A framework for geometric modeling based on physical fields

Fokko P. Beekhof[*]and Guntram Berti[†]

April 15, 2011

1 Introduction

This document describes a framework to generate smooth geometries in 3D space as level sets of field-strength functions. In addition, it outlines a reference implementation of this method in a class library. The strength of the field-based approach is the ease with which to obtain blends of unions or intersections, which is particularly attractive in application domains where exact set-theoretic operations are not required. Mathematically, our basic approach is equivalent to that introduced by Ricci [1], but we offer additional methods to control global effects of local changes.

One application domain where our approach proves particularly apt is the field of medical simulation.

Using geometries based on real medical data for fluid flows in the scientific community has proven to be difficult, because such data cannot be generated at arbitrary precision, cannot be freely shared for the purpose of cross-validation, may suffer from defects, and does not come in a format that is convenient for all simulation platforms. In contrast, synthetically defined yet realistic geometries suffer none of these drawbacks. The purpose of this package is to provide the research community with a way to standardize benchmarks by having the ability to define a 3D geometry with infinite precision, without privacy-issues and with a natural form of smoothing. In any publication that reports CFD results, the XML to create a geometry used in CFD simulations can be included or referred to so that others can easily run simulations using exactly the same geometry.

Shapes is 1) a mathematical model to implicitly describe smooth 3D geometries; 2) a standardized way to define such geometries in human-readable XML files, and 3) software to convert such XML descriptions to several existing formats to interoperate with other software.

2 Mathematical model

We use a definition of geometry that is loosely based on the idea of physical field strength. If a field strength is given by a function F, we define its associated shape or volume as

$$V_F = \{p \in \mathbb{R}^n | F(p) \leq 1\}$$

[*]University of Geneva

[†]Consulting in mathematical methods (CMM), Bonn, Germany

For instance, the shape V_F associated to

$$F(p; R, c) = (R/\|p - c\|)^2 \tag{1}$$

is a sphere centered at c with radius R, and corresponds to the volume where the field strength of an electric particle centered at c exceeds a certain threshold.

In general, for F to be admissible in our system, we require F to be smooth, $F(x) \geq 0$ and $F(x) \to 0$ as $\|x\| \to \infty$ where $x = p - c$.

A *blended union* $V_f \sqcup V_g$ of two volumes V_f and V_g in this approach is in its simplest form given by the volume V_{f+g}, intuitively by adding the field strengths. The ease of obtaining smooth blendings is one of the most attractive properties of this function system.

Likewise, we can define a *blended intersection* $V_f \sqcap V_g$ as $V_{(1/f+1/g)^{-1}}$, noting that $V_{1/f}$ is the complement of the interior of V_f and using the identity $A \cap B = \overline{\bar{A} \cup \bar{B}}$, if $\bar{A}$ denotes the complement of the set A.

Finally, a *blended difference* can be defined based on the identity $V_f \setminus V_g = V_f \cap \overline{V_g}$ as $V_f \setminus_1 V_g = V_{(1/f+g)^{-1}}$.

However, there are some problems with this simple approach:

1. The amout of blending is difficult to control
2. The "natural" field functions (corresponding to physical fields), such as for the sphere in Eqn. 1, have non-compact support and thus influence the shape everywhere, making it impossible to add a small detail with only local effects.
3. These same "natural" field functions have a singularity in the center, which may cause problems in implementations.

Fortunately, these problems can be solved by appropriate modifications to the underlying functions. In Section 2.1 we address the blending control issue, and in Section 2.2 we introduce a way to control and reduce the global influence of a shape. A graceful method to deal with singularities is proposed in Section 4.1.

2.1 Controling the amount of blending

Instead of taking the sum $u(f, g) = f + g$ of two functions to yield the blended union $V_{u(f,g)} = V_f \sqcup V_g$ of V_f and V_g, we can use

$$u_q(f, g) = (f^q + g^q)^{1/q} \tag{2}$$

for some parameter $q > 0$, giving the parameterised blended union $V_f \sqcup_q V_g = V_{u_q(f,g)}$. Here $q = 1$ yields the original definition, larger values of q reduce the amount of blending (cf. Fig. 1), and for $q \to \infty$, we obtain the usual set theoretic union:

$$\begin{align} u_\infty(f, g) &:= \lim_{q\to\infty} u_q(f, g) = \max(f, g) \tag{3} \\ V_f \sqcup_\infty V_g &= V_{\max(f,g)} = V_f \cup V_g \tag{4} \end{align}$$

Similarly, we define

$$\begin{align} i_q(f, g) &:= (f^{-q} + g^{-q})^{-1/q} \tag{5} \\ V_f \sqcap_q V_g &:= V_{i_q(f,g)} \tag{6} \\ i_\infty &:= \lim_{q\to\infty} i_q(f, g) = \min(f, g) \tag{7} \\ & \tag{8} \end{align}$$

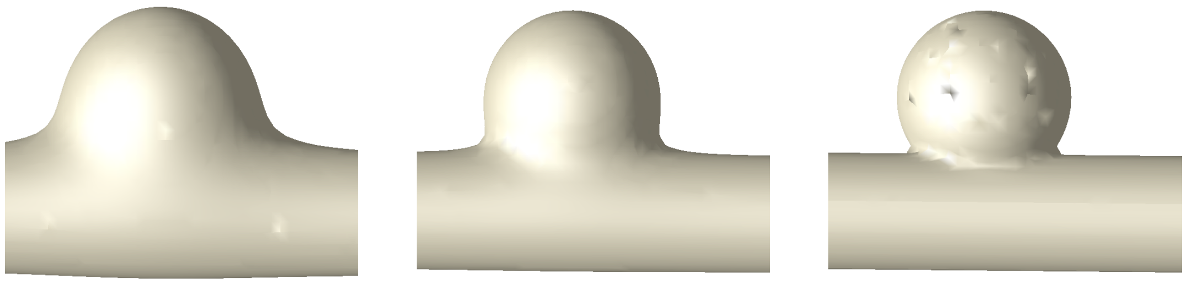

Figure 1: Controling the blending of the union of a sphere and a tube. Cases $q = 1$, $q = 2$ and $q = 10$ (left to right).

and we have $V_f \sqcap_\infty V_g = V_f \cap V_g$.

Finally, we can generalize the *blended difference* $V_f \setminus_q V_g$ as $V_f \sqcap_q V_{1/g}$, where again $V_f \setminus_\infty V_g = V_f \setminus V_g$:

$$\begin{aligned} d_q(f,g) &:= (f^{-q} + g^q)^{-1/q} && (9) \\ V_f \setminus_q V_g &:= V_{d_q(f,g)} && (10) \\ d_\infty &:= \lim_{q\to\infty} d_q(f,g) = \min(f, 1/g) && (11) \end{aligned}$$

We note that for $q < p$

$$\begin{aligned} V_f \sqcup_q V_g &\supset V_f \sqcup_p V_g && (12) \\ V_f \sqcap_q V_g &\subset V_f \sqcap_p V_g && (13) \\ V_f \setminus_q V_g = V_f \sqcap_q V_{1/g} &\subset V_f \sqcap_p V_{1/g} = V_f \setminus_p V_g && (14) \end{aligned}$$

and in particular

$$\begin{aligned} V_f \sqcup_q V_g &\supset V_f \cup V_g && (15) \\ V_f \sqcap_q V_g &\subset V_f \cap V_g && (16) \\ V_f \setminus_q V_g &\subset V_f \setminus V_g && (17) \end{aligned}$$

See the appendix for a proof.

In comparison to the usual approach using R-functions (see e. g. [2]), we can obtain the set-theoretic operations only at the expense of letting $q \to \infty$ and thus loosing smoothness. However, the strict set-theoretic operations are often not necessary or not wanted. For instance, if the task is to generate artifical anatomical structure, sharp features are in general not realistic, and thus a certain amount of blending is necessary, which can be obtained naturally using the method presented here.

2.2 Controling the range of influence

If the functions f have non-compact support, it means that they influence also parts of the geometry far outside V_f, which is often unwanted. Therefore, what is needed is a way to *decay* such a function to zero outside a given range around the volume.[1]

A simple approach is to modify the function f with a positive, monotonic damping function d operating on the range of f:

$$\begin{aligned} d:\mathbb{R}^+ &\mapsto \mathbb{R}^+ \\ \tilde{f}(x) &= d\circ f(x) = d(f(x)) \end{aligned}$$

The damped function $\tilde{f}$ should fulfill the following requirements:

1. $V_{\tilde{f}} = V_f$, that is, $\tilde{f}-1$ has the same sign as $f-1$. In particular, $\tilde{f}(x)=1 \Leftrightarrow f(x)=1$, and thus $d(1)=1$.

2. $\tilde{f}(x)=0$ if $f(x)<\epsilon$ for some ϵ with $0<\epsilon<1$, so $d(y)=0$ for $y<\epsilon$.

3. $\tilde{f}$ and hence d should be smooth, i.e. two times continously differentiable, and be as monotonic as f (without unnecessary turning points etc.).

4. $\tilde{f}$ should be close to f near the boundary of V_f, i.e. where $f(x)=1$, so preferably $d'(1)=1$. This reduces the effect of damping on the combinations of volumes using blended unions and intersections.

If ϵ is fixed, a simple approach would be to use $d=d_1$ with

$$d_1(y) = d_1(y;\epsilon) = \begin{cases} 0 & \text{if} \quad y\le\epsilon \\ \frac{y-\epsilon}{1-\epsilon} & \text{else} \end{cases} \tag{18}$$

This, however, is not smooth at $y=\epsilon$. In order to ensure smoothness, we can exponentiate the expression:

$$d_2(y;\epsilon,p) = d_1(y;\epsilon)^p \quad \text{with} \quad p\ge 1 \tag{19}$$

The larger p, the better the smoothness. However, near $f(x)=1$, a large p will interfere with the blending strength. Therefore, we introduce a variable exponent $p=p(y)$ depending on the argument y of d. We would like to have

$$p(1)=p_1 \quad \text{and} \quad p(\epsilon)=p_\epsilon>1 \tag{20}$$

In order to do this, we introduce the auxilliary function

$$s(t) = 3/2t^2 - 1/2t^3 \tag{21}$$

fulfilling $s(0)=s'(0)=s''(0)=0, s(1)=1, s''(1)=0$. We set $S(y,\epsilon)=s((1-y)/(1-\epsilon))$ and

$$p(y;\epsilon,p_1,p_\epsilon) = \begin{cases} p_1 & \text{if} \quad y\ge 1 \\ p_1+(p_\epsilon-p_1)S(y) & \text{else} \end{cases} \tag{22}$$

fulfilling 20. The final damping function $d=d(y;\epsilon,p_1,p_\epsilon)$ is defined as

$$d(y;\epsilon,p_1,p_\epsilon) = d_1(y;\epsilon)^{p(y;\epsilon,p_1,p_\epsilon)} \tag{23}$$

[1]If we adopt formally the notions $1/\infty=0$ and $1/0=\infty$, we can apply the blended intersection operation also on the modified functions. In an implementation we have to use a case analysis to check for a vanishing function.

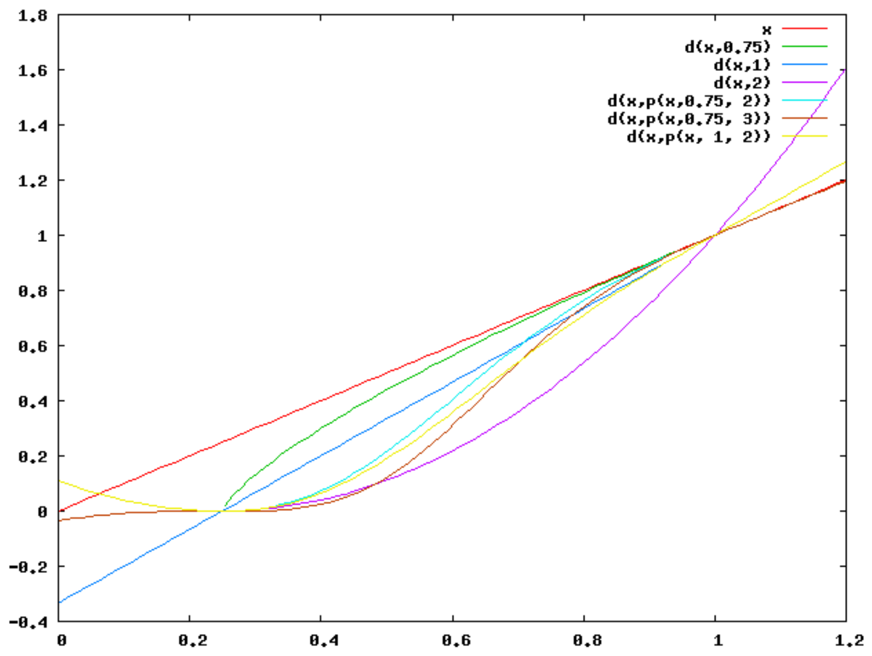

Figure 2: Damping function d ($\epsilon = 0.25$)

We would like to have $d'(1) = 1$ in order to preserve the amount of blending as much as possible. We have

$$\begin{aligned} d'(y) &= \frac{d}{dy} d_1(y)^{p(y)} = d_1(y)^{p(y)} \left(p'(y) \log d_1(y) + p(y) \frac{d_1'(y)}{d_1(y)} \right) && (24) \\ d'(1) &= \underbrace{d_1(1)^{p(1)}}_{1} \left(p'(1) \underbrace{\log d_1(1)}_{0} + p(1) \frac{d_1'(1)}{d_1(1)} \right) \\ &= p(1) d_1'(1) \\ &= p_1 \cdot 1/(1-\epsilon) \\ \text{so} \quad & d'(1) = 1 \Rightarrow p_1 = 1 - \epsilon && (25) \end{aligned}$$

Setting $p_1 = 1 - \epsilon$ will ensure $d'(1) = 1$, see Fig. 2.

An application of this damping to the function $f(x) = 1/x$ is shown in figure 3.

3 Practical Shape Representation

The top level structure is a (abstract) `Shape`, which holds a tree of concrete structures. The distance field generated by a shape in a point $\vec{p}$ is given by the distance field function F_s of p of this tree. A shape can be defined in an XML-file or built programmatically using combinations of individual primitives.

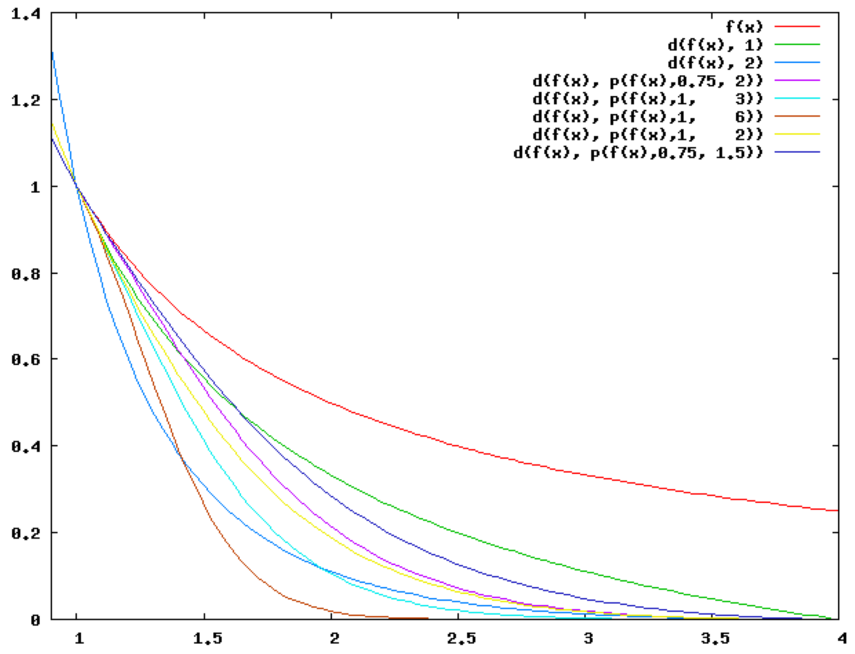

Figure 3: Damping result for $f(x) = 1/x$ ($\epsilon = 0.25$). Note how a $p_1 = 1 - \epsilon = 0.75$ ensures tangency at $f(x) = 1$.

3.1 The Shape and Structures

There are several types of structures: primitives such as Spheres and Tubes, and combinators such as Unions, Intersections and Differences, which are the blended versions of unions, intersections, and difference defined above. Although there are only two primitives currently defined, it is trivial to extend the framework with new primitives. One can imagine, for example, primitives based on cubes.

Each structure generates a distance field. Each structure has a function f to compute its *raw value*, and may have a damping field to allow for a smooth decay to zero, which is given as in Equation 23. If such a field is active, parameters d_{low} and d_{high} must be defined with $0 < d_{low} \leq d_{high}$. If d_{low} and d_{high} are both defined as one, the damping field is not active. Here d_{low} corresponds to p_1 and d_{high} corresponds to p_ϵ in Eqn. 23, and ϵ is set to $1 - p_1$.

3.2 Point

A (control) Point is not a structure in itself, but it is the building block of the Spheres and Tubes. A Point defines a center $\vec{c}$, weights $\vec{w}$ ($w_i = 1$ by default), an orientation $\vec{v_1}$, $\vec{v_2}$ and $\vec{v_3}$ (where $\vec{v_i}$ are the Cartesian unit vectors $\vec{e_i}$ by default), a radius R, and an exponent e ($= 2$ by default).

The center is the location of the Point in space. The weights are normalized and control the shape of the Point. The radius determines the size.

The orientation is a tuple of a rotation axis and an angle, so that the orientation of the point can be controlled by a local rotation R. Alternatively, the orientation can be seen as a set of three direction vectors, linked to R by the convention $e_i = Rv_i$, with the Cartesian unit vectors e_i. (If we rotate an ellipsoid E with the main axes e_1, e_2, e_3 given by $f(x, y, z) = 0$ to an ellipsoid E_R with main axes v_1, v_2, v_3, E' is given by the function f defined by $f_R(xv_1 + yv_2 + zv_3) = f(x, y, z)$, because $f(w_i e_i) = 0 \Leftrightarrow f_R(w_i v_i) = 0$. Defining R by letting $Re_i = v_i$, we have $f_R(R\vec{x}) = f(x)$ or $f_R(\vec{x}) = f(R^{-1}x)$.)

These representations are fully equivalent[3]. In the mathematical model, the representation in the form of direction vectors is used. In the XML standard and implementation, the representation in the form of a rotation axis and angle is used as it is was found to be more convenient for users.

Using the exponent, one can control the rate of decay of the field strength. Using a higher exponent will result in a much more rapid decay towards zero. Nonetheless, the influence of the field reaches out to infinity unless a dampening field is used.

The weight factor w_i gives the scaling of the Point's diameter in the direction of its main axis $\vec{v}_i$.

3.3 Spheres

A Sphere is formed by a single Point and an optional damping field. For a given point $\vec{p}$ in space, the raw value is computed as:

$$f(\vec{p}) = \left(\frac{R^2}{\sum_{i=1}^{3} \left(\vec{v}_i \cdot (\vec{c} - \vec{p})/w_i\right)^2} \right)^{e/2}$$

3.4 Tubes

A Tube is formed by taking a natural cubic spline through at least three Points, interpolating the spatial coordinates as well as all parameters of the Points, and an optional damping field. This results in a set of at least two Segments between the consecutive Points. The parameters in Segments are determined by the splines, which are functions of t, where $t \in [0, 1]$, so the center line of a Segment is given by the function $\vec{c}(t)$. The distance field generated in a certain point $\vec{p}$ in space by a Tube is the maximum of the distance fields generated by its individual Segments.

For a given Segment and a given point $\vec{p}$ in space, a corresponding t_p is found as the value of t for which the distance between $\vec{p}$ and $\vec{c}(t)$ is minimal, of course respecting the restriction $t \in [0, 1]$. For points not projecting to any Segment, in order to ensure continuity, t is set to 0 or 1, i.e. the distance function to the first or last point of the Tube is used.

We assume that t_p is uniquely defined for all $\vec{p}$ insided the tube, that is, the tube is thin enough with respect to its curvature. In order to avoid discontinuities in parts of the geometry farther away from the tube, one should use a damping function to decay the tube pseudo-distance f to zero in regions where t_p might jump.

This strategy cannot be used for the region near the endpoints, if both ends are closer than the sum of the corresponding distances. In this case, the surface will be intersected by the locus of points which are exactly at the same distance of both end points, and hence the function f as the maximum of two local function will not be smooth there, leading to a sharp edge. This can be avoided only be introducing closed (periodic) splines, which is a possible future extension.

Given these limitations, the value of f is computed as follows: Let $\vec{x} = \vec{c}(t_p) - \vec{p}$. The computation of the value of the Segment is then almost equivalent to the computation of the value of a Sphere, except that all parameters are functions of t:

$$f(\vec{p}) = \left(\frac{R^2(t_p)}{\sum_{i=1}^{3} \left(\vec{v}_i(t_p) \cdot \vec{x})/w_i(t_p)\right)^2} \right)^{e/2}$$

3.5 Union

A Union implements the blended union of one or more structures and a damping field. A blended union has the parameters to set up the damping field, an exponent q and a set of structures S. The raw value f_u generated by a blended union in a given point $\vec{p}$ in space is defined as:

$$f_u(\vec{p}) = \left(\sum_{s \in S} [f_s(\vec{p})]^q \right)^{1/q}$$

Note that Unions are recursive structures, a Union may contain another Union. The number of structures in a Union is not limited to two.

3.6 Intersection

An Intersection has the same parameters as a Union and implements a blended intersection. The raw value f_i of the Intersection is computed as

$$f_i(\vec{p}) = \left(\sum_{s \in S} [f_s(\vec{p})]^{-q} \right)^{-1/q}$$

3.7 Difference

A Difference implements the blended difference of exactly two structures, but otherwise has the same parameters as a Union. The raw value f_d of a Difference is computed as

$$f_d(\vec{p}) = \left(f(\vec{p})^{-q} + g(\vec{p})^q \right)^{-1/q}$$

4 Implementation

The purpose of an implementation of the proposed method in software is to allow conversion from the infinitely precise implicit definition of a shape to an explicit representation suitable for processing by, for example, software that is part of a Computational Fluid Dynamics pipeline.

When converting from an implicit definition to an explicit representation using numerical methods and sampling, the introduction of errors can be expected. The implementation has two obvious sources of error; first the user-supplied sampling interval, and second the issue of singularities. Singularities refer to areas where computers are unable to correctly express the values required for the evaluation of the field, such as infinity.

In the reference implementation, singularities are dealt with by capping the values in an elegant and smooth manner as detailed in Section 4.1.

4.1 Avoiding singularities

Singularities are not a big issue when we only deal with unions, because then all singularities are well inside the defined volume. They become more critical however in the presence of intersections or differences, where they may occur close to or even on the boundaries.

A simple, though not smooth, way would be to cap f at a large positive constant γ:

$$\tilde{f}(x) = \begin{cases} f(x) & \text{if } f(x) < \gamma \\ \gamma & \text{if } f(x) \geq \gamma \end{cases} \tag{26}$$

We can smooth this simple definition by using a transition interval $[\gamma - \delta, \gamma + \delta]$ around γ for values of f in that range. If we are given a smooth, monotonic function s with

$$s(f) = \begin{cases} f & \text{if } f < \gamma - \delta \\ \gamma & \text{if } f \geq \gamma + \delta \end{cases} \tag{27}$$

we simply choose the composition $\tilde{f} = s \circ f$. The quartic $S(x) = x^3 - x^4/2$ satisfies $S(0) = S'(0) = S''(0) = 0 = S''(1), S'(1) = 1, S(1) = 1/2$. If we define s as

$$s(f) = \begin{cases} f & \text{if } f < \gamma - \delta \\ f - 2\delta S((f - \gamma + \delta)/2\delta) & \text{if } \gamma - \delta \leq f \leq \gamma + \delta \\ \gamma & \text{if } f > \gamma + \delta \end{cases} \tag{28}$$

we get a C^2-smooth capping function for f. This approach is implemented in the reference implementation.

4.2 Reference Implementation

A reference implementation is available, written in C++, licenced under the GPL. All information about the reference implementation is maintained on: `http://tech.unige.ch/shapes`.

5 Conclusions

5.1 Possible extensions

There are a number of potentially useful extension. First, there should probably support for automatically computing a damping function to ensure tubes do not generated non-smooth features far away. Similarly, closed splines should be supported to avoid problems with close endpoints.

When defining a system of directions v_i for tubes, it is likely the intention to set them up in a way that they are swept orthogonally along the spline curve. This could be supported by defining them in term of the curve tangent, normal and binormal.

Finally, an number of other primitives are conceivable. A complete system of function permitting to approximate just about any shape could be useful to mimic real geometries to any desired level of accuracy.

References

[1] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[2] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11:429–446, 1995. 10.1007/BF02464333.

[3] Serge Belongie. Rodrigues' rotation formula. `http://mathworld.wolfram.com/RodriguesRotationFormula.html`. MathWorld.

A File Format

A.1 Example: Union

Here is an example of an XML file describing a shape.

```
<?xml version="1.0" ?>

<Shape>
    <Union>
        <Exponent>4</Exponent>

        <Tube>
    <Name>Vessel</Name>
            <Point>
                <Center>-50 0 0</Center>
                <Radius>4</Radius>
            </Point>
            <Point>
                <Center>0 0 0</Center>
                <Radius>4</Radius>
            </Point>
            <Point>
                <Center>50 0 0</Center>
                <Radius>4</Radius>
            </Point>
        </Tube>

        <Sphere>
    <Name>Aneurysm</Name>
            <Center>0 10 0</Center>
            <Radius>6</Radius>
        </Sphere>

        <Sphere>
            <Center>0 10 6</Center>
            <Radius>2</Radius>
        </Sphere>
    </Union>
</Shape>
```

A.2 Example: Difference

Here is an example of an XML file describing a shape.

```
<?xml version="1.0" ?>
<Shape>
    <Difference>
        <Exponent>16</Exponent>
        <DampLow>1.0</DampLow>
        <DampHigh>1.0</DampHigh>

<Plus>
<Tube>
    <DampLow>1.0</DampLow>
    <DampHigh>1.0</DampHigh>

    <Point>
<Center>-50 0 0</Center>
<Weight>1 1 1</Weight>
<Orientation>
    <Axis>1 0 0</Axis>
    <Axis>0 1 0</Axis>
    <Axis>0 0 1</Axis>
</Orientation>
<Radius>4</Radius>
<Exponent>2</Exponent>
    </Point>
    <Point>
<Center>0 0 0</Center>
<Weight>1 1 1</Weight>
<Orientation>
    <Axis>1 0 0</Axis>
    <Axis>0 1 0</Axis>
    <Axis>0 0 1</Axis>
</Orientation>
<Radius>6</Radius>
<Exponent>2</Exponent>
    </Point>
    <Point>
<Center>50 0 0</Center>
<Weight>1 1 1</Weight>
<Orientation>
    <Axis>4 0 0</Axis>
    <Axis>0 3 0</Axis>
    <Axis>0 0 4</Axis>
</Orientation>
<Radius>8</Radius>
<Exponent>2</Exponent>
    </Point>
</Tube>
```

```
</Plus>

<Minus>
<Sphere>
    <DampLow>1.0</DampLow>
    <DampHigh>1.0</DampHigh>

    <Center>0 5 0</Center>
    <Weight>1 1 1</Weight>
    <Orientation>
<Axis>1 0 0</Axis>
<Axis>0 1 0</Axis>
<Axis>0 0 1</Axis>
    </Orientation>
    <Radius>6</Radius>
    <Exponent>2</Exponent>
</Sphere>
</Minus>
    </Difference>
</Shape>
```

B Proof of Equation (12)

We prove that

$$(x^q + y^q)^{1/q} > (x^p + y^p)^{1/p} \tag{29}$$

if $x, y > 0$ and $q < p$. Let without loss of generality $x \geq y$ then $(x^q+y^q)^{1/q} = x(1+(y/x)^q)^{1/q}$, and letting $y/x = t$, we have to prove

$$(1 + t^q)^{1/q} > (1 + t^p)^{1/p} \tag{30}$$

for $0 < t \leq 1$ and $q < p$. We show that the derivative of $f(q) = (1 + t^q)^{1/q}$ is negative, from which follows the conclusion.

$$\begin{aligned}
f(q) &= (1 + t^q)^{1/q} = \exp(1/q \log(1 + \exp(q \log t))) \\
f'(q) &= f(q)(-1/q^2 \log(1 + \exp(q \log t) + 1/q((1 + \exp(q \log t))^{-1} \log t \exp(q \log t)) \\
&= f(q)(\underbrace{-1/q^2 \log(1 + t^q)}_{<0} + 1/q((1 + t^q)^{-1} \underbrace{\log(t)}_{\leq 0} t^q))) \\
&< 0
\end{aligned}$$