AN ABSTRACT OF THE THESIS OF

Sanaz Golbabaei for the degree of Master of Science in Computer Science presented on June 06, 2016.

Title: Branched Covering Space (BCS) Construction and Visualization

Abstract approved: ____________________

Eugene Zhang

Branched covering spaces are a mathematical concept which originates from complex analysis and topology and has found applications in tensor field topology and geometry re-meshing. Given a manifold surface and an N-way rotational symmetry field, a branched covering space is a manifold surface that has an N-to-1 map to the original surface except at the so-called *ramification points*, which correspond to the singularities in the rotational symmetry field. Understanding the notion and mathematical properties of branched covering spaces is important to researchers in tensor field visualization and geometry processing. The theory of covering space is used in many research areas in computer science such as vector and tensor field topological simplification, parametrization, and re-meshing.

In this research, we provide a framework to construct and visualize the branched covering space (BCS) of an input mesh surface and a rotational symmetry field defined on it. In our framework, the user can visualize not only the BCSs but also their construction process. In addition, our system allows the user to design the geometric realization of the BCS using mesh deformation techniques. This enables the user to verify important facts about BCSs such as they are manifold surface around singularities as well as the *Riemann-Hurwitz formula* which relates the Euler characteristic of the BCS to that of the original mesh. By enabling the design of BCS, our system can be used to generate meshes with a relatively large number of self-intersections.

Branched Covering Space (BCS) Construction and Visualization

by

Sanaz Golbabaei

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 06, 2016
Commencement June 2016

Master of Science thesis of Sanaz Golbabaei presented on June 06, 2016.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

Sanaz Golbabaei, Author

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Prof. Eugene Zhang, for his long-standing support of my study and research, For his motivation, and friendship. Hi guidance helped me all the time of this research. I will always be grateful for his seemingly infinite patience. Without his help and insight, none of this would have been possible.

I would like to thank my committee members: Prof. Bailey, Prof. Nayyeri, Prof. Tyler, for all their help during this research. I'd like to thank my fellow graduate students, collaborators, who contributed to this research. I am very grateful to all of you.

I would especially like to thank my amazing family for the love, support, and constant encouragement I have gotten over these years. In particular, I would like to thank my mother Afsaneh, undoubtedly I could not have done this without you and your endless support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF ALGORITHMS

Chapter 1: Introduction

Branched covering spaces is a mathematical concept in topology, which has found applications in computer graphics and tensor field topology.

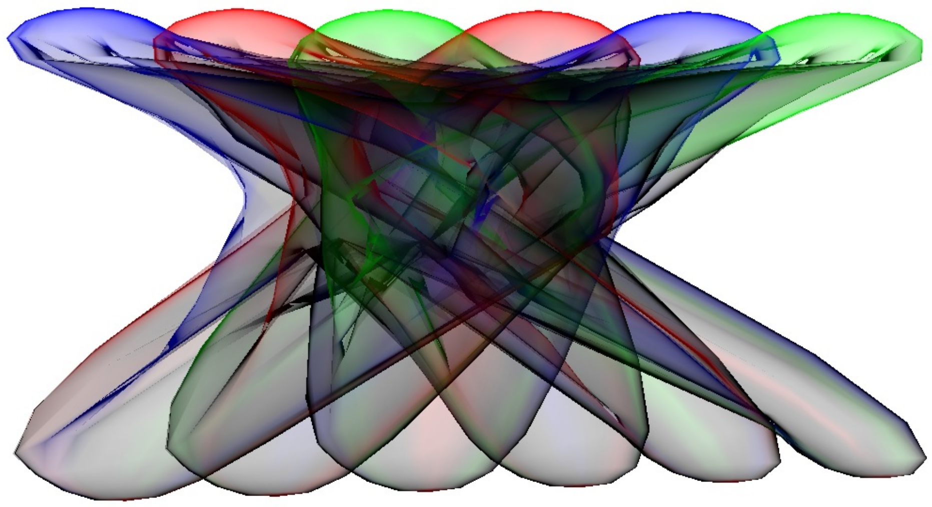


Figure 1.1: 6-fold branched cover of sphere with 12 ramification points.

1.1 Introduction to BCS

Tricoche [22] (Figure 1.3) describes the behavior of a *degenerate point* in a 2D symmetric tensor field defined on some domain by converting the tensor field into a vector field, defined on the branched covering space of the domain, and studying the behavior of the singularity in the vector field that corresponds to the degenerate point in the tensor field.

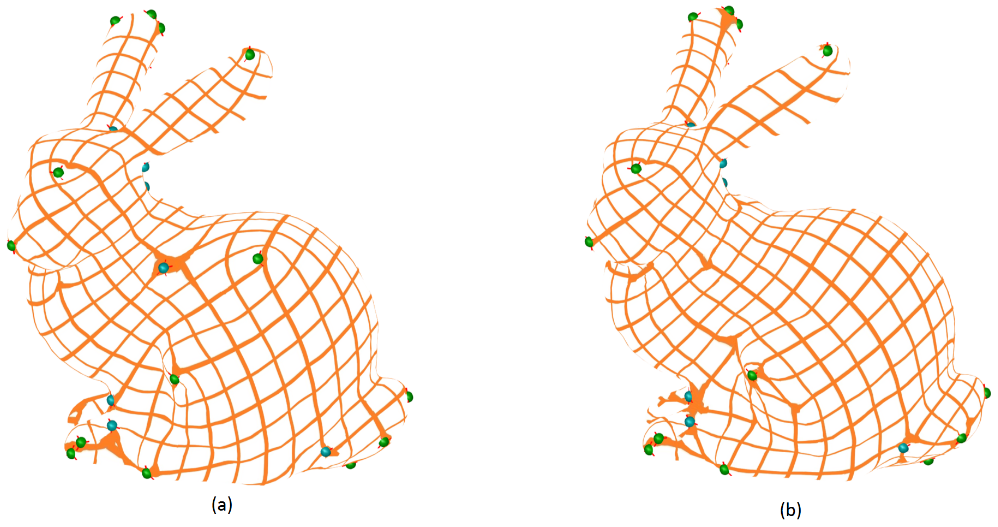


Figure 1.2: a) Parametrization and re-meshing done on bunny before cancelling a pair of singularity. b) Parametrization and re-meshing done on bunny after cancelling a pair of singularity. As you can see by cancelling a singularity we improved the underlying field which led to a better re-meshing with fewer T-junctions.

In geometry processing, the problem of quadrangular re-meshing, i.e., the generation of a mesh of quads from an input triangle mesh, has gained much attention by the graphics community. In quadrangular re-meshing, the edges in the quads are often required to be approximately aligned with a given *cross* field that is usually derived from the principal curvature directions in the underlying surface. Having a better field would lead to a better parametrization and re-meshing. In order to have a better field, we can cancel singularities, As shown in figure 1.2 cancelling singularities would help. Unfortunately we can not cancel all the singularities on a mesh since the number of singularities on a mesh is based on the topology of the object. To prevent T-junctions from occurring in the quad mesh, Kälberer et al. [9] proposed to lift the cross field in the input mesh to a vector field on the four-fold branched covering space of the surface. They then perform Hodge-decomposition to remove the divergence-free part of the vector field, thus preventing T-junctions in the re-mesh.

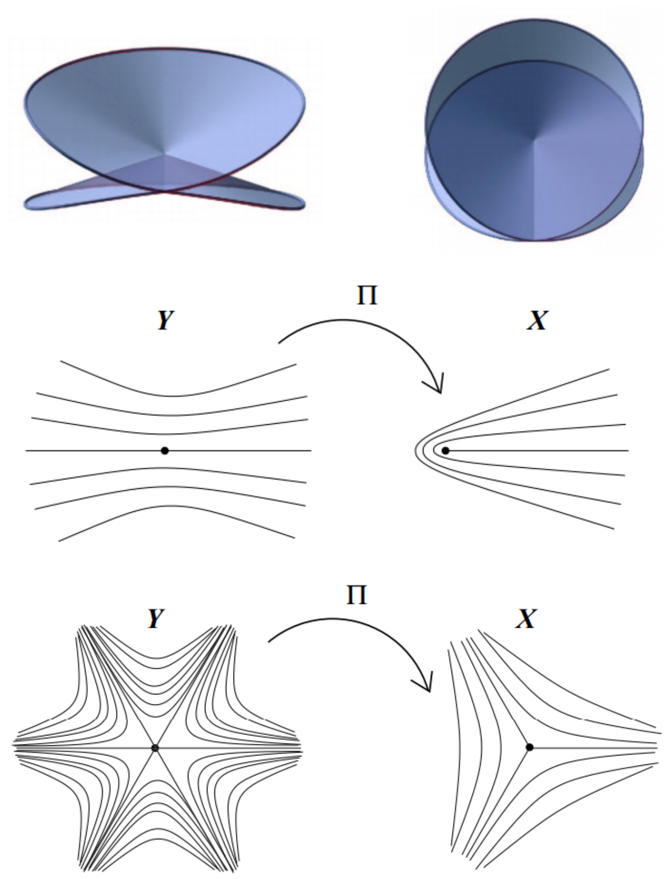


Figure 1.3: Tricoche [22] describes the behaviors of a tensor field near a degenerate point by *lifting* the tensor field on the domain to a vector field on the two-fold branched covering space of the domain and inspecting the behaviors of corresponding singularity. The top row shows the branched covering space around a singularity from the side view and the top view, respectively. The middle and bottom rows show how the double cover by the branched covering space relates the tensor field behavior around a degenerate point (middle-right: wedge; lower-right: trisector) to the vector field behavior around the corresponding singularity (middle-left: regular point; lower-left: monkey saddle).

The concept of branched covering space, which we will refer to as *BCS* in the remainder of the thesis, is both important and challenging. Mathematically, a BCS is the extension of the concept of covering space, with the additional notion of *ramification points*. While this addition can seem minor, the behaviors of a BCS are significantly more complex than a covering space and thus more difficult to understand. The additional complication of the BCSs has led to misconceptions such as that the BCS of a manifold surface is no longer a manifold surface due to the presence of ramification points.

In addition, existing research that requires some visual descriptions of a BCS often does so with hand-drawn illustrations of some patch on the BCS, usually around a ramification point (Figure 1.3). To the best of our knowledge, there is no published algorithm to explicitly compute the BCS given an input mesh, nor to visualize the BCS. This can make it difficult to grasp for graphics researchers who are interested in re-meshing but have not acquired sufficient mathematical background in topology and differential geometry. In this research, we address this by providing efficient algorithms to construct the BCS given an input surface. Moreover, we provide functionalities that enable the user to deform the BCS as well as visualize them in a number of visualization modes, thus enabling the user to see and interact with the BCS would result in gaining intuitions and verify important known results about BCS. In particular, we strive to enable the understanding of the following known facts about BCSs through our system:

1. An N-way rotational symmetry (N-RoSy) field on the input mesh leads to an N-fold branched covering space of the original mesh.

2. The singularities in the N-RoSy field become the ramification points of the BCS.

3. Away from the ramification points, every point in the original mesh corresponds to N points in the BCS, each of which is assigned one of the vectors in the N-RoSy at the base point.

4. If the input mesh represents a manifold surface, then the BCS is also a manifold surface, i.e., the points in the BCS corresponding to a ramification point are manifold points.

5. The index of a singularity in the vector field on the BCS of the mesh is decided by the index of the corresponding singularity in the N-RoSy field.

6. *Riemann-Hurwitz formula*, which states that the Euler characteristic of the BCS is related to that of the base surface and the number of ramification points in the BCS.

7. The BCS is independent of the way in which it is constructed.

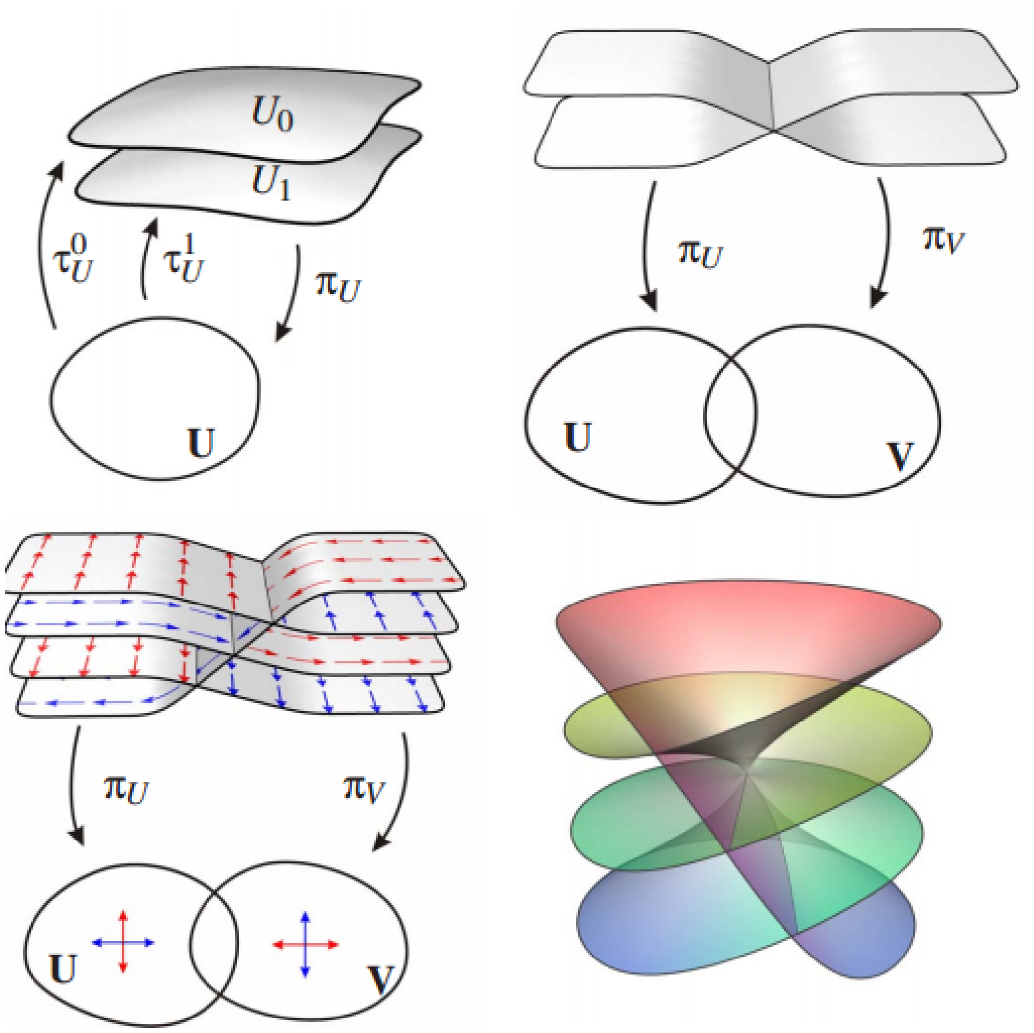


Figure 1.4: Kälberer et al. [9] use a combination of hand-drawn illustrations and rendering to visualize important properties of the BCS, such as that away from ramification points, every point in the original mesh has N corresponding points in the BCS (upper-left), and that the N-RoSy at the base point becomes N vectors, each at a corresponding point in the BCS (lower-left). Around a ramification point in the mesh, the BCS is still a manifold (lower-right).

The remainder of this thesis is organized as follows. In Section 1.2 we will review related work in computer graphics and visualization in which the concept of BCS has been applied. We then provide a review of the needed mathematical concepts in chapter 2. In chapter 3 and 4, we describe our algorithms to construct the BCS from an input mesh surface as well as interactive visualization framework with which the user can see and interact with the BCS to gain intuition. in chapter 5 we describe our user interface and show some results of BCSs generated by our system with different fields. We then in chapter 6 go through our construction and visualization methods to talk about time complexity of different stages of our algorithms and also we will talk about future research opportunities in this area.

1.2 Related Works

The notion of branched covering spaces is first introduced into computer graphics in the context of quadrangular remeshing of surfaces [9].

Alliez et al. [1] point out the importance of quadrangular remeshing from a triangular mesh where the edges of the quads in the remesh follow the principal curvature directions of the underlying surface. To generate such a mesh, they adapt the evenly-spaced-streamlines approach of Jobard and Lefer [8] for vector fields to the curvature tensor field of the surface. Ray et al. [14] point out that the distortion in the resulting quad mesh can be greatly improved if the quads in the mesh are oriented according to the curvature tensor field, which has two mutually perpendicular directions (major and mior principal directions), can be modeled as a four-way rotational symmetry (i.e., a cross). They further point out the difficulties associated with the singularities in the rotational symmetry field. Palacios and Zhang [11] provide a rotational symmetry field design system, with the ability to control the number and location of the singularities in any N-way rotational symmetry field ($N \geq 1$). Ray et al. [16] introduce a mathematically rigorous algorithm that generates a smooth, per-face N-way rotational symmetry field given desired index and location of singularities in the mesh. They later extend this approach to automatically generate such a field that is "geometry-aware", i.e., it adapts to the principal curvature directions in the surface [15]. Still, approaches such as tracing streamlines following an N-way rotational symmetry field will lead to *T-junctions*, leading to *quad-dominant* but not pure quad meshes. Tong et al. [20] provide a framework in which a quadrangulation is designed on a surface. Dong et al. [5] uses the *Morse-Smale complex* of a shape-aware Morse function to generate a quadrangulation with T-junctions.

Kälberer et al. [9] employ a global parameterization approach to quadrangulation that can eliminate T-junctions. The core of their approach is to lift the N-way rotational rotational symmetry field on the input surface to a vector field on the branched covering space and then remove the divergence-free part in the vector field through Hodge-decomposition [13, 21]. This approach is reformulated into a mixed-integer quadrangulation approach [3]. Nieser et al. [10] apply a similar approach to produce high-quality triangular meshes with control over the irregular vertices.

In scientific visualization, the notion of branched covering spaces has been introduced by

Xavier Tricoche [22] to explain the behaviors of the *degenerate points* in 2D symmetric tensor fields. These behaviors are understood by lifting the tensor field locally to a vector field on its branched covering space and examining the behavior of the singularity in the vector field corresponding to the degenerate point in the tensor field.

To the best of our knowledge, there is no published algorithm for constructing the branched covering spaces given an input surface. Existing research that uses the notion of branched covering spaces [9, 3, 10] does not actually need to construct the branched covering spaces explicitly. Consequently, there are no known visualization techniques for branched covering spaces. We address this issue in this research.

Chapter 2: Mathematical Background

In this chapter we briefly review the definition of covering space, branched covering space and rotational symmetry and we discuss the connections between them. BCS is an extension of the notion of covering spaces [2], which we describe next.

2.1 Covering Space Definition

Let X be a topological space. A **covering space** of X is a topological space C together with a **continuous surjective** map: $p : C \to X$ such that for every $x \in X$, there exists an open neighborhood U of x, such that $p^{-1}(U)$ is a union of disjoint sets in C, each of which is mapped homeomorphically onto U by p. The map p is called the **covering map**, the space X is the **base space** of the covering, and the space C is called the **total space** of the covering. The pre-image of x is necessarily a set of discrete points in C, which are referred to as the **fiber** over x. The neighborhood U is referred to as an **evenly covered neighborhood**. Each homeomorphic copy in C of U is a *sheet* over U.

An example covering space is $\mathbb{R}^1$, which provides an infinite cover of $\mathbb{S}^1$ (the unit circle in $\mathbb{R}^2$) through the map: $p(\theta) = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$. Here $\mathbb{S}^1$ is the base space, while $\mathbb{R}^1$ is the total space, as shown in figure 2.1.

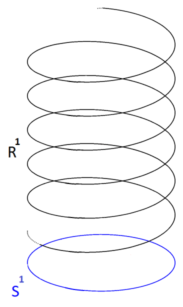


Figure 2.1: A circle $\mathbb{S}^1$ is covered with a helix $\mathbb{R}^1$.

Another example covering space is $\mathbb{S}^n$ (n-sphere), which double covers $\mathbb{RP}^n$ (the n real project space), with the map that takes every pair antipodal points in $\mathbb{S}^n$ to the same point in $\mathbb{RP}^n$.

A BCS extends the notion of covering spaces in the following fashion.

2.2 Branched Covering Space Definition

Let X and C be two topological spaces and $p : X \to C$ be continuous surjective map. X is said to be **branched covering space** of C under p if there exists a nowhere dense set $\Delta \subset C$ such that $p|_{p^{-1}(C\setminus\Delta)} : p^{-1}(C \setminus \Delta) \to C \setminus \Delta$ is a covering mapping. The set $C \setminus \Delta$ is a **regular set** of the branched covering p, whereas Δ is the **singular set**.

Figure 2.2 shows an example of a two-fold branched cover of $\mathbb{S}^1$ wich one branched point. Note that every covering mapping is also a branched covering mapping with an empty singular set. A less trivial example is $\mathbb{R}$ which covers the set of non-negative numbers $\mathbb{R}^+$ under the map: $p : \mathbb{R} \to \mathbb{R}^+$ ($p(x) = |x|$). The singular set consists of the number 0 which has only one pre-image under p while other elements in $\mathbb{R}^+$ has two pre-images.

In geometry remeshing and tensor field topology, the branched covering mappings are usually induced from an input N-RoSy field on some base surface S.

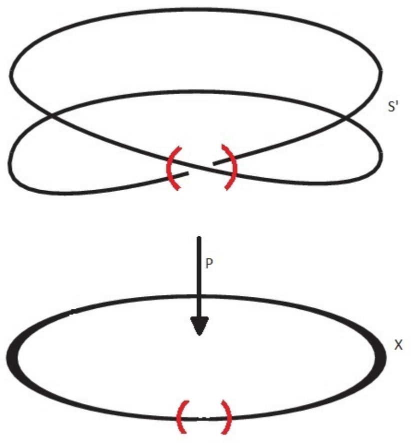


Figure 2.2: 2-Fold branched cover of sphere with one branched point.

2.3 Rotational Fields

An N-RoSy is a set of N-vectors $s = \{\begin{pmatrix} R\cos(\theta + \frac{2k\pi}{N}) \\ R\sin(\theta + \frac{2k\pi}{N}) \end{pmatrix} \mid 0 \leq k \leq N-1\}$ which each of the vectors called member vectors and R is the length of the member vectors and θ is the angular component of one of the member vectors, as shown in figure 2.3. An N-RoSy field is a continuous N-RoSy valued function on the surface. A **singularity** in an N-RoSy field is a point in the domain where $R = 0$.

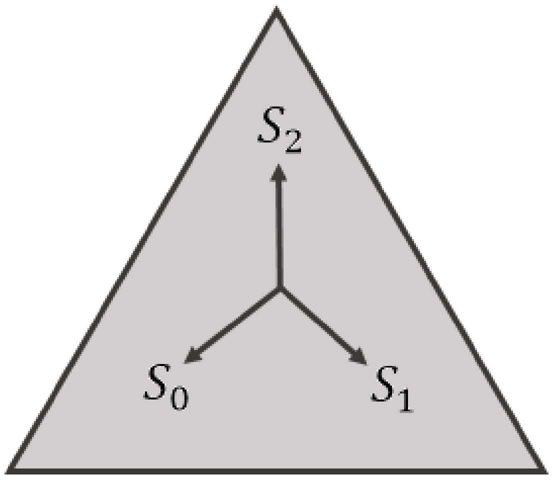


Figure 2.3: 3-RoSy defined on a triangle. All the vectors have the same size and they are evenly spaced with 120 degree seperation.

A singularity can be characterized by its *singularity index*, which is defined in terms of the *winding numbers* and is a multiple of $\frac{1}{N}$. *winding numbers* is an integer representing the total number of times that field travel counter-clockwise around a point.

Intuitively, at every point in the manifold S except singularities, there are N evenly-spaced vectors. A natural question to ask is: is it possible to find a N-fold covering space C for S with a vector field defined on C such that:

1. Every regular point p in S corresponds to N points in the covering space.

2. The collection of the vectors at points corresponding to p is exactly the N-RoSy at p.

For example, Figure 2.4 shows such a scenario when S is a torus with a 4-RoSy field derived from the its principal curvature directions.

However, when the N-RoSy field has singularities, a covering space is not possible as each singularity in the field has fewer than N corresponding points in the covering space. Such points are referred to as *ramification points*, and thus the cover is a branched covering space.

Branched covering spaces have a number of important properties:

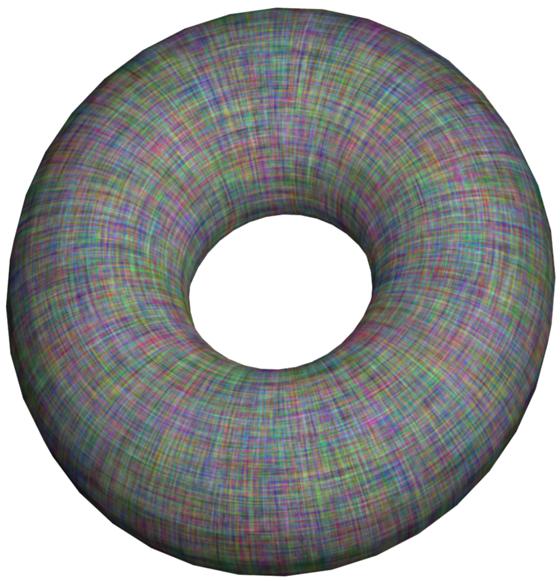
1. A branched covering space is a manifold surface if the base surface is a manifold.

2. A branched covering space is orientable if the base surface is orientable.

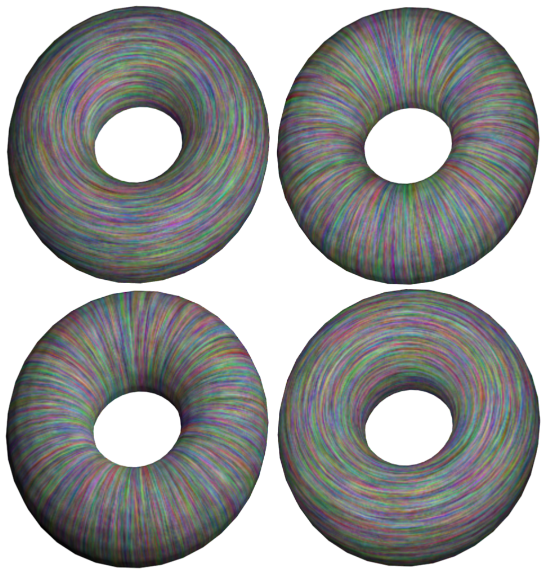
3. Every singularity of index k in the N-RoSy field is mapped to a singularity of index $Nk - (N-1)$ in the vector field in the branched covering space.

4. The Euler characteristic of the branched covering space is described by the *Riemann-Hurwitz formula* [6]: $\chi(B) = N \cdot \chi(S) - \sum_{p \in B}(e_p - 1)$, where $\chi(B)$ and $\chi(S)$ are the Euler characteristics of the branched covering space B and the base space S, respectively, and the summation is over the set of pre-images of the singularity set where e_p is the index of each ramification point in the singularity set.

In the next chapters we will describe our algorithm to construct the branched covering space given an input surface with an N-RoSy field. In addition, we will provide details on a number of visualization and interaction techniques that allow users gain intuitions about the aforementioned mathematical properties.



a



b

Figure 2.4: A 4-RoSy field without a singularity on the torus (a) leads to a covering space (b). Notice that the four directions at a point in the base mesh (a) are maintained by the four vectors, each at one of the corresponding points in the BCS (b).

Chapter 3: Branched Covering Space Construction

In this section, we describe the construction of the branched covering space given an input orientable manifold surface represented by a triangular mesh with a per-face N-RoSy field defined on it.

Our algorithm is based on the properties that:

1. The branched covering space is an N-fold covering space of the base manifold except around singularities, when there are fewer pre-images.
2. The N-RoSy field is a continuous vector field on the branched covering space such that the vector values at points that are the pre-image of a point in the base surface should collectively form the original N-RoSy.

Our algorithm consists of four stages:

First, we compute a subgraph G on the input surface M such that G contains all the singularities of the N-RoSy field and M can be cut open into a topological disk along G. In addition, we compute the gap for each edge in G, which is an integer between 0 and $N-1$ that describes how the vectors on different layers are assigned and how the layers are connected during the BCS construction process.
Second, we cut M open along G and obtain a topological disk M'.
Third, we replicate M' so that there are N layers $M_0, M_1, ..., M_{N-1}$, each of which is identical to M'. We then assign appropriate vector values at each face of each layer.
Fourth, we stitch M_0, M_1, M_{N-1} along their edges based on the gap computed for G, thus ensuring a manifold surface with a continuous vector field. Figure 3.2 the 2-fold BCS construction. As you can see in figure 3.2 torus is a 2-fold branched cover for sphere.

3.1 Cut Graph Calculation and Gap Computation

In this step, we compute a cut graph G which contains all the singularities in the N-RoSy field and along which the base surface S can be cut into a topological disk. A number of methods

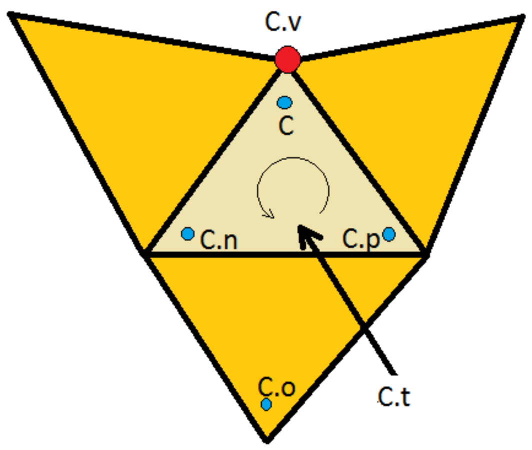
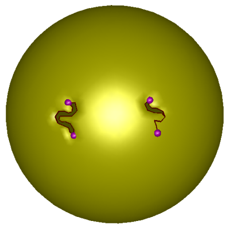


Figure 3.1: The corner data structure.

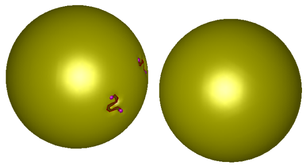
have been proposed to compute cut graphs [9, 3]. BCS has property which indicates that the final BCS does not depend on the actual cut graph, but only the N-RoSy field. Therefore, we employ a method based on region growing which is adapted from the Edgebreaker technique [17] including a data structure called *corners* from their work.

The corners is a data structure that, among other things, facilitates region growing. Every triangle contains three corners, each of which is associated with a vertex of the triangle. A triangular mesh of K triangles therefore has $3K$ corners. Note that many corners may be associated with the same vertex. Additional data (Figure 3.1) is stored for each corner c, such as the corner vertex $c.v$, the triangle containing c ($c.t$), the next and previous corners in the same triangle (respectively $c.n$ and $c.p$), the associated edge $c.e$ whose vertices are $c.p.v$ and $c.n.v$, and the opposite corner $c.o$ which is the only other corner satisfying $c.o.e = c.e$. Note that $c.e$ and $c.o.e$ can be considered as two half edges since $c.p.v = c.o.n.v$ and $c.n.v = c.o.p.v$. For a corner whose associate edge is on the boundary, its opposite corner is NULL. A useful property of the corners is that the boundary of a region on a triangular mesh, which is an ordered list of half edges, can be encoded by their associated corners.

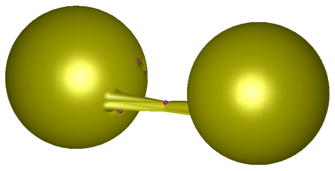
Our region growing process is as follows. Starting a seed triangle t, we iteratively add one triangle at a time across the boundary of our region. When adding a triangle across a corner c on the boundary corner list, we replace c with $c.o.p$ and $c.o.n$ on the boundary list, reflecting the new boundary. At the end of the region growing process, we will have covered all the triangles in



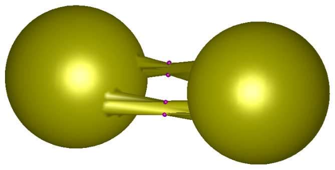
a



b

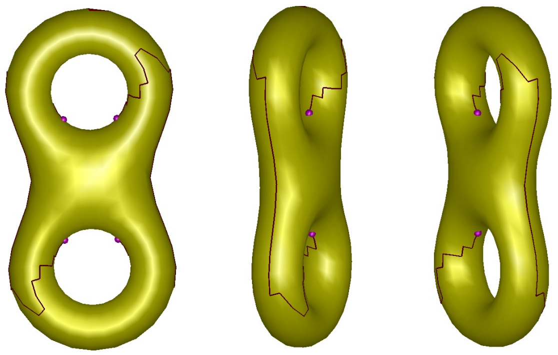


c

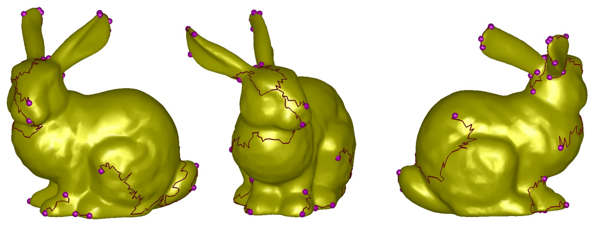


d

Figure 3.2: The construction of a BCS from an input mesh with a field (not shown): (a) mesh cutting along the cut graph, (b) layer replication, (c) layer stitching, and (d) final BCS.



(a)



(b)

Figure 3.3: The calculated cut graphs based on the 2-RoSy field after reduction for double torus (a) and for bunny (b).

the mesh since it is a connected manifold surface. Algorithm 1 demonstrate the region growing technique. This leads to a corner list of $2+|T|$ corners. Note that every vertex in the mesh is also part of the cut graph at this point. Cutting along this cut graph will lead to a topological disk.

However, this boundary list is usually unnecessarily long since every vertex in the mesh is part of the cut graph. While this does not present a problem to our BCS construction and visualization algorithms, it can lead to performance issues as every edge in the cut graph will be used during the cutting and stitching process. Note that a minimal cut graph only requires that it contains all the singularities in the N-RoSy field and all the homological generators. Consequently, we iteratively identify adjacent corners who form an opposite pair, i.e., in the boundary

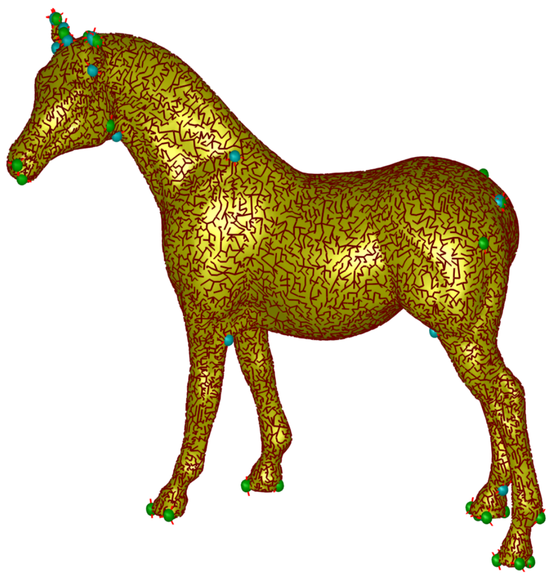
Algorithm 1 Boundary Edge Detection

```
Select a random seed triangle $t$
Add t's corners to the Boundary list $B$
**while** Num_of_Triangles in $Region$ $<$ Num_of_Triangles in M **do**
  choose a gate $g$ from $B$
  **if** $g.opposite.triangle = visited$ **then**
    choose another gate
  **end if**
  Parallel transport the field in $g.triangle$ to $g.opposite.triangle$
  **if** Field in $g.opposite.triangle$ is compatible to field in $g.triangle$ **then**
    **if** $Orientation = Clockwise$ **then**
      Swap $g$ with {g.opposite.next , g.opposite.previous}
    **else**
      Swap $g$ with {g.opposite.previous , g.opposite.next}
    **end if**
  **else**
    choose another gate
  **end if**
**end while**
```

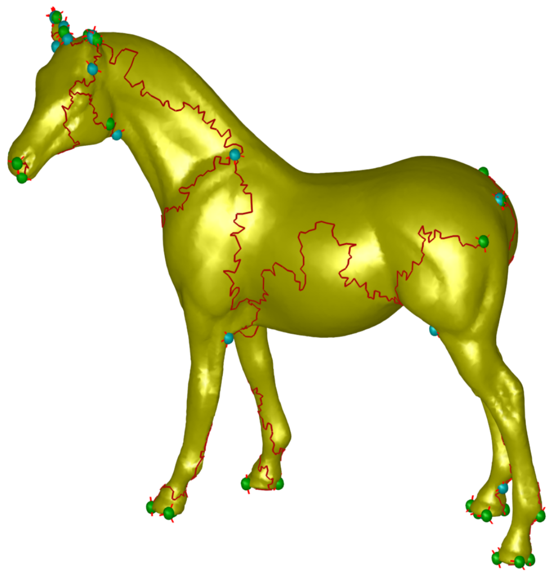
list $\{c_1,...,c_K\}$ where $c_{i+1} = c_i.o$. We will remove the corner pair from the boundary list if doing so does not lead to a singularity in the N-RoSy field to be disassociated with the cut graph. Repeating this process can lead to significant reduction to the number of edges on the cut graph without losing the singularities in the field and homological generators. Figure 3.4 compares the cut graphs before and after this reduction process.

Figure 3.3 shows the generated cut graph of bunny and double torus. This cut graphs are generated based on the 2-RoSy field on the models.

For generating the BCS we need to replicated the cutted open layer and stitch these layers together. This stitching process is based on how fields around the cut edges change; hence, we also compute the *gap* along the cut edges, which decides how different layers in the BCS are stitched together. This is computed by first numbering the N vectors in the seed triangle and then propagating the numbering to later added triangles through their corners. For example, when adding a triangle from a corner c on the current boundary, we consider the triangles $c.t$ and $c.o.t$. We take the first vector in the N-RoSy inside $c.t$, parallel transport it to $c.o.t$, and find among the N vectors inside $c.o.t$ the closest one and number it the first vector for $c.o.t$. Figure 4.1 shows this process. Since $c.t$ and $c.o.t$ are not in the same plane, we need to somehow map these two



(*a*)



(*b*)

Figure 3.4: The cut graph at the end of the region grow process (a) and after reduction (b).

faces to a plane and then compare the vectors in the triangles. This process is called parallel transport.

At the end of the region growing process, we have numbered all the vectors inside every triangle in M such that if two triangles shared an edge that is not on the cut graph, then their respective first vectors will be the closest to each other than any other $N-1$ vectors. However, if two triangles share an edge that is on the cut graph, then if means that their first vector in the N-way rotational symmetry field has more than $\frac{2\pi}{N}$ difference. In this case, the first vector in one triangle can be closest to any of the $N-1$ vectors in the other triangle. The gap from the first triangle to the second triangle is therefore $K-1$ where v_K is the k's vector inside the second triangle that is closet to the first vector in the first triangle. Note that if the gap from one triangle to the other is L, then the gap from the second triangle to the first will be $N-L$.

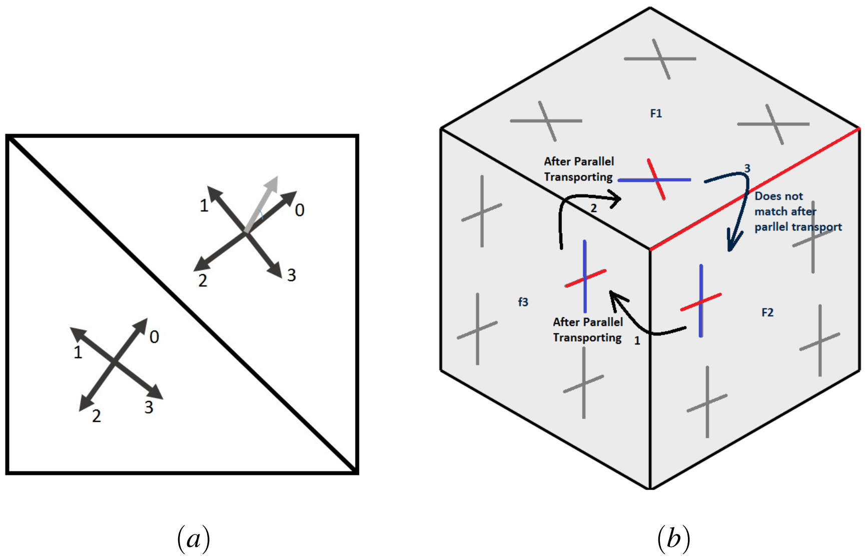

1
0
2
3
1
0
2
3
F1
After Parallel Transporting
2
3
Does not match after parllel transport
f3
After Parallel Transporting
1
F2
(a)
(b)

Figure 3.5: (a) shows An 0-indexed vector is propagated from one triangle to the next during region growing, and (b) shows the propegation process of this technique.

3.2 Mesh Cutting

In the second stage of our BCS construction, we cut the mesh open along the cut graph computed in the first stage.

We define the *cut degree* of a vertex on the cut graph as the number edges (not half-edges) on the cut graph that are incident to the vertex. These edges will divide the 1-ring neighborhood of the vertex into P sectors where P is the cut degree. During mesh cutting along the cut graph, the sectors will be disassociated from each other. That is, every vertex needs to be replicated P times.

Once the vertices have been duplicated, we traverse the cut graph. If two consecutive corners satisfy that the end vertex of the half edge corresponding to the first corner is the starting vertex of the half edge corresponding to the second corner, this indicates that the halfedges are the boundary of a sector for the common vertex. We therefore iteratively find all the triangles between the two half edges and replace their common vertex with the next available duplicated vertex. At the end of this process, the mesh has been cut open along the cut graph.

3.3 Mesh Replication

Once the mesh is cut open, we replicate the cut-open mesh N times, resulting in N co-located meshes each of which is given an index between 0 and $N-1$. For a triangle in the original mesh, its N-RoSy vectors are assigned to the corresponding triangles in each duplicated layer based on the aforementioned layer index. That is, the first vector is assigned to the triangle on the first layer, and the second triangle to the second layer, and so on. This results in a triangle mesh with N connected components and a vector field. Figure 3.6 demonstrate this process.

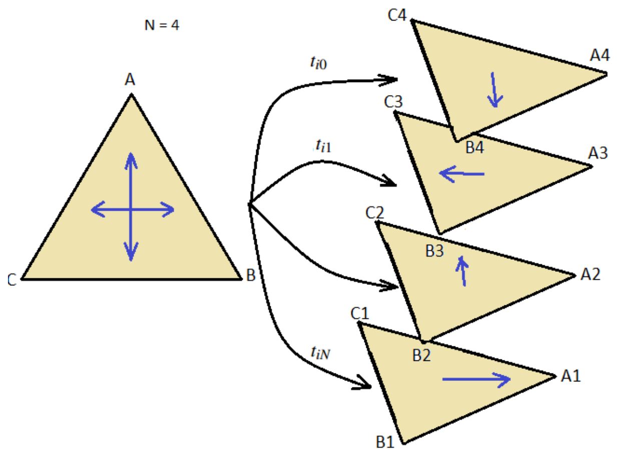


Figure 3.6: Replication of one triangle for 4-RoSy field.

3.4 Mesh Stitching

In the last stage, we stitch the N layers together to form the BCS. This is essentially an inverse process to the cutting stage except that there are more layers to stitch. Given an edge on the original cut graph, the two triangles incident to the edge correspond to $2N$ triangles in the replicated mesh. We will stitch triangle $t_{1,j}$ to $t_{2,L+j}$ where L is the gap from triangle t_1 to t_2 and $1 \leq j \leq N$. Once all the edges have been glued together, we remove redundant vertices. This results in the BCS.

3.5 Cut Graph Improvement with Essential Cut Graphs

The above process can be further improved with our observation that it is unnecessary to cut the mesh open along an edge in the cut graph that has a zero gap. This leads to the notion of *essential cut graph* which is a subgraph of the cut graph without edges with a zero gap. Figure 3.7 compares the cut graph and the essential the graph. Note that the essential cut graph is usually not connected, and cutting the mesh open along the essential cut graph is usually not a topological disk. However, we note that cut-open mesh is a mesh with multiple boundaries, and with minor changes to our cutting and stitching algorithms (assuming only one boundary), we can construct the BCS using the essential cut graph.

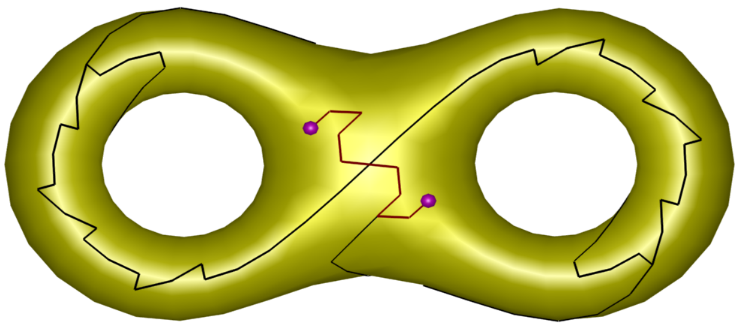


Figure 3.7: The cut graph contains many edges over which the gap is 0 (black edges). The essential cut graph is a subgraph of the cut graph that consists of only edges whose gap index are not zero (red).

The orientation of the constructed BCS depends on the orientation of the base surface. If we

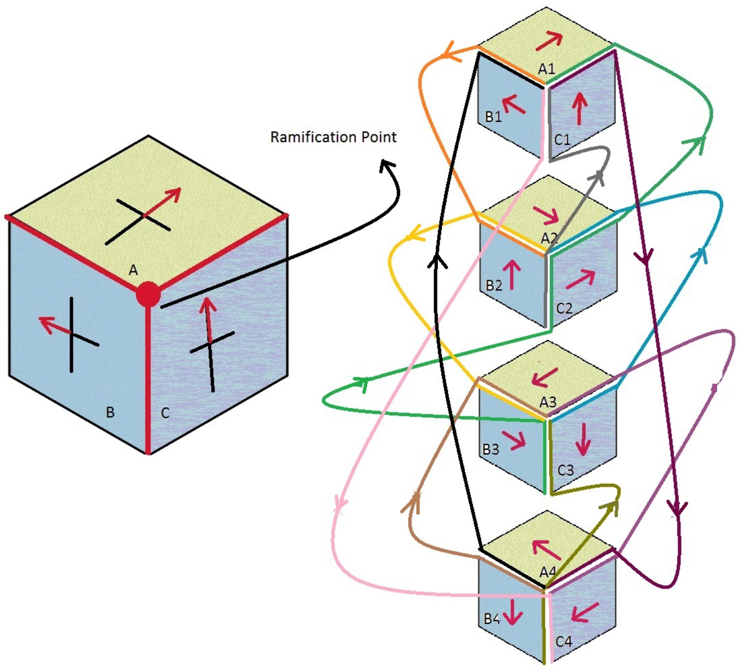


Figure 3.8: different layers have different directional field.

apply BCS construction algorithm to an orient-able surface, the generated BCS will be orient-able, otherwise, if we apply BCS construction algorithm to a non-orient-able surface, the BCS will be non-orient-able.

Figure 3.8 shows how different layers have their own vector field. It shows how the stitching process is done around a ramification point. Red lines on the left cube are the cut edges and cubes on the right are different layers.

Chapter 4: Branched Covering Space Visualization

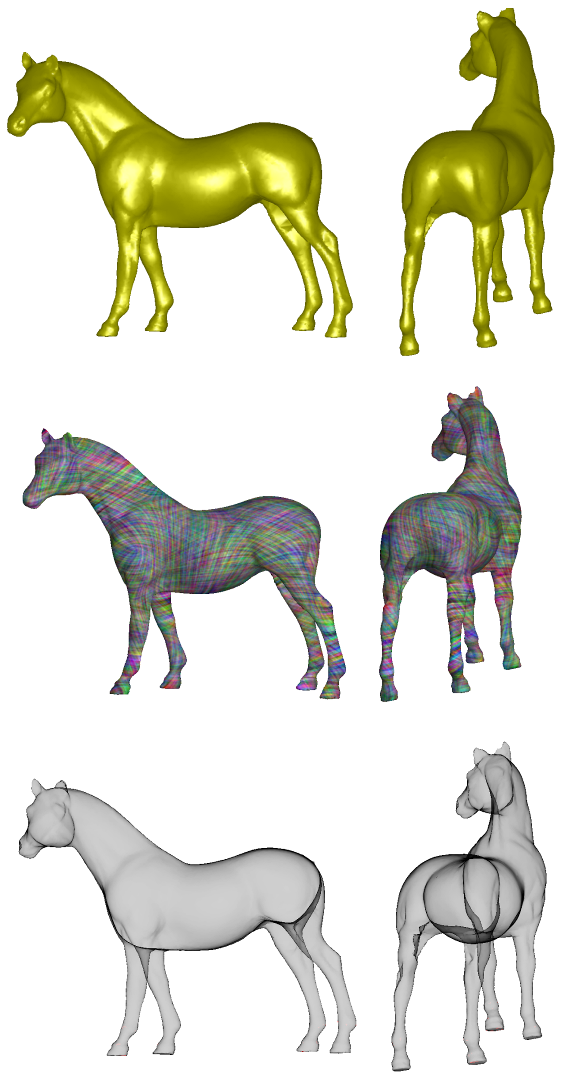


Figure 4.1: Top image shows A horse rendered in regular rendering, middle image shows a horse rendered in flow visualization and botton image show a horse rendered translucently.

The BCS constructed by our algorithm results in a topological valid but geometrically uninteresting BCSs as all the layers are co-located. This makes it difficult to see that the BCS is indeed a *N*-fold cover of the original mesh, nor is it obvious why the BCS has more handles than the original mesh and that *N*-RoSy field is lifted to a vector field on the BCS. We address these difficulties by combining a number of strategies, such as translucent rendering (SmokeSurface rendering mode) , flow visualization (Line integral convolution), and user-guided mesh deformation.

BCS has various interesting properties:

1. N-fold branched covers are used to convert N-way rotational symmetry fields to vector fields by lifting process.
2. N-fold branched covers are N layer of the base space which are connected in case of singularities.
3. N-fold branched covers are closed manifold.
4. N-fold branched covers are orient-able if the base surface is orient-able.
5. Topology of the BCS is related to the topology of the base space.
6. BCSs are dependant on the topology and the field of the base space and they are independent of the cut graph.

In our visualization tool we have added different functionalities to cover these concepts. As said earlier we are using different strategies such as:

- Translucent rendering
- Flow visualization
- Mesh deformation

4.1 Visualization Schemes

Different surfaces with various geometry features make it impossible to have a unique visualization pipeline for all the surfaces. We have presented an interactive BCS visualization tool so that users can design a BCS based on their needs.

4.1.1 Translucent Rendering

Branched Covering Space contains multiple self intersection geometries and with the regular solid surface rendering, it is impossible to visualize all the layers. We need to have a transparency visualization tool to see all the layers and their connectivity. Rendering a correct transparency material requires sorting fragments in depth order. Once we have the BCS we can visualize it by translucent rendering [23] This allows the user to see how the layers are connected and gain intuitions on why self-intersections tend to occur.

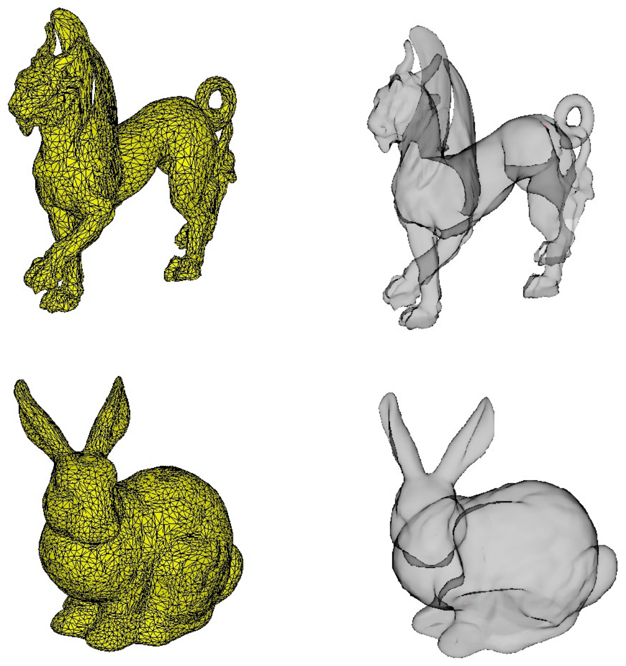


Figure 4.2: Feline and bunny rendered in translucent mode.

Figure 4.2 shows feline and bunny rendered in regular and SmokeSurface techniques. Regular and translucent techniques have their own advantages and disadvantages. In regular rendering technique, adjacency information can be visualized clearly. However, if we want to see the back side and front side of the model simultaneously or to see the self-intersections, SmokeSurface works better.

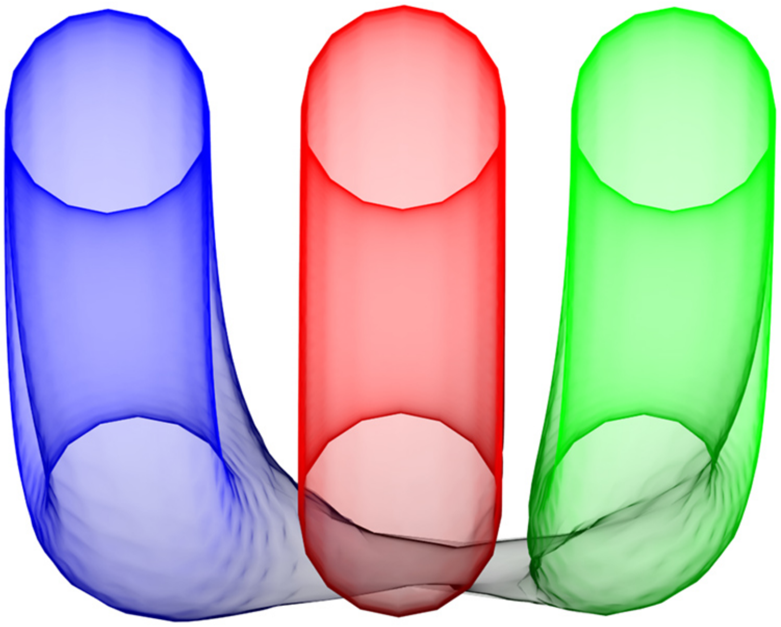
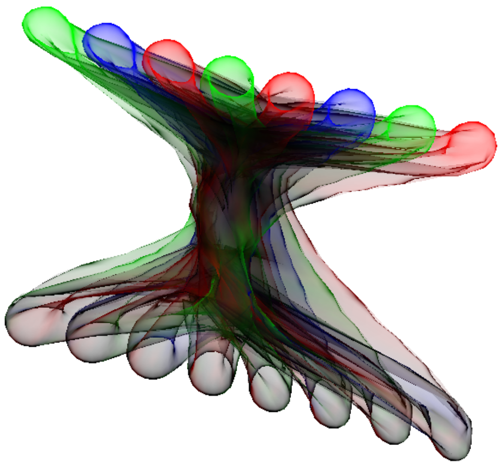


Figure 4.3: A 3-fold branched cover of torus.

Unfortunately SmokeSurface visualization can not work effectively for higher Ns due to the complicated geometry. As you can see Figure 4.3 SmokeSurface for relatively small Ns works perfect and shows the connectivity very clearly. Figure 4.4 shows an 8-fold branched cover of double torus and as you can see this translucent rendering is not effective in showing the inside geometry.

4.1.2 Flow Visualization

The main reason that we have implemented this technique is to show the lifting property of the BCS. It is important to see the field on each of these layers. We have implemented two flow visualization techniques, static and dynamic. Dynamic flow visualization technique is very helpful if we want to show the lifting process from a line field to a vector field. Line field and vector field look the same in regular points except singularities.



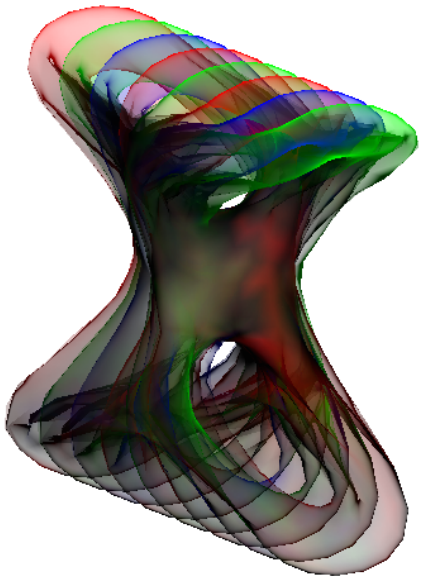


Figure 4.4: Two different viewpoints of a 8-fold branched cover of double torus.

Cabral, and Leedom [4] proposed a method called ”Line Integral Convolution” for rendering particle flows following the vector field by imaging two or tree dimensional vector fields. Given a vector field, LIC algorithm would trace the vectors ac each point. Palacios, and Zhang [12] have adapted the LIC algorithm for N-way rotational symmetry fields.
Figure 4.5 shows 3-RoSy field on torus and also the respective vector field on the BCS.

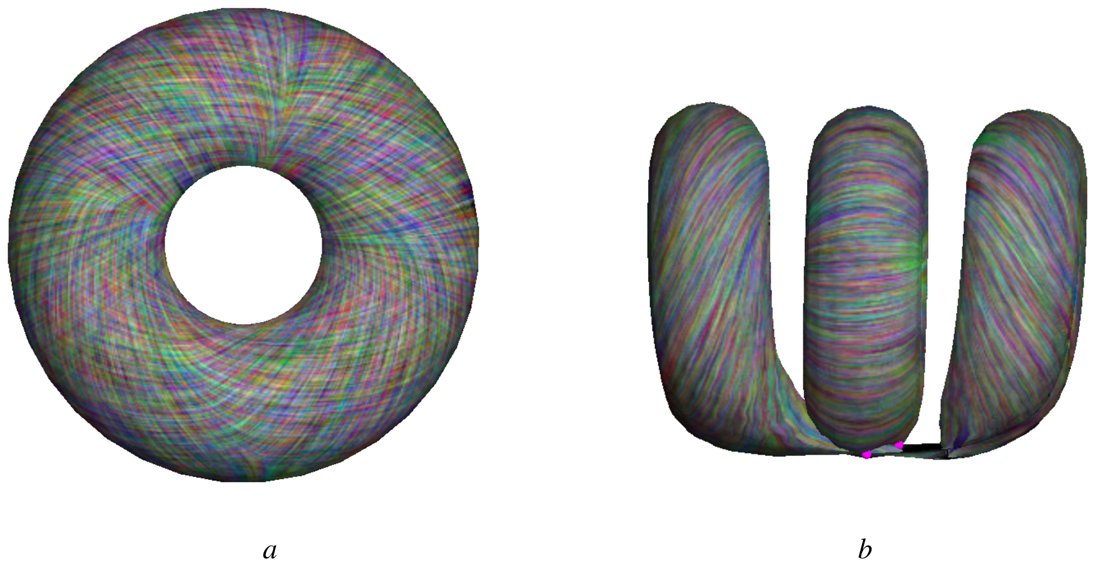


Figure 4.5: a) A torus with 3-RoSy field and 2 singularities, b) 3-Fold branched cover of torus, As you can see the 3-RoSy field have been lifted to vector field in the BCS and each of the layers have just one direction.

4.1.3 surface deformation

Initially after construction all N-folded layers in BCS co-locate. We have to deform the layers to distinguish each layer clearly. Interactive surface deformation techniques were used to locally and globally deform the layers and show the geometrical connections clearly. Our system allows the BCS to be deformed as a whole, or per each layer.

Our deformation should be based on these two main criteria:

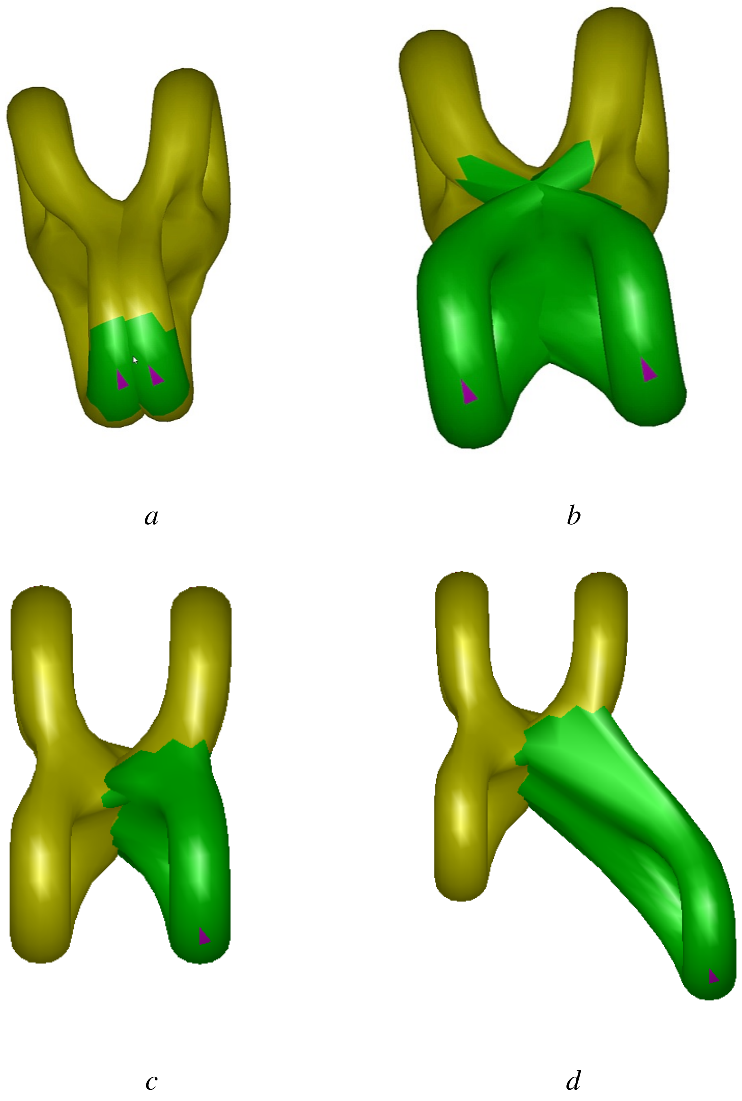


Figure 4.6: Our system allows the BCS to be deformed interactively: (a) regions to deformed (green) around the deformation handle (purple triangles) symmetrically, (b) the deformed BCS, (c) regions to deformed (green) around the deformation handle (purple triangles) asymmetrically, and (d) deformed BCS.

1. Each layer needs to look like the original surface (base space).

2. All the layers should be separated to be seen clearly.

During the deformation process, we strive to maintain the shape of each layer in order to maintain their recognizability. To achieve this, we reuse the framework of [19], which transfers the deformation from one surface (source) to another (target). Given an initial pose of the source mesh $P_{s,1}$ and the pose after deformation $P_{s,2}$, the framework seeks to extract the deformation and apply it to the initial pose of the target mesh $P_{t,1}$ to produce $P_{t,2}$. The vertex locations for $P_{t,2}$ are computed by minimizing an energy function which is the total squared difference between the deformation of every triangle in $P_{s,1}$ and its corresponding triangle in $P_{t,1}$.

In our case, we do not have two meshes. Therefore, to apply the framework, we let the user select a subset of triangles in the mesh which can serve as the deformation handle. The user can translate and rotate these triangles. The affine transformation from this subset of triangles and the rest of the triangles in the mesh will be maintained as much as possible during the deformation of the rest of the triangles. This leads to a similar framework.

When deforming each layer, we automatically compute a seed in the base mesh and use its pre-images in the BCS as the seed triangle for each layer. Since the self-intersections occur most around singularities, we wish to select a seed that is farthest away from the singularities in the base mesh.

We can translate and rotate each layer independently as seen in figure 4.6, leading to the spatial separations of the layers. The users can clearly see that there are N shapes similar to the base mesh that are connected. To help with seeing the connection between the layers, we employ the SmokeSurfaces technique which shows all the layers in a translucent fashion. We have found that the way in which the layers connect is made strong if the user can interact with the BCS by moving, rotating each layer.

4.2 Feature Visualization

As mentioned in chapter 4, BCSs have various important features. In this section we talk about how our visualization present these futures:

1- BCS is indeed N layers of the base space connected in some points (Ramification points).

This property can been shown with all the visualization techniques, but the smoke surface is the best way to show this property since you can actually see all the layers and their intersections.

Figure 4.3 is a good example how of how our system shows all different layers.

2- BCS is a manifold even around singularities.

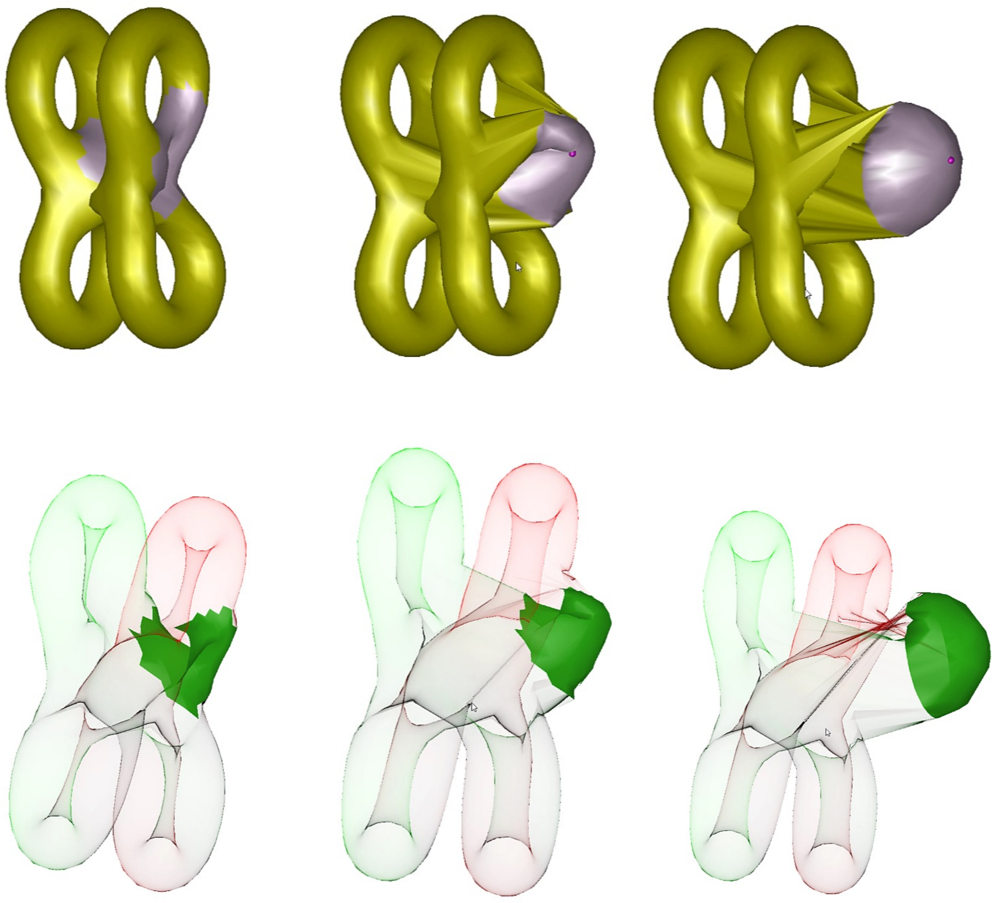


Figure 4.7: Three frames from an animation of the unfolding near a singularity both in regular and SmokSurface rendering mode.

BCS is a manifold around the singularity in the *N*-RoSy. Due to the self-intersections around singularities, it often leads to misconceptions that the singularities are non-manifold points on the BCS. In our visualization, we compute a neighborhood of a user-selected singularity, cut it open from the rest of the BCS, and unfold it onto a hemisphere. The unfolding is achieved by first unfolding the neighborhood onto a disk in the plane using parameterization [18] and then projecting the vertices on the hemisphere using stereographic projections. We then perform animation from the original patch to its image on the hemisphere by interpolating their vertex location. The user can view the animation either in solid rendering or translucent rendering.

We can also show this unfolding process while the neighborhood is connected to the rest of the BCS. In this case, the latter is divided into two regions, a fixed region and a transition region. The fixed region will not be moved during the BCS deformation process while the transition region will deform to connect the hemisphere with the fixed region on the BCS. In this case, the positions of the vertices inside the transition region are computed by solving the Laplacian equations with the vertex positions in the neighborhood and fixed regions as the boundary conditions. Figure 4.7 shows a few frames during the animation. The animation can be visualized using solid and translucent rendering as well as static and dynamic LIC. In our system, the user can select the size of the neighborhood and the size of the transition point. Furthermore, the user can rotate and translate the hemisphere in order to reduce stretch in the transition region.

3- The topology of the BCS is related to the topology of the base space.

The Riemann-Hurwitz formula is an important result of the BCS, which relates the Euler characteristic of the base mesh to that of the BCS. Our system provides visualizations for this in a number of different ways. we compute a homological generator per handle in the BCS using the technique of [7]. The homological generators are visualized on the BCS. However, some handles are difficult to see due to the self-intersections in the BCS. In this case, we allow the user to deform the BCS in order to make the handle visible using the aforementioned deformation framework. Moreover, we provide a more direct way of visualizing a handle in a fashion similar to visualizing a neighborhood of a singularity. First, we grow a topological cylinder from the homological generator by performing region growing from the a region of zero triangles and two boundaries (the homological generator is treated as two co-locating boundaries). The region is grown by adding one triangle at a time until a user-specified distance between the two boundaries is reached. The topological cylinder is then mapped to a canonical half torus (also a topological cylinder) of a user-given outer and inner radii. This is achieved by finding a shortest path between the two boundaries on the topological cylinder on the BCS and unfolding it onto a rectangle

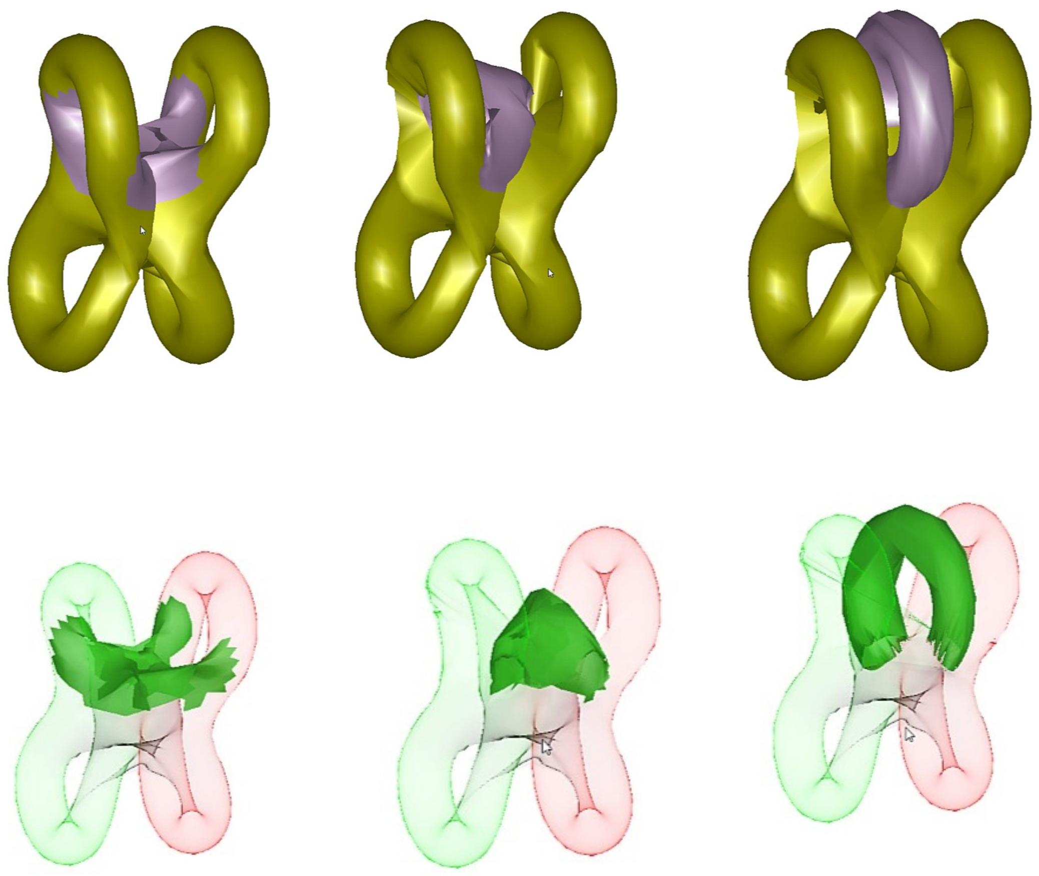


Figure 4.8: three frames in the animation of inflating a handle in regular and also SmokeSurface that is otherwise difficult to see due to the self-intersections in the BCS.

such that the top and bottom sides correspond to the shortest cut and the left and right sides correspond to the two boundaries of the cylinder. We ensure that each vertex on the shortest path is mapped to two points on the top and bottom sides with same the X-coordinates. The interior vertices are then solved using the parameterization technique [18]. Since a mapping from a rectangle to a half torus is known, we have now mapped the topological cylinder to the half torus. Again, the user can choose to only see the topological cylinder deforming into a half torus, i.e., disconnected from the rest of the BCS, or deforming a transition region between the topological cylinder and the fixed region on the BCS. The user can also change the size of the topological cylinder, the transition region, as well as the inner and outer radii, orientation, and location of the half torus. Solid and translucent rendering as well as static and dynamic LIC are used to see the deformation. Figure 4.8 shows intermediate results of the animation.

Another reason that handle visualization is important is that we want to show how singularity cancellation impacts the number of handles. The number of handles in a BCS is calculated based on this formula:
$g(S') = \frac{M*(N-1)-N\chi(S)}{2} + 1$ where M is the number of ramification points in the BCS, S is the base space, S' is the N-fold branched cover, and also $g(S')$ represent the number of handles in branched cover. This shows that by cancelling two singularities we are increasing the Euler Characteristic of the BCS which means we lose handles.
For this comparison we need the BCSs of two N-RoSy fields f_1 and f_2, where f_2 is obtained from f_1 through singularity pair cancellation. Figure 4.10 shows two such fields. The user can then compare their respective BCSs and see that the Euler characteristic of the BCS corresponding to f_2 is increased by $2(N-1)$, which in this case indicates one more connected component due to the removal of the bridge that used to connect the two layers. Furthermore we show the Riemann-Hurwitz formula by contrasting the visualization the construction of the BCS for an N-RoSy field f_1 and a second N-RoSy field f_2 obtained by removing a singularity pair.

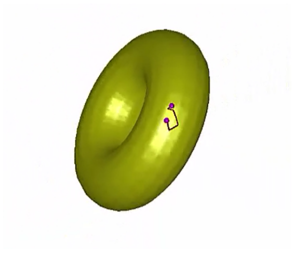
This formulation also shows that by moving a pair of singularity and not cancellling them we are not changing the topology of the BCS; hence, the number of handles should remain the same but the width of the handles would change which you can see in figure 4.10 that (b) and (e) are topologically the same but they are different in the width of the handle.

4- Singularity of index k ***would transfer to ramification of index*** $(N*K)-(N-1)$**.**

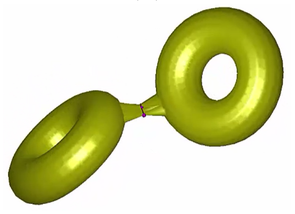
To show this property we can inflate that specific singularity and apply flow visualization techniques to it. we use LIC to show the relationship between a singularity's index I_p and the index of its corresponding ramification point I'_p, i.e., $I'_p = NI_p - (N-1)$. Figure 4.11 shows this in LIC. You can see that in the base mesh we have a trisector with three sectors which in the BCS we have monkey saddle with six sectors which represent another property of the BCS that multiplies the number of sectors by N.

5- BCS is independant of the cut graph.

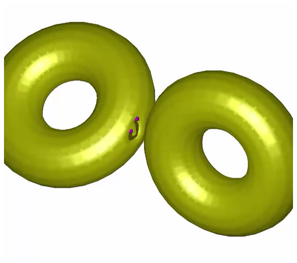
This is shown by computing the BCS using two cut graphs and show that the resulting BCSs are identical. For this purpose, We construct a BCS and do a reverse construction bu the new cut graph and we see that the resulting surfaces are identical to each other. We provide animations that show the computation of the cut graph, the cutting stage, the layer replication stage, and the stitching stage. Figure 4.6 shows some frames.



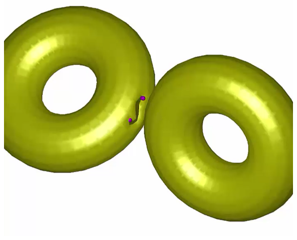
(a)



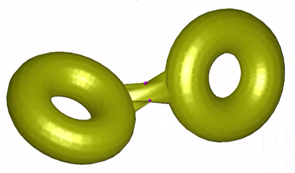
(b)



(c)



(d)



(e)

Figure 4.9: (a) Cut garph generated on the torus with 2 singularities. (b) 2-Fold branched cover of torus. (c) 2-Fold branched cover of torus with unstitched singularities. (d) 2-Fold branched cover of torus with unstitched moved singularities. (e) 2-Fold branched cover of torus.

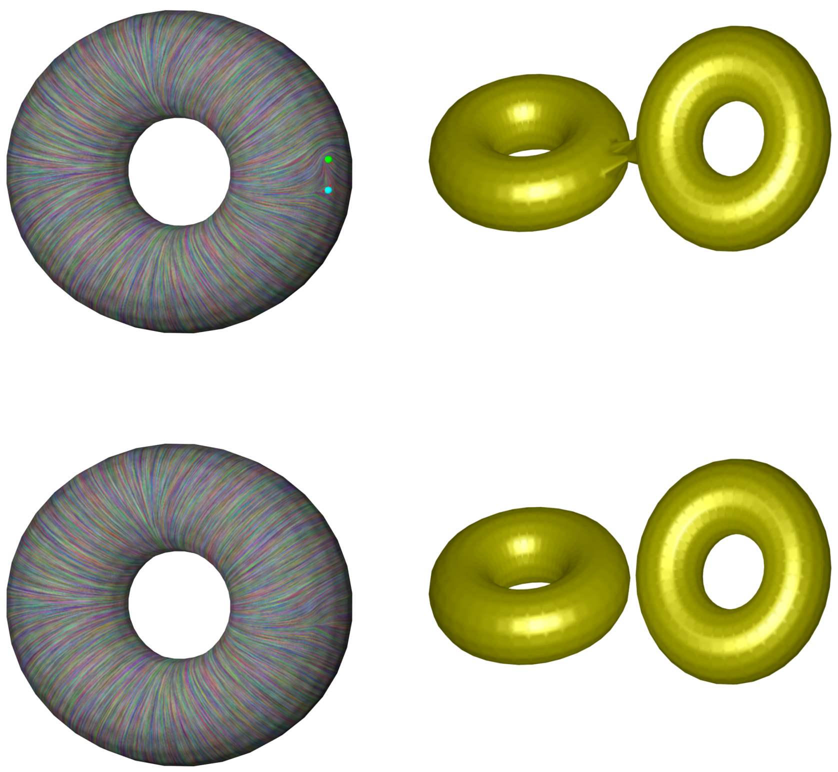


Figure 4.10: An 2-RoSy field on the torus (upper-left) leads to a BCS of Euler characteristic of -2 (upper-right). After cancelling the singularity pair (lower-left), the bridge connecting the two layers in the BCS disappears which results in a BCS of Euler characteristic of 0.

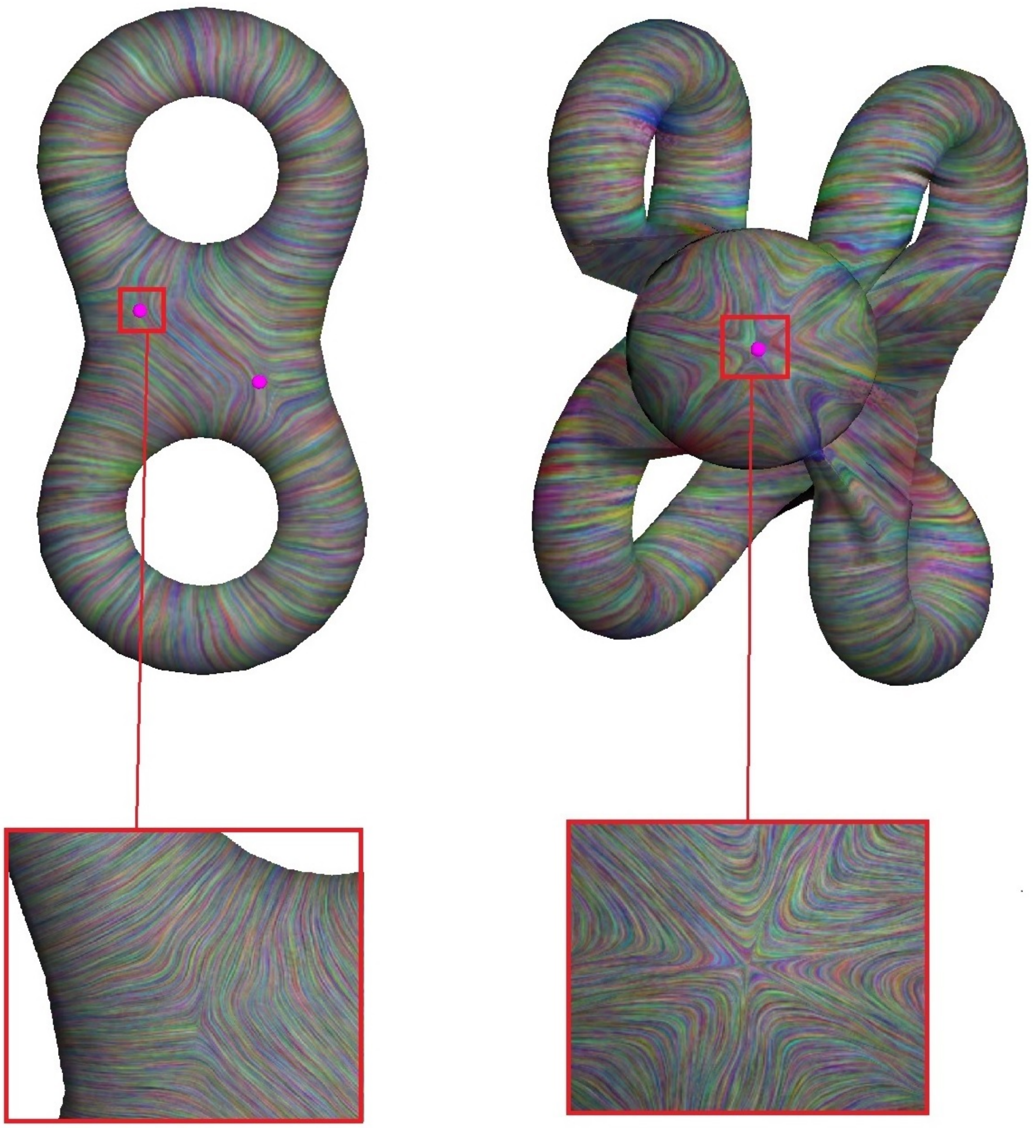


Figure 4.11: A $-\frac{1}{2}$ indexed singularity in the 2-RoSy field on the double torus (left) corresponds to a -2 indexed singularity in the vector field in the BCS (right).

Chapter 5: Result

Figure 5.1 shows the controls available for the users to interact with the BCS construction and also visualization. As seen in the figure, our system would give a full control to the users for local and also global deformation. This deformation can be applied as whole BCS or just to one specific layer. Deformation handle selection can be controlled by the slides provided in the system. User can select different models and different fields. Users can edit the loaded field by adding or cancelling singularities. User can select the visualization scheme. Inflation animations for singularities and hidden handles are controllable by the sliders.

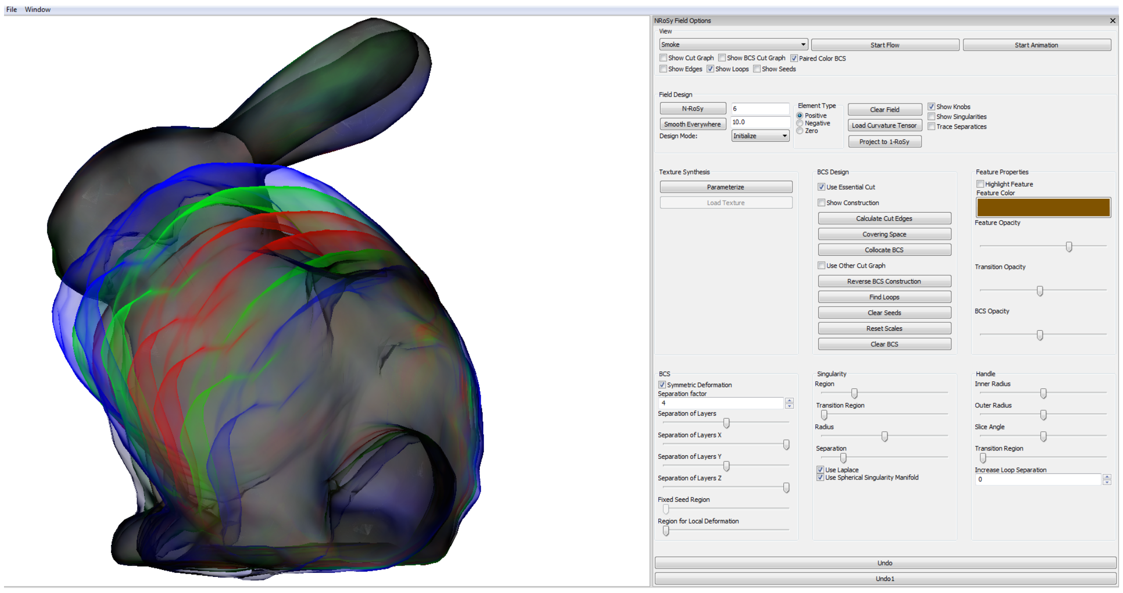


Figure 5.1: The controls available to the users for BCS construction and also visualization.

Our system can create N-fold branched covers for any N and any size mesh. Some images has been added to this chapter to represent the robustness of our construction and visualization algorithms.

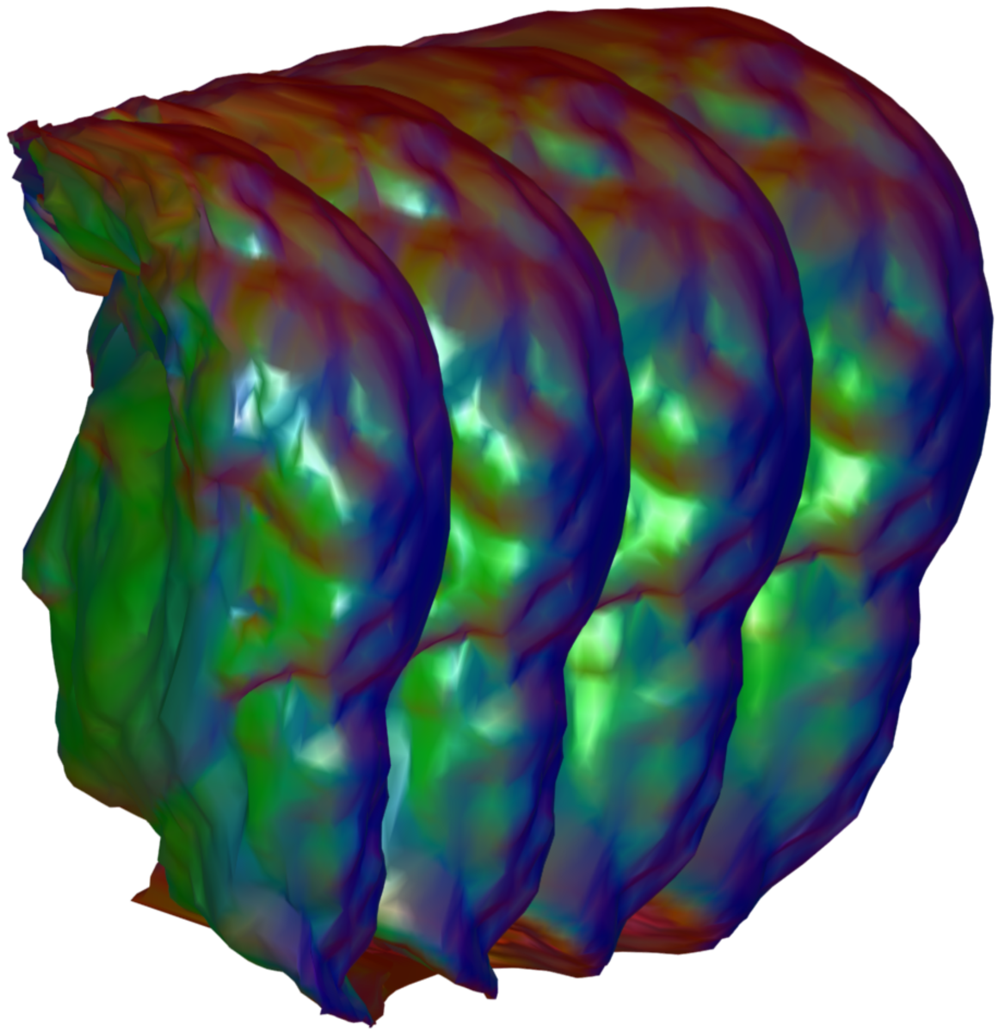


Figure 5.2: 4-Fold Branched Covering for David statue, color coded with normal map.

Figure 5.2 and 5.4 are rendered in regular visualization scheme but color coded in normal map. Normal map can easily show the property of the covering space that it consists of N identical layers that are connected in some points.

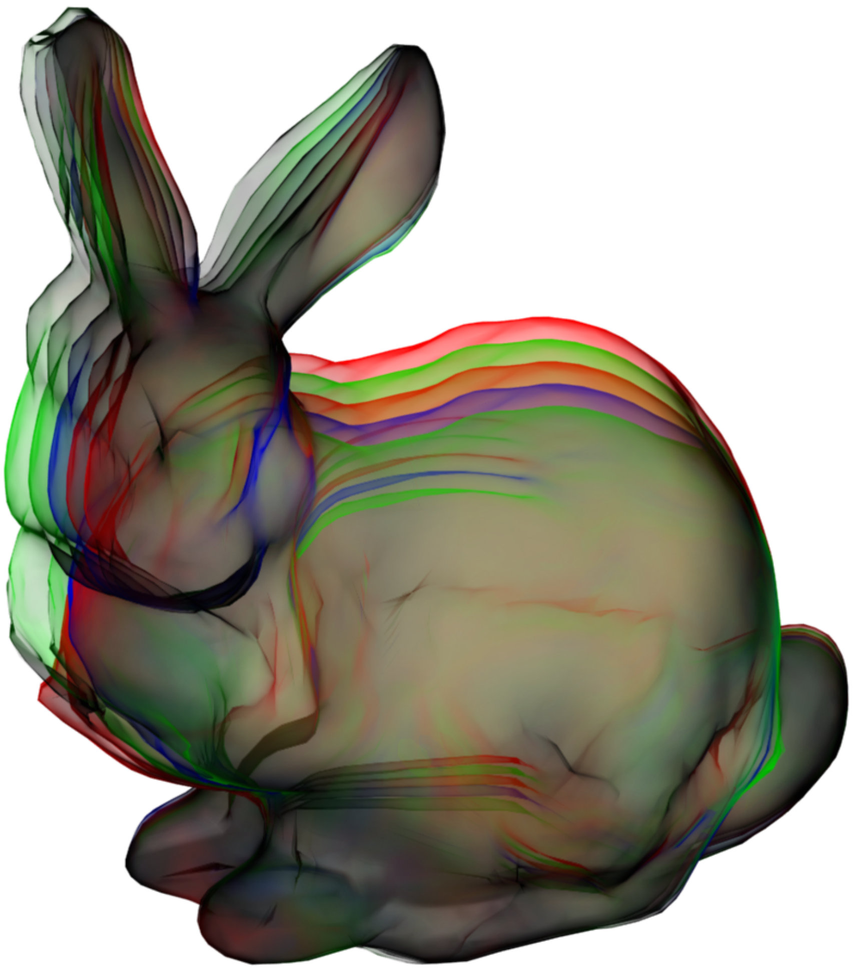


Figure 5.3: 5-Fold Branched Cover for bunny rendered with SmokeSurface.

Figure 5.3 and 5.5 are rendered in translucent mode which makes it easier to see the different layers and their connections. Figure 5.4 has been deformed and the fact that Buddha has two heads is showing that it is a 2-fold branched cover.

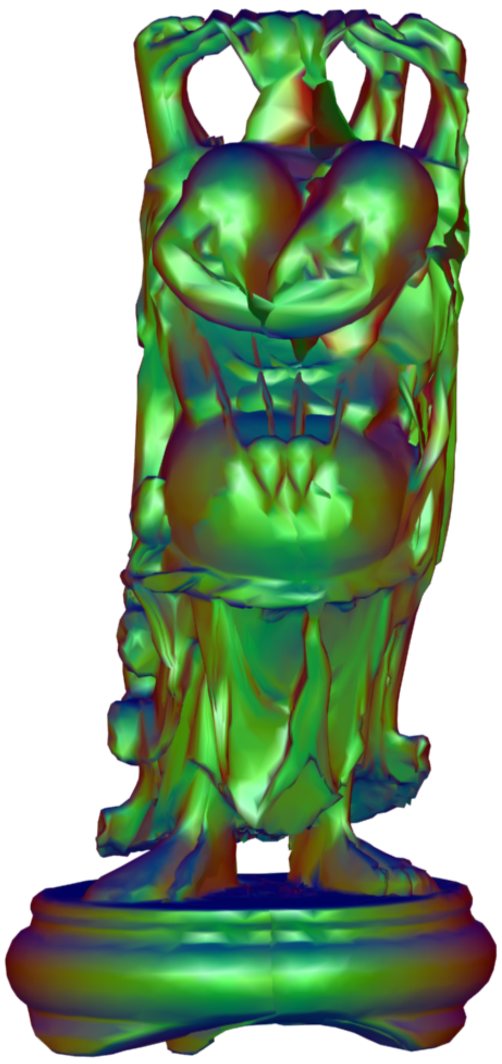


Figure 5.4: 2-Fold Branched cover for Buddha, color coded with normal map.

Our BCS construction can be used to create self-intersected surfaces. In computer science we have different algorithms that have been tested in non-self intersected surface and having a tool to easily create one is very helpful.

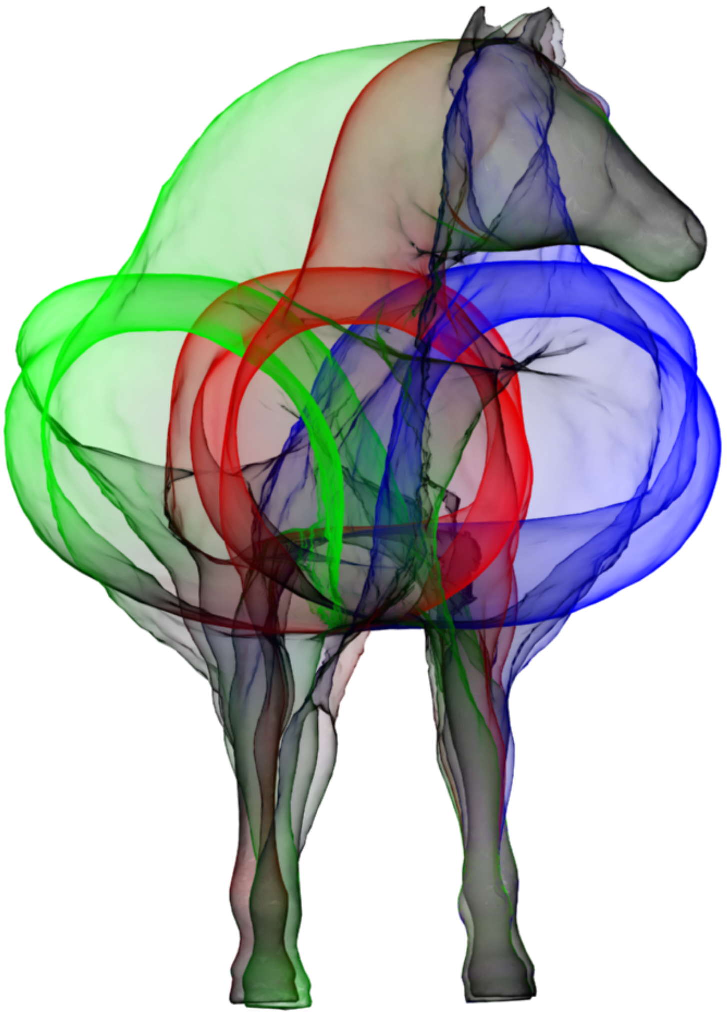


Figure 5.5: 3-Fold Branched cover for horse rendered with SmokeSurface.

Chapter 6: Conclusion and Future Work

6.1 Performance

Our system has been tested on a system of with 4 $i7$ cores with 2.8 Ghz speed and an NVidia GeForce 310 graphics card. Our LIC and SmokeSurface are implemented using the Shaders while the mesh deformation is achieved on the CPU. The time to perform one step in mesh deformation depends on the size of the input mesh and ranges from a few frames per second (e.g., sphere, torus, double torus, which have up to $2,000$ triangles) to two seconds (e.g., bunny, horse, and buddha, which have $10,000-40,000$ triangles). In chapter 5 we presented various BCSs designed by the user using our system. The running time is impacted by the mesh size and also the number of layers in covering spaces. The computation time for each step is as follows:

- **Cut Edges Calculation** $\rightarrow O(|F|)$
  This section is completed by a region growing technique that uses the Breadth-first search algorithm to proceed. Since it is using the Breadth-first search algorithm, the time complexity can be expressed as $O(|V|+|E|)$ in the dual graph of the mesh or $O(|F|+|E|) \rightarrow O(4|F|) \rightarrow O(|F|)$ for the actual mesh.

- **Cutting** $\rightarrow O(|H|)$ **which H is the number of cut edges in the system**
  For the cutting part, the time complexity can be expressed as $O(|F|)$ since we need to travel through the boundary edges. The number of boundary edges in the mesh is initially equal to $|F|+2$. After Construction the cut edges we perform a cut edge reduction which remove all the dangling edges form the cut graph and make it shorter.

- **Replication** $\rightarrow O(|F|)$
  In this process we go through each face and replicate it N times, so the time complexity will be $O(|F|)$.

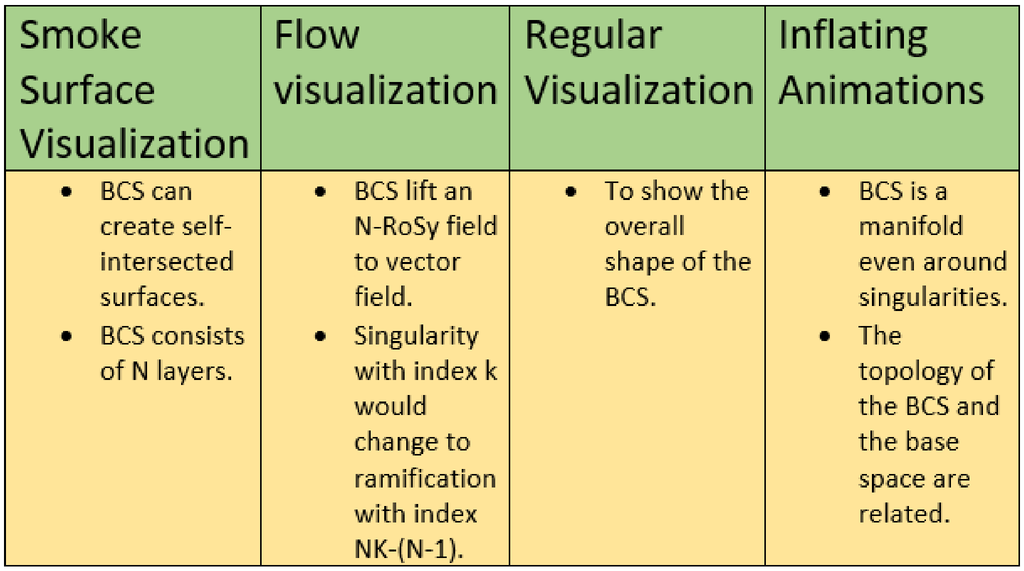
- **Stitching** $\rightarrow O(|NH|)$ **which N as the number of layers and H is the number of cut edges**

Time complexity of this section is similar to the time complexity of the Cutting section since we travel through the boundary of patches and connect the patches together.

6.2 Comparison on Visualization Methods

We have presented our system to 40 graduate and undergraduate students of Mathematics and Computer Science as well as a number of Mathematics faculty members teaching classes in topology, differential geometry, and complex analysis. We have received positive feedbacks on the effectiveness of our system in demonstrating the mathematical properties of the BCSs using mesh deformation as well as SmokeSurface, Normal Maps, and LIC visualizations. In fact, visualizing BCS construction using animation was suggested by our audience and thus implemented. They also have provided suggestions for reducing the amount of self-intersections in the BCSs, which we plan to leverage in our future research on the subject.

In conclusion each of the presented visualization methods can cover some of the properties of the covering spaces as you can see in figure 6.1.



Smoke Surface Visualization	Flow visualization	Regular Visualization	Inflating Animations
• BCS can create self-intersected surfaces. • BCS consists of N layers.	• BCS lift an N-RoSy field to vector field. • Singularity with index k would change to ramification with index NK-(N-1).	• To show the overall shape of the BCS.	• BCS is a manifold even around singularities. • The topology of the BCS and the base space are related.

Figure 6.1: Visualization techniques comparisons.

As you can see SmokeSurface visualization is helpful for showing the self-intersection areas and multiple layers due to the transparency effect. Static and Dynamic flow visualization is helpful in showing the lifting process. In our system you can actually see the transformation from a singularity to a ramification point with flow visualization and also inflating animation combined. Inflating animations are helpful in showing areas around singularities and also handles which are difficult to see.

6.3 Future Work

In this research, we introduced the problem of constructing and visualizing BCSs given an input manifold surface with an *N*-RoSy field. We also provided algorithms and practical implementations to achieve this. The system allows a user to see the construction of the BCS as well as examine the BCS near a singularity and a handle through user interaction and animation. The system employs visualization techniques such as solid and translucent rendering and static and dynamic LIC. In addition to the system, we improved the construction of the BCS by using essential cut graphs instead of cut graphs. Our system can generate meshes with self-intersections, which can be useful to produce models to test the robustness of existing geometry processing tools. We have also sought suggestions and feedback from Mathematicians and graduate and undergraduate students in Mathematics and Computer Science on our approach.

Our system is not without limitations. We have noticed that the SmokeSurface technique does not work well when *N* is higher than ten. In addition, it is difficult to design a BCS that has relatively few self-intersections and low stretch. These are all directions that can be pursued in the future, Specifically, automatic placement of each layer of the BCS that would lead to minimal stretch and self-intersections.

Bibliography

[1] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 485–493, New York, NY, USA, 2003. ACM.

[2] M.A. Armstrong. *Basic topology*. McGraw-Hill Book Co., 1979.

[3] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 77:1–77:10, New York, NY, USA, 2009. ACM.

[4] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 263–270, New York, NY, USA, 1993. ACM.

[5] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1057–1066, New York, NY, USA, 2006. ACM.

[6] Robin Hartshorne. *Algebraic geometry*. Graduate texts in mathematics. Springer, New York, 1977.

[7] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 203–212, New York, NY, USA, 2001. ACM.

[8] Bruno Jobard and Wilfrid Lefer. Creating evenly-spaced streamlines of arbitrary density. pages 43–56, 1997.

[9] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Comput. Graph. Forum*, 26(3):375–384, 2007.

[10] Matthias Nieser, Jonathan Palacios, Konrad Polthier, and Eugene Zhang. Hexagonal global parameterization of arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):865–878, June 2012.

[11] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[12] Jonathan Palacios and Eugene Zhang. Interactive visualization of rotational symmetry fields on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 17(7):947–955, 2011.

[13] Konrad Polthier and Eike Preu. Identifying vector field singularities using a discrete hodge decomposition. pages 112–134. Springer Verlag, 2002.

[14] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, October 2006.

[15] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. Geometry-aware direction field processing. *ACM Trans. Graph.*, 29(1):1:1–1:11, December 2009.

[16] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):10:1–10:13, May 2008.

[17] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, January 1999.

[18] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 409–416, New York, NY, USA, 2001. ACM.

[19] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 399–405, New York, NY, USA, 2004. ACM.

[20] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 201–210, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[21] Yiying Tong, Santiago Lombeyda, Anil N. Hirani, and Mathieu Desbrun. Discrete multiscale vector field decomposition. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 445–452, New York, NY, USA, 2003. ACM.

[22] Xavier Tricoche. *Vector and Tensor Topology Simplification, Tracking, and Visualization*. PhD thesis, University of Kaiserslautern, 2012.

[23] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2008)*, 14(6):1396–1403, November - December 2008.