Extracting Curves from Subdivision Surfaces

Written by Bryce Summers (BryceSummers.com), Advised by Keenan Crane

Last Updated: April 28, 2016

Contents

1 Abstract

Effective communication of technical ideas often demands compelling mathematical diagrams and visualizations. Just as TeX makes the best practices of professional mathematical typesetters accessible to everyday users, we aim to codify and automate best practices of professional mathematical illustrators. Currently, however, there is a functionality gap between 3D modeling software and 2D illustration software. The former allows users to manipulate 3D geometric information, while the latter allows users to manipulate aesthetic and stylistic information. In our work we wish to bridge the gap between these two types of software by extracting relevant curves from views of 3D Catmull-Clark subdivision surfaces that visually communicate geometric relationships in the form of projected 2D Bezier curves amenable to traditional aesthetic design.

2 Introduction to the Problem

2.1 Software Paradigms

Our work seeks to bridge a functionality gap between 3D Modelling software and 2D Illustration software. We will start out by describing each of these software paradigms and their strengths and weaknesses.

2.1.1 3D Modeling Software

Traditional 3D modeling software, such as the open source Blender program, are used primarily for creating defining the geometry and visual appearance of models to be used in various applications such as 3D animation, the automative industry, and architectural visualization. These programs use the latest and greatest algorithms in the fields of computational geometry, rendering, and Computer Aided Design, but they do not have the capability to stylize their models with prescision. Many of them are geared towards rendering the models using realistic models of light transport. As discussed in [1], realisitic lighting is not necessarily the best stylistic descision for communicating geometric information about an object. Please see 1 for an example realistically rendered image. Please see 2 for a typical 3D model as might be seen during manipulation in a 3D Modeling program. The visual style communicates the individual discrete vertices, edges, and faces that a user can modify which is important to a person constructing a 3D model, but it does not emphasize visual information that would be important to a geometer or a person concerned with aestetics.

Figure 1: 3D Modeling system are often used to produce photorealistic imagery, which is not always the proper best at communicating geometric information as discussed in [1]. Image Credit: Gilles Tran on Wikipedia.

2.1.2 2D Illustration Software

2D illustration software, such as the open source Inkscape, are used primarily by designers and communicators to create 2D svg illustrations that communicate ideas, rather than realistic visual artifacts. They have a lot of capabilties for modifying the colors and stroke sizes of lines and interiors, labeling important features with textual boxes and arrows, and

compositing different visual objects on top of each other through blending. While they are great at manipulating the aestetics of images, they do not necessarily understand 3D geometry and take it into account in the manipulations that they support. Please see Figure : 11, which is an example 2D illustration that we created in Inkscape that communicates light transport within a traditional Cornell Box scene.

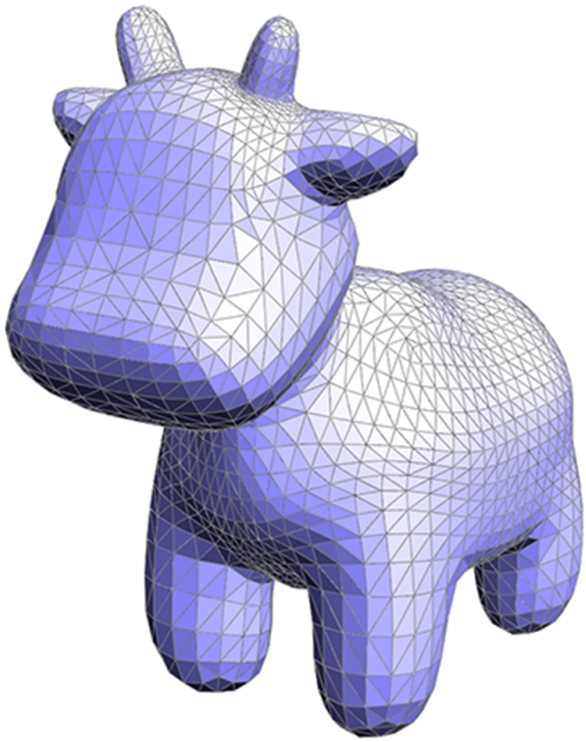

Figure 2: Typical 3D model that is manipulated in a 3D modeling program.

3 Background/Prior Work

3.1 Important 2D curves for projected 3D models.

Various curves on surfaces communicate important geometric information. Here is a list of some relevant curves that folks have been trying to extract from surfaces in the past.

- Integral Lines: lines that follow the gradient of a function defined on a surface from one critical point to another. These may be used to form the Morse - Smale complex which segments the model into regions with similar monotonic functional behavior.
- Parameterization curves are useful in showing lines along a collection of quadrilaterals and showing global coordinate systems.
- Silhouette curves communicate the visual extent of the model and the boundary.
- Minnimum - Maximum curvature lines communicate the curvature of the model and locally intuitive coordinate systems.
- Geodesic curves communicate the path on the surface of minnimum distance between two points on a surface.

3.2 2D Curve Extraction Methods

Many people have extracted silhouette curves by simply tracing the exterior boundary of a 2D rasterization of the surface. This approach suffers from discretization and sampling problems and does not provide much information about the 3D geometric structure of the silhouette curves. For instance, they can't be used to compute direct shadows. It also may be possible to extract silhouette curves using an algorithm akin to the marching squares, but again it suffers from the same sorts of problems.

3.3 3D Curve Extraction Methods

Stroila et. al. were able to extract silhouette, shadow, gleam, and other curves via particle chain systems. They have also studied practical ways to utilize these curves for illustration by describing how to sort, orient, identify, and fill them as closed regions. [6] // FIXME: Should I rephrase this, because I am pretty much parroting the abstract of this paper.

3.4 Catmull - Clark Subdivision Surfaces

Naively we could directly evaluate these curves over the linear patches defined on standard polyhedral 3D models, but much like the results found in [2], such lines would jitter back and forth over boundaries and would not converge to smooth natural and correct looking curves even after substantial subdivision of the surface. Please see Figure 3 for example of Eisemann et Al's attempt to extract silhouette curves from linear patches. We therefore wish to work on Catmull - Clark Subdivision surfaces that allow us to take a discrete quadrilateral control mesh and perform calculations on its limit defined subdivision surface, instead of any intermediate discrete representation. Please see Figure : 4 for an illustration of control meshes and limit surfaces.

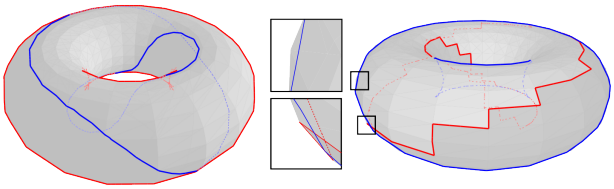

Figure 3: Linear Patches Silhouette curves either produce staircase patterns or don't properly lie on the geometry, even in the limit. Image Credit: [2].

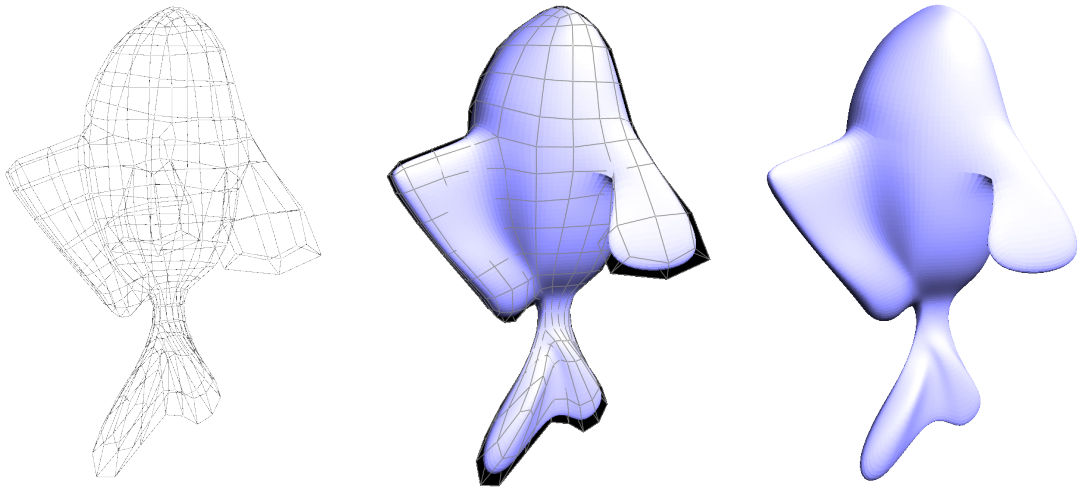

Figure 4: A model of a fish represented from on the left by its control mesh and on the right by a geometry patch approximation of its Catmull-Clark limit surface. Please note the discontinuity in shading at the top of the fish's back finn. This is caused by the vertices of degree 5 in that vicinity and is fixed by using tangent patches to interpolate the tangents across boundaries.

3.5 Loop - Shaeffer Approximations

No matter how finely we refine a traditional Catmull - Clark surface, it will still be a collection of linear patches which suffer from the same problems as in [2]. We therefore get around this problem by directly computing a continuous and differentiable approximation of the limit surface via the scheme described in [4] We can create a one-to-one correspondence between faces in the control mesh and bicubic bezier surfaces (a.k.a "patches") that approximate the limit Catmull - Clark subdivision surface that the faces represent. To do so, we follow the procedure outlined in [4]. We therefore are able to derive control points for a geometry patch which we will denote G_{ij} (Figure: 12) and two tangent patches, one for each principal parameter direction along the bicubic patch (Figure: 13 and Figure: 14). The tangent patches are necessary, because the geometry patches only exhibit G0 continuity in the presense of extraordinary vertices. Since any quadrilateral mesh that is not homeomorphic to a torus must contain an extraordinary vertex, it is essential that we use the tangent patches to ensure effective differentiability everywhere along the surface formed by the union of the bicubic patches.

For the remainder of the paper, we will be using these Loop - Schaeffer patch defined surfaces defined on quadrilateral meshes and refer to them simply as "Surfaces".

4 Methodology

4.1 Definitions and Calculus on Bicubic Patches

In this section we will discuss some fundamental calculus computations involving the patches that will be used in curve extraction algorithms.

4.1.1 Bezier Surfaces

The beauty of the bicubic patch approximations is that they allow us to transition from discrete math to well defined continuous math.

The Bernstein Polynomials are defined as follows:

$$\mathcal{B}_{i,n}(x) = \mathcal{B}_i^n(x) = \binom{n}{i} x^i (1-x)^{n-i} \textbf{ for } i \in \{0, \cdots, n\}$$

Two-dimensional Bezier surfaces are defined parametrically as follows:

$$[0,1] \times [0,1] \to \mathbb{R}^3 : \sum_{i=0}^{n} \sum_{j=0}^{m} \mathcal{B}_i^n(u) \mathcal{B}_j^m(v) C_{i,j}$$

where n, m is the degree of the surface, which is represented by $(n+1) \cdot (m+1)$ control points denoted genarically here by C_{ij}.

4.1.2 Geometry Patches

The geometry patch surfaces are of degree (3, 3) and are therefore represented by the 16 control points G_{ij} as follows:

$$g(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathcal{B}_i^3(u) \mathcal{B}_j^3(v) G_{i,j}$$

4.1.3 Partials Defined by Geometry Patches

We can easily take partial derivatives of a Geometry Patch as follows:

$$g_{u^m v^n} = \frac{\partial g}{\partial u^m \partial v^n} = \sum_{i=0}^{3} \sum_{j=0}^{2} \mathcal{B}_{i,3}^{(m)}(u) \mathcal{B}_{j,3}^{(n)}(v) G_{i,j}$$

where m and n are the degrees of differentiation in u and v respectively.

4.1.4 Partials Defined by Tangent Patches

We can easily take partials of g in u and v by differentiating the relevant u or v parameterized bezier function, but for the most part we will not make use of this pleasantry, because the geometry patches may be non differentiable on the boundaries. We will instead evaluate the partials from the tangent patches.

$$g_u := \frac{\partial g}{\partial u} = \sum_{i=0}^{3}\sum_{j=0}^{2} \mathcal{B}_i^3(u)\mathcal{B}_j^2(v)U_{i,j} \tag{1}$$

$$g_v := \frac{\partial g}{\partial v} = \sum_{i=0}^{2}\sum_{j=0}^{3} \mathcal{B}_i^2(u)\mathcal{B}_j^3(v)V_{i,j} \tag{2}$$

4.1.5 Normal

The normal direction is defined for any point on the surfaces as follows:

$$N(u,v) = g_u \times g_v(u,v)$$

4.1.6 Visibility

Given a fixed yet arbitrary viewing direction E we can define the visibility function as follows:

$$f(u,v) = N(u,v) \cdot E$$

A location (u,v) in parameter space on a surface is visible if and only if $f(u,v)$ is negative. It is important to note that we are assuming an orthonormal fixed viewing direction, instead of a perspective projection, because it simplifies our mathematics. [5]

4.1.7 Silhouette Curves and Points

A Silhouette point *sp* is any point such that the visibility function is zero. In other words the following must hold:

$$sp = g(u,v) \textbf{ and } f(u,v) = 0$$

A silhouette curve contains a closed ordered set of silhouette points. In other words they are defined as the boundary between the visible and non visible regions of the surface.

4.1.8 Behavior of Visibility Function

Assuming that E is normalized, the following properties hold for the visibility function:

- f has a bounded range as follows: $-|N| \leq f(u,v) \leq |N|$
- f has a Global Minimum when $f(u,v) = -|N|$.
- f has a Global Maximum when $f(u,v) = |N|$.
- Silhouette curves are defined when $f(u,v) = 0$.

4.1.9 Partials of the Visibility Function

The first order partial derivatives for the visibility function may be derived as follows using applications of the product rule for derivatives of cross products:

$$\begin{aligned} N(u,v) &= g_u \times g_v(u,v) \\ f &= E \cdot (P_u \times P_v) \\ \frac{\partial f}{\partial u} = f_u &= E \cdot (g_{u^2} \times g_v + g_u \times g_{uv}) \\ \frac{\partial f}{\partial u} = f_v &= E \cdot (g_{uv} \times g_v + g_u \times g_{v^2}) \end{aligned}$$

Similarly, the second order partial derivatives are as follows:

$$\begin{aligned} \frac{\partial f}{\partial u \partial v} = f_{uv} &= E \cdot (g_{u^2 v} \times g_v + g_{u^2} \times g_{v^2} + \cancel{g_{uv} \times g_{uv}} + g_u \times g_{uv^2}) \\ \frac{\partial f}{\partial u^2} = f_{u^2} &= E \cdot (g_{u^3} \times g_v + 2 g_{u^2} \times g_{uv} + g_u \times g_{u^2 v}) \\ \frac{\partial f}{\partial v^2} = f_{v^2} &= E \cdot (g_{uv^2} \times g_v + 2 g_{uv} \times g_{v^2} + g_u \times g_{v^3}) \end{aligned}$$

4.1.10 Gradient Descent

// FIXME : Should I remove this section from the paper and just reference Keenan's handout? It seems convenient to have it here.

This section is mainly a restating of information contained in Keenan Crane's *Line Search for Non-Smooth Optimization* [7].

Many of our algorithms rely on gradient descent to optimize various functions on surfaces. A proper selection of step sizes is important for these operations. If the steps are too slow, then performance will suffer due to slow convergance. If the steps are too large, then the optimization procedure may overshoot local minnima and fail to converge. Gradient descent on a function $f : \mathbb{R}^n \to \mathbb{R}$: requires that f be differentiable. A direction $u \in \mathbb{R}^n$ is a *search direction* if the directional derivative of f along u is negative. In other words for a search direction u, the following holds:

$$D_u f = \nabla f \cdot u < 0$$

Given an initial point x_0 and a search direction u, we want to find a new point $x_1 := x_0 + \tau u$ that is of a step size that is guranteed to make progress towards finding a local minnima. The step size should therefore satisfy two conditions as follows:

1. The **Armijo condition** $f(x_0 + \tau u) \leq f(x_0) + c_1 \tau D_u f(x_0)$

2. The **Wolfe condition** $|D_u f(x_0 + \tau u)| \leq c_2 |D_u f(x_0)|$.

where $c_1, c_2 \in \mathbb{R}$ are any two constants such that

$$0 < c_1 < c_2 < 1.$$

We can then compute the step size via the following algorithm:

---

begin
$\alpha \leftarrow 0$
$\beta \leftarrow +\infty$
$\tau \leftarrow 1$
repeat
  if Armijo is not satisfied **then**
    $\beta \leftarrow \tau$.
  else if Wolfe is not satisfied **then**
    $\alpha \leftarrow \tau$
  else
    BREAK.
  end if
  if $\beta < +\infty$ **then**
    $\tau \leftarrow (\alpha + \beta)/2$
  else
    $\tau \leftarrow 2\alpha$
  end if
until BREAK
end

For the purposes of this thesis, we will label gradient descent operations as follows:

$$x_1 = \mathrm{GD}(f, x_0)$$

where x_0 is the initial point, f is a differentiable function, and x_1 is the resultant minnimization of the function.

4.2 Extracting Parameterization Curves

Extracting parameterization curves is quite straightforward. Start corner of a patch and then procceed in either the u or v direction as desired until you either get back to the original point or you encounter an extraordinary vertex. Please see a more formal description in Algorithm: 4.2.0.1, noting that the computation is symmetric for the u or v direction. Please see Figure: 5 for example of parameterization curves extracted from a torus.

4.3 Extracting the Morse-Smale Complex for the Visibility Function

We now have ridge sets and valley sets, which may be singletons. We then proceed to union all disconnected sets together and add their levels to the final output.

Algorithm 4.2.0.1 Given a surface, a point, and a fixed yet arbitrary u direction, this procedure computes the visible and nonvisible portions of an **axis aligned parameter curve**.

Require: The surface must be **continuous**, and implied by this document be made of quadrilaterals.

Ensure: Returns a set of visible and nonvisible portions of the parameter curve going through the input point in the u direction. // FIXME : This procedure description is incomplete.

begin

repeat

$(dv, -du) \Leftarrow \nabla f(u, v)$

normalize (du, dv).

$(u', v') \Leftarrow (u, v) + \epsilon \cdot (du, dv)$

if Line $(u, v) \Leftrightarrow (u', v')$ crosses a patch boundary **then**

Remove the closest root on the boundary from S to prevent duplicate tracings.

end if

$(u, v) \Leftarrow \mathrm{GD}(c \cdot \nabla f^2, (u', v'))$, where c is any positive number.

until $(u, v) = (u_0, v_0)$ {This should be checked within some ϵ dependant bounds.}

end

Algorithm 4.3.0.2 For a given height h, finds exactly 1 point on all level sets for a 2-form function with no boundary that is morse, i.e their critical points are non degenerate and seperate.

Find all minima, maxima, and saddle points. We will call these features. Note that it is possible to calculate the equivalence to two features, because we can follow the gradient to theoretically arbitrary precision. We assume in this problem that all features are isolated as in the definition of a morse function.

Connect every saddle point to its 2 neighboring maxima and 2 neighboring minima using gradient descent to follow the integral curves. Remember the locations on the level sets where they are intersected by the integral curves.

We now have a Morse-Smale complex?) that looks something like Figure : 6.

We will use a union find algorithm to separate G into disjoint regions that I will call ridges and valleys. A ridge is a connected set of features that are all above the level h. A valley is a connected set of features that are all below the level h.

Initialize a Union Find Structure UF

Make a singleton set for every initial feature.

for $(c_1, c_2) \in Edges(G)$ **do**

if $h \notin [f(c_1, f(c_2)]$ **then**

UF.union(c_1, c_2) Union sets together if they are they do not cross the h level.

end if

end for// FIXME : I need to simplify the prose for this pseudo-code algorithm.

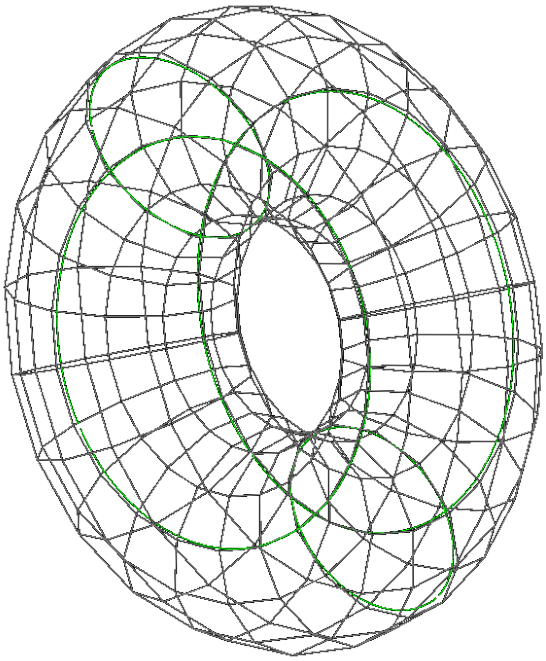

Figure 5: Some Example Parameterization curves along the major axis of a torus. FIXME : Replace this with a better picture with thicker lines.

4.4 Extracting Silhouette Curves

4.4.1 Finding Exact representations of Silhouette Curves

// FIXME: Discuss finding roots of multinomial roots, and morse smale complex.

4.4.2 Curve Tracing Method

Because it is intractable to find the exact representations for silhouette curves, we use the practical curve tracing algorithm 4.4.2.1. This algorithm is similar to the work of [5].

// We are integrating a 1st order ODE over the mesh, see Keenan's email.

We find silhouette curves as follows:

1. Find all silhouette points that lie on patch boundaries. This may be acomplished via a 1D root finding algorithm in one parameter along the surfaces. Please see section 4.4.3 for more information.

2. Trace the curves Move in the direction perpendicular to the gradient of the function, then move back to the silhouette curve by optimizing the following function:

$$\nabla f^2(u, v)$$

when doing the optimization, use appropiate step bounding as shown in Crane's pamphlet.

Algorithm 4.4.2.1 Given a surface, this algorithm computes a set of **sillhouette curve discretizations**, each consisting of silhouette points and cooresponding tangent vectors pointing along the silhouette curve. We use an ϵ value of .1 which yields roughly $\frac{1}{\epsilon} = 10$ steps per patch, because each patch constitutes a 1 by 1 square in parameter space.

Require: The surface must be **continuous**, **differentiable**, and contain disjoint silhouette curves.

Ensure: If the surface has no boundaries, then the output curves are closed loops.

begin
$S \Leftarrow$ Silhouette Point Finding Algorithm 4.4.2.2.
for Point $p \in S$ **do**
$(u_0, v_0) \leftarrow (p.u, p.v)$
$(u, v) \leftarrow (u_0, v_0)$
repeat
$(dv, -du) \leftarrow \nabla f(u, v)$
normalize (du, dv).
$(u, v) \leftarrow (u, v) + \epsilon \cdot (du, dv)$
$(u, v) \leftarrow$ GD $(\nabla(f^2), (u, v))$.
Output point g(u, v) and tangent (du, dv).
until $(u, v) == (u_0, v_0)$
end for
end

Algorithm 4.4.2.2 Given a surface, this algorithm finds the set of all silhouette points that lie on patch boundaries.

Require: The silhouette points on the boundaries need to be sufficiently far apart to prevent numerical instabilities.

Ensure:

begin
$S \Leftarrow \varnothing$
for $e \in Edges$ **do**
Compute the polynomial p representing the silhouette function f along e.
Add all roots $\in [0, 1]$ of p to S.
end for
return S.
end

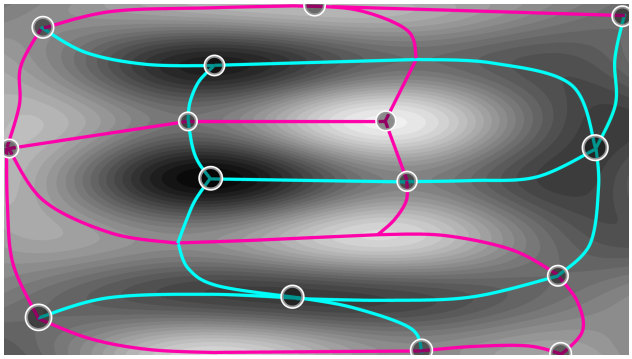

Figure 6: Morse Smale Complex of a 2-form function. Every saddle point connects to 2 minnima and 2 maxima.

4.4.3 Finding Silhouette Points

// Discuss Morse Smale Algorithm. // Discuss our method and concerns over sampling. // Look up my description of the algorithm to Keenan.

Because it is intractable to find the exact representations for sillhouette curves, we instead resort to finding a Silhouette points that lie along the boundary of patches. Ideally we would like to only find 1 silhouette points per disjoint silhouette curve, because otherwise we run the risk of tracing the same silhouette curves multiple times without protective measures that increase the computational overhead of our algorithms.

4.4.4 Finding Silhouette Points Via 1D Root Finding on Geometry Patches

To find the silhouette points along a boundary of a geometry patch, we first expand the definition of the visibility function as follows:

$$f = E \cdot (G_u \times G_v)$$

$$G_u(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} \mathcal{B}'^{3}_{i}(u)\mathcal{B}^{3}_{j}(v)G_{i,j}$$

$$G_v(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} \mathcal{B}^{3}_{i}(u)\mathcal{B}'^{3}_{j}(v)G_{i,j}$$

If we assume that u = 0, then we can simplify these partial derivatives as follows:

$$P_u = -3\sum_{j=0}^{3} \mathcal{B}^{3}_{j}(v)G_{0,j} + 3\sum_{j=0}^{3} \mathcal{B}^{3}_{j}(v)G_{1,j} \tag{3}$$

$$P_v = \sum_{j=0}^{3} \mathcal{B}'^{3}_{j}(v)G_{0,j} \tag{4}$$

To compute the silhouette points along a boundary or any other 1 - dimensional axis aligned slice of a patch, evaluate the partial derivative with either u or v set to a constant value, such as $u = 0$ in Equations 3 and 4. This results in partials represented by 3-dimensional vectors containing single variable polynomials for each dimension. The visibility

function polynomial may then be computed directly form the visibility function formula applying the cross product and dot product operations as as normal, but with polynomial algebra, instead of scalar algebra. The resulting single variable polynomial represents the value of the visibility function along the given axis aligned slice within the input variable domain [0, 1]. The roots of this polynomial coorespoind to Silhouette points. The roots may be computed using any single variable real root finding algorithm. We decided to implement root finding based on Sturm's theorem, although numerical algorithms may have trouble discerning between roots that are sufficiently close to each other. These Silhouette points work well, except when the silhouette curves cross boundaries between patches that contain extraordinary vertices. Please see Figure : 7 for an example of the types of problematic behavior the extraordinary edges cause in our curve tracing algorithms. These problems provide the motivation for us to use tangent patches.

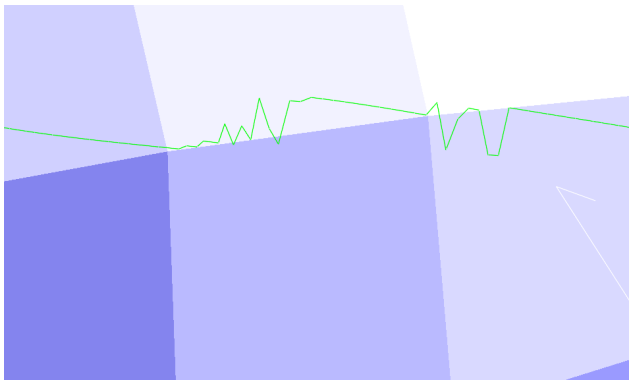

Figure 7: Along extraordinary boundaries, the geometry patches are non differentiable and therefore the visibility function is discontinuous. This causes curve tracing to fail due to an invalidation of its assumptions.

// FIXME : Should I call 'slices' something else.

4.4.5 Finding Silhouette Points Via 1D Root Finding on Tangent Patches

To compute the roots of the tangent patch defined visibility curve, please use Equations 1 and 2 for the partials.

zzz

4.4.6 Degenerate surface views

When a surface is oriented such that its silhouette curves are not disjoint, such as viewing a torus in a manner perfectly orthogonal to its hole. Please see Figure : 8 for an example near degenerate view of a torus.

4.4.7 Projecting 3D Discretizations onto 3D Planes

// FIXME : I need to discuss the conversion between 3D curves into 2D svg paths.

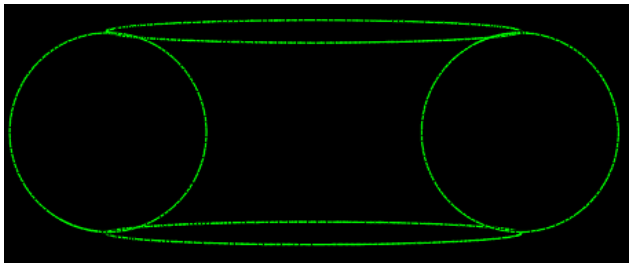

Figure 8: As a torus becomes oriented with its hole perpendicular to the vieing direction, its silhouette curves intersect each other. This is one of several degenerate cases that present challenges to our curve tracing algorithms.

5 Results

We made a system that represents quadrilateral mesh defined Catmull - Clark subdivision surfaces through the the Loop - Shaeffer approximate via geometry and tangent patches. We have developed some calculus for extracting curves on these surfaces, including the silhouette curves, parameter aligned curves, integral curves in Morse - Smale complexes. We have developed algorithms for finding the location of critical points for the Silhouette function **F** Please see Figure: 9 to see some extracted silhouette curves from a pig model.

Please see the following url for the research code that we wrote for this thesis: `https://github.com/Bryce-Summers/GeometricSurfaceCurves`.

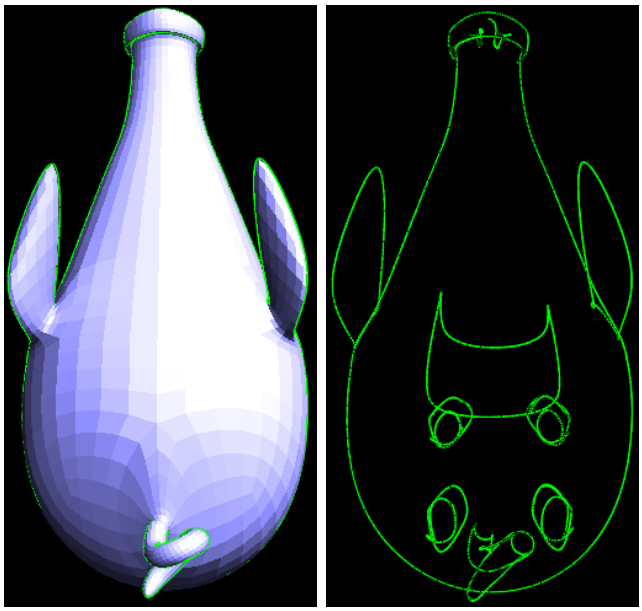

Figure 9: A view of a pig model and some silhouette curves extracted via our system.

6 Comparison with prior work

Past work including [2] has extracted silhouette curves from linear patches. They suffer from discontinuity and a lack of accurate interpolation of the points on the surface. Please see

Figure: 3. Please see Figure: 10 for an example silhouette curve that we extracted using our methods that is continuous everywhere and properly follows the surface.

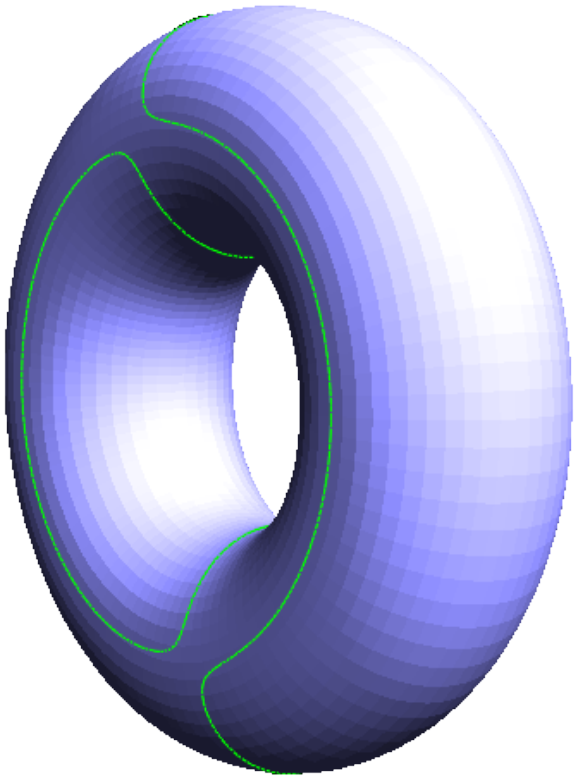

Figure 10: The curves extracted by our system are continuous even when not viewed by their defining viewing direction.

7 Future Work

In this discussion, we enumerate several problems that should be addressed in the future, categorized into those that we feel can be immediatly tackled, those that may need to wait for Mathematics to progress a bit, and those whose realization is tied to the development of Artifical Intelligence.

7.1 Near Term Problems

In this section we will describe several coscisely stated problems similar to the silhouette curves problem that may be immediatly tackled in the near future.

7.1.1 Extracting the exterior silhouette curve

The actual visual exterior for a surface may include subsets of several silhouette curves. The exterior silhouette curve may be computed by projecting the visible

7.1.2 Calculation of shadows.

We can compute direct shadows by computing the exterior silhouette curves from a light source viewing in the direction of the surface and then projecting this exterior silhouette onto a plane representing the groud that the surface is resting on.

7.1.3 Minnimum and Maximum Curvature Curves

Minnimum and maximum curvature curves may be used to communicate information about geometrically intuitive local coordinate systems in the neighborhood of specific points on an object. It would be very useful to be able to derive a 2D coordinate grid given a point on a surface.

7.1.4 Geodesic Curves

In the future, a user should be able to specify two points on a surface and receive the curve that represents the minnimum distance path between those two points. This is known as the curve of minnimum geodesic distance.

7.1.5 User Geometric Stylization Scheme

My advisor Keenan liked giving presentations with an orange style back in the day, but now he favors blue. It would be very useful for users to be able to automatically convert entire presentations, including the images from one style to another. A user should be able to define their own figure color scheme in something like a CSS file and be able to automatically convert their figures between styles.

7.1.6 Occlusion

In our extracted curves, in addition to those points whose normals face away from the camera viewpoint, they may also include points that are not visible due to occlusion by other regions of the surface. There might be some interesting topological properties of closed curves that could be used for this task, especially if the homogenous depth of each of the points from the viewport was taken into account.

7.2 Moderate Term Problems

In this section, we describe several problems that are more difficult, mainly because they involve geometric computations of a higher degree than the current mathematics of our day can handle.

7.2.1 Perspective Correct Silhouette Curves

Right now, we are assuming that the user is viewing the surface with an orthonormal view perspective where the eye is looking in one uniform direction. This approximation leads to visually acceptable silhouette curve computations, but it is not accurate in terms of the actual perspective projection that the figures are rendered in. To compute the curves in a perspective correct manner would require higher degree geometric computations

7.2.2 Extracting Exact Geometric Curves

Right now we are extracting points and tangents along curves, but if people were solve the problem determining the root cuves of multinomials, then we could represent these curves without discretization.

7.2.3 Labelling Geometry

It would be interesting to develop algorithms for properly placing textual labels for a given figure view where the labels are aware of the geometry. The labels placing would have to take into account desirable properties, such as avoiding overlapping lines, avoiding intersections with other labels, and encouraging visual orientation coherence, whereby the labels would all face roughly the same way. Arrows could also be investigated.

7.2.4 2D Segmenting and Labelling of Ray Traced Imagery

A system could hypothetically be built that segments a 2D projection of a ray traced scene into different regions based on light transport phenomena. Please see Figure 11.

7.3 Long Term Problems

In this section, we will describe some long term grand problems that synthesize our work with Artificial Intelligence.

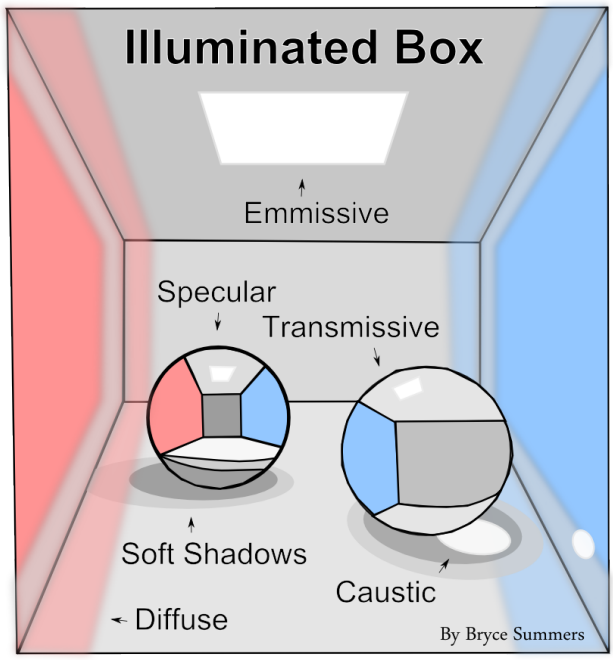

Figure 11: A labeled illustration of the idea of converting a ray tracable scene into an SVG file with regions segmented by light transport phenomena.

7.3.1 Automatic Paper Interpretter

In the future, it would be amazing if a user could take a confusing research paper or any other work of communication feed it into a system and get a perfectly clear version of the paper back that contains automatically generated illustrations of the ideas contined therin.

References

[1] Tilke Judd, Frdo Durand, and Edward Adelson. 2007. *Apparent ridges for line drawing.* ACM Trans. Graph. 26, 3, Article 19 (July 2007). DOI=http://dx.doi.org/10.1145/1276377.1276401

Elmar Eisemann, Holger Winnemller, John C. Hart, and David Salesin. 2008. *Stylized vector art from 3D models with region support.* In Proceedings of the Nineteenth Eurographics conference on Rendering (EGSR '08). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1199-1207. DOI=http://dx.doi.org/10.1111/j.1467-8659.2008.01258.x

[2] Elmar Eisemann, Holger Winnemller, John C. Hart, and David Salesin. 2008. *Stylized vector art from 3D models with region support.* In Proceedings of the Nineteenth Eurographics conference on Rendering (EGSR '08). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1199-1207. DOI=http://dx.doi.org/10.1111/j.1467-8659.2008.01258.x

[3] E. Catmull and J. Clark. 1998. *Recursively generated B-spline surfaces on arbitrary topological meshes.* In Seminal graphics. ACM, New York, NY, USA 183-188. DOI=http://dx.doi.org/10.1145/280811.280992

[4] Charles Loop and Scott Schaefer. 2008. *Approximating Catmull-Clark subdivision surfaces with bicubic patches.* ACM Trans. Graph. 27, 1, Article 8 (March 2008), 11 pages. DOI=http://dx.doi.org/10.1145/1330511.1330519

[5] Li Xuejun, Sun Jiaguang, and Yang Changgui. 1998. *Extracting Silhouette Curves of NURBS Surfaces by Tracing Silhouette Points.* Tsinghua Science and Technology. ISSN 1007-0214, 13/22, pp1005 - 1008, Volume 3, Number 2. (June 1998), 4 pages.

[6] Matei Stroila, Elmar Eisemann, and John Hart. 2008. *Clip Art Rendering of Smooth Isosurfaces.* IEEE Transactions on Visualization and Computer Graphics 14, 1 (January 2008), 135-145. DOI=http://dx.doi.org/10.1109/TVCG.2007.1058

[7] Keenan Crane *Line Search for Non-Smooth Optimization.* Keenan's Notes.

[8] http://www2.cs.uh.edu/ chengu/Teaching/Spring2013/Lecs/Lec8.pdf

Keenan Crane's Powerpoint presentation about 3D illustration.

8 Apendix A: 3rd Order Bernstein Basis Functions and Derivatives.

The Bernstein Basis functions of the third order are defined as follows:

$$B_i = \binom{3}{i} x^i (1-x)^{3-i}, \text{for } i \in \{0, \cdots, 3\}$$

Here is a listing of the four 3rd order bernstein polynomials along with their 1st, and 2nd order derivatives:

$$\begin{array}{lll} B_0 = (1-x)^3 & B_0' = -3(1-x)^2 & B_0'' = -6x+6 \\ B_1 = 3x(1-x)^2 & B_1' = (x-1)(9x-3) & B_1'' = 18x-12 \\ B_2 = 3x^2(1-x) & B_2' = (6-9x)x & B_2'' = -18x+6 \\ B_3 = x^3 & B_3' = 3x^2 & B_3'' = 6x \end{array}$$

Here is a listing of the 3rd order bernstein polynomials in standard polynomial form along with their 1st derivatives in standard polynomial form:

$$\begin{array}{ll} B_0 = -x^3 + 3x^2 - 3x + 1 & B_0' = -3x^2 + 6x - 3 \\ B_1 = 3x^3 - 6x^2 + 3x & B_1' = 9x^2 - 12x + 3 \\ B_2 = -3x^3 + 3x^2 & B_2' = 9x^2 + 6x \\ B_3 = x^3 & B_3' = 3x^2 \end{array}$$

// FIXME: Consider putting a lovely picture here illustrating the Bernstein Polynomials.

9 Apendix B: 2nd Order Bernstein Basis Functions and Derivatives

The Bernstein Basis functions of the third order are defined as follows:

$$B_i = \binom{2}{i} x^i (1-x)^{2-i}, \text{for } i \in \{0, \cdots, 2\}$$

Here is a listing of the 2nd order bernstein polynomials along with their 1st, and 2nd order derivatives:

$$\begin{array}{lll} B_0 = (1-x)^2 & B_0' = 2x - 2 & B_0'' = 2 \\ B_1 = 2x(1-x) & B_1' = -4x + 2 & B_1'' = -4 \\ B_2 = x^2 & B_2' = 2x & B_2 = 2 \end{array}$$

Here is a listing of the three 2nd order bernstein polynomials in standard polynomial form along with their 1st derivatives in standard polynomial form:

$$\begin{array}{lll} B_0 = x^2 - 2x + 1 & B_0' = 2x - 2 & B_0'' = 2 \\ B_1 = -2x^2 + 2x & B_1' = -4x + 2 & B_1'' = -4 \\ B_2 = x^2 & B_2' = 2x & B_2'' = 2 \end{array}$$

// FIXME: Consider putting a lovely picture here illustrating the Bernstein Polynomials.

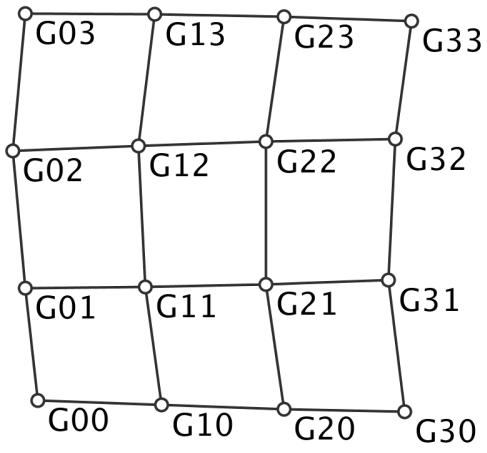

Figure 12: Geometry Patch Control Points for a single control mesh face.

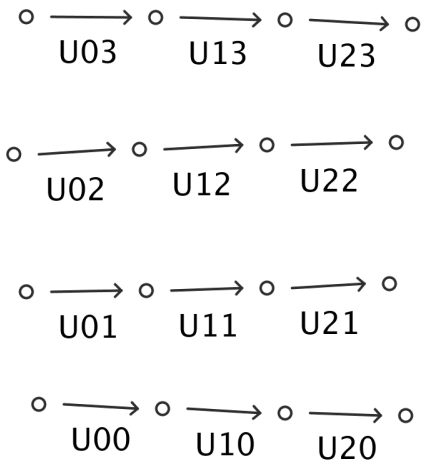

Figure 13: ∂u tangent patch control vectors for a single control mesh face.

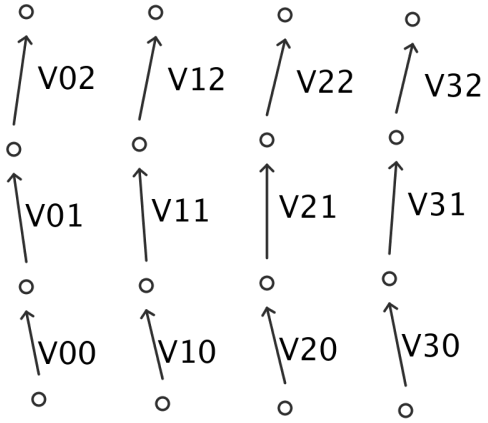

Figure 14: ∂v tangent patch control vectors for a single control mesh face.