is associated to one sub-mesh. Therefore, the number of sub-meshes will be equal to the number of bones.
- Samples with position and normals: for each sub-mesh is required a set of points evenly distributed on the surface. The points will be pairs of position and normals.

Once the inputs are loaded additional information are calculated and stored in opportune structures. The most important information calculated at this stage is the scalar field that approximated each submesh. Other examples are: the skin weights, which calculated by using the Bounded biharmonic weights technique and the joints' adjacency list, which stores information about the bones' structure.

A description of the pipeline follows.

Step 1. The mesh is bound to the skeleton through a skinCluster object. (the name is borrowed from Autodesk 3d applications, Maya in particular uses the skin cluster node to connect geometry to joints)Autodesk (2016) The skinCluster deforms the geometry as the skeleton moves by using the Linear Blending Skinning, algorithm. As described in the chapter about skinning methods, dual quaternion is generally preferable to LBS because it conserves the volume of the mesh during the deformation and, in general, it produces more plausible results which require less work to be fixed. The reason why was choosen LBS instead of dual quaternion is because the corrective action operated by implicit skinning is made more evident and clearly visible.

Step2. The output mesh which comes from the skinCluster is then given as input to the blend function. At the same time, the sub-meshes' scalar filed are rigidly transformed to follow the skeletal animation. The blend function operates on these scalar fields. Scalar fields relative to adjacent sub-meshes are blended together, two by two. The resulting fields are then compound into a global scalar field by using a union operator. At this point iso-values around mesh's vertices are changed. There is a mismatch between the vertices' initial iso-values and the iso-values after the transformations.

Step 3. The vertex projection step, moves the mesh's vertices along the scalar field's gradient, until they reach their initial iso-value. To ensure a good quality of the deformed skin a tangential relaxation step is performed every two projection steps. After this step a local Laplacian smoothing is applied.

Step 4. Visualization. The last step consists in rendering the deformed mesh and skeleton.

4.2 Class implementation

4.2.1 Class diagram overview

The UML diagram shows the class design used for the implementation of the implicit skinning standalone program. External libraries and classes are not shown in the diagram and their instances are shown as common types.

The overall design consists mainly in two different application of the Observer Pattern. The first one is between the Controller, View, Model classes, the second one is between View, SkinCluster and Skeleton.

The Model class is an interface that must be implemented by every class that has content to render. The Observer interface is not displayed in the class diagram for sake of legibility. Nonetheless, View and SkinCluster implement the interface to participate to the Observer Pattern. Subject is the other interface part of the Observer pattern. Since the class skeleton is the only class which implements Subject interface, the implementation is made directly inside the skeleton class itself.

The displayable object are the ones which implement the Model interface. In particular, the classes derived from Basemesh and Skeleton contain the information to be rendered.

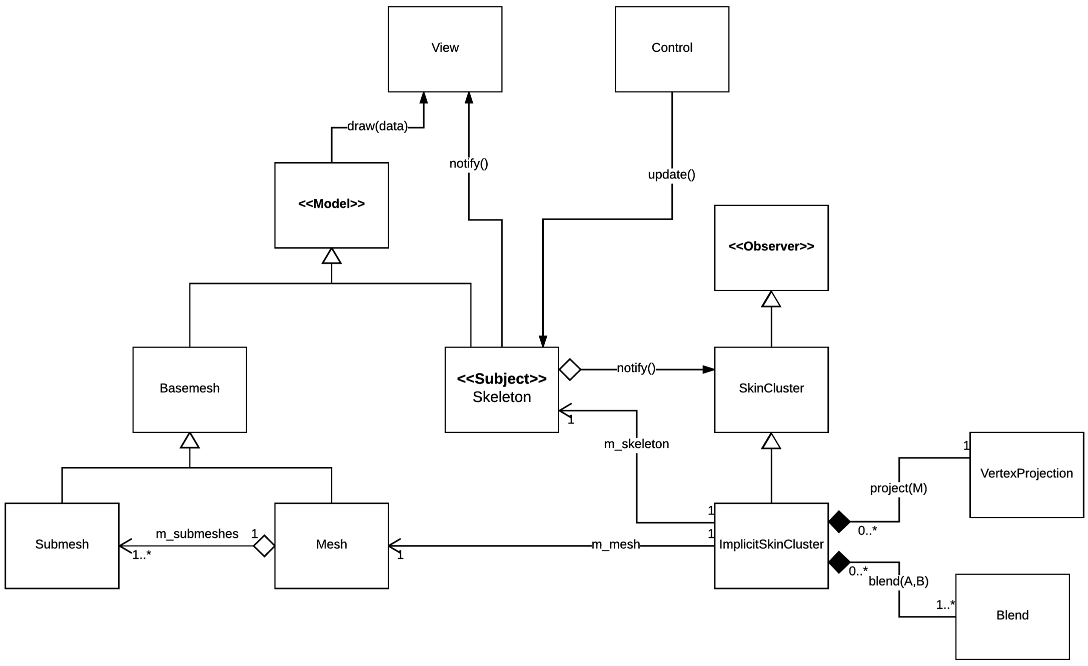

Figure 3: Class diagram.

4.2.2 MVC

Model-View-Control E. Gamma (1994). regulates the interaction between the three relative classes and assigns specific responsibilities to each one of them. In this class design Model is an interface to be implemented by any class which has content to be displayed. The View class has a set of models assigned to itself. At every frame, View read the information from each model-class and displays the content. Control is responsible for uploading the information to the models, make the connection between the View and the models to be rendered and manage the user interaction.

4.2.3 Observer

Every change in the data to be displayed is driven by the skeleton animation. If the skeleton doesn't move, nothing changes. The Observer pattern E. Gamma (1994). allows the subject (the Skeleton) to notify the observers every time a change happened. The observers in this case are the View class, the Mesh class and the SkinCluster class. When the view class is notified that a change happened, it updates its buffers. Otherwise the buffers are reused as long as nothing has changed. Mesh class is responsible to generate and update a grid which contains the sub-meshes' grids. The update is performed only in case something changed in the sub-meshes, i.e. the skeleton has moved. Finally, the SkinCluster update. The skinCluster binds the mesh to the skeleton, and its specialization (ImplicitSkinCluster) performs many other operations in order to modify the mesh's shape. All of those updates are really expensive from a computational point of view. Their execution must be limited as much as possible.

4.2.4 Basemesh, mesh, submeshes

A sub-mesh is a partition of a mesh. The most important difference between mesh and sub-mesh is that the sub-meshes are rigidly transformed and never distorted. They also have a set of sample points used to calculate the scalar field which describes their volume. The mesh is the main geometry, the one that has corrected by the implicit skinning algorithm and modified by LBS or dual quaternions. The scalar field which represent the mesh is computed by blending the sub-meshes' scalar fields. Also the mesh's grid must contain the sub-meshes' grids. But except for these differences, the underlying representation is very similar. They both have vertices, bounding boxes, grids and scalar fields. All this common information, as well as the function to access and manipulate them, are included in the Basemesh class. This setup prevents code duplication and simplifies the debug and code maintenance. The Basemesh class also implements the Model interface to give both the mesh and submesh classes the ability to send data to the View.

4.2.5 SkinCluster and ImplicitSkincluster

SkinCluster and ImplicitSkinCluster are responsible of altering the mesh. SkinCluster connects the skelton with the geometry and modifies the latter by applying LBS. The specialization on SkinCluster, ImplicitSkinCluster, is able to deal with scalar fields. ImplicitSkinCluster combines three classes together. The first one is of course SkinCluster, from which ImplicitSkinCluster derives. The other two classes are Blend, responsible of blending the scalar fields and their gradients; and VertexProjection, responsible for the projection of the mesh's vertices as well as for the smoothing of the mesh. The purpose of ImplicitSkinCluster is to enhance the modifications introduced by its base class. It basically implements the scalar fields blending and the surface tracking algorithm of implicit skinning.

4.3 Results

This implementation of implicit skinning runs entirely on the CPU and uses matrices as primary data structure to store information.

The advantage of this approach is that it permitted rapid prototyping of the application. Results, algorithms and features have been improved ad extended during software development. Matrix representation led to the development of code closely related to the original mathematical concepts and formulas. The choice of using matrices also guaranteed high performances due to the efficient data management that the Eigen library Eigen(2017) provides.

This approach while allowing the development of the entire algorithm, introduce restrictions which heavily affect the final outcome.

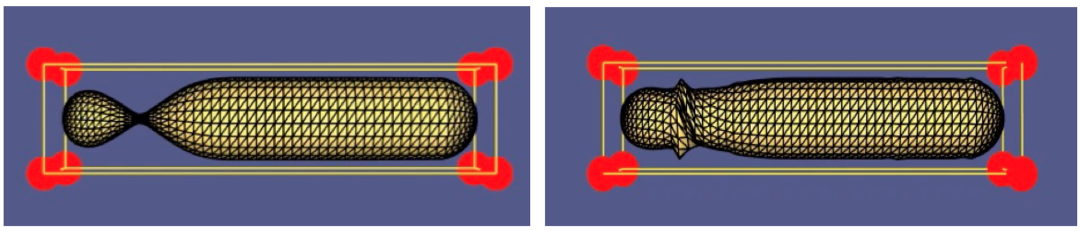

Figure 4: On the left twist with LBS, on the right after the surface tracking.

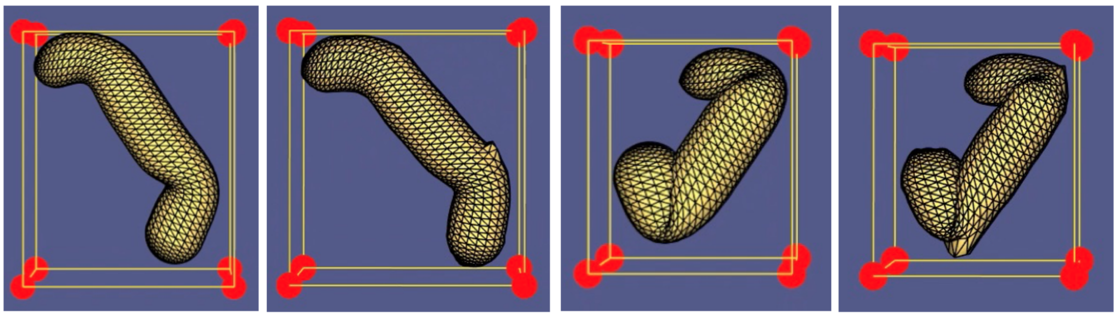

Figure 5: Before and after surface tracking at two different frames.

As is clearly shown in the pictures, several artefacts are introduced by the algorithm. The mesh suffers from severe flickering, and often vertices travel far away from the geometry. The surface produced is generally rough and noisy.

The presence of flaws in the animation is mainly due to the usage of matrices for storing the scalar field and manipulating the data. The original paper implementation relayed on textures of different dimensions to deal with the scalar field, store its values and compute the blending functions R. Vaillant, et al. (2013). .

For my implementation I decided to use simple matrices for a couple of reasons. The first one was simplicity. But also, matrices are efficiently implemented in Eigen library Eigen(2017) and provide a large set of operation fundamental to develop algorithms. I based my design on the assumption that a 1D texture can be naturally translated into a vector, or an array; a 2D texture would be a bi-dimensional matrix; and so on.

The difference between matrices and textures is that, while they are both able to store floating point values at each location, textures offer more important features fundamental for an accurate representation of the scalar field. In particular, most implementations, like OpenGL (2013). support different interpolation methods, which give the ability to manipulate the data, while maintaining a smooth image.

4.3.1 Geometric spatial transformation and image registration

When the skeleton is animated the scalar field changes accordingly. This change is computed onto matrices. Basically the new vertex position of the transformed sub-mesh grid is "expressed" as a position in the mesh grid which

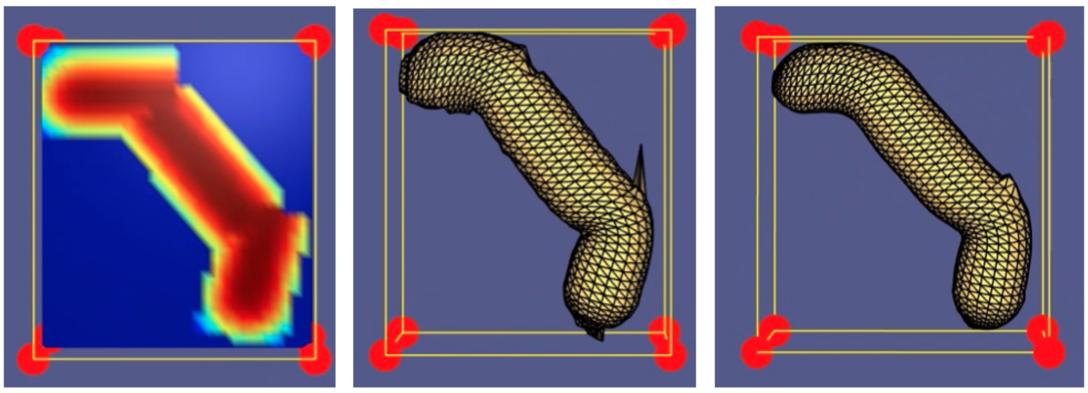
Figure 6: From left to right. Global scalar field, Vertex Projection, Laplacian smoothing

contains all the sub-meshes grids. Another matrix stores the field value and it is used as a look-up table to colour the target grid cell.
When geometric transformations are applied, the spatial relationship between pixels, changes. The transformation induced by the joint rotation, creates a mismatch between the source and the destination grid. The source "pixel" does not overlap to a single "pixel" in the destination matrixR. C. Gonzalez (2008). .

To deal with this situation textures provide different interpolation methods. Linear interpolation is a really common method which produces smooth results when an "image" is transformed. R. C. Gonzalez (2008).

The approach implemented in the application is called forward mapping. It consists in scanning the "pixels" of the "input image" and, for each cell, finding the spatial location of the corresponding pixel in the "target image". A common scenario is that two or more "pixels" would share the same "target pixel", alternatively it is also possible that target locations may not be assigned any values. R. C. Gonzalez (2008).

These problems are intrinsic of the spatial transformations in sampled spaces like matrices. Higher resolutions make the problem less visible, but alone are not enough to solve it. In facts increasing the resolution can only take the same effect to smaller scale.

It remains the problem of a lack of smoothing among cells. In textures this smoothing can be performed in different ways and a "pixel" in the "source image" does not necessarily correspond to a single "pixel" in the destination, resulting in a smooth "image".

The smoothness is a key aspect of the implicit skinning algorithm. The robustness of it is strictly related to how smooth the scalar field is. Despite Vaillant takes all the precautions to prevent sharp features, some operations (like Ricci's max operator) inevitably introduce them. R. Vaillant, et al. (2013). This sharp features cause flickering and faults at troublesome locations of the animated mesh. For this reason, Vaillant introduces Laplacian smoothing in addition to tangential relaxation as final step.

The effect of the above described geometric transformations is to generate sharp features at every spatial location. It results in severe flickering, false updates (the algorithm tries to fix a mesh already smooth), and a general noisy surface.

The surface tracking algorithm, vertex projection in particular, makes every effort to maintain the vertices in their original iso-values. Being the scalar field corrupted as described, it results in unwanted updates and spurious vertex locations. For these reasons the implementation extends the action of Laplacian smoothing to the whole mesh and performs a tangential relaxation step after every vertex projection step. This modification differs from the original implementation, but uses the same tools to address the problems of flickering and noise.

To summarize, low resolution grids and a lack of interpolation in the registration of the scalar field produce sharp edges at every position in the sampled space. The surface tacking algorithms are triggered and move the vertices along the gradient field until they reach their original iso-value. The effect produced reflects the sharp representation of the scalar field, generating flickering and noisy surfaces.

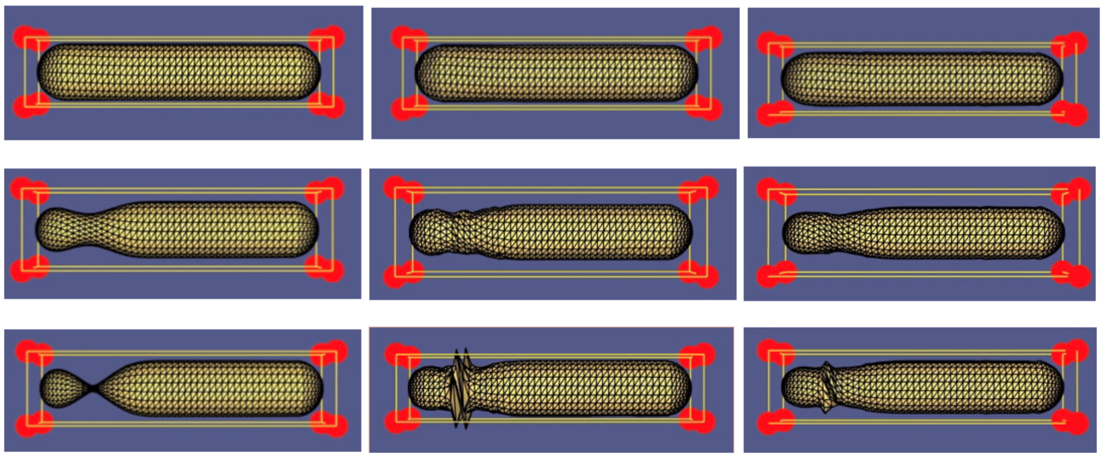
Figure 7: From left to right LBS, after Vertex projection, after Laplacian smoothing.

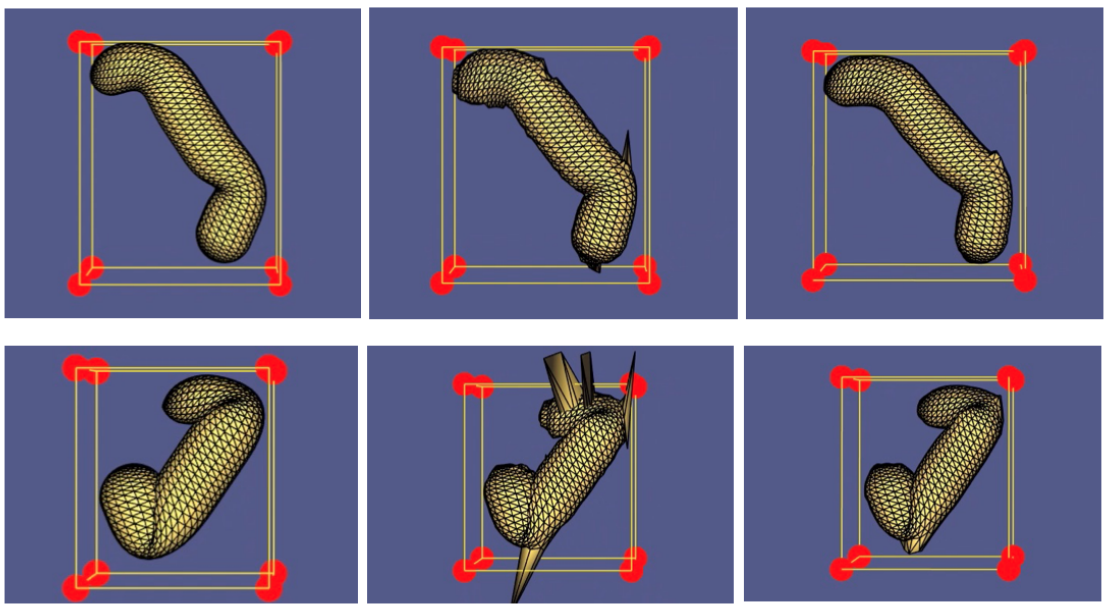
Figure 8: From left to right: LBS, after Vertex projection, after Laplacian smoothing

5 Conclusion

A lot of work can be done to improve the results achieved. The most important problem to address is the scalar field storage. In order to produce a good quality result, working with a smooth scalar field is imperative. Developing am implementation based on textures might give more tools to preserve the smoothness of the function. The most important future development might be a plug in for Maya which would make the tool usable by animators.

5.1 New applications for implicit skinning

Implicit skinning also offers a wide range of possibilities for further improvements and future work. Among the possible applications are the followings.

5.1.1 Implicit skinning for mesh substitution

An interesting aspect of this approach is that, Implicit skinning is a methodology which allows to modify a mesh irrespective of the mesh itself. The mesh is approximated by a scalar field. This step decouples the mesh used to generate the volume from the mesh to be rendered. In character rigging mesh substitution is really common task, and generally transferring weights between meshes doesn't always produce good results. Implicit skinning could provide an effective way to overcome these problems.

5.1.2 Implicit skinning for mesh substitution

An additional interesting possible application is in face rigging. Face rigging is generally based on blend shapes, a technique which allows a fine control over the final result. The drawback of blend shapes is the high number of models to be generated and combined in order to produce plausible results.E. Miller(2008) Using implicit skinning could be possible to produce a broad range of deformations, while providing real time feedback. It would be possible to subdivide the face mesh into areas and connect them with the action of the principal sets of muscles. The partitions would represent the underlying masses which move and create wrinkles (forehead) and bulges(cheeks). The modification required to this approach would mainly be in regard with Gourmel's operator. O. Gourmel, et al. (2013). Gourmel's operator O. Gourmel, et al. (2013). is based on angles, it would be necessary to remap the circle which controls the deformation to a distance range. The muscle range of motion could be a starting point to explore this possibility.

6 References

Autodesk (2013). 'Autodesk Maya: 3D Animation, Visual Effects and Compositing Software'. Available from: https://www.autodesk.co.uk/products/maya/overview [Accessed 21.08.2017].

L. Barthe, et al. (2004). 'Controllable Binary Csg Operators for "soft Objects"' In International Journal of Shape Modeling , World Scientific Publishing Company.

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.

Eigen (2017). 'Eigen Library'. Available from: $http://eigen.tuxfamily.org/index.php?title=Main_page[Accessed 21.08.2017]$.

E. Gamma (1994). 'Design patterns : elements of reusable object-oriented software'. Addison- Weasley professional.

R. C. Gonzalez (2008). ' Digital Image Processing'. Pearson International Edition.

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

B. Huy Le, et al.(2016). 'Real-time skeletal skinning with optimized centers of rotation'. ACM Trans. Graph.

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

Libigl (2017). ' libigl - A simple C++ geometry processing library'. Available from: http://libigl.github.io/libigl/ [Accessed 21.08.2017].

I. Macedo, 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.

MeshLab (2017). 'MeshLab'. Available from: http://www.meshlab.net/ [Accessed 21.08.2017].

E. Miller(2008) . 'Maya Hyper-realistic Creature Creation: A Hands-on Introduction to Key Tools and Techniques in Autodesk Maya (Autodesk Maya Techniques: Offical Autodesk Training Guides)'. Autodesk.

OpenGL (2016). 'OpenGL - The Industry Standard for High Performance Graphics'. Available from: http://www.opengl.org/ [Accessed 21.08.2017].

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

F.Turchet et al. (2015). 'Extending implicit skinning with wrinkles'. ACM Trans. Graph.

R. Vaillant (2013a). 'Recipe for implicit surface reconstruction with HRBF'. Available from: http://rodolphe-vaillant.fr/?e=12 [Accessed 21.08.2017].

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

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.