FLUID ANIMATION ON DEFORMING SURFACE MESHES

By

Xiaojun Wang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2017

ABSTRACT

FLUID ANIMATION ON DEFORMING SURFACE MESHES

By

Xiaojun Wang

We explore methods for visually plausible fluid simulation on deforming surfaces with inhomogeneous diffusion properties. While there are methods for fluid simulation on surfaces, not much research effort focused on the influence of the motion of underlying surface, in particular when it is not a rigid surface, such as knitted or woven textiles in motion. The complexity involved makes the simulation challenging to account for the non-inertial local frames typically used to describe the motion and the anisotropic effects in diffusion, absorption, adsorption. Thus, our primary goal is to enable fast and stable method for such scenarios.

First, in preparation of the material properties for the surface domain, we describe textiles with salient feature direction by bulk material property tensors in order to reduce the complexity, by employing 2D homogenization technique, which effectively turns microscale inhomogeneous properties into homogeneous properties in macroscale descriptions. We then use standard texture mapping techniques to map these tensors to triangles in the curved surface mesh, taking into account the alignment of each local tangent space with correct feature directions of the macroscale tensor. We show that this homogenization tool is intuitive, flexible and easily adjusted.

Second, for efficient description of the deforming surface, we offer a new geometry representation for the surface with solely angles instead of vertex coordinates, to reduce storage for the motion of underlying surface. Since our simulation tool relies heavily on long sequences of 3D curved triangular meshes, it is worthwhile exploring such efficient representations to make our tool practical by reducing the memory access during real-time simulations as well as reducing the file sizes. Inspired by angle-based representations for

tetrahedral meshes, we use spectral method to restore curved surface using both angles of the triangles and dihedral angles between adjacent triangles in the mesh. Moreover, in many surface deformation sequences, it is often sufficient to update the dihedral angles while keeping the triangle interior angles fixed.

Third, we propose a framework for simulating various effects of fluid flowing on deforming surfaces. We directly applied our simulator on curved surface meshes instead of in parameter domains, whereas many existing simulation methods require a parameterization on the surface. We further demonstrate that fictitious forces induced by the surface motion can be added to the surface-based simulation at a small additional cost. These fictitious forces can be decomposed into different components. Only the rectilinear and Coriolis components are relevant to our choice of local frames. Other effects, such as diffusion, adsorption, absorption, and evaporation are also incorporated for realistic stain simulation.

Finally, we explore the extraction of Lagrangian Coherent Structure (LCS), which is often referred to as the skeleton of fluid motion. The LCS structures are often described by ridges of the finite time Lyapunov exponent (FTLE) fields, which describe the extremal stretching of fluid parcels following the flow. We proposed a novel improvement to the ridge marching algorithm, which extract such ridges robustly for the typically noisy FTLE estimates even in well-defined fluid flows. Our results are potentially applicable to visualizing and controlling fluid trajectory patterns. In contrast to current methods for LCS calculation, which are only applicable to flat 2D or 3D domains and sensitive to noise, our ridge extraction is readily applicable to curved surfaces even when they are deforming.

The collection of these computational tools will facilitate generation of realistic and easy to adjust surface fluid animation with various physically plausible effects on surface.

To my dear family

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help from my collaborators, my advisor, my guidance committee, and the support from my family and my friends.

TABLE OF CONTENTS

LIST OF FIGURES viii

CHAPTER 1 INTRODUCTION 1
1.1 Homogenization 2
1.2 Angle-based Representation for Triangle Meshes 4
1.3 Fluid Simulation on Moving Surfaces 5
1.3.1 Ink Simulation over Planar Fibrous Medium 5
1.3.2 Fluid and Droplet Simulation on Curved Surfaces 6
1.3.3 Flows on Deforming Surfaces 7
1.3.4 Contributions 7
1.4 Surface Lagrangian Coherence Structure Extraction 9
1.5 Organization of Dissertation 10

CHAPTER 2 HOMOGENIZATION 11
2.1 Introduction 11
2.2 Homogenization for Bulky Diffusion Property 12
2.2.1 Input Inhomogeneous Tensor 13
2.2.2 Homogenization from Weaving Pattern 14
2.3 Results and Discussion 15

CHAPTER 3 ANGLE-BASED SURFACE REPRESENTATION 19
3.1 Introduction 19
3.2 Related Work 20
3.3 Angle-based Shape Construction 22
3.3.1 Linear System 24
3.3.2 Spectral Method 25
3.3.3 Gradient Descent in 12D 27
3.3.4 Implementation Details 28
3.4 Results 30
3.5 Conclusion and Future work 33

CHAPTER 4 FLUID SIMULATION 36
4.1 Introduction 36
4.2 Semi-Lagrangian Fluid Simulation 37
4.2.1 Eulerian vs Lagrangian representations 38
4.2.2 Fictitious Force 39
4.3 Simulation on 3D Surface 39
4.3.1 Advection 40
4.3.2 Momentum Advection 42
4.3.3 Diffusion 43

4.3.4 Diffusion for Anisotropic Material 44
4.4 Fictitious Forces 45
4.5 Stain-Surface Interaction 47
4.6 Results 49
4.7 Conclusion 51

CHAPTER 5 SURFACE LAGRANGIAN COHERENT STRUCTURE 52
5.1 Introduction 52
5.2 Mathematical Background 53
5.3 Related Work 57
5.3.1 FTLE Ridges 57
5.3.2 Other LCS Definitions 59
5.4 Overview 60
5.5 Implementation Details 61
5.5.1 FTLE Evaluation 61
5.5.2 Candidate Ridge Regions 62
5.5.3 Marching Triangle Ridge Extraction 63
5.5.4 Cleanup of Noisy Ridges 65
5.5.5 Spurious Valleys 65
5.6 Numerical Experimental Results 66
5.7 Conclusion and Future Work 67

CHAPTER 6 CONCLUSION 72

BIBLIOGRAPHY 73

LIST OF FIGURES

Figure 1.1 Fluid simulation on a rotating Bunny shape. The initial density field is a blob on the back of the bunny. (a) Without the Coriolis effect, the stain develops under the centrifugal force. (b) With the Coriolis effect, the stain turns into the more realistic spiral shape. 2

Figure 1.2 Simulation based on textile's bulk diffusion properties. 3

Figure 1.3 Flowchart of the simulation. 8

Figure 1.4 Stain on a waving flag. Visualized on the undeformed flag to avoid occlusion. 8

Figure 1.5 Stain on dropping tablecloth. 9

Figure 2.1 Schematic drawing of the homogenization process. 12

Figure 2.2 Illustration of 2D homogenization. 15

Figure 2.3 Comparison between effective tensor and average tensor on an isotropic layout. 16

Figure 2.4 Comparison between effective tensor and average tensor on an anisotropic layout. 17

Figure 3.1 Illustration of local construction through triangle angles and a dihedral angle. This pair of adjacent triangles contains 4 vertices v_1, ... v_4. The left figure shows that we can fix the location of $v_1 = (0,0,0)^\top$ and $v_2 = (1,0,0)^\top$, the vertices v_3 ($\bar{v}_4$) can be constructed in the XY-plane using angles α_{123} and α_{312} (angles α_{214} and α_{421} respectively). The right figure shows that the last vertex position v_4 can be adjusted to get the correct dihedral angle θ_{ij} between the two triangle normals. . . 22

Figure 3.2 Non-orthonormal frame 24

Figure 3.3 Top: Results for adding different levels of noises to dihedral angles. Middle: Results for adding different levels of noises to triangle interior angles. Bottom: Results for adding different levels of noises to both angles. From left to right: the maximum random noise magnitude is at the level of 0%, 10%, 20%, and 30%, respectively. The result in this example is resilient to such noises. Distortion only becomes obvious when interior angles reached 30% noise level. 31

Figure 3.4 Random 0.01% noise level 32

Figure 3.5 We show in this collection of reconstructed models a variety of shapes, including surfaces with high curvature regions, with high genus (22), or with irregular sampling. The first number is the percentage of max dihedral angle noise, and the second number is that of interior angle noise. 34

Figure 3.6 The results for deforming flags with 1% noises added to both dihedral and interior angles. The reconstruction produces large deviation from the original shape if larger noises are added, especially for interior angles. We postulate that the deviation from isometry as the original embedding curvature can have a large impact when the mesh has a boundary. The boundary vertex positions are determined only from their relation to the neighbors from certain directions, whereas in the closed mesh case the vertex positions are determined by neighbors from all directions. 35

Figure 3.7 The angle deviation decreases over iteration in our 12D gradient descent. However, the result after gradient descent remains quite similar to the original guess. The noise level on dihedral angle is at 20%. 35

Figure 4.1 Stain on a bunny rotating around a vertical axis. 37

Figure 4.2 The comparison shows that both translational (missing in a) and rotational (missing in a and b) inertia forces are necessary for the comoving and corotational frames. A solid ball is indicates the motion of the center of mass (without diffusion) in an inertial coordinate system. 45

Figure 4.3 The stain induced by various initial fluid velocities. 48

Figure 4.4 Simulation on a waving flag. 49

Figure 4.5 Stain on a dropping tablecloth. 50

Figure 4.6 Anisotropic diffusion without (a) and with (b,c) the influence of absorption, adsorption, and evaporation. 50

Figure 5.1 Spiral eddies containing both hyperbolic and elliptic Lagrangian Coherent Structures. Image by Paul Scully-Power/NASA 53

Figure 5.2 Attracting and repelling LCSs in space-time (left), and one time slice (right). Image by George Haller. 54

Figure 5.3 Hyperbolic LCS (attracting red curves and repelling blue curves) and elliptic LCS (green region boundaries) in a 2D turbulence simulation. Image by Mohammad Farazmand 55

Figure 5.4 Attracting (red) and repelling (blue) LCSs visualized through height field with FTLE as the height for a von Karman vortex street. Image by Jens Kasten. 58

Figure 5.5 From left to right: FTLE calculated from the double gyre velocity field, S field computed by second derivative-based ridge conditions, and S computed based on thresholding followed by smoothing. 62

Figure 5.6 Left: ridges (black curves) generated from the gradient of the FTLE field. Right: ridges generated from the gradient of the indicator function S. 65

Figure 5.7 Left: before cleanup based on the graph structure. Right: after the cleanup. 66

Figure 5.8 Air flow above an airfoil forms a vortex behind the LCS ridge line. 67

Figure 5.9 LCS for double gyre flow at various starting times. 68

Figure 5.10 LCS for quad gyre flow at various starting times. 69

Figure 5.11 LCS on the Bickley jet. 70

Figure 5.12 LCS on the torus octuple gyre. 71

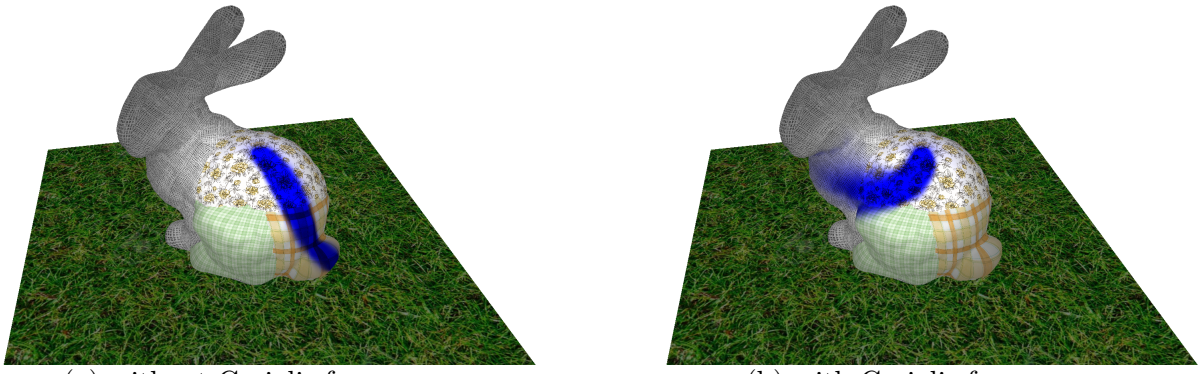
CHAPTER 1

INTRODUCTION

Fluid simulation has been a long standing problem in computation. In fact, progress towards a mathematical theory of the solution to Navier-Stokes equations, the governing equations of fluid motion is listed as one of the unsolved six among the seven Millennium Prize Problems, along side the "P=NP" problem [1]. Nevertheless, with the advent of graphics simulation and Graphics Processing Unit (GPU) technologies, people nowadays may enjoy high quality fluid animation even at interactive rates [2].

However, even if we focus only on these visually plausible results, not much research effort has been directed towards the simulation of fluid motion on curved deforming surfaces. Such motions are abundant in real world. For instance, coffee or milk spilled on one's sweater when walking along the aisle, painters painting on textiles, coloring of elastic products on assembly lines, or blood or mud stains on characters in action packed scenes in movies.

Accurate simulation of such scenarios can be even harder to resolve than the basic Navier-Stokes equations. Thus, we focus on creating physically plausible effects of such motions, with an emphasis on the overall behavior. To this end, we propose a collection of computational tools to simplify the problem, each focusing on a different aspect of it. The first tool is the calculation of bulk anisotropic diffusion tensors, extracted from the material property patterns (e.g., knitting pattern) based on the homogenization theory, with which the highly inhomogeneous surface material can be expressed effectively as one tensor for each patch with same patterns for the diffusion process. Next, we propose a novel angle-based representation of surface meshes to reduce the storage for each frame of the deforming surface, whose deformation drives the motion of the fluid flow on it. Last, we propose a framework that handles fluid simulation using local reference frames on vertices to extend typical 2D fluid simulation performed on regular grids to deforming



(a) without Coriolis force (b) with Coriolis force

Figure 1.1 Fluid simulation on a rotating Bunny shape. The initial density field is a blob on the back of the bunny. (a) Without the Coriolis effect, the stain develops under the centrifugal force. (b) With the Coriolis effect, the stain turns into the more realistic spiral shape.

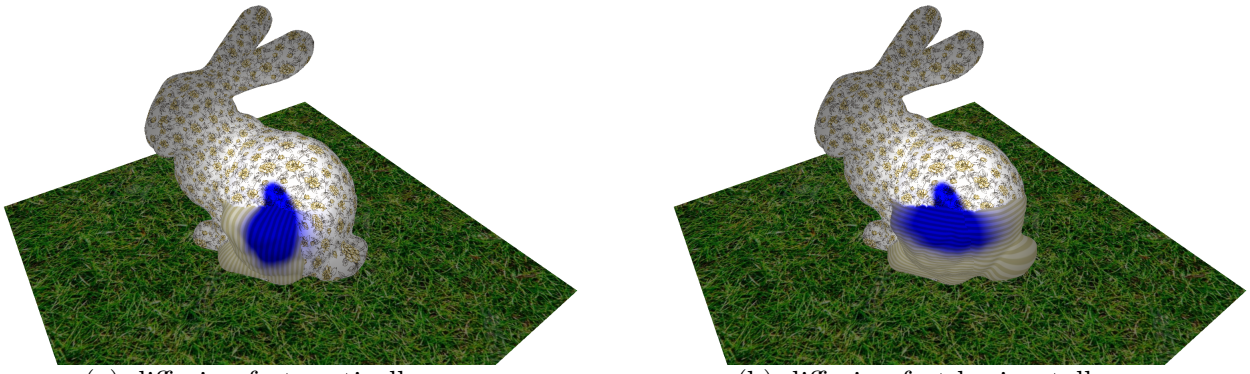
surfaces. Absorption, adsorption and evaporation effects are also accounted for, following their formulation in physics. The governing equations of motion are calculated directly on deforming curved surface meshes for efficiency and robustness. The influence of the surface motion on the shape of the stain developed in such motion is determined by using the inertial forces experienced in comoving and corotating frames attached to the deforming surface. Figure. 1.1 shows the effects of our treatment of inertial forces.

These tools make the targeted simulation tractable. Our numerical experiments demonstrate the plausibility of our results in practical applications, e.g., stains on fabrics. In future, we plan to use the concept of Lagrangian Coherent Structure to provide effective control over the bulk motion of fluid simulation on surfaces.

We now provide a brief introduction to each of these three computational tools in the following section.

1.1 Homogenization

In computer graphics, yarn-level modeling of the details of textiles has been used in physically based rendering and simulation [3]. While we also seek the effects of the knitting



(a) diffusing fast vertically (b) diffusing fast horizontally

Figure 1.2 Simulation based on textile's bulk diffusion properties.

pattern on the fluid motion, we focus on the efficiency of generating the bulk motion. See, *e.g.*, Figure. 1.2.

One popular method to achieve equivalent bulk motion with homogeneous material replacing highly inhomogeneous material is through homogenization theory[4, 5], which replaces the microscopic structures by an effective locally homogenous material with similar macroscopic properties. Homogenization has aroused attention of applied mathematicians when two-phase media, or disperse media, has been discovered to posses property that is stable and can be represented by simple matrices. Based on such an observation, we proposed a homogenization procedure for evaluating bulk diffusion tensors of textiles with arbitrary knitting patterns, as diffusion on knitting patterns can be approximated by spatially varying diffusion tensor fields which are periodic. Most relevant to the diffusion process in our fluid simulation is a homogenization technique of elliptic equations in divergence form proposed by Owhadi et al. [6]. Our technique can be applied to curved surface meshes, while existing methods are mostly restricted to 2D or 3D Euclidean space. While textiles and garments made of textiles are indeed piecewise-flat, materials such as cured leather on the back of a couch may need a homogenization procedure that can account for the curvature.

In animation, an elasticity homogenization [7] was introduced to approximate a deformable object made of arbitrary fine structures of various linear elastic materials with a dynamically similar coarse model. We envision that our technique can potentially be

extended to handle thin-shell simulation with inhomogeneous elasticity.

1.2 Angle-based Representation for Triangle Meshes

One major delay in the simulation of fluid on deforming mesh is caused by loading the updated locations of the vertices. Methods for compressing temporal sequences of meshes exist [8]. However, in our case, the underlying surface often undergoes deformations that are close to isometries, i.e., mapping under which the edge lengths are almost fixed. In the discrete setting, it means that the dihedral angles of adjacent faces are the only updated values. While the number of edges is roughly three times the number of vertices, which is the same as the number of coordinates for the entire mesh. However, as noted in Angle Analyzer, the distributions of angles, including triangle interior angles and dihedral angles typically leads to more efficient entropy coding, leading to superior compression ratio both in terms of the connectivity (how vertices are connected to form triangles) and the geometry (vertex locations) for triangle, quadrilateral, or hybrid meshes [9]. For isometric mesh sequences, the connectivity is fixed, and the majority of the angles (triangle interior angles) is also fixed, which indicates the dihedral angles provide a highly efficient representation of the vertex coordinates.

However inaccuracy in aggressive lossy compression can lead to noise in the mesh position, which may have large impacts the acceleration, since it is the second order temporal derivative of positions. As we depend on the acceleration of each vertex to calculate the inertial force acting on the local fluid motion, we seek a stable reconstruction of mesh positions based on the angles. Inspired by a recent paper on dihedral angle-based tetrahedral mesh representation [10], we propose a spectral method for the reconstruction of the mesh, avoiding accumulated errors. This may seem like a simplification from 3D to 2D, but the curved 2D turns out to be more complicated than 3D. In 3D, a vertex of a tetrahedron can be expressed as a linear combination of vertices of an adjacent tetrahedron, with coefficients

computed based solely on angles; on a surface, the 3D coordinates of a vertex cannot be expressed linearly by 3 other vertices. Even if we include more vertices near a given vertex to construct a linear map, it can be highly degenerate, since a fine smooth mesh would have all nearby vertices nearly coplanar.

We address the issue by introducing one reference frame per triangle as intermediate variables to turn the problem into a linear system, based on which we build an eigenvalue problem to efficiently reconstruct the system without error accumulation. We show that our method is robust even with large perturbation in the input angle data. An iterative method is introduced to further reduce the error in reconstructed angles.

We propose a novel algorithm for determining the vertex coordinates of triangle meshes from angles with given triangle connectivity. Our algorithm is capable of handling noise interior and dihedral angles by employing spectral analysis. More precisely, we formulate the shape reconstruction as an eigenvalue problem. The reconstructed shape is determined up to an affine transformation. We fix the affine transformation through a simple voting strategy. Finally, we offer an optional gradient descent for further reducing the deviation of the reconstructed angle from the input. We show the efficiency and effectiveness of our method in numerical tests.

1.3 Fluid Simulation on Moving Surfaces

Fluid simulation on moving surfaces, such as stain formation, relates to many research efforts in computer graphics. We briefly cover the most relevant works here to motivate our technique and its various components.

1.3.1 Ink Simulation over Planar Fibrous Medium

Over the years, researchers have developed numerous efficient techniques for 3D fluid simulation in graphics [11]. Recently, reduced models for fluid simulation on surfaces have

received increased attention due to their efficiency compared to a full-blown simulation of the 3D Navier-Stokes equations. In particular, work has been dedicated to simulate watercolor painting and Chinese ink painting, both of which involving diffusion of pigments in paper. Curtis et al. [12] simulated watercolor effects such as dry-brushing, intentional backruns, and flow patterns. Kunii et al. [13, 14] modeled the interactions between ink and paper with partial differential equations, which are essentially Fick's law of diffusion. Chu et al. [15] simulated ink dispersion in absorbent paper by solving the lattice Boltzmann equation. Van Laerhoven et al. [16, 17] presented a physically based technique for creating images with watery paint in real time. Simulating Chinese calligraphy and Chinese landscape painting based on ink diffusion on paper has also been proposed by various researchers [18, 19, 20, 21, 22]. Morimoto et al. [23] presented a method for visualization of cloth dyeing. However, their method focused on simulating the dyeing process, and is not suited for the stain formation on textile surfaces. Liu et al. [24] proposed a simulation of stains on flat anisotropic media, including a simplified treatment of diffusion, infiltration, and evaporation. While all these methods can generate realistic simulation of pigment distributions in flat 2D domains, they often use no or very simplified fluid advection, and cannot be directly extended to model stain evolution on moving and deforming cloth.

1.3.2 Fluid and Droplet Simulation on Curved Surfaces

Fluid motion on non-flat surfaces was simulated by Stam [25] through the use of 2D parameterization, which can induce noticeable artifacts. Intrinsic (i.e., parameterization-free) methods were later proposed to simulate surface flows directly on triangle meshes[26, 27, 28], and a model-reduced approach based on eigenvector bases has been recently offered as well [29]. Auer et al. [30] leveraged the Closet Point Method (CPM) to numerically approximate the wave equation and the incompressible Navier-Stokes equations on arbitrary surfaces in real-time on GPU. Wang et al. [31] presented a technique to simulate shallow water equations on the surfaces, targeting surface tension driven effects, after introducing a simulation of water

drops on hydrophobic or hydrophilic leaves [32]. Zhang et al. [33] offered a fast Lagrangian method for such droplet simulations using triangle meshes to represent the drops, while Jung and Behr [34] introduced GPU-based real-time simulation of droplet flows. Finally, Djado et al. [35] simulated the motion of water drops on a surface, realistically capturing condensation on a surface or human sweating in real time.

1.3.3 Flows on Deforming Surfaces

Angst et al. [36] proposed a method for wave simulations on deforming meshes. Neill [37] developed a framework for simulation of fluid flow on interacting deformable surfaces, where only the average acceleration of the underlying surface is considered for the calculation of inertia force. Hegeman et al. [38] presented an approach to solve Navier-Stokes equations on surfaces based on the unique properties of conformal cube maps. Recently, Jeong and Kim [39] introduced a combustion model of heat transfer and fuel consumption for the propagation of a fire front on a point cloud surface. They proposed angular Voronoi weights for a discrete Laplace-Beltrami operator that shows better isotropic diffusion on the inhomogeneous distribution of point clouds than the cotangent or moving least-squares schemes. All the above methods treat the curved surface composed of either homogeneous or isotropic media, without accounting for the full inertial effects of the possibly deformation of the surface, thus not directly applicable for stain formation on moving surfaces with inhomogeneous and anisotropic materials.

1.3.4 Contributions

Figure 1.3 shows an overview of our framework. We preprocess the inhomogeneous (and possibly anisotropic) material through the homogenization process to get effective diffusion tensors. Then we specify the material type directly on curved surfaces on a per triangle basis. The fluid motion is governed by a number of processes that are treated in a split step

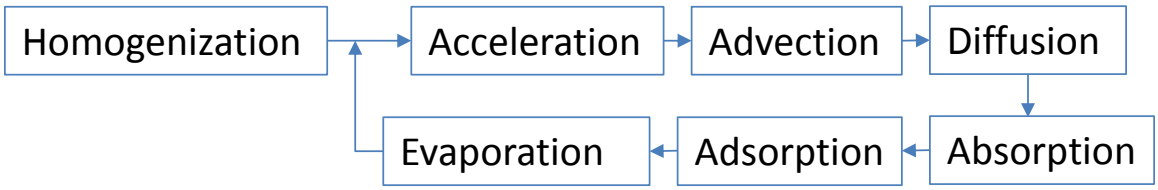
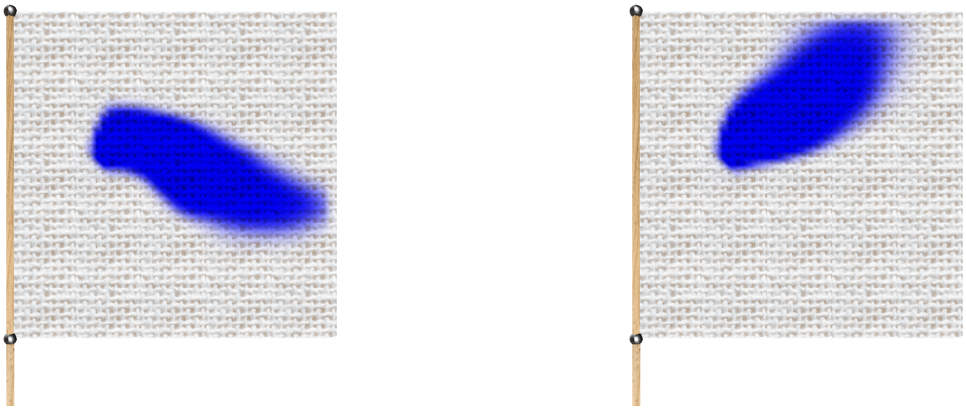


Figure 1.3 Flowchart of the simulation.



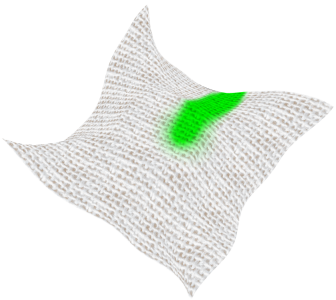
(a) with Coriolis force (b) without Coriolis force

Figure 1.4 Stain on a waving flag. Visualized on the undeformed flag to avoid occlusion.

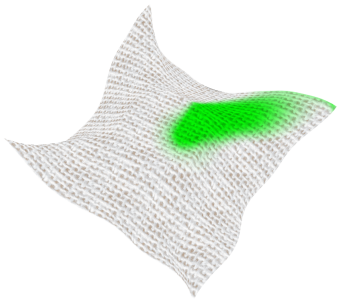
fashion as in semi-Lagrangian fluid simulation [40], including acceleration by fictitious forces and gravity, advection, diffusion, infiltration, adsorption, and evaporation. We use a mixed explicit-implicit time integration, in which only the parabolic diffusion process involves a global implicit integration. The main contributions of our method include:

- A modified Laplacian to solve the diffusion process directly on curved surfaces.
- An efficient fictitious force evaluation induced by the one-way coupling from surface to fluid, including the Coriolis effects, based on local comoving and corotational frames.
- A complete framework incorporating realistic physical staining processes, including absorption, adsorption, and evaporation.

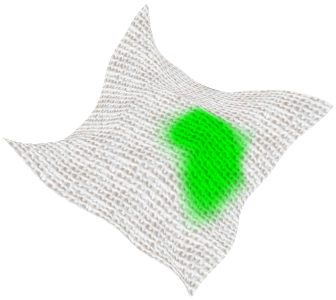
Figures. 1.4 and 1.5 show some of the effects produced by our framework.



(a) without inertia forces



(b) with only rectilinear force



(c) with both rectilinear force and Coriolis force

Figure 1.5 Stain on dropping tablecloth.

1.4 Surface Lagrangian Coherence Structure Extraction

In computer animation, artistic guidance is often required to control the motion instead of directly using purely physical simulation. While there are methods for artistic guidance for smokes or even water-air interface, they do not apply to surface fluid simulations directly. Toward that end, we propose to extract the Lagrangian Coherent Structure (LCS) that serves as a coarse description of the fluid motion. One way to define LCS is through ridges of the finite time Lyapunov exponent (FTLE) scalar field, which describes how stretched an infinitesimal fluid parcel will become by following the flow. Such ridges are like watersheds that divide the domain into regions that move with the fluid, but among which little exchange of fluid particles happens. LCS is, thus, often referred as skeleton of fluid dynamics. It offers a coarse representation to observe fluid flows. As such, LCS has been used to allow high resolution fluid motion to follow the overall motion of a low resolution simulation. We present a robust method for calculating LCS on curved surfaces. Concrete applications in

artistic guidance is beyond the scope of this dissertation. However, with the clean output from our method, it is potentially easy to incorporate to existing LCS-preserving simulators.

1.5 Organization of Dissertation

The rest of document is organized as follows. We provide details of homogenization of inhomogeneous materials, in particular, knitted patterns, in Chapter 2. We then discuss the novel reconstruction of surfaces from angle data in Chapter 3. We describe the fluid simulation framework with fictitious acceleration on deforming surface in Chapter 4. We describe the robust Lagrangian Coherence Structure extraction method for triangle meshes in Chapter 5, before we conclude in Chapter 6 with a summary of contributions and possible future work.

CHAPTER 2

HOMOGENIZATION

2.1 Introduction

Homogenization is a process that converts a region with repeated patterns highly diverse material components into a region composed of a single type of material which exhibits similar bulk physical behavior[4, 5]. The original region contains heterogeneous materials, while the converted region is homogeneous with a single but often anisotropic material. Often the homogenization plays an important role in computation, because it allows the handling of material characteristics that are rapidly varying spatially as if it is its simplified homogeneous counterpart. For instance, it is used in [41] as a method to analyze stress variations in elastic fiber composites. Hollister and Kikuchi [42] show that homogenization method outperforms standard mechanics of materials approaches even for locally periodic material.

The procedure of homogenization method can be seen as solving for the homogeneous material that best approximate the behavior of the original material, i.e., for some predefined measurements, simulation in the homogeneous material and that in the original material should exhibit minimum residual. For instance, Lukkassen et al. [41] outlined a typical four-step procedure, assuming microscopic heterogeneous material distribution is periodic:

1. Within a cell (representative volume element) containing a few periods in each direction, solve the cell problem of strain tensor under periodic boundary conditions;

2. Obtain homogenization coefficients based on the solution from step 1;

3. Incorporate homogenization coefficients into macroscale equilibrium equations;

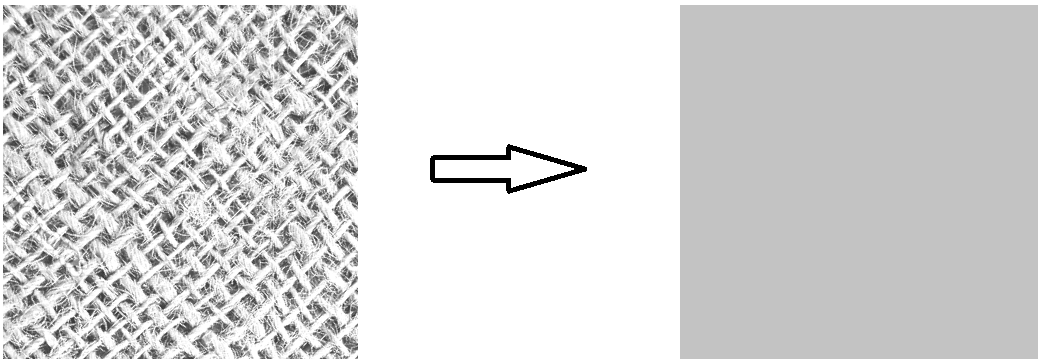


Figure 2.1 Schematic drawing of the homogenization process.

4. Compute local fluctuation of strain field based on the average strain field obtained in step 3.

Elasticity homogenization introduced to computer animation in [7] can be seen as using the first three steps in the structure to get the bulk motion. In our case, we focus on obtaining the bulk diffusion tensors based on knitting patterns or other material patterns that are also periodic. Since diffusion equation is a typical elliptic equation, we use the a simplified version of homogenization described in [6]. We essentially ignore the local fluctuation as the microscale phenomena typically would not be salient. However, for applications such as simulation of Chinese painting, the microscale fluctuation in the fourth step can and should be applied.

2.2 Homogenization for Bulky Diffusion Property

We represent textile (or other inhomogeneous material) by patterns of microscale cells for efficient simulation. As described in [23], weaving pattern in a textile model can be approximated by three types of cells, namely warp cell, weft cell, and gap cell. Figure 2.2b shows an example distribution of different cells for a small piece of textile (Figure 2.2a). A prescribed weaving pattern determines whether each warp (weft) cell is orientated up or down. By changing the arrangement of warp cells and weft cells, we can approximate the yarn/thread-level structure of textiles with different weaving styles such as plain weave,

satin weave and twill weave. Gap cells denote the gaps among warps and wefts. The diffusion of stain can be different between different pair of cells. For instance, with hydrophobic fibers, stain may diffuse quickly between gap cells but slowly between warp cell and weft cell.

We can thus describe the diffusion procedure by the following diffusion equation,

$$\dot{f}(u,v,t) = \nabla \cdot [S(u,v)(\nabla f(u,v,t))], \tag{2.1}$$

where f is a scalar or vector field being diffused, $\dot{f}$ is its time derivative, and $S(u,v)$ is the highly oscillatory diffusion tensor field, which is piecewise constant with different types of cells.

We seek a bulk diffusion tensor to effectively average the diffusion coefficients of different cells on a piece of textile. Figure 2.2c gives the basic layout of an input diffusion tensor field. A 2×2-matrix is used to denote the diffusion property of each cell. Each element in the matrix can be obtained according to the layout of the cells and the diffusion characteristics of the textile such as the tortuosity and porosity. In contrast to modelling the diffusion using resolutions matching the numerous threads in textile, the homogenization technique is efficient, but can still simulate the overall anisotropic diffusion of stains in textile. More precisely, we seek an effective constant tensor S^{eff} for each type of textile, such that the solution $\tilde{f}$ of the following equation is an approximation of f,

$$\dot{\tilde{f}}(u,v,t) = \nabla \cdot [S^{\text{eff}}(\nabla \tilde{f}(u,v,t))]. \tag{2.2}$$

2.2.1 Input Inhomogeneous Tensor

One method to obtain the input tensor field is to use the Weisz-Zollinger model (see, e.g. [43]), which defines five types of connectivity based on cell positions and porosity: a) fibers in different layers, b) perpendicular fibers in the same layer, c) fiber and gap, d) gap and gap, and e) parallel fibers in the same layer. The permeability (diffusion) coefficient between two textile cells is computed as $D = K_{\text{diff}}T$, where K_{diff} is a coefficient determined by

the properties of the both stain (dynamical viscosity) and textile (porosity), and T is the tortuosity, which can be different for different types of cell connectivity. We allow the user to choose specifying a pattern of diffusion tensors, or specifying cell pattern and D. In the latter case, we average the boundary permeability to construct the diffusion tensor,

$$S = 1/2\ [D_{\text{left}} + D_{\text{right}}, 0; 0, D_{\text{top}} + D_{\text{bottom}}].$$

2.2.2 Homogenization from Weaving Pattern

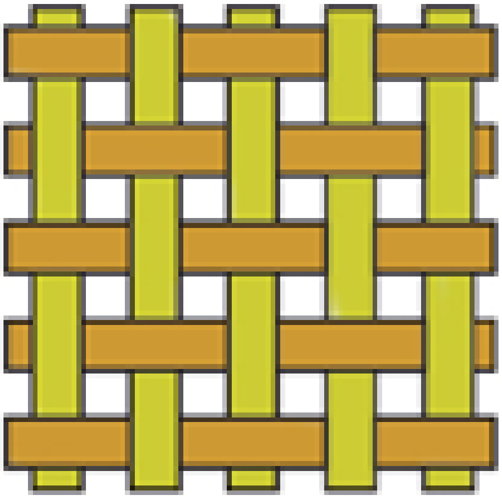
The original highly inhomogeneous material property can only be resolved with an extremely high mesh resolution. By using the homogenization theory [4], we can simulate the long-time behavior effectively and avoid the high frequency tensor field. We assume that the material pattern is periodic, and the user input is one representative tile (or copies of several tiles). We construct a mesh of a topological torus with the specified tensor field, i.e., we create a mesh for a rectangular region, and then identify the vertices and edges on the left and right boundaries at the same height and those on the top and bottom boundaries at the same distance from the left boundary. An effective constant tensor, one capable of producing similar diffusion in a region containing a large number of tiles, can be evaluated as

$$S_{ij}^{\text{eff}} = 1/A \int_{\Omega} [S(x,y)(e_i + \nabla f_i(x,y))] \cdot e_j dxdy, \tag{2.3}$$

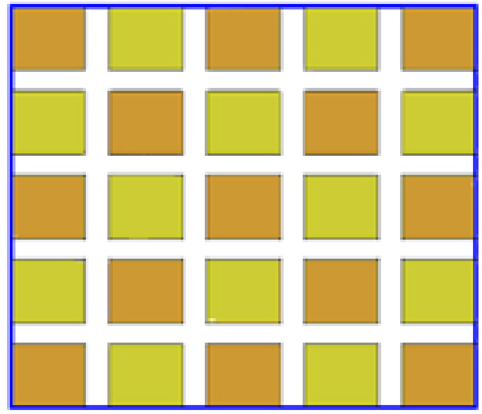
where A is the total area of the domain Ω, S^{eff} is the effective bulk diffusion tensor, $S(x,y)$ is the diffusion tensor field, e_i is the unit vector in i-th direction ($i = 0, 1$), and f_i is the solution to the following Poisson equation

$$\nabla \cdot [S(x,y)(\nabla f_i(x,y) + e_i)] = 0. \tag{2.4}$$

Roughly speaking, $(f_1 + x, f_2 + y)$ is a new (harmonic) parameterization of the domain, in which the diffusion tensor is properly stretched, and can thus be directly averaged. In 1D, this is the same as taking a simple harmonic mean, but in 2D, it has to be evaluated



(a) Weaving pattern. Weft: dark. Warp: light.



(b) Cell-based approximation.

$$\left[\begin{array}{ccc} \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \\ \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \\ \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \end{array}\right]$$

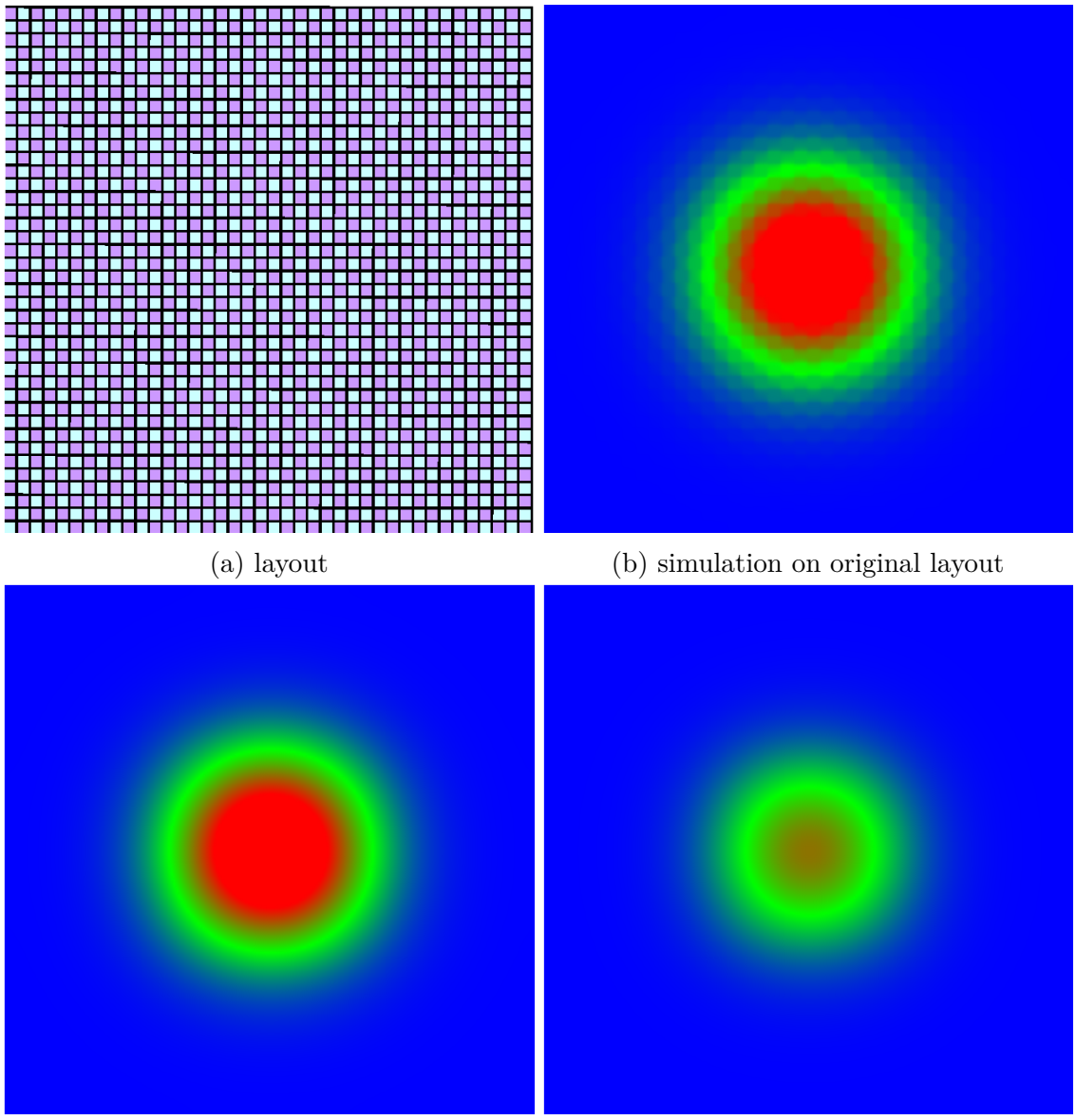
(c) Layout of diffusion tensor for one block of the period domain.

Figure 2.2 Illustration of 2D homogenization.

numerically for various patterns in general. The discretization of Equation 2.4 is the same as that of the diffusion process in Sec. 4.3.3.

2.3 Results and Discussion

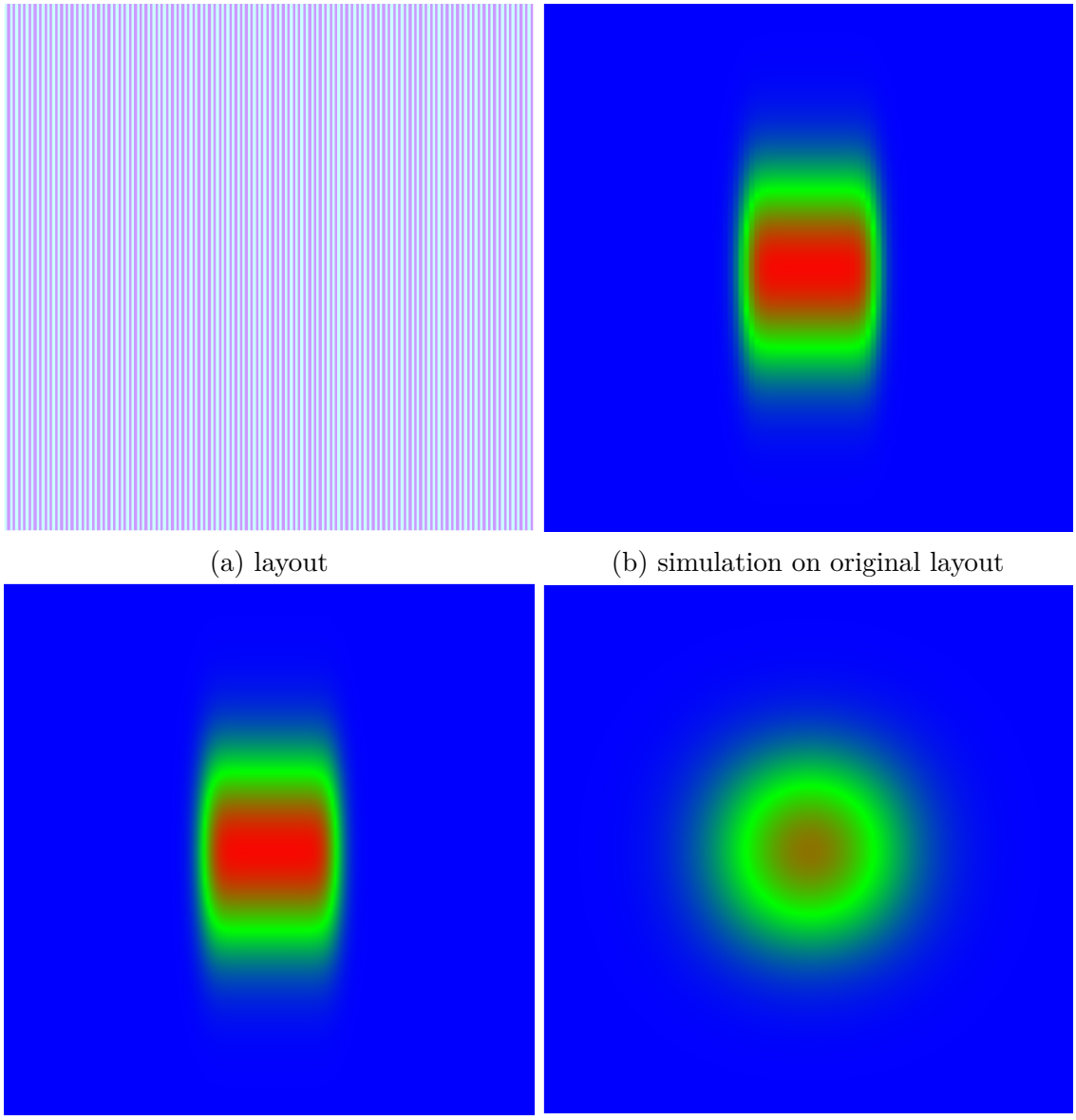
We tested the homogenization of the diffusion/permeability tensor on an inhomogeneous rectangle. As shown in Figure. 2.3, the homogenized diffusion tensor produces diffusion results that closely resemble the original tensor field, while the direct averaging of the tensor fields in the original domain results in a completely different stain. We used the knitting pattern to compute the tensor field in the top row with D set as 1 for gap to gap, 100 weft to weft, 500 warp to warp, 10 gap to weft, 50 gap to warp, and 200 for weft to



(a) layout (b) simulation on original layout

(c) simulation with effective tensor (d) simulation with average tensor

Figure 2.3 Comparison between effective tensor and average tensor on an isotropic layout.



(a) layout (b) simulation on original layout

(c) simulation with effective tensor (d) simulation with average tensor

Figure 2.4 Comparison between effective tensor and average tensor on an anisotropic layout.

warp. The tensor field in Figure. 2.4 has alternating columns of isotropic material, one with diffusion coefficient 1 and the other 100, the effective bulk tensor is anisotropic unlike the average tensor.

As expected, the results produced by following the homogenization theory lead to the correct behavior for the diffusion procedure. We expect our use of homogenization to be applicable to the elastic properties of the textile, in which case, the cloth simulation of the underlying surface can also become efficient without changing the overall motion.

CHAPTER 3

ANGLE-BASED SURFACE REPRESENTATION

3.1 Introduction

Triangle angle meshes contain two sets of data, one for the geometry, i.e., vertex locations, and the other for the connectivity, the triplets of the vertex indices denoting triangle faces. Given fixed connectivity, the global coordinates of vertex locations provide a unique and non-redundant description of the shape. However, such a representation is also coordinate system dependent, which can be vastly different with even a simple rotation and translation. It also does not directly provide the local description of surface shape, such as the principal curvatures. Proper choice on the use of coordinate system independent descriptions can often facilitate the specific applications, such as parametrization, deformation, animation, and compression[10].

Although convenient for the target application, these angle, metric, or curvature based representations are often redundant as shape descriptions, and it is necessary to establish the mapping from them to a closest vertex coordinates for embedding in 3D Euclidean space. We propose a fast spectral surface reconstruction from triangle angles and dihedral angles, inspired by the dihedral angle-based maps of tetrahedral meshes[10].

While seemingly simpler than the 3D case, the curved 2D case is actually more challenging. In the tet mesh case, each vertex can be expressed as a linear combination of vertices of the tet incident to the opposite face but not incident to the vertex itself. This linear combination forms the basis of the spectral method that assembles all these mappings. For the triangle mesh case, such a linear combination does not exist, so we introduce local frames as intermediate variables. In addition, the final map still contains a global affine transformation, which we resolve using a simple voting strategy.

Contributions Our main contributions include:

- a spectral framework that extract the 12 dimensional null space satisfying the linear relations between each vertex and its 2-ring triangles;
- an affine transformation correction by voting on each pair of triangles;
- and an optional gradient descent method that further reduces reconstruction error within the 12D space.

Our angle-based reconstruction can be potentially attractive for near isometry deformation, shape retrieval and analysis, and compression.

3.2 Related Work

Inferring local shape descriptors from global vertex coordinates can be performed in a neighborhood as in the definitions of their continuous counterparts in most cases, for instance, the computation of curvature as in [44]. See [45] for a survey on the link to the continuous differential theory on polygonal meshes.

The other way around is typically nontrivial. In fact, the fundamental theory in classical differential geometry theory of surfaces deals with the determining the shape of a surface from the first and second fundamental forms[46]. Mimicking this theory, a number of geometry processing methods establish this mapping, e.g., [47, 48, 49]. In particular, [49] is using a linear construction very similar to ours by introducing the frames associated with triangles as intermediate variables. However, they rely on edge lengths to construct the first fundamental form, while we use only triangle angles and allow the scaling to be left flexible. Functional characterization that are independent of the triangulation also exists[50], when the fine details are not required.

In some applications such as shape classification, only intrinsic lengths, angles, and/or areas are used as local shape descriptors, which do not determine a unique embedding in

3D. In such cases, ideas similar to multidimensional scaling[51] can be used to reconstruct a "most stretch" shape, as in[52]. Our method focuses on the cases when the extrinsic dihedral angles are available, which is desirable for constructing unique shapes within the same isometry class.

Triangle angles and dihedral angles in theory determines the shape up to a global scaling, thus they can be used to encode the geometry part of the surface. In fact, the angle analyzer[9] does use these angles to sequentially assemble the entire mesh, it also uses the geometry as a cue to encode the connectivity of triangle, quadrilateral, or hybrid meshes. Since angle values have a distribution that is easier to encode than the global coordinates, this encoding can be used in compression despite the much larger number of angles than the vertex coordinates (dihedral angles alone would need the degrees of freedom equaling three times the vertex count). 3D mesh compression has a vast literature. See recent survey in, e.g., [53]. In most systems, geometry of a mesh is represented by vertex locations. Compared to vertex-based methods like Deering [54], Lee et al. [9] with a single-rate compression algorithm for triangle-quad hybrid mesh was able to accomplished an average of 40% better result. Methods for compressing temporal sequences of meshes also exist [8]. We speculate that angle based method can further lower the compression rate by encoding only the dihedral angles for near-isometry cases like cloth and thin-shell simulation.

Inaccuracy in aggressive lossy compression can lead to noise in the mesh position, which makes the sequential assembly impossible to avoid drifting in the vertex locations for those traversed later in the sequence and produces visible artifacts. We extend the spectral method proposed by Paillé et al. [10], which proposes a map from dihedral angles in a tet mesh to vertex locations. Similar spectral methods have also been used in parameterization[55], deformation[56], mesh processing[57], vector field design[58], etc. We differ from these methods in that we use the space spanned by multiple eigenvectors associated with the lowest eigenvalues and reduce the angle deviation by a gradient descent in this low dimensional space.

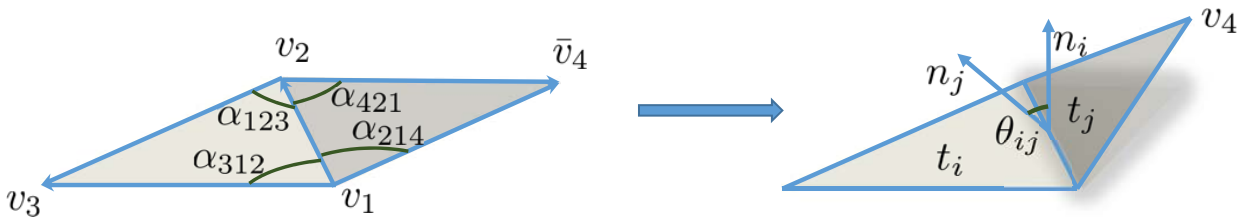


Figure 3.1 Illustration of local construction through triangle angles and a dihedral angle. This pair of adjacent triangles contains 4 vertices v_1, ... v_4. The left figure shows that we can fix the location of $v_1 = (0,0,0)^\top$ and $v_2 = (1,0,0)^\top$, the vertices v_3 ($\bar{v}_4$) can be constructed in the XY-plane using angles α_{123} and α_{312} (angles α_{214} and α_{421} respectively). The right figure shows that the last vertex position v_4 can be adjusted to get the correct dihedral angle θ_{ij} between the two triangle normals.

3.3 Angle-based Shape Construction

We assume the connectivity of a surface triangle mesh is given as $\mathcal{T}$ containing triples of vertex indices (i,j,k) with its cyclic order indicating counterclockwise orientation. The mesh is assumed to be an orientable 2-manifold (non-degenerate surface), i.e., a consistent unit normal direction can be assigned to all the triangles, typically the outward pointing normal for the closed boundary surface of a non-degenerate 3D object.

The orientability implies that we can assume that each edge (i,j) shared by two triangles appears as (i,j) in one triangle, and as (j,i) in the other. Given the interior angle α_{ijk} of triangle (i,j,k) at j, and dihedral angle θ_{ij} denoting the angle between the normals of the triangles (i,j,k) and (j,i,l) for each nonboundary edge, we compute $\mathbf{x}_i$, the position of vertex i, such that the mesh with these vertex positions will reproduce the given angles. Note that we only need two angles for each triangle, with the third angle determined by

$$\alpha_{jki} = \pi - \alpha_{kij} - \alpha_{ijk}. \tag{3.1}$$

As angles are rotation, translation and uniform scaling invariant, the solution can only be determined up to a rotation R, a translation $\mathbf{t}$ and a scaling factor s. One straightforward way of constructing the mesh would then be to choose $\mathbf{x}_i$ for an arbitrary vertex i as the origin $(0,0,0)^\top$, and the other end of one edge (i,j) incident to vertex i as $\mathbf{x}_j = (1,0,0)^\top$,

and one triangle (i, j, k) incident to that edge to have the counterclockwise unit normal aligned to Z-axis, i.e., that triangle lies in XY-plane. Given the interior angles of that triangle, we can uniquely determine position of the other vertex $\mathbf{x}_k$ in the oriented triangle. See Figure. 3.1.

Following a breadth-first search of the dual graph, formed by the dual graph of the mesh, i.e., a graph with triangles as nodes and triangle adjacency (defined as triangles sharing two vertices) as edges, we can traverse all the triangles. Each time we go from a visited triangle to another, if the opposite vertex k across edge (i, j) does not have a known position, we can first determine the plane containing the unvisited triangle (i, j, k) through the dihedral angle θ_{ij} and the normal of the visited triangle also incident to (i, j), and then compute the vertex location $\mathbf{x}_k$ based on the interior angles. When the traversal is complete, all the vertex positions are determined.

However, the straightforward sequential approach would only work if the angles are not noisy or inaccurate (e.g., due to quantization in compression). Otherwise, the deviation in the vertex position would accumulate as we move away from the starting vertex. We show in Sec. 4.6 that the resulting mesh of the straightforward reconstruction based on angle data with very little noise already leads to huge distortion for the Stanford Bunny model. The distortion is minimal near the starting triangle, but deteriorates quickly as the traversal proceeds.

Angle Analyzer [9] essentially follows the sequential approach. Angle Analyzer has shown that the angle-based representation can benefit the encoding of connectivity, through a front advancing strategy prioritizing the traversal order based on eliminating sharp angles of the active front, and benefit the encoding of geometry as the interior angles have distribution concentrated around $\pi/3$, and dihedral angle concentrated around 0. We, instead, focus on the geometric part given noisy or incompatible input angles. We intend to make our method capable of pushing the average bit per vertex in the representation to be lower than Angle Analyzer, which can benefit applications such as encoding deforming meshes with

fixed connectivity.

Inspired by a recent dihedral angle-based tetrahedral mesh representation [10], we propose to solve for the vertex positions simultaneously through a global approach. First, we build a system of linear equations that a mesh with the prescribed angles have to satisfy. Second, we solve for eigenvectors of the least-squares quadratic energy of the linear system. Finally, we use the eigenvectors to assemble the vertex positions, which produce minimal distortion of the angles. This way, we can avoid the accumulation of error in regions far from the beginning vertex in sequential approach. Our approach also removes the bias in regions far from constraints in least-squares with constraints on the initial triangle(s).

3.3.1 Linear System

We build a linear system based on the local reconstruction of triangle pairs. However, this is not as straightforward as in the tetrahedral case, because vertex positions of a triangle do not span the 3D space, which means that given the positions of one triangle, the non-shared vertex in the other triangle cannot be expressed as a linear combination of the vertices in the given triangle.

We introduce a non-orthonormal frame for each triangle to address the problem. See inset figure. Our choice of frame is similar to that of [59], where the first axis is aligned to one edge of the triangle, and the second axis is aligned to the reverse direction of the previous edge, and the last axis is the normal of the triangle with the norm being the square root of the area to keep the construction scaling-invariant. For a triangle $t = (i, j, k)$, the associated frame is the following 3×3 matrix,

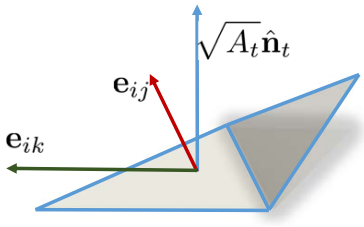


Figure 3.2 Non-orthonormal frame

$$F_t = \left(\mathbf{e}_{ij}, \mathbf{e}_{ik}, \sqrt{A_t}\mathbf{n}_t\right),$$

where $\mathbf{e}_{ij}$ is the vector along edge (i, j), A_t is the area of triangle t, and $\mathbf{n}_t$ is its unit normal.

We can then formulate the linear map in two parts. The first part is the relation between the frames of adjacent triangles t_i and t_j. They must satisfy the affine transformation R_{ji} uniquely determined by the interior angles of these two triangles and the dihedral angle between them,

$$F_{t_i} - F_{t_j} R_{ji} = \mathbf{0}. \tag{3.2}$$

This matrix can be constructed by first creating a local reconstruction for the 4 vertices of t_i and t_j based on angles as in Figure 3.1, then evaluating $\bar{F}_{t_i}$ and $\bar{F}_{t_j}$ in the local reconstruction, and finally $R_{ji} = \bar{F}_{t_i} \bar{F}_{t_j}^{-1}$.

The second part is the relation between the vertices in a triangle (i, j, k) and its frame F_{ijk},

$$\begin{bmatrix} \mathbf{x}_j - \mathbf{x}_i & \mathbf{x}_k - x_j & \mathbf{x}_i - \mathbf{x}_k \end{bmatrix} - F_{ijk} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{0}. \tag{3.3}$$

Note that with accurate angle data, the system of linear equations of the above two types is underconstrained, since it can be satisfied not just by the original mesh, but also by an affine transformation of the original mesh. In other words, the linear system has a kernel with 12 degrees of freedom, 9 for linear transformation, and 3 for translation.

As mentioned above, if we have boundary constraints, the system is typically overconstrained and can produce a reasonable result in the least-squares sense. However, when noise is present, the regions far from the constraints may still suffer from numerical errors and drift far from the original shape. Thus, we propose to use a spectral method similar to [10].

3.3.2 Spectral Method

We first assemble the linear system as $\mathrm{L}\tilde{\mathbf{x}} = \mathbf{0}$, where $\tilde{\mathbf{x}}$ is a column vector of dimension $3|\mathcal{V}|+9|\mathcal{T}|$, containing the coordinates for all $|\mathcal{V}|$ vertices, and frames for all $|\mathcal{T}|$ triangles, and L contains the coefficients for the above two types of equations. The original mesh

would thus minimize the least-squares error

$$\tilde{\mathbf{x}}^\top \mathrm{A} \tilde{\mathbf{x}},$$

where $\mathrm{A} = \mathrm{L}^\top \mathrm{L}$.

Since the kernel is 12 dimensional, we cannot use only the eigenvector associated with the lowest eigenvalue as the solution for noisy input angle data. Instead, we build a 12 dimensional space spanned by the eigenvectors associated to the lowest 12 eigenvalues $\mathrm{Span}\{\mathbf{e}_i\}_{i=1\ldots12}$.

Even in the ideal case, the result is in the invariant space of A spanned by the 12 eigenvectors of eigenvalue 0. So we need a method to find the right linear combination such that the position part of $\tilde{\mathbf{x}}$ actually corresponds to a mesh with the prescribed angles.

Without loss of generality, we start with a random initialization (or the mean of the 12 eigenvectors) $\bar{\mathbf{x}} = \sum_{i=1}^{12} \lambda_i \mathbf{e}_i$. While it is possible that the vertex positions in this initialization is degenerate (e.g., inside a plane, along a line), the possibility is low compared to using a single eigenvector $\mathbf{e}_1$.

However, the linear system does not prevent the vertex positions and frames from undergoing an arbitrary affine transformation. Thus, we need to estimate a translation and a linear transformation T_{ij} that can restore one of the adjacent triangle pair t_i and t_j to their local reconstruction with the correct angles. For noise-free angle data that originate from a real mesh, we can simply apply T_{ij} to the whole mesh to get all angles restored.

However, for noisy angle data, T_{ij} may be different for different pairs of adjacent triangles. It is highly nonlinear to solve for the best global linear transformation T that minimizes the sum of angle deviations. In practice, we found that voting for a best linear transformation is sufficient to produce a reasonable guess, which can be further improved by gradient descents if necessary.

However, there is one necessary preprocessing before each pair of triangles can cast a vote. This is because T_{ij} is computed as the map from the pair of triangles in $\bar{\mathbf{x}}$ to a local

reconstruction based on angle, which can be rotated and rescaled. So the direct average of T_{ij} for all triangle pairs is likely to be rather random since each vote favors a different rotation and scaling. This issue can be resolved if we remove the scaling and the rotation by first perform a polar decomposition, $T_{ij} = R_{ij}S_{ij}$, where R_{ij} is a rotation and S_{ij} is a symmetric matrix. We then normalize S_{ij} by its Frobenius norm $\|\cdot\|$ to remove the bias introduced by scaling for each triangle pair,

$$S_{ij} \leftarrow \frac{1}{\|S_{ij}\|} S_{ij}.$$

We then take the sum of the normalized S_{ij},

$$S = \sum_{t_i \cap t_j = e \in \mathcal{E}} S_{ij},$$

where $\mathcal{E}$ is the set of all edges. The normalized version $T = (1/\|S\|)S$ is practically a global linear transformation that restores the original angles as much as possible. In our tests, employing further gradient descent in the 9D space of 3×3-matrices does not introduce much improvements.

With ideal input angles, any randomly initialized point $\Lambda = (\lambda_1, \ldots, \lambda_{12})^\top$ in the 12D space has a near 1 probability of producing the original mesh up to a rigid transformation and a scaling. With noisy data, it is not necessarily the shape with the smallest angle distortion. So we offer an optional step of gradient descent in the 12D space of all possible Λ by using squared angle residuals as the energy to minimize.

3.3.3 Gradient Descent in 12D

We restrict our gradient descent in the space spanned by the eigenvectors associated with the lowest 12 eigenvalues, as we assume that this model has the right capacity to prevent underfitting to the input data, and avoid overfitting to the angles when they are not compatible with an actual mesh. We show in the next section, that numerical evidence seems to validate this assumption empirically.

We formulate the problem as the optimizing l_2-distance between the angles of the resulting mesh and the corresponding input angles.

$$\hat{\Lambda} = \arg\min_{\Lambda,T}[\sum_{(i,j,k)\text{or}(j,k,i)\text{or}(k,i,j)\in\mathcal{T}}(\alpha_{ijk}(\Lambda,T)-\bar{\alpha}_{ijk})^2+\sum_{(i,j)\in\mathcal{E}}(\theta_{ij}(\Lambda,T)-\bar{\theta}_{ij})^2],$$

where Λ is the vector containing the 12 coefficients in $\bar{\mathbf{x}} = \sum_{i=1}^{12}\lambda_i\mathbf{e}_i$, T is an arbitrary linear transformation, $\alpha_{ijk}(\Lambda,T)$ and $\theta_{ij}(\Lambda,T)$ are the corresponding triangle angle and dihedral angles of the mesh determined by $\bar{\mathbf{x}}$ under transformation T, respectively, $\bar{\alpha}$ and $\bar{\theta}$ are the input angles, $\mathcal{T}$ is the set of triangles, and $\mathcal{E}$ is the set of edges.

We use block cyclic descent for the optimization, i.e., we first find the best T given Λ, then we fix T and perform gradient descent on Λ, and we iterate the procedure until convergence. We can use the above described voting method to find the linear transformation T, with an optional gradient descent in 9D space containing T. Gradients of the target function with respect to T and to Λ depend on the derivative of triangle angles and dihedral angles with respect to vertex coordinates, which can be found in physically based thin-shell simulation algorithms, such as [60]. The final gradients can then be assembled by chain rule. The derivative of vertex coordinates with respect to either T or Λ are straightforward, since they are linear functions of T and Λ.

3.3.4 Implementation Details

Preprocessing of angle data. With noisy input, angles can be degenerate. We first clamp all triangle interior angles to be in $[\epsilon, \pi-\epsilon]$, and all dihedral angles to be in $[-\pi+\epsilon, \pi-\epsilon]$, where $\epsilon = 0.02$. If the angle of the triangle not in input that is evaluated through Eq. 3.1 is not in the range, we clamp it and then rescale all three angles to make the sum π.

Quadratic energy construction. The target function representing the error in reconstruction can be formulated as the total penalty of violating the relation between adjacent

faces Eq. 3.2 and the relation between an edge and its incident face Eq. 3.3,

$$\sum_{t_i \cap t_j = e \in \mathcal{E}} \left\| F_{t_i} - F_{t_j} R_{ji} \right\|^2 + \sum_{(i,j,k) \in \mathcal{T}} \left\| \begin{bmatrix} \mathbf{x}_j - \mathbf{x}_i & \mathbf{x}_k - x_j & \mathbf{x}_i - \mathbf{x}_k \end{bmatrix} - F_{t_{ijk}} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \right\|^2,$$

where R_{ji} is completely determined by the angle input data. With $\mathbf{x}_i$'s and F_t's stacked into a vector $\tilde{\mathbf{x}}$, the quadratic energy above can be rewritten as $\tilde{\mathbf{x}}^\top \mathrm{A} \tilde{\mathbf{x}}$.

Pseudocode. The overall implementation can be summarized in the following pseudocode.

Data: Inner and Dihedral Angles α_{ijk} and θ_{ij}

Result: Vertex Location $\mathbf{x}_i$

Preprocess input angles;

Construct Transition Matrices R_{ij} for each edge;

Assemble quadratic form A of penalty function for construction error;

Evaluate lowest 12 eigenvectors of A;

Construct a linear combination $\tilde{\mathbf{x}}$;

for *each pair of adjacent triangles* t_i *and* t_j **do**

Perform local construction of the four vertices;

Evaluate frames $\bar{F}_i$ and $\bar{F}_j$ in local construction;

Find the best linear transformation from F_i and F_j in $\tilde{\mathbf{x}}$ to the local construction through $T_{ij} = [\bar{F}_i \bar{F}_j][F_i F_j]^\top ([F_i F_j][F_i F_j]^\top)^{-1}$;

Perform polar decomposition $T_{ij} = U_{ij} S_{ij}$;

$T+ = S_{ij}/\|S_{ij}\|$;

end

Apply $T/\|T\|$ to all vertex positions in $\tilde{x}$ to produce output;

Optional Gradient Descent in 12D for further reduce angle deviation;

Algorithm 1: Angle-based Shape Evaluation

3.4 Results

We tested our procedure on various shapes including closed meshes and open meshes with noisy input angles. Figure 3.3 top row shows reconstruction results with different level of noises in dihedral angle. In this test we add a random noise uniformly distributed within $[-n\%, n\%]$ with a given percentage of the magnitude of the angle to simulate quantization error and other noises in deformation algorithms. Figure 3.3 second row shows reconstruction results with different level of noises in triangle interior angles. The behavior is similar when we add noises to both interior and dihedral angles as shown in Figure 3.3 bottom row.

We confirm that our spectral method combined with affine correction leads to robust reconstruction results for closed mesh. We also found that noises in triangle angles can have a larger impact than those in dihedral angles. In our test, the effect of noise level of $n/4\%$ for triangle interior angles seems to be similar to the effect of noise level of $n\%$ in dihedral angles.

For open meshes, such as the flags shown in Figure 3.6, our method is, however, more sensitive to noise than the closed mesh case. The reconstruction result will deviate far when the noise level is larger than 2%. Nevertheless, it is far better than sequential construction.

We assembled a gallery of reconstruction results for closed mesh models of various types in Figure 3.5. The approach is capable of handling meshes with high curvature regions, and those with high genus.

Results for gradient descent. We also tested the further optimization through gradient descent in the 12D space spanned by the eigenvectors. It is shown that the total angle deviation indeed decreases as shown in Figure. 3.7, although the improvement does not lead to much difference visually when the meshes are rendered. We have also performed tests on multiple random initialization, and found that they may lead to different local minima in terms of total angle deviation, but the impact on mesh coordinates seems hardly discernable. We postulate that the information about the shape signal is already lost with large noise, so

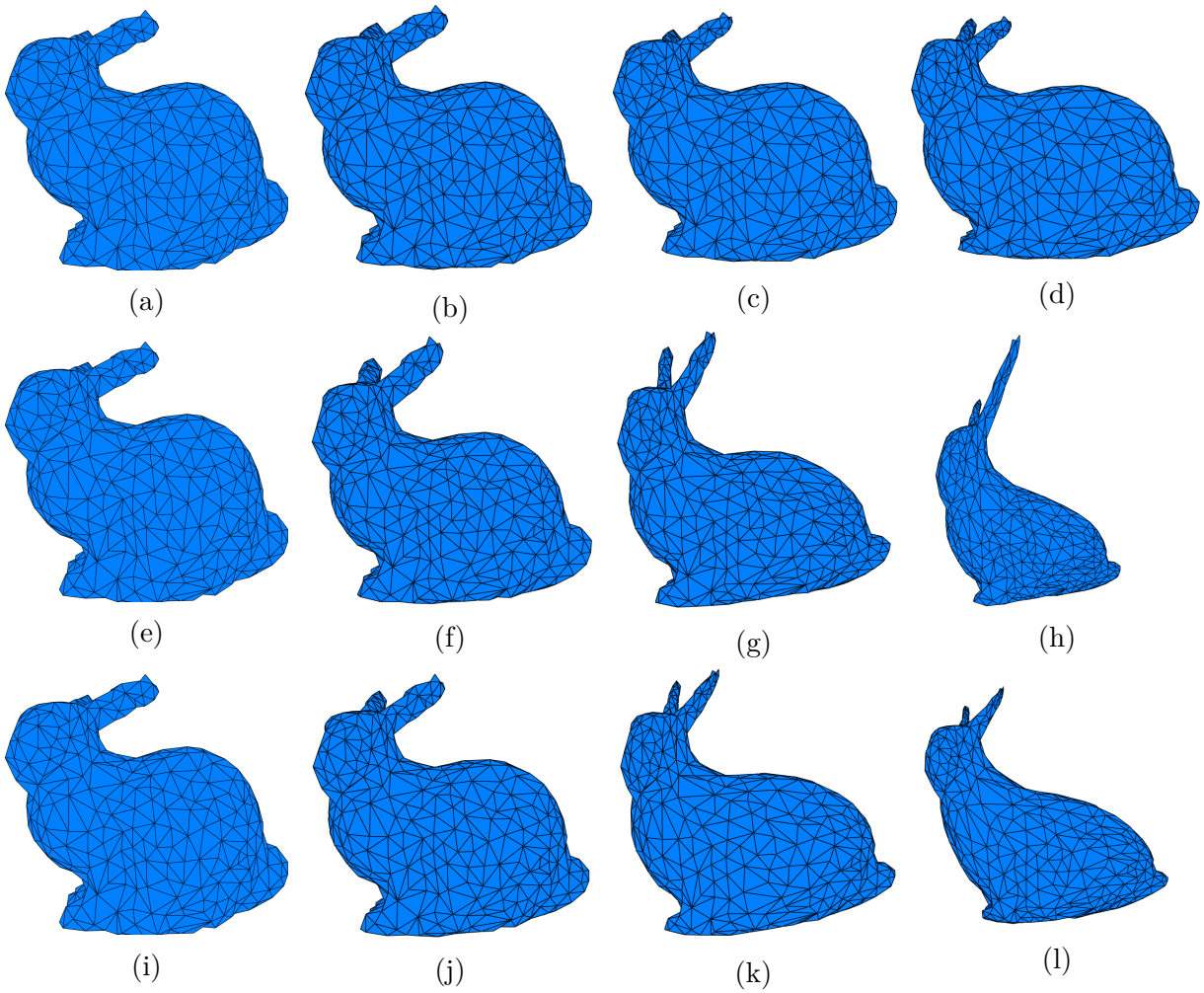


Figure 3.3 Top: Results for adding different levels of noises to dihedral angles. Middle: Results for adding different levels of noises to triangle interior angles. Bottom: Results for adding different levels of noises to both angles. From left to right: the maximum random noise magnitude is at the level of 0%, 10%, 20%, and 30%, respectively. The result in this example is resilient to such noises. Distortion only becomes obvious when interior angles reached 30% noise level.

further improving the fitting to the noisy input does not improve the shape quality much.

Comparison to sequential construction. We follow a similar strategy to Angle Analyzer in this test, i.e., we sequentially construct the vertex locations by starting from a seed triangle, fix its translation, rotation, and scaling, and traverse the entire mesh by moving from the visited triangles to adjacent unvisited triangles.

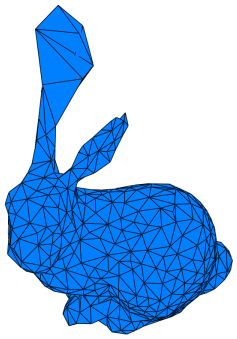


Figure 3.4 Random 0.01% noise level

As seen in the inset figure, random noise with a maximum of just 0.01% leads to discrepancy when faraway vertices are reached following different paths in the dual graph. In contrast, our method will distribute noise evenly through the whole mesh, making the reconstructed shape continuous. For the result in the comparison figure, we traversed the mesh by breadth first search, and our results with depth first search led to even worse artifacts on the reconstructed shapes.

Performance. The step with the highest time complexity in our approach is solving the eigenvalue problem. We simply use the eigensolver in Matlab for sparse symmetric matrices. For a mesh with a few thousands of vertices, it takes less than a minute to produce the lowest 12 eigenvectors. The assembly of A and the voting for the affine transformation are both linear time complexity in terms of vertex count. However, even though the complexity is linear, for large meshes, the voting for best affine correction matrix can take a few minutes due to our unoptimized implementation. The optional gradient descent in the 12D space depends on convergence criterion, as it determines the number of iterations, but for each iteration, the evaluation of all the derivatives is with linear time complexity as well.

Discussion and Limitations Our method can robustly reconstruct closed surfaces with large noises (up to 40% on some models). With such noises, the simple sequential reconstruction will lead to meshes distorted beyond recognition. For open surfaces that are isomorphic to a flat region, noises larger than 2% can already lead to visible distortions (still better than the 0.01% noise for sequential construction though). We postulate that the problem is with the lack of boundary constraints, on the other hand, our linear system combined with boundary constraints could be used if such constraints are available, and the frame variables can even be used to specify Neumann or Cauchy types of boundary constraints.

Another limitation of our approach is the large number of additional variables $9|\mathcal{F}| \approx 27|\mathcal{V}|$. We leave it as future work to see if the linear map can be modified to use only the normal. The basic idea is that the vertices of one triangle together with its normal is sufficient to span the 3D space and be used to represent the vertices in adjacent triangles.

3.5 Conclusion and Future work

We propose a spectral method for mapping triangle angles and dihedral angles, as one type of local shape descriptors, to the global shape, vertex locations of a surface. For this purpose, we construct a quadratic energy measuring the deviation from a local linear reconstruction relation determined by these angles. The quadratic energy function is then optimized with a spectral method to prevent bias in the presence of noise, as well as to prevent collapsing of the surface to a single point due to the rescaling invariance of the angle-based representation. An affine correction is obtained by voting from each pair of triangles, where each vote restores the angles in the triangle pair from the distortion due to the invariance of the local linear reconstruction relation. We offer an optional step of gradient descent to further reduce reconstruction angle deviation.

While we tested the robustness of our reconstruction under high levels of noise or compression loss (far higher than the level that can be tolerated by sequential construction), we did not explore the coding of connectivity with noisy angle data. In future, we plan to incorporate a strategy similar to angle analyzer, and test the effectiveness of our representation through common metrics, such as L_2-distance and Hausdorff distance, and to use an actual entropy coder to measure the bit rate per vertex.

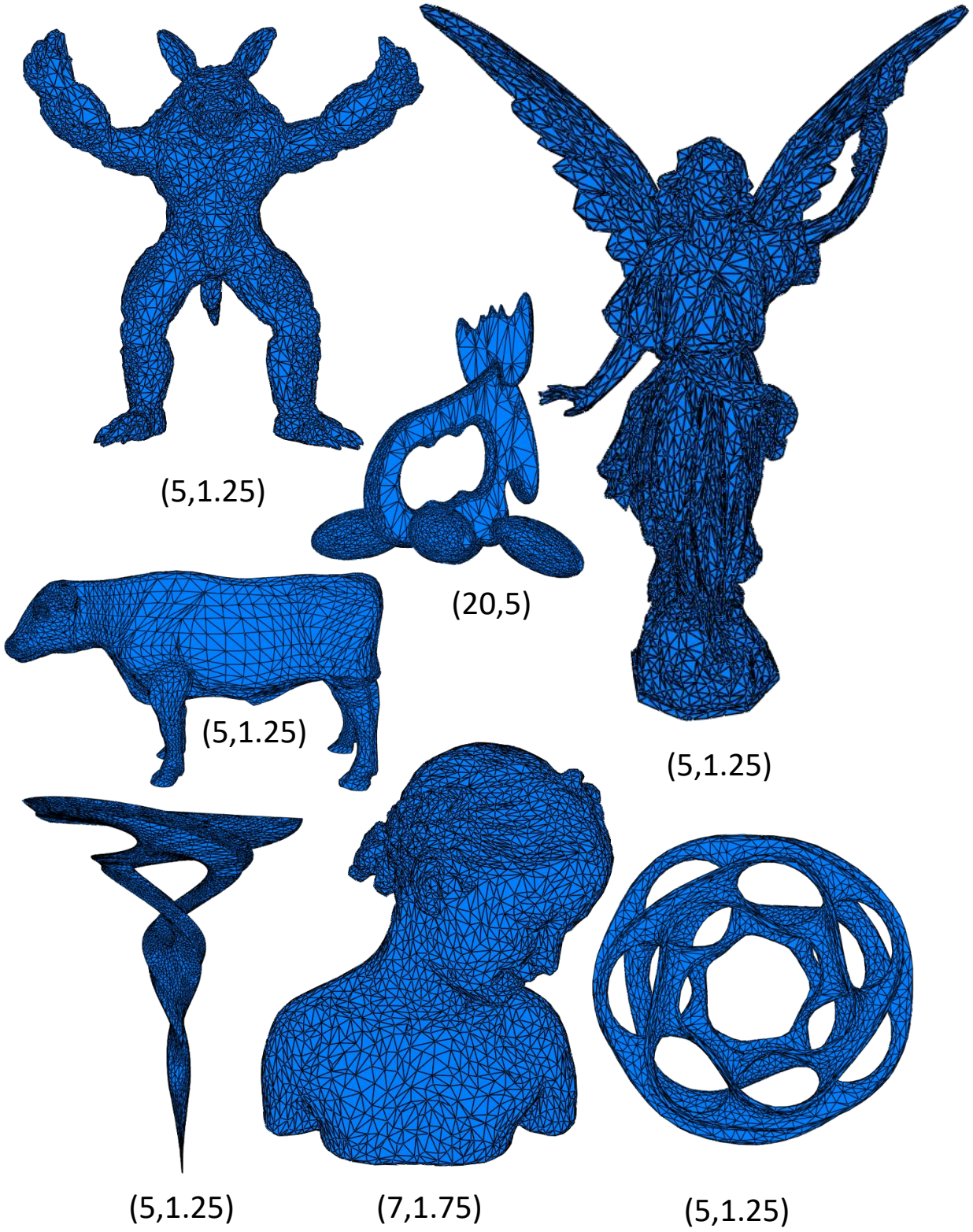
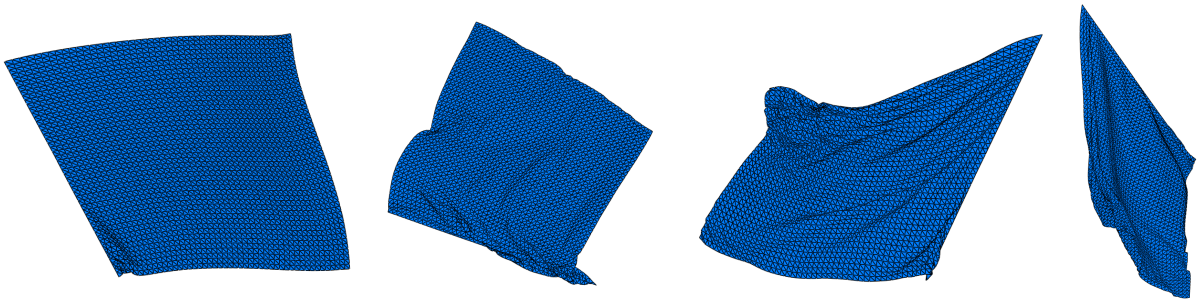
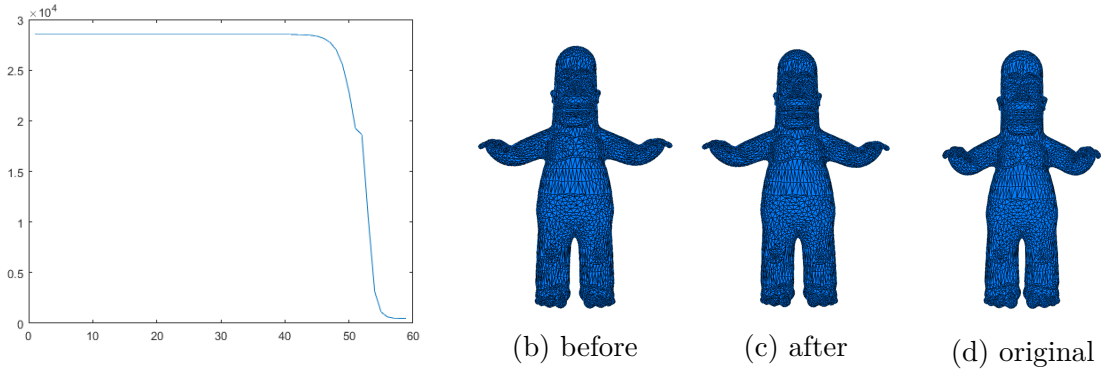


Figure 3.5 We show in this collection of reconstructed models a variety of shapes, including surfaces with high curvature regions, with high genus (22), or with irregular sampling. The first number is the percentage of max dihedral angle noise, and the second number is that of interior angle noise.



(a) (b) (c) (d)

Figure 3.6 The results for deforming flags with 1% noises added to both dihedral and interior angles. The reconstruction produces large deviation from the original shape if larger noises are added, especially for interior angles. We postulate that the deviation from isometry as the original embedding curvature can have a large impact when the mesh has a boundary. The boundary vertex positions are determined only from their relation to the neighbors from certain directions, whereas in the closed mesh case the vertex positions are determined by neighbors from all directions.



(a) (b) before (c) after (d) original

Figure 3.7 The angle deviation decreases over iteration in our 12D gradient descent. However, the result after gradient descent remains quite similar to the original guess. The noise level on dihedral angle is at 20%.

CHAPTER 4

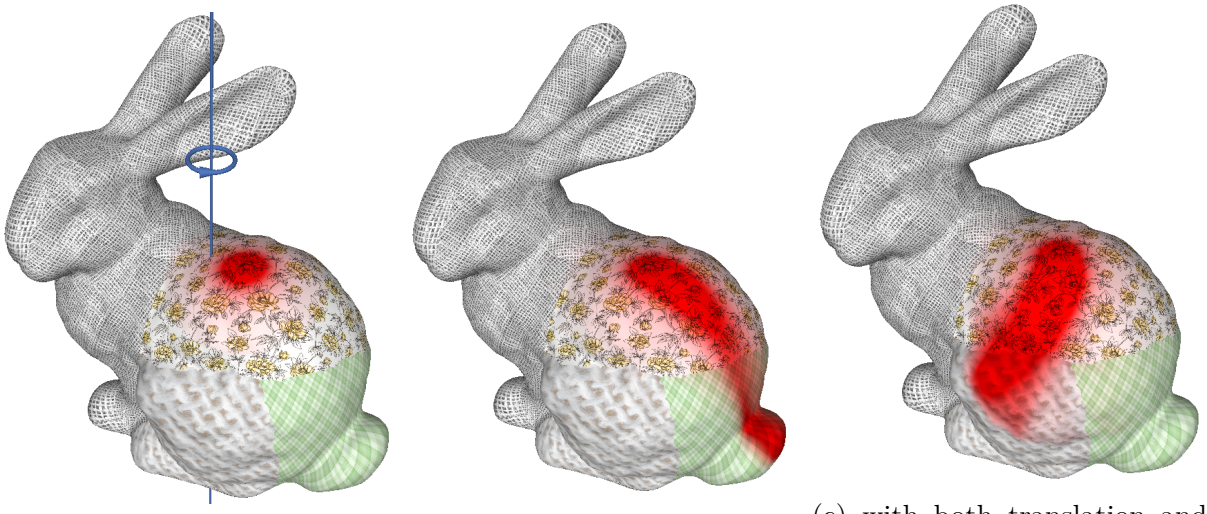
FLUID SIMULATION

4.1 Introduction

Despite the large number of physically-based methods for both cloth animation and fluid simulation, very limited effort has been devoted to the simulation of stain formation on cloth in motion: the complexity in accounting for inertial forces and anisotropic effects that the absorption, adsorption, and diffusion of a liquid on knitted or woven textiles involve has thwarted their efficient simulation. However, with the rapid improvement in computing power and the advent of virtual worlds, simulating the formation and evolution of stains on moving cloth could have a wide range of applications in computer games, special effects in movies, digital arts, and augmented reality.

Stains on fabric, textile, or even paper are formed when a fluid such as ink, wine, or water gets into the fibers of the medium, starting a process of diffusion, absorption, adsorption, and evaporation. Since knitted or woven cloth materials are often highly anisotropic and inhomogeneous, the evolution of a stain in time depends heavily on the pattern of threads or yarns of the textile. Moreover, the motion of the cloth itself can have a dramatic effect on the stain formation: inertial forces (be they centrifugal or Coriolis) that the fluid experiences will bias the propagation of the stain, adding further complexity to the phenomenon.

We present a simple physically-based approach to capture the evolution of a fluid density (i.e., stain) over a moving and deforming cloth. Using homogenization theory, we construct the bulk anisotropic diffusion tensor based on the composition and the weaving pattern of the fabric. The resulting diffusion tensor is mapped onto the cloth by specifying the local alignment of the fabric. The fluid motion is then calculated directly on the mesh by storing the velocity field as a tangent vector field. The velocity field is influenced by the movement



(a) without fictitious force (b) without Coriolis forces (c) with both translation and rotational forces

Figure 4.1 Stain on a bunny rotating around a vertical axis.

of the deforming cloth through locally estimated inertia forces. Interactions between stain and textile including absorption, adsorption and evaporation are also modeled by keeping track of the density fields of various components of the solute and solvent. Our approach generates realistic visual simulation of stains, which we demonstrate by showing complex stain evolutions on deforming cloth made of a variety of inhomogeneous and anisotropic materials.

4.2 Semi-Lagrangian Fluid Simulation

In a typical simulator for particles, both particle locations and particle velocities need to be updated. The particle coordinates $\mathbf{x} = (x^1, x^2, x^3)^\top$ and velocity $\mathbf{u} = (u^1, u^2, u^3)^\top$ are expressed through components in an inertial orthonormal frame in 3D Euclidean space. If we map 2D surface to a parameter domain, we can do our computation in 2D space.

4.2.1 Eulerian vs Lagrangian representations

Eulerian and Lagrangian representations are widely used in simulating fluids in computational science. The former assumes that we set an observer at a fixed point, or grid point, to keep tracking the fluid variables, while the latter track single fluid particle. In other words, in Eulerian representation, $\mathbf{u}$ is expressed as a function of $\mathbf{x}$, and are sampled at grid points $\mathbf{x} = (ih, jh, kh)^\top$, where i, j, k are integers and h is the grid spacing; while in Lagrangian representation, both position and velocity for each particle representing a parcel of fluid are kept as $\mathbf{x}_i, \mathbf{v}_i$ for particle i.

As all particles are deemed as the same, in Eulerian representation, only the velocity field needs to be updated for the dynamics. For our purposes, the Eulerian representation is more efficient. In updating the velocity, a semi-Lagrangian method is often used, it updates the velocity by following the particle along its path backward within each step to perform a stable update.

The Navier-Stokes equation can be seen as directly the result of an Eulerian representation of the field $\mathbf{u}(\mathbf{x})$ based on Newton's second law. Since acceleration $d\mathbf{u}/dt = \partial\mathbf{u}/\partial t + \mathbf{u} \cdot \nabla\mathbf{u}$, we conclude that

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{1}{\rho}(-\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}),$$
$$\nabla \cdot \mathbf{u} = 0,$$

where ρ is the mass density, p is the pressure, μ is the dynamic viscosity, and $\mathbf{f}$ is body force density, such as gravity. The divergence freeness comes from the assumption that the characteristic velocity in the simulation is much lower than the speed of sound. These equations have 4 components to update the 4 numbers in $\mathbf{u}$ and p.

4.2.2 Fictitious Force

For Eulerian representation on deforming surfaces, we note that Newton's second law has to be modified as the grid on the surface is deforming over time, and cannot serve as inertial frames. However, it is still convenient to use coordinate frames attached to each vertex, which is comoving with the vertex, and corotating with the one-ring neighborhood of the vertex. However, the inertia forces are often ignored in many existing fluid dynamic system for easier implementation either entirely, or with only rectilinear acceleration being considered.

We show that this challenging problem for fluid flow on deforming mesh with Coriolis and rectilinear forces can be solved with the registration of the comoving corotating frame aligned with the tangent space. The difference is not to be neglected if there is non-trivial relative acceleration or rotation between the global frame and the local frame.

4.3 Simulation on 3D Surface

The motion of the solution fluid on surface is governed by restricted Navier-Stokes equations, in which the normal velocity can be ignored, i.e., the normal acceleration is balanced by the constraint for the fluid to stay on the surface:

$$\dot{u} + u \cdot \nabla u = \nabla \cdot (S_v \nabla u) - \nabla p_{\text{ext}}/\rho + a_{\text{ext}}, \tag{4.1}$$

$$\dot{\sigma}_f + \nabla \cdot (\sigma_f u) = \nabla \cdot (S_f \nabla \sigma_f) + \text{interact}_f, \tag{4.2}$$

$$\dot{\sigma}_s + \nabla \cdot (\sigma_s u) = \nabla \cdot (S_f \nabla \sigma_s) + \text{interact}_s, \tag{4.3}$$

where $\dot{x} = \partial x/\partial t$ denotes the time derivative, u is the tangential velocity field, ρ is the density of the incompressible fluid, p_{ext} is the external pressure (ignored in our tests), a_{ext} is the external body force (including tangential components of the gravity and inertial forces detailed in Sec. 4.4), σ_f is the solvent fluid density (per unit area), σ_s is the solute density, and interact_f (interact_s) is the interaction terms detailed later (Sec. 4.5). Aside from the solute

density, the above equation is equivalent to the generalized shallow wave equations in [31] with $\sigma_f = H\rho$, where H is the height of the solvent in the normal direction. We also assume that the motion of the solvent is not influenced by the solute, and the viscosity/permeability tensors are different for the tangent velocity field (S_v) and the densities(S_f). These tensors are evaluated through the aforementioned homogenization procedure, as we assume that the fluid is partially flowing on the surface and partially permeating through the porous media.

As shown in Figure 1.3, we follow a typical split step time integration as in [40]. In each iteration, we first compute the external forces (Sec. 4.4); then, we advect the velocity and solvent and solute densities (Sec. 4.3.1); next, we perform the diffusion processes (Sec. 4.3.3); and finally, we handle the additional interaction terms, adsorption and evaporation (Sec. 4.5).

4.3.1 Advection

Our velocity advection essentially follows the procedure in [37]. The surface is triangulated and the velocity field is discretized as one 2D vector $u_i = (u_i^x, u_i^y)^T$ per vertex in the XY-plane of the tangent frame F_i stored at vertex i. F_i is defined by setting z_i to area-weighted average unit normal of the one-ring, $y_i = (z_i \times e_{ij})/|z_i \times e_{ij}|$, where e_{ij} is one of the incident edges of vertex i, and $x_i = y_i \times z_i$.

The key issue in the velocity advection is the calculation of ∇u, which, on a curved surface, is the covariant derivative of u. It provides a way to compare u_i and u_j for adjacent vertices, since a direct differencing of the components would not produce the true coordinate-frame independent "vector gradient". One approximation of ∇u is to map the 2D tangent field to 3D through F_i for comparison. However, much numerical diffusion would be introduced as long as the edges are not infinitesimal. Thus, we follow the practice in [37] and use a 2×2 rotation matrix R_{ij} to align the vector expressed in F_j to F_i.

The 2×2 rotation matrix R_{ij} can be regarded as parallel transporting a vector from

vertex j to vertex i along the edge connecting the two. Thus $R_{ij}u_j - u_i$ produces the covariant derivative ∇u integrated along the edge. To obtain the rotation angle for the 2D rotation matrix, we map the 1-ring of vertex i to a flat 2D topological disk through the geodesic polar map by rescaling the sum of tip angles to 2π. If the local edge e_{ik} is the chosen edge to construct F_i, the direction of the edge e_{ij} can be represented by the angle α_{ij} that it forms with the chosen x-axis in the 2D domain. Assuming the sum of tip angles for all the triangles adjacent to vertex i is γ_i, α_{ij} is the sum of all the tip angles of the triangles between e_{ik} and e_{ij} in the counterclockwise order rescaled by $2\pi/\gamma_i$. Likewise, e_{ji} makes an angle α_{ji} with the x-axis in the local frame of vertex j. Thus, the rotation matrix R_{ij} can be expressed through the angle $\theta_{ij} = \alpha_{ij} - \alpha_{ji} + \pi$,

$$R_{ij} = \begin{bmatrix} \cos\theta_{ij} & -\sin\theta_{ij} \\ \sin\theta_{ij} & \cos\theta_{ij} \end{bmatrix}. \tag{4.4}$$

Following the semi-Lagrangian advection method [40] on the surface, we first backtrack from vertex i in its flattened one-ring using u_i by a time step h, evaluating the barycentric coordinates $(\lambda_i, \lambda_j, \lambda_k)$ within the triangle t_{ijk}, and update the velocity u_i by

$$u_i^{t+h} = \lambda_i u_i^t + \lambda_j R_{ij} u_j^t + \lambda_k R_{ik} u_k^t.$$

It is possible to extend the method to include the case when the backtracked point is outside the one-ring by following the strip of the triangles traversed.

For density field advection, we use a finite-volume approach instead for mass conservation of both solvent and solute. Since σ_f and σ_s advect in the same fashion, we use σ_s to illustrate the process. We discretize the solute density by assigning a value $\sigma_{s,i}$ for the Voronoi region of vertex i, and we update it by

$$\sigma_{s,i}^{t+h} = \sigma_{s,i}^t - h/A_i \sum_{j \in N(i)} F_{ij},$$

where $N(i)$ is one-ring of vertex i, A_i is the Voronoi region area, and F_{ij} is the flux through

the interface between the Voronoi regions of i and j,

$$F_{ij} = \sigma_{ij}^{\uparrow}(u_{ij} \cdot e_{ij})w_{ij},$$

where $u_{ij} = 1/2(u_i^x x_i + u_i^y y_i + u_j^x x_j + u_j^y y_j)$, with (x_i, y_i) denoting the local tangent plane in frame F_i, $\sigma_{ij}^{\uparrow}$ is the upwind density, i.e., it is $\sigma_{s,i}$ if $u_{ij} \cdot e_{ij} > 0$ and $\sigma_{s,j}$ otherwise, and w_{ij} is the usual cotan weights, ratio between the dual Voronoi edge length and $|e_{ij}|$,

$$w_{ij} = -\frac{1}{2}(\cot\alpha + \cot\beta), \tag{4.5}$$

where α and β are the opposite angles of e_{ij} in the two incident triangles.

4.3.2 Momentum Advection

An alternative approach is to use momentum advection instead. First, we update fluid density through advection, then we can update the momentum in the same fashion as density, from which we get the velocity of next time step using the updated momentum.

Assume A is the dual area. Define $flow_{ij}$ to be how much fluid across the border, We have the following discretized update equation:

$flow_{ij} = \rho * (e_{ij} \cdot v_{ij}(t)) * cotan() * h$

$\rho(t+h) = \rho(t) - flow_{ij}/A$

$M(t+h) = v(t) * \rho(t) * A + v(t) * flow_{ij}(t)$

$v(t+h) = \frac{M(t+h)}{\rho(t+h)*A}$

The pseudo-code can be formulated as:

Algorithm: Momentum Advection

FOR EACH $vertex_i$ DO

FOR each vertex j around vertex i DO

$velocity_{ij} \leftarrow e_{ij} \cdot (v_i + v_j)/2$

IF flow will move from vertex i to vertex j

$flow_{ij} \leftarrow density_i \times velocity_{ij} \times cotan()_{ij} \times \delta t$

ELSE

$$flow_{ij} \leftarrow density_j \times velocity_{ij} \times cotan()_{ij} \times \delta t$$

$$density_i \leftarrow density_i - flow_{ij}/A$$

$$momentum_i \leftarrow velocity_i \times density_i \times A + velcity_i \times flow_{ij}$$

$$velocity_i \leftarrow momentum_i/(density_i \times A)$$

4.3.3 Diffusion

As our diffusion is performed directly on the curved surface, we need to specify the material (the homogenized diffusion tensors) on the triangle mesh. The textile can be specified with weft and warp directions, so we partition the surface into a few patches, each with a smooth weft direction field e, which induces the warp direction as its 90° rotation $e^\perp$. We delay the discussion on the calculation of these fields to the end of this section, since it is essentially using the same discretized operators involved in the diffusion process. The main operator in this process is the modified Laplacian operator $\nabla \cdot S\nabla$, which, in piecewise linear finite element method, can be discretized as a linear operator (matrix) L_S,

$$L_{S,ij} = \sum_{e_{ij} \in T} \int_T \nabla \phi_i \cdot (S_T \nabla \phi_j),$$

where ϕ_i is the linear basis function for vertex i, where $S_T = (e, e^\perp) S (e, e^\perp)^T$ is the diffusion tensor aligned to the specified direction field within triangle T. For S equals to identity, L_S reduces to the usual cotan formula.

The temporal discretization of the diffusion process is performed using an implicit integration,

$$M(\sigma^{t+h} - \sigma^t)/h = L_S \sigma^{t+h},$$

where M is the mass matrix with $M_{ij} = \sum_{e_{ij} \in T} \int_T \phi_i \phi_j$, which is often simplified through mass lumping to just a diagonal matrix with $M_{ii} = A_i$, with A_i being one third of the one-ring area for vertex i. Thus, the above process is turned into a linear system $(M + hL_S)\sigma^{t+h} = M\sigma^t$.

The diffusion process of the velocity field involves the covariant derivative ∇u as the one described in the advection process. However, using the same discretization of covariant derivative through the use of R_{ij} for aligning the tangent vectors at vertices i and j, we can create essentially the Bochner Laplacian in the metric of S_v by replacing each entry in the $N \times N$-matrix (N is the number of vertices in the patch) $L_{S,ij}$ with a 2×2-matrix $L_{S,ij} R_{ij}$ with R_{ii} set to the identity matrix, and obtain a $2N \times 2N$-matrix. The implicit integration results in the following linear equations

$$A_i(u_i^{t+h} - u_i^t) = h \sum_{j \in N(i)} L_{S_v,ij} \, (R_{ij} \, u_j^{t+h} - u_i^{t+h}). \tag{4.6}$$

In order to obtain the smooth direction field, we simply use (at least one) user specified direction for each patch, and solve the discretized $\nabla \cdot \nabla e = 0$ under the user constraint.

4.3.4 Diffusion for Anisotropic Material

Assume we have tent functions at vertices of triangle ijk, denoted as

$\nabla \phi_i$ and $\nabla \phi_j$, $A = \frac{|e_{kj}||e_{ki}| sin(a)}{2}$

We incorporate the tensor matrix T into the dot product of the two tent functions.

Algorithm: Computation of new cotangents

FOR EACH $halfedge_{ij}$ DO

$\vec{e'_{ki}} \leftarrow R_{pi/2} \times e_{ki}$

$\vec{e'_{kj}} \leftarrow R_{pi/2} \times e_{kj}$

$\nabla \phi_i \leftarrow \frac{|e_{kj}|}{2A} \vec{e'_{kj}}$

$\nabla \phi_j \leftarrow \frac{|e_{ki}|}{2A} \vec{e'_{ki}}$

$s_{ij} \leftarrow 2A \nabla \phi_i^T T \nabla \phi_j$

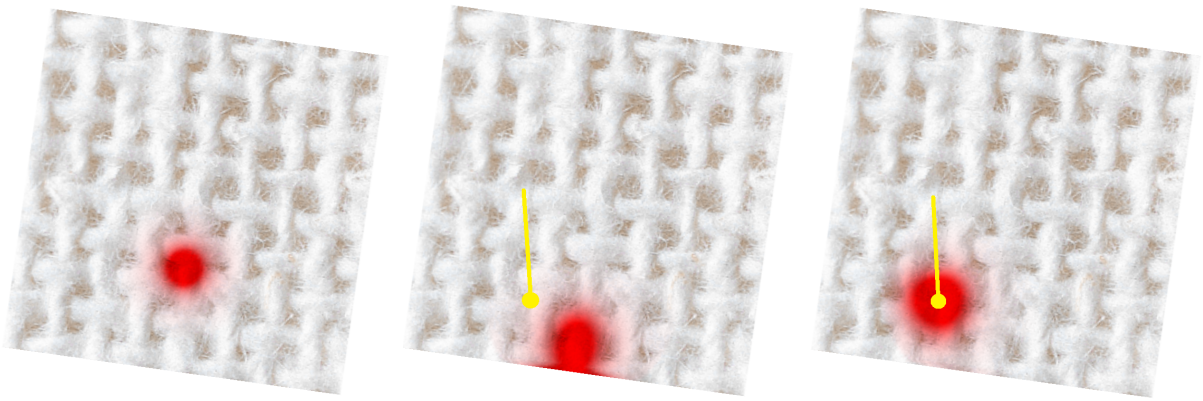


Figure 4.2 The comparison shows that both translational (missing in a) and rotational (missing in a and b) inertia forces are necessary for the comoving and corotational frames. A solid ball is indicates the motion of the center of mass (without diffusion) in an inertial coordinate system.

4.4 Fictitious Forces

For a deforming underlying surface, each frame F_i attached to vertex i can be time dependent. It can be seen as a comoving and corotating frame for the one-ring neighborhood. Similar to [61], we find $F_i = U_i \bar{F}_i$ by calculating the best rotation aligning the original one-ring to the deformed one-ring U_i, where $\bar{F}_i$ denotes the original frame at vertex i,

$$U_i = \text{argmin}_{R \in SO(3)} \sum_{j \in N(i)} [(v_j - v_i) - R(\bar{v}_j - \bar{v}_i)]^2,$$

where $SO(3)$ is the set of all 3D rotations, v_i and $\bar{v}_i$ are respectively the current and original locations of vertex i. Instead of performing a polar decomposition as [61], we directly find the minimum using the Kabsch procedure in [62]. Assuming that the position of vertex i is p_i, the inertial force (acceleration) experienced by a moving object at location r in the frame F_i centered at p_i can be expressed in this frame as the tangential components of

$$a_{\text{inertia},i} = -F_i^T \ddot{p}_i - F_i^T \ddot{F}_i r - 2F_i^T \dot{F}_i \dot{r},$$

where the second term (the sum of the centrifugal force and the Euler force) on the right hand side vanishes since the fluid velocity is measured at the vertex $r = 0$, the first term is called the linear inertia force due to the translation of the local frame, and the last term is

the Coriolis force due to the rotation of the local frame and the local velocity $\dot{r} = u$. The total body force is given by

$$a_{\text{ext}} = a_{\text{inertia}} + (g - (g \cdot n)n),$$

where g is the gravitational acceleration and n is the surface normal.

Given the motion of the mesh stored in a sequence of vertex locations, we can easily calculate the linear inertia force by centered differencing,

$$\ddot{p}_i^t = \frac{p_i^{t+h} - 2p_i^t + p_i^{t-h}}{h^2}.$$

As $F^T \dot{F}$ for F in the rotation group $SO(3)$ is in the Lie algebra $\mathfrak{so}(3)$, i.e., antisymmetric 3×3 matrices, we evaluate the Coriolis force as $2\omega \times u_i$, where ω is evaluated by

$$\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \frac{(F^t)^T F^{t+h} - (F^{t+h})^T F^t}{2h}.$$

Here, to get a proper angular velocity ω, we antisymmetrize the estimate of $F^T \dot{F} \approx (F^t)^T (F^{t+h} - F^t)/h$.

The pseudo-code for fictitious force calculation is given below:

Algorithm: Fictitious Force

Input: Mesh deforming data $p(t - \delta t), p(t), p(t + \delta t)$, $p(0)$, and local frame (U)

Output: Acceleration in local frame a

FOR EACH $vertex_i$ DO

$x \leftarrow p_j(t) - p_i(j)$

$y \leftarrow p_j(0) - p_i(0)$

Generate transformation that is best match(x, y), $B \leftarrow match(x, y)$

$C \leftarrow B \times U$

$\omega \leftarrow C^T \times (C(t) - (Ct - \delta t))/\delta T$

$$CoriolisAcceration \leftarrow -2 \times \omega \times velocity_i$$
$$RectilinearAccerlation \leftarrow C^T \times (2 \times p(t) - p(t - \delta t) - p(t + \delta t))/(\delta t)^2$$
$$a = CoriolisAcceration + RectilinearAccerlation$$

4.5 Stain-Surface Interaction

In addition to diffusion, the solution is influenced by the surface through penetration, adsorption and evaporation. Both solvent and solute are involved. The solute is separated into three parts, one dissolved in the solution, one absorbed by the textile fibers, and the other deposited in the fibers. The solvent is partially absorbed by the textile, and it evaporates.

In the process of the textile absorbing the solution, the penetration depth and the absorption speed can be evaluated through the Lucas-Washburn equation [63], which shows that the speed is inversely proportional to the penetration depth. We use a simplified formula with the absorption speed dependent on how far the surface is becoming saturate, assuming that the surface is thin and thus reaches the saturation capacity σ_{sat} quickly. We calculate the absorbed solvent density σ_p as

$$\dot{\sigma}_p = K_p(\sigma_{\text{sat}} - \sigma_p),$$

where K_p is the absorption speed constant.We can use a simple implicit scheme at each vertex,

$$\sigma_p^{t+h} = \sigma_p^t + \min(K_p h/(1 + K_p h)(\sigma_{\text{sat}} - \sigma_p^t), \sigma_f^t),$$

where σ_f is the maximum amount of solvent left to be absorbed. We also maintain the absorbed solute density $\sigma_{s,p}$, which evolves as

$$\dot{\sigma}_{s,p} = K_p(\sigma_{\text{sat}} - \sigma_p)\sigma_s/\sigma_f.$$

The absorbed solvent and solute go through the same adsorption and evaporation process as their free flowing counterparts, but they are assumed to be held in place and not diffusing much.

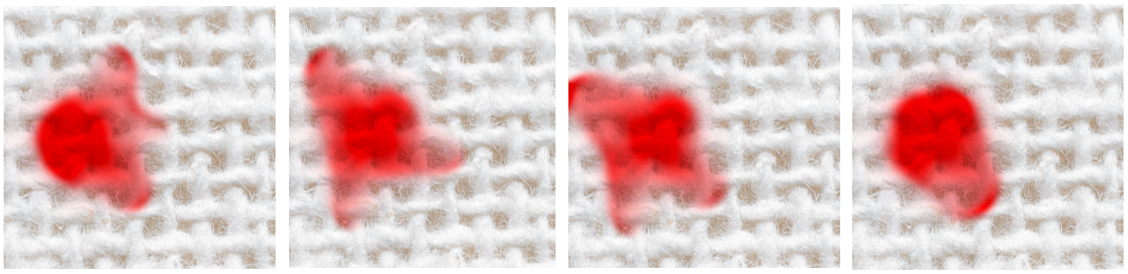


Figure 4.3 The stain induced by various initial fluid velocities.

Adsorption is the process of solute gradually being deposited into the textile fibers and becoming difficult to be dissolved again by the solvent. We model it by the Langmuir adsorption theory [64], which assumes monolayer adsorption, so the adsorbent will not adsorb further after the adsorbate is covered. Before reaching adsorption equilibrium, the adsorption rate is proportional to the area of the blank surface, while the desorption rate is proportional to the coverage. We can calculate the adsorption rate and the desorption rate as $K_a V_d(1–\theta)C$ and $K_d V_d \theta C$ resp., where V_d is the total adsorption capacity determined by the material and porosity, θ denotes the coverage of the surface, C represents the concentration of the single layer adsorbed solute, K_a and K_d are the adsorption rate and desorption rate, respectively. When the adsorption is balanced with desorption, we obtain the maximum adsorption capacity $A_d = V_d K/(1 + K)$, where $K = K_a/K_d$. We calculate the change of adsorbed solute $\sigma_{s,a}$ as the difference between adsorption and desorption, which is equal to

$$\dot{\sigma}_{s,a} = K_d(A_d - \sigma_{s,a}), \tag{4.7}$$

which can be discretized exactly as in the absorption process.

The solvent in the solution evaporates. The change of the amount of solvent is directly proportional to the surface area, $\dot{\sigma}_p = -K_{\text{evap}}$, where K_{evap} is the evaporate coefficient. However, the effective exposed area of the boundary cells is greater than that of the inside cells. So the evaporation of the boundary cell will be faster, and the above calculation should be changed as

$$\dot{\sigma}_f = -a_{\text{evap}} K_{\text{evap}}, \tag{4.8}$$

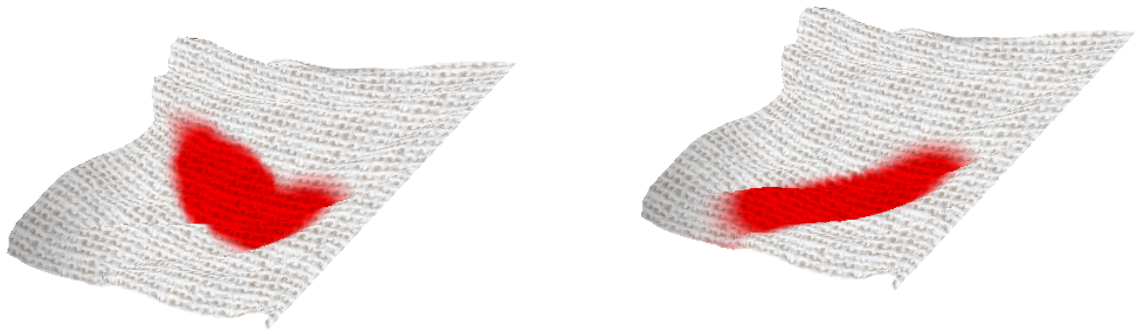


Figure 4.4 Simulation on a waving flag.

where a_{evap} is boundary coefficient (we use 1.2 in our tests). Note that the amount evaporated in each step is bounded by the total amount left.

The combined effects can be formulated as

$$\text{interact}_f = -K_p(\sigma_{\text{sat}} - \sigma_p) - a_{\text{evap}} K_{\text{evap}} \tag{4.9}$$

$$\text{interact}_s = -K_p(\sigma_{\text{sat}} - \sigma_p)\sigma_s/\sigma_f - K_d(A_d - \sigma_{s,a}). \tag{4.10}$$

The entire state of the system includes the velocity field, solvent and solute on surface, absorbed solvent, absorbed solute, and adsorbed solute, all of which is updated in each time step.

4.6 Results

In the rotating planar region test (Figure 4.2), we show that without fictitious force, the fluid motion is not influenced by the motion of the underlying mesh. If we ignore the influence of rotation, as was often done in graphics, the fluid still deviates far from the trajectory without the Coriolis effects.

When the bunny model undergoes rigid motion (Figure 4.1a), our procedure leads to realistic fluid motion, taking inertia force as well as gravity into account. When the un-

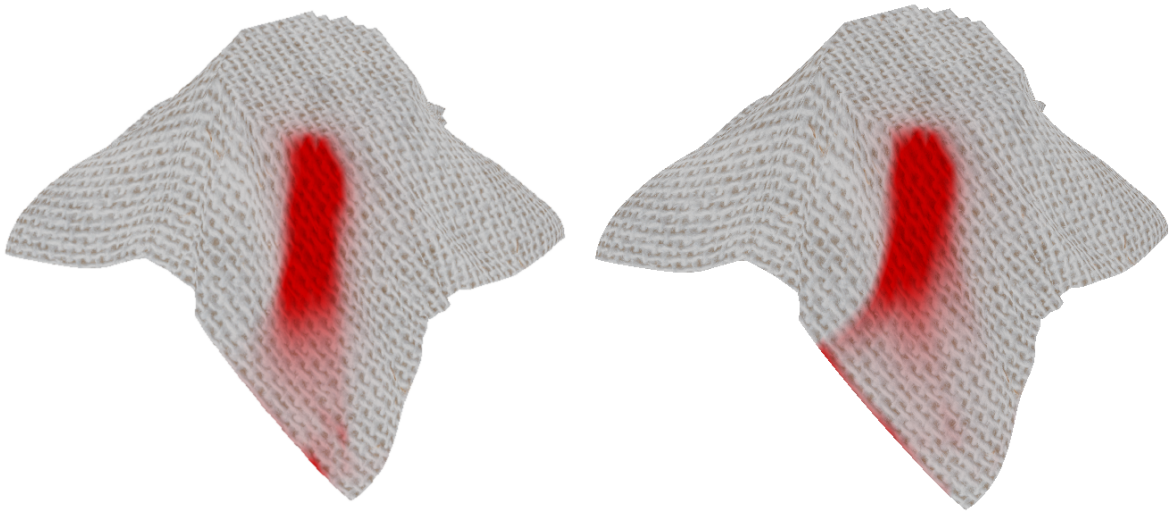


Figure 4.5 Stain on a dropping tablecloth.

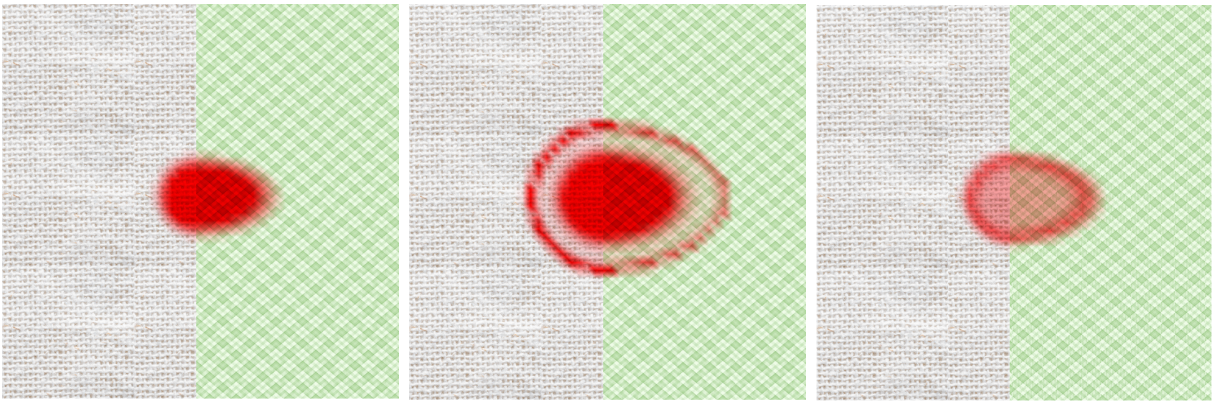


Figure 4.6 Anisotropic diffusion without (a) and with (b,c) the influence of absorption, adsorption, and evaporation.

derlying surface is deforming, we can still capture the translation and rotation of the local frames, as shown in Figures 4.1b, 4.1c, and 4.5. In the flag example, we also tested the influence of the initial velocity (random outgoing velocity fields to simulate a splash) has on the stain shape.

In Figure 4.6, we can also see the necessity of absorption, adsorption, and evaporation for the halo ring-like effects of staining common in certain materials . We implemented our framework and performed our tests on a Windows 7 system with Intel Core i7@2.8GHz and 12GB RAM. The cloth simulation are loaded as a dynamic sequence of meshes with

the same connectivity. While our code is not optimized for speed, the tests can run at interactive rates, as with most other 2D surface flow simulations.

4.7 Conclusion

We present an efficient framework for the simulation of the evolution of solution fluid that creates stains on textile surface. Instead of resorting to the computationally intensive direct simulation using the threads or yarns, we analyze the pattern of the textile, and use effective bulk diffusion tensors on the surface to replace the actually highly inhomogeneous material. Our system handles the resulting anisotropic diffusion process with a simple modification to the (Bochner) connection Laplacian. The one-way coupling form the mesh motion to the fluid is modeled by fitting the one-ring neighborhoods to create comoving and corotating frames with inertia forces (including acceleration of the frame and the Coriolis effects of the rotation of the frame). We also examine the absorption, adsorption, and evaporation processes, and model them as independent ODEs for each vertex. The resulting stain simulation is visually plausible.

In future, we wish to explore multi-layered textile model for diffusion, the effects of possible chemical reaction, two-way coupling, wetting, possible use of dynamical texture for adding back high frequencies taken out during homogenization, and learning parameters from real stains on textile.

CHAPTER 5

SURFACE LAGRANGIAN COHERENT STRUCTURE

5.1 Introduction

Due to the inherent complexity of fluid dynamics, the overall motion of fluid flow are difficult to analyze, predict, or control—even in flat 2 dimensional space. The trajectory of individual fluid particle or parcel is sensitive to initial conditions, noise in data, and error accumulated as simulation proceeds. In fact, these trajectories are often chaotic, i.e., a small perturbation in the initial positions of a fluid particle can lead to vastly different end positions in a finite amount of time. The concept of Lagrangian Coherent Structures (LCSs) was proposed to describe the transport of particles within and between different regions, which avoids the need to determine accurate locations of individual particles [65]. LCS describes regions that are coherent, or equivalently it describes the boundary between such regions. Across such boundaries, the particle trajectories diverge quickly (see, e.g., Figure 5.1).

These structures are called Lagrangian because these regions or boundaries evolve over time, so they are not fixed regions in space. Unlike watersheds, which are fixed topographical ridges separating neighboring drainage basis, LCS moves with fluid flow over time. That is why LCS is sometimes called the skeleton of fluid motion, in analogy to the skeleton used in the character animation, which is the deformation of character skin controlled by the motion of an underlying articulated rigid body. Skeletons are ubiquitously used in robot, human or animal motion. With LCS serving as a skeleton, the qualitative motion of the fluid in a domain involving the movement of these regions can be visualized and analyzed. Since fluid particles do not cross the boundaries of these regions, salient features of the coherent sets of particle trajectories can be identified and the overall flow geometry becomes predictable.

While the mathematical conditions for a point to be on LCS are well-defined, such ridges

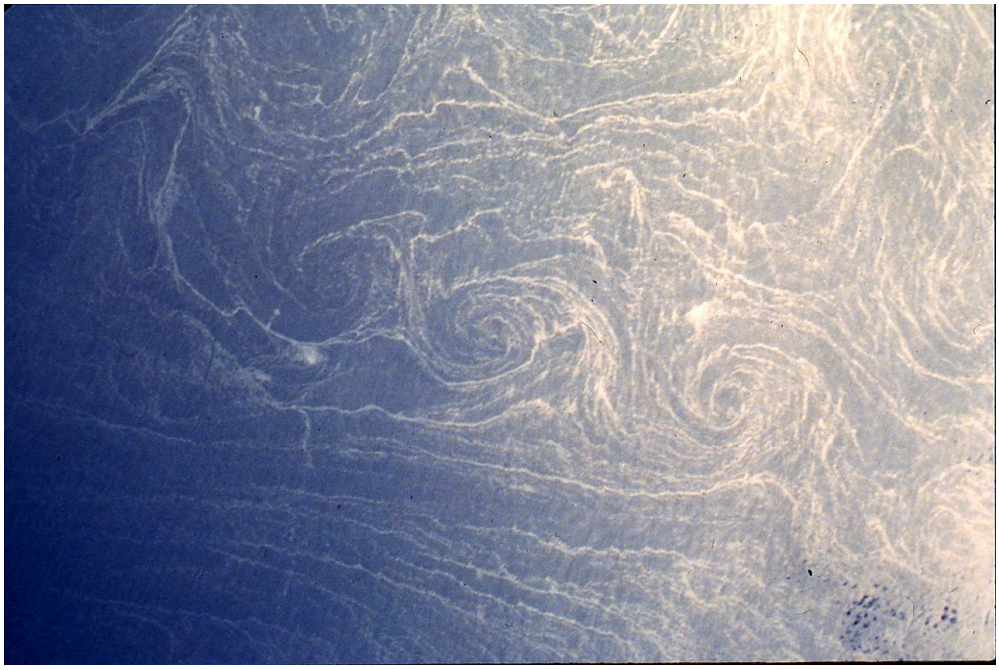


Figure 5.1 Spiral eddies containing both hyperbolic and elliptic Lagrangian Coherent Structures. Image by Paul Scully-Power/NASA

are actually visualized often through simple thresholding of the underlying scalar field. This simple thresholding results in noisy sets of grid points, which indicate the region containing the ridge, but not a curve separating fluid particles nearby into two disjoint sets. Direct ridge detection using image processing also leads to noisy output. Moreover, these methods do not extend to curved surface meshes straightforwardly.

We propose a novel robust framework to extract LCS, which works both in flat 2D and curved or even deforming surfaces. If our LCS extraction is used in conjunction with LCS-preserving fluid animation software, fine simulation can be made to have the same LCS as a coarse preview, enabling fast animation adjustment necessary for efficient artistic guidance.

5.2 Mathematical Background

Given a dynamical velocity field $\mathbf{v}(\mathbf{x}, t)$ that depends on spatial location $\mathbf{x}$ and time t, the flow map $\phi_{t_0}^t : D \to D$, where D is the domain for fluid motion, is defined as through the

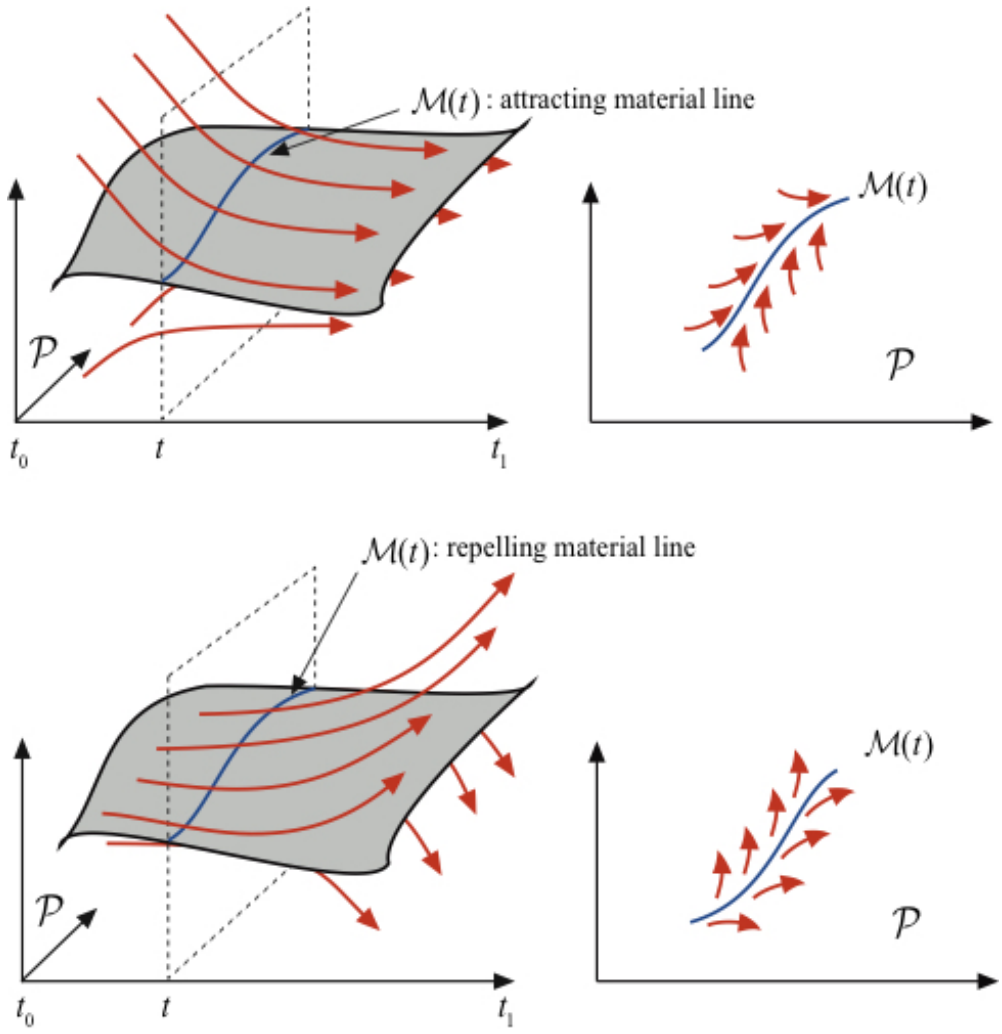


Figure 5.2 Attracting and repelling LCSs in space-time (left), and one time slice (right). Image by George Haller.

solution of the ordinary differential equation:

$$\frac{d\phi_{t_0}^t(\mathbf{x})}{dt} = \mathbf{v}(\phi_{t_0}^t(\mathbf{x}), t).$$

Given the flow map, a layer of fluid particles M_0 (a surface in 3D fluid or a curve in 2D fluid) advected by the fluid motion can be computed as $M(t) = \phi_{t_0}^t(M_0)$, which is called a material surface. Lagrangian coherent structures can be seen as exceptional material surfaces inducing strong coherent patterns, such as attracting towards the surface or repelling from the surface. See Figure 5.2.

Other types of LCSs also exist, such as elliptic LCSs, which form the boundary of

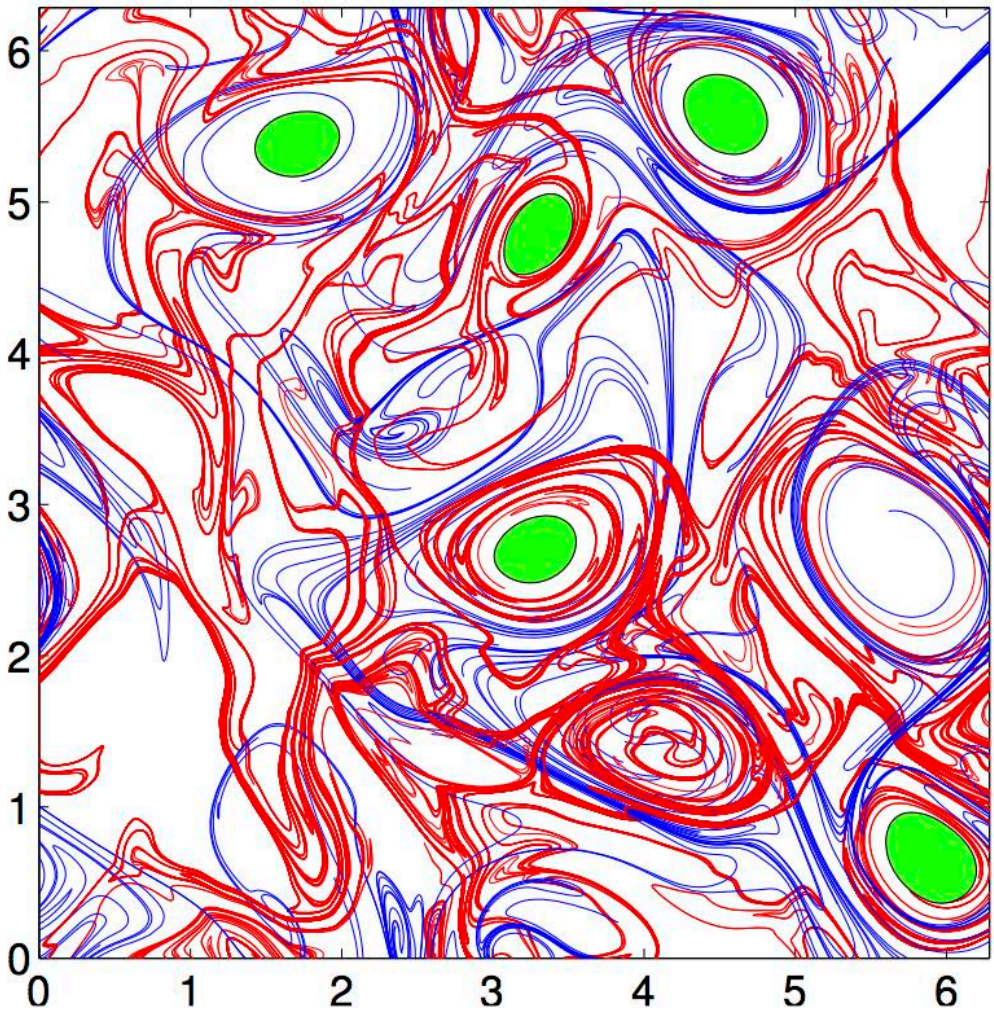


Figure 5.3 Hyperbolic LCS (attracting red curves and repelling blue curves) and elliptic LCS (green region boundaries) in a 2D turbulence simulation. Image by Mohammad Farazmand

vortices, i.e, rotation-dominated regions with homogeneous material rotations[66]. See Figure 5.3. Even for hyperbolic (attracting/repelling) and shearing LCSs, different definitions exist. A thorough survey of various definitions is beyond the scope of this dissertation. Interested readers can refer to [65].

Here, we focus on the definition of LCS based on Finite-Time Lyapunov Exponent (FTLE) ridges. The intuition is based on computing the stretching of a small fluid parcel over a fixed time window $[t_0, t_0 + T]$ with a span of T. the maximum stretching induced by

the flow map is the maximum eigenvalue $\lambda_{max,T}$ of the Cauchy-Green deformation tensor

$$C = (\nabla \phi_{t_0}^{t_0+T})^\top \nabla \phi_{t_0}^{t_0+T}.$$

The above formula can be derived from the fact that a small perturbation δ in $M(t_0)$ is mapped to a perturbation $\nabla \phi_{t_0}^{t_0+T} \delta$ in $M(t_0+T)$. The latter perturbation has the length $\sqrt{|\nabla \phi_{t_0}^{t_0+T} \delta|} = \sqrt{\delta^\top C \delta}$, which means the length ratio $\sqrt{\delta^\top C \delta}/\sqrt{\delta^\top \delta}$ is has a maximum $\lambda_{max,T}$.

In a static velocity field, λ_{max} grows approximately exponentially with T. The limit $\sigma_\infty = \lim_{T\to\infty} \frac{1}{T} \ln \sqrt{\lambda_{max,T}}$ is called the Lyapunov exponent. For a time dependent velocity field, such a number is only useful for a finite time window T. Thus, the finite time Lyapunov exponent is defined as

$$\sigma = \frac{1}{T} \ln \sqrt{\lambda_{max,T}}.$$

In time-independent velocity fields, separatrices between coherent regions are given by the stable and unstable manifolds of hyperbolic singularities, and they are typically the ridges in the Lyapunov fields. Extending to the time-dependent case, the ridges of the FTLE field σ are often used as one definition of LCS, as the point on such ridges reaches the local maximum stretching in any transverse directions of the ridges. An injective curve $c : I \to D$, where $I = (a, b) \subset \mathbb{R}$ is an interval, is a ridge whenever the following two conditions are satisfied:

$$\frac{dc(s)}{ds} \times \nabla \sigma(c(s)) = 0, \tag{5.1}$$

and

$$n^\top H n = \min_{|u|=1} u^\top H u, \tag{5.2}$$

where n is a unit normal to the curve $c(s)$, and $H = \nabla \nabla^\top \sigma$ is the Hessian of FTLE σ. Intuitively, the first condition means that the ridge should be parallel to the gradient direction right along the edge, and the second condition means that across the ridge, height drops rapidly on both sides.

This LCS is, however, not technically a material surface, since sliding the time window by changing t_0 typically does not lead to ridges advected from ridges from previous time windows[67]. However, it is shown that the material flux over such ridges is small[68]. We follow this commonly used definition of LCS in our approach for its simplicity and efficiency.

5.3 Related Work

There are several different existing approaches to extraction of LCSs. As mentioned above, even the mathematical definitions in these methods may differ significantly. We only review often used categories of these LCS definitions and extraction algorithms, with a focus on those closely related to our approach.

5.3.1 FTLE Ridges

One of the most highly cited methods is to extract ridges of finite-time Lyapunov exponent (FTLE) fields[68, 69]. As described in the previous section, an FTLE field is a scalar field that characterizes the amount of stretching experienced by a fluid parcel along the trajectory determined by the fluid flow over a given time interval. Such ridges are similar to watersheds separating flows to different eventual locations (drainage basins).

In analogy to image segmentation in computer vision, extraction methods of LCS defined as such ridges are similar to edge detection rather than clustering of coherent regions bounded by them. On the other hand, for time-dependent velocity fields, FTLE ridges do not necessarily form closed loops, so fluid particles on different sides of the ridges can actually mix, but they do not mix in the neighborhood across the ridges. Thus, LCSs based on FTLE ridges are more flexible than those based on regions. See, e.g., the open ended ridges in Figure 5.4.

The visualization in Figure 5.4 is a rendering of the graph of 2D FTLE as a terrain map, with FTLE serving as the elevation [70]. Similar LCS visualization was also used in

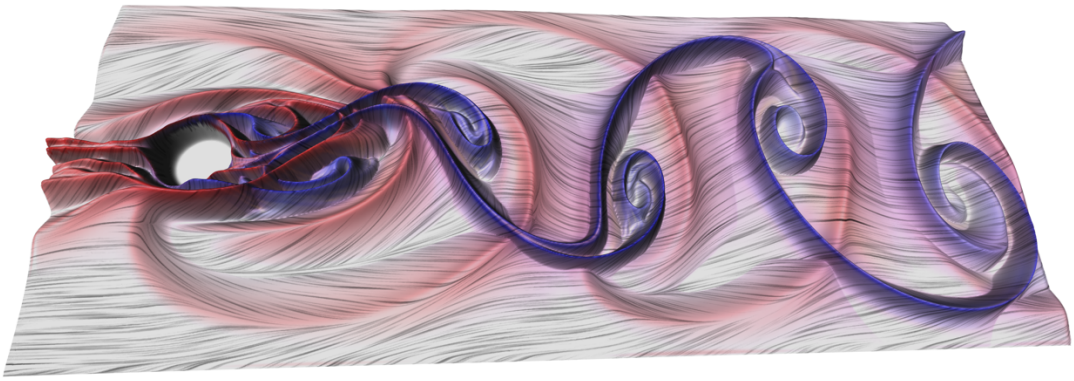


Figure 5.4 Attracting (red) and repelling (blue) LCSs visualized through height field with FTLE as the height for a von Karman vortex street. Image by Jens Kasten.

computer vision for crowd flows such as the pilgrims in Mecca[71]. The vision paper also proposed a pairwise measurement similar to FTLE, which is called the Lyapunov divergence, to segment the flow regions.

We differ from these existing FTLE ridge extraction algorithms in that we try to find a thin curve along the true FTLE ridges instead of the pixelated discrete set description through thresholding FTLE. We also want our method to be as robust as the marching cubes algorithm in level set extraction[72].

For this purpose, we adapted the Marching Ridges algorithm[73]. Our algorithm differs from the original Marching Ridges method in a number of ways. First, we adapted it for triangle meshes instead of regular grid so that we can apply it to surface meshes. Second, we have an estimate of the transverse direction to the ridges following the definition in [69], instead of the estimate based on local sampling density functions. Last, we created an improved edge intersection detection criterion, that precisely mimics the marching cubes algorithm in regions with coherent transverse direction, while allowing open curves to be formed.

The definition of ridge structures of scalar fields, whether used for FTLE fields in LCS or other scalar fields. For example, two of the ridge definitions by Eberly and Lindeberg

can be altered to narrow down the set of raw feature points [74]. In such a process, the explicit calculation of eigenvalues and eigenvectors of the Hessian of the scalar fields can be avoided to further enhance the efficiency.

One major drawback of using FTLE to extract LCS is the low performance due to the need for computation of a large number of sample flow trajectories in space-time to obtain the Cauchy-Green tensor. Sadlo et al. [75] proposed to alleviate the problem by searching for LCS only in predefined regions such as the boundaries, related to flow attachment and flow separation.

Although LCS is mostly computed in flat 2D domains represented by time-dependent boundary curves, extensions of the method were proposed to calculate LCSs in 3D volumes, where they are described by evolving 2D surfaces. Sulman et al. [76] presented a reduced LCS form in 3d volume to improve accuracy for 2D LCS surfaces. Raben et al. [77] further improved the reliability and accuracy from noisy image data by using pathline flow map (PFM) and flow map compilation(FMC).

5.3.2 Other LCS Definitions

Even if we restrict our discussion to hyperbolic LCSs, there are various definitions other than FTLE ridges. Most notably the definition based on stretch (and strain) lines, which in 2D are along the maximum (and minimum, respectively) eigenvector directions of the above mentioned Cauchy-Green tensors, which encode the pointwise length distortion in every direction. Starting form seeds located of local maximum and minimum points, one can solve ordinary differential equations by following those eigenvector directions to find such LCSs. This analytic criterion can be applied to high precision data sets for find precise LCSs [67]. Sadlo and Peikert [78] have employed adaptive mesh refinements on variants of the definition and sped up the process by avoiding seeding regions with no ridges. However, even with analytic expressions for velocity fields, the Cauchy-Green tensors following rank-4 Runge-Kutta (RK4) integration can still be noisy. Seeding on a uniform grid is often

necessary. In comparison, FTLE ridges are less rigorous definitions of LCSs in terms of material surfaces, but they can be found more efficiently without seeding for noisy input data. Thus, we follow the FTLE ridges definition in our calculation.

5.4 Overview

In this section, we provide an overview of our pipeline in producing ridges for a scalar field stored on a triangle mesh. Our rationale behind our design is that the scalar field is as noisy as an FTLE field, and that it usually contain ridge structures. We provide the implementation details and the reasons for the design decision in the next section.

Our framework assume that FTLE is given as an input computed on a triangulated domain for a certain time t_0 over a period of time T, which can be positive or negative. Such an input is typically noisy even when the time-dependent velocity field is smooth.

We first apply a thresholding of FTLE as typically done in visualization of LCS as FTLE ridges[69]. The thin set of vertices with FTLE above the threshold is indicated by a binary scalar field S on the triangle mesh,

$$S(v) = \sigma(v) > \sigma_0 ?\ 1000 :\ 0, \tag{5.3}$$

for each vertex v.

We then smooth the scalar field S through Laplacian smoothing by implicit integration of the diffusion equation identical to the approach described in Chapter 4. We treat the regions with $S > S_0$, where S_0 is a threshold as the candidate locations for detecting a ridge structure.

Next, we evaluate the Hessian of the S,

$$H = \nabla\nabla^\top S.$$

We again smooth the resulting H field before we extract the two eigenvectors n_{max} and n_{min} of this symmetric matrix associated with the maximum eigenvalue λ_{max} and the minimum

eigenvalue λ_{min}. We extract the ridge by finding the zero crossings of $\nabla\sigma \cdot n_{min}$ on each edge, since the minimum eigenvalues λ_{min} is typically less than 0, in regions satisfying $S > S_0$, and n_{min} is transverse to the ridge direction. Note that $\pm n_{min}$ are both eigenvectors of H, so we flip one of the n_{min} if the two minimum eigenvectors have a negative dot product before checking for zero crossing.

Once the zero crossing are found, we proceed with an improved version of ridge marching (similar to marching cubes) that is straightforward on triangle meshes, i.e., connect the possible two or three intersection points on the three edges of each triangle containing edges with zero crossings.

Finally, we perform a clean-up on the resulting ridge structures to produce an LCS output that can be used in downstream applications, such as visualization or LCS-preserving animations[79].

5.5 Implementation Details

5.5.1 FTLE Evaluation

While our main task is to extract LCS from input FTLE, we need to prepare the data ourselves since we are using it on triangle meshes. As done in Chapter 4, we trace the particle located at a vertex at time t by an amount of time T either forward ($T > 0$) or backward ($T < 0$). For simplicity, we trace the vertex first inside the one-ring flattened through geodesic polar map if it is not flat along the velocity direction to determine which triangle the particle will first enter. When tracking inside each triangle, we use a constant velocity formed by averaging the velocity stored at the vertex and projecting to the tangent plane containing the triangle. When crossing an edge to an adjacent triangle, we switch to a new velocity. It is possible to use Runge-Kutta or simple mid-point rules to get a better trajectory, but either way, the produced FTLE fields are still typically noisy.

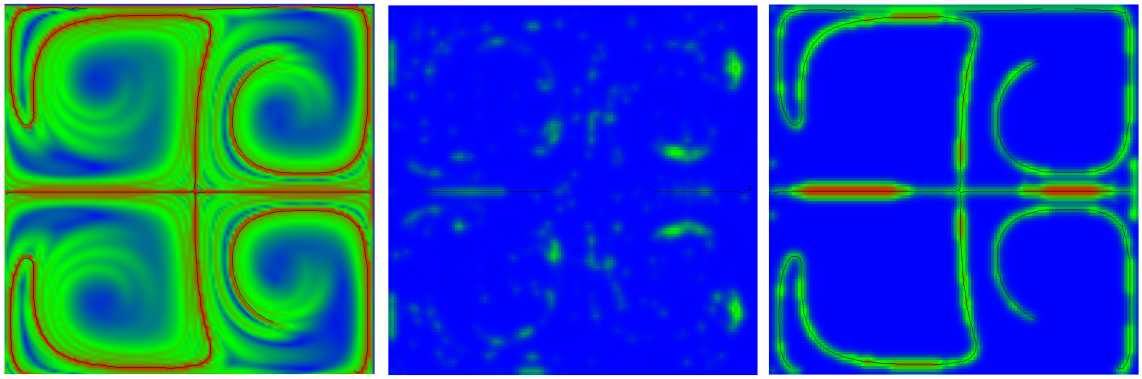


Figure 5.5 From left to right: FTLE calculated from the double gyre velocity field, S field computed by second derivative-based ridge conditions, and S computed based on thresholding followed by smoothing.

5.5.2 Candidate Ridge Regions

FTLE can be optionally smoothed through an implicit integration of the diffusion equation as in Chapter 4. We then use Eq. 5.3 to find potential ridge regions. We compute the threshold σ_0 based on the maximum and minimum simply as $\sigma_0 = 0.75\sigma_{max} + 0.25\sigma_{min}$. A better threshold can be automatically selected based on the histogram as often used in computer vision.

Note that using the second derivative-based ridge conditions directly on the noisy data on a mesh does not work, probably due to the matching of directions cannot be achieved precisely on any vertices. To test the direct use of the two ridge conditions, we reformulated Eqs. 5.1 and 5.2 into the following equivalent conditions for the ridge region indicator function S:

$$S = \lambda_{min} < 0 ?\ 1/(\epsilon^2 + \frac{\nabla\sigma}{|\nabla\sigma|} \cdot n_{min}) \ :\ 0,$$

where ϵ is a small number to prevent division by 0, λ_{min} is the minimum eigenvalue of Hessian of FTLE field σ and n_{min} is the associated eigenvector. As seen in Figure. 5.5, given the same FTLE field, the simple thresholding yields much better indicator S field after smoothing.

5.5.3 Marching Triangle Ridge Extraction

We actually believe the second ridge condition given in [69] are overly strong. Since ridge point is a local maximum across the ridge, so the first derivative along the direction normal n to the ridge should be 0. Thus, the first ridge condition for the gradient to be parallel to tangential to the ridge is reasonable, and we can formulate it equivalently as $\nabla\sigma \cdot n = 0$. However, the second derivative along n only needs to be negative, i.e., $n^\top \nabla\nabla^\top \sigma n < 0$, and there is no need for n to align to the eigenvector associated with the smallest eigenvalue of $\nabla\nabla^\top\sigma$.

Thus, we propose to estimate the transverse direction from eigenvectors of Hessian of the indicator function $H = \nabla\nabla^\top S$ instead of that of Hessian of σ. Assuming the FTLE contains strong stretching or shearing LCS structures, the ridges of FTLE tend to be sharp and the thresholding can produce an S with similar but smooth values for points near the ridges, which means its first derivatives tend to be similar along the ridge. This behavior of the first derivatives forces the second derivatives to have an eigenvalue across the ridge direction to be negative with a large absolute value. Thus we assume n_{min} of H to represent the normal direction to the ridge.

Next, we flag the candidate edge (between vertices v_i and v_j) that contains an intersection point with the ridge based on the following condition:

$$(n_i \cdot (\nabla\sigma)_i)(n_j \cdot (\nabla\sigma)_j)(n_i \cdot n_j) < 0, \tag{5.4}$$

where the third factor is checking the consistency between the choice on the eigenvector orientations at the two end vertices. The intersection point is marked to be location

$$\frac{(n_j \cdot (\nabla\sigma)_j)v_i + (n_i \cdot (\nabla\sigma)_i)v_j}{(n_i \cdot (\nabla\sigma)_i) + (n_j \cdot (\nabla\sigma)_j)}.$$

If the normals can be made consistent in a region, we are indeed extracting the 0 level set of $n \cdot \nabla\sigma$.

Our estimate of the normal direction is smoother than the transverse direction proposed in the marching ridge algorithm[73], which essentially uses the covariance matrix of Voronoi

regions around sample points on the ridge to determine the normal direction. We also differ from their intersection point detection, where they used $(\nabla\sigma)_i^\top \; nn^\top \; (\nabla\sigma)_j < 0$, with $nn^\top$ being the average of $n_i n_i^\top$ and $n_j n_j^\top$, which is less like level set in our test for the noisy FTLE fields.

We also note that true level set finding algorithms do not work for ridge detection, since level set can only find closed curves or curves that end on the boundary of the computational domain. In fact, we tried the algorithm in [80], which uses a variational approach to minimize the oscillation of the level set and the deviation from the sample points while maximizing the alignment to the estimated normal direction. We found that the level set finding algorithm always forces the ridges to deviate from the open curve in the attempt to form closed loops, which is equivalent to finding a globally consistent orientation to the unsigned normal field. In contrast, our method only needs to check the local consistency between the normals.

Once the intersection points are dumped. We "march" through each triangle as in marching cubes to store the ridges as a graph with the intersection points as its nodes. If we have only one candidate edge, no edge (a pair of intersection point indices) is produced. If we have two candidate edges, one edge is added. If there are three candidate edges, (which cannot happen in level set method) we produce two edges with the largest angle in between, among the three possible edges.

We also tried to use ∇S instead of $\nabla\sigma$ in the estimate of ridge field, and found that using the FTLE gradient produces more accurate results. See Figure. 5.6.

Optionally, we can force local maxima of FTLE to appear on the ridges we detect. This way, three or more ridges may join at a local maximum. In order to do that, we output the local maximum vertices along with the intersection points. Then, we shortcut the intersection points on the boundary of the one ring of the local maximum vertex to connect to that vertex instead of the intersection points on the edges incident to the local maximum vertex. As a postprocessing, we shift the location of these local maxima to the average of their neighbors in the graph.

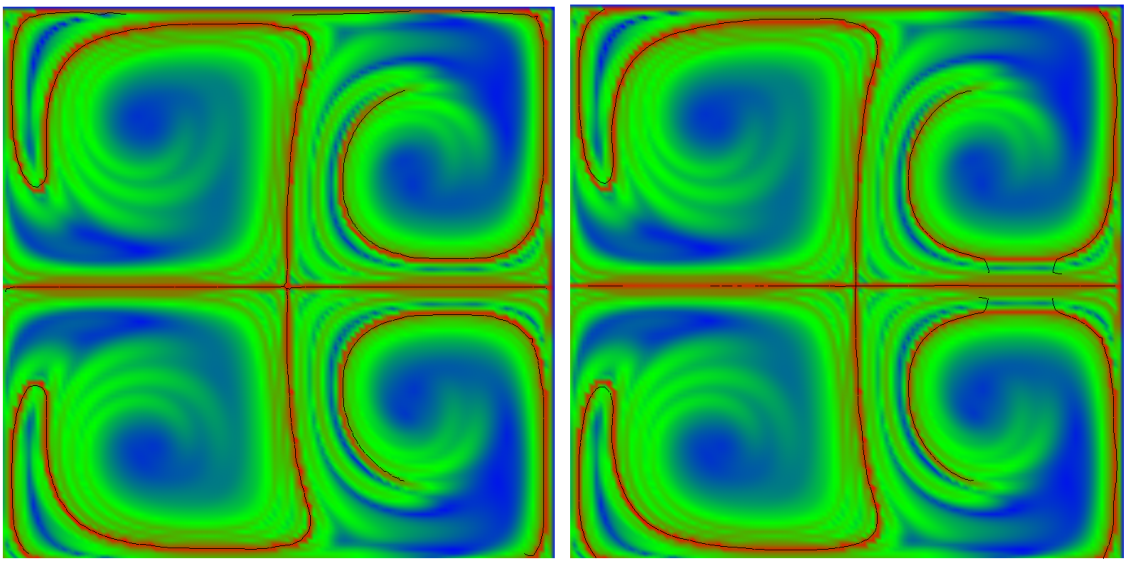


Figure 5.6 Left: ridges (black curves) generated from the gradient of the FTLE field. Right: ridges generated from the gradient of the indicator function S.

5.5.4 Cleanup of Noisy Ridges

With noisy input data, even after smoothing, there are artifacts in the detected ridge edges. We employ a simple strategy to remove these artifacts.

We first use union-find on the ridge graph to find connected components, and remove any connected components containing fewer than a preselected number of edges (in our test, we use 20 as the threshold).

Next, we find T-junctions with one of the branches short. If there are two short curves connecting the same two T-junctions, we remove the larger curve. We also remove short loops connecting a T-junction to itself.

See Figure. 5.7 for a comparison before and after cleanup.

5.5.5 Spurious Valleys

While thresholding σ restricts us to regions most likely to contain ridges instead of valley. With the noisy input, valleys are occasionally detected by Eq. 5.4. We provide an option to remove these valleys by checking the following conditions: if the normals are making a

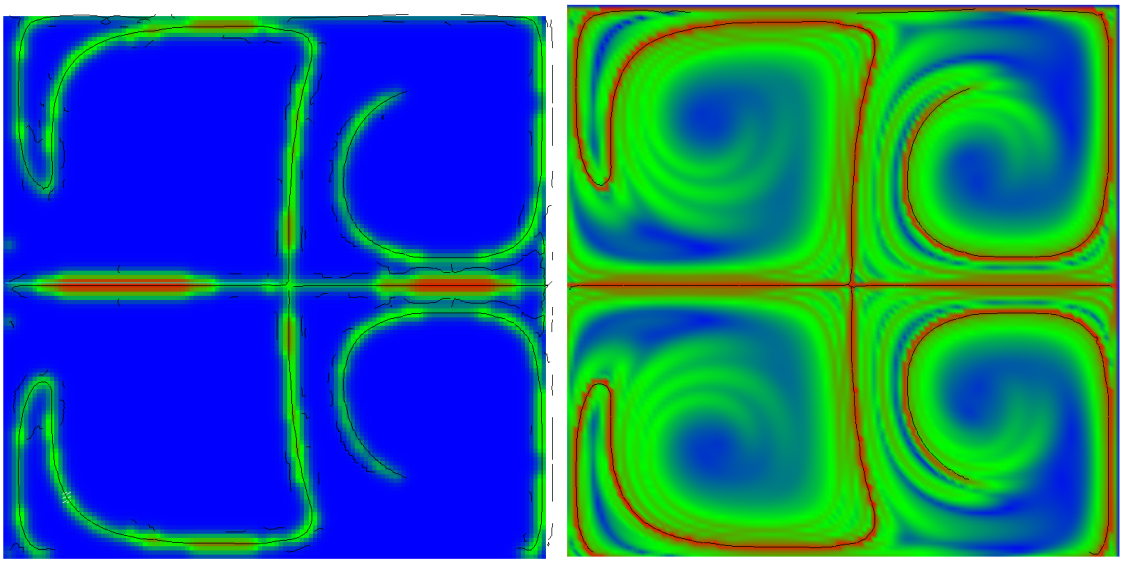


Figure 5.7 Left: before cleanup based on the graph structure. Right: after the cleanup.

small angle with the edge itself, we ask the projections of the gradients along the normal direction to point towards the interior of the edge.

$$((\nabla\sigma)_i \cdot n_i)(n_i \cdot e_{ij}) > 0.$$

Another optional criterion we use for intersection candidate edge is to see whether H has two similar negative eigenvalues, which indicates the vicinity of a local maximum, and we try both eigenvectors of H as n to find potential intersections.

5.6 Numerical Experimental Results

Our first simple test data is the GLAS-II airfoil data from [69]. With the 50x25 grid size, we can already detect the LCS ridge, with the FTLE evaluated from $t = 8$ to $t = 7$. See Figure. 5.8.The color indicates FTLE value.

Next, we tested our algorithm on the double gyre [69], described by the velocity field

$$u = -\pi A \sin(\pi f(x)) \cos(\pi y),$$

$$v = \pi A \cos(\pi f(x)) \sin(\pi y) \frac{df}{dx},$$

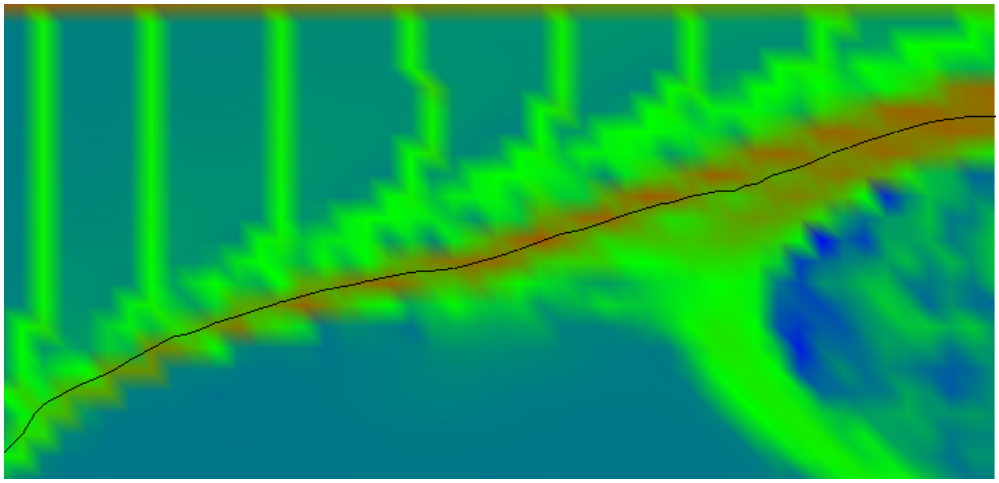


Figure 5.8 Air flow above an airfoil forms a vortex behind the LCS ridge line.

where $f(x,t) = a(t)x^2 + b(t)x$, $a(t) = \epsilon \sin(\omega t)$, $b(t) = 1 - 2\epsilon \sin(\omega t)$.

Over the domain $[0,2] \times [0,1]$, two gyres form. Figure. 5.9 shows our results for $A = 0.1, \omega = 2\pi/10, \epsilon = 0.25$.

Extending the domain to $[0,2] \times [-1,1]$, we can find four gyres. See Figure. 5.10.

We tested also on the Bickley jet, fequently used as a model of zonal jets in the Earth's atmosphere[81]. See Figure. 5.11

Since we did not have standard test data for LCS on surfaces, we created the octuple gyre by mirror imaging of quad gyre from left to right to form a periodic velocity field and mapped it onto a torus. We ran our algorithm on the torus mesh to produce the LCS in Figure. 5.12

5.7 Conclusion and Future Work

The existing methods for LCS curve extraction even in flat 2D domains are often based on discrete samples or seeds, akin to highlighted pixels in images based on FTLE fields. While it is possible to fit curves to these detected ridge pixels, the resulting curves are not robust and may contain duplicates. We propose to use an improved marching ridges algorithm that is applicable to triangle mesh to extract local level set-like clean ridge structures.

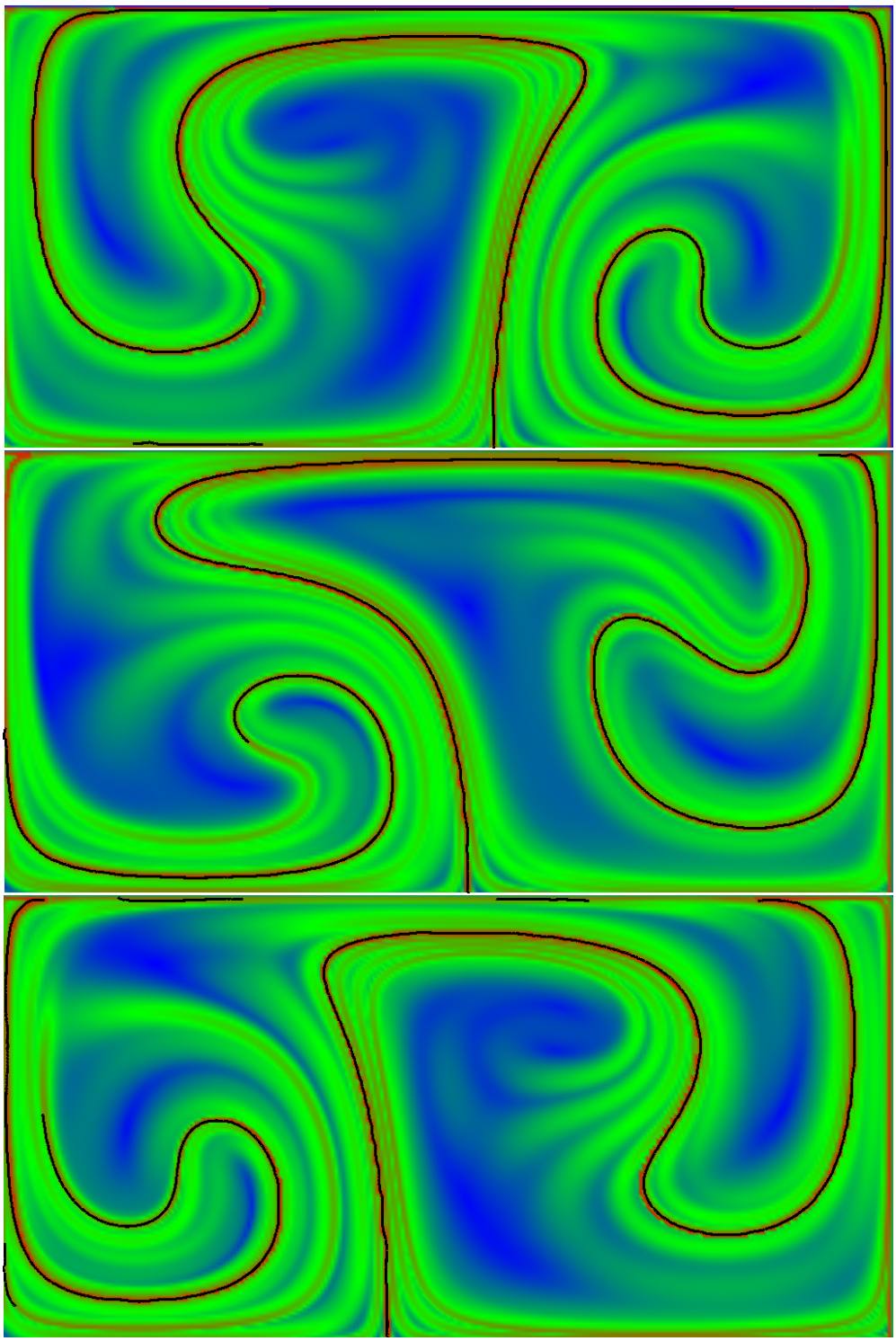


Figure 5.9 LCS for double gyre flow at various starting times.

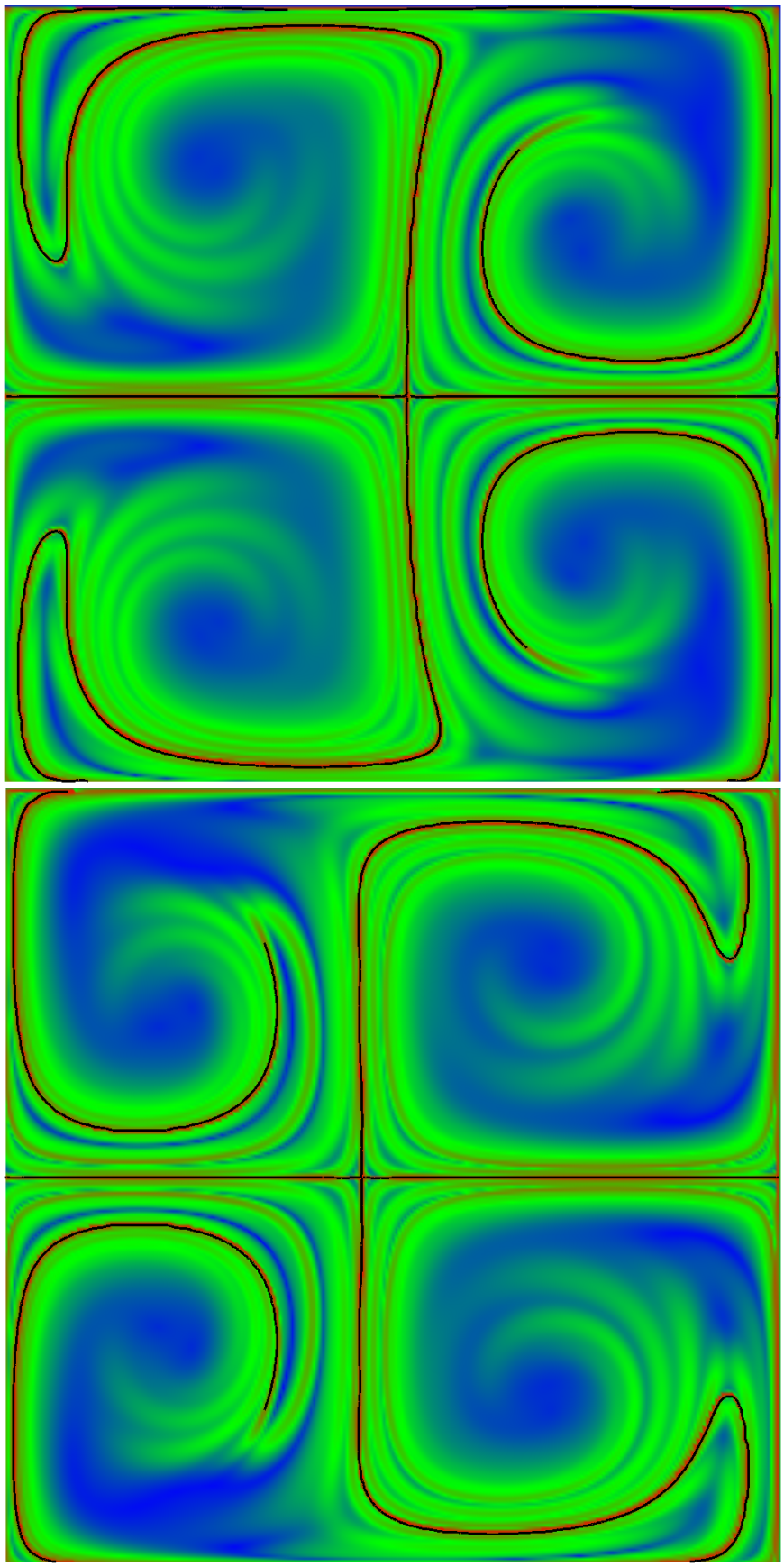


Figure 5.10 LCS for quad gyre flow at various starting times.

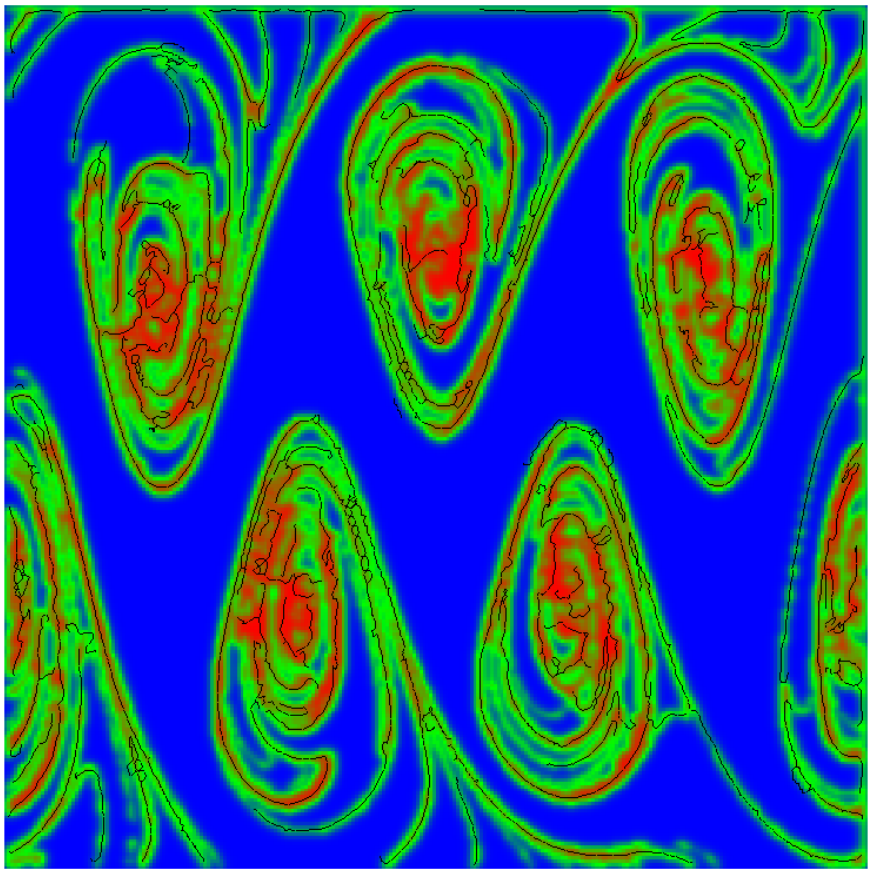


Figure 5.11 LCS on the Bickley jet.

Our test results are promising, but are limited to the often used LCS illustration examples. We leave further evaluation of our method on real-time incompressible surface flow on deforming surfaces as future work. Other applications of our ridge detection algorithm is also worth exploring.

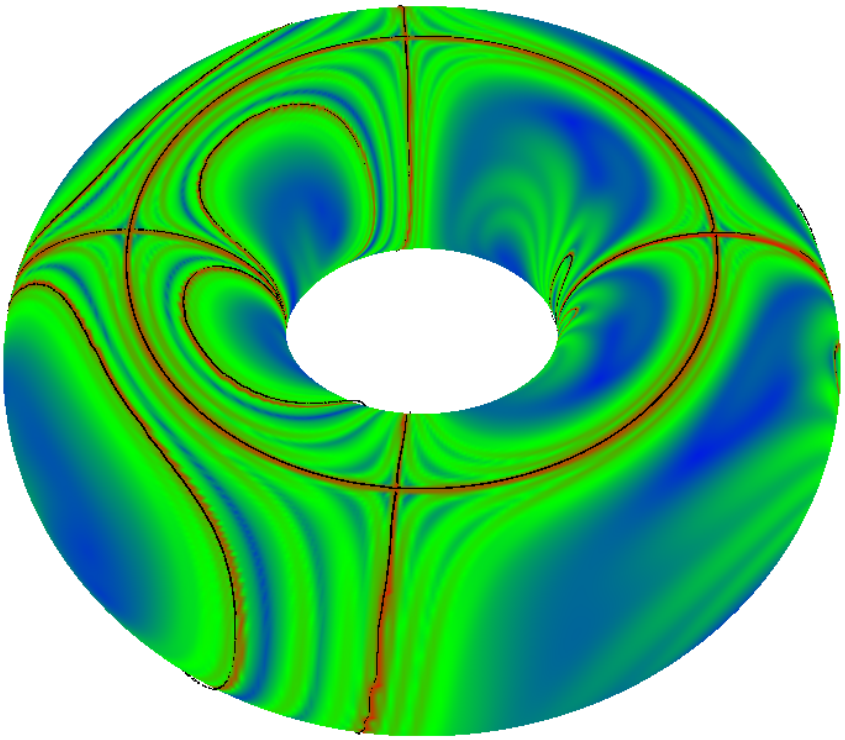


Figure 5.12 LCS on the torus octuple gyre.

CHAPTER 6

CONCLUSION

In this dissertation, we provided a collection of computational tools for flow simulation on possibly moving and deforming surfaces. These tools are shown to be effective for various tasks in the context of surface fluid simulation. However, they can potentially be applied in other related domains.

For instance, the homogenization for textile materials can potentially be useful for 3D printing of thin shell objects for fast evaluation of printed inhomogeneous micro-structures. The stain simulation may be extended for efficient atmospheric simulations. The angle-based representations can be used in elasticity or shape analysis. The LCS extraction algorithm can essentially be directly used where ridges of scalar fields are needed, including shape features used in shape analysis.

Various parts of these tools have plenty of room for improvement and extensions. We believe artistic guidance for surface fluid simulation is worth exploring. Extending ridge features of codimension-1 for 3D domains can also be promising by switching from marching triangles to marching tetrahedra.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] A. M. Jaffe, "The millennium grand challenge in mathematics," *Notices of the AMS*, vol. 53, no. 6, 2006.

[2] K. Crane, I. Llamas, and S. Tariq, "Real-time simulation and rendering of 3d fluids," *GPU gems*, vol. 3, no. 1, 2007.

[3] Y.-Q. Xu, Y. Chen, S. Lin, H. Zhong, E. Wu, B. Guo, and H.-Y. Shum, "Photorealistic rendering of knitwear using the lumislice," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 391–398, ACM, 2001.

[4] A. Bensoussan, J.-L. Lions, and G. Papanicolau, *Asymptotic analysis for periodic structures*. Elsevier, 1978.

[5] V. V. Jikov, O. Oleinik, and S. M. Kozlov, *Homogenization of differential operators and integral functionals*. Springer, 1994.

[6] H. Owhadi and L. Zhang, "Metric-based upscaling," *Communications on Pure and Applied Mathematics*, vol. 60, no. 5, pp. 675–723, 2007.

[7] L. Kharevych, P. Mullen, H. Owhadi, and M. Desbrun, "Numerical coarsening of inhomogeneous elastic materials," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 51, ACM, 2009.

[8] K. Muller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand, "Predictive compression of dynamic 3d meshes," in *IEEE International Conference on Image Processing 2005*, vol. 1, pp. I–621, IEEE, 2005.

[9] H. Lee, P. Alliez, and M. Desbrun, "Angle-analyzer: A triangle-quad mesh codec," in *Computer Graphics Forum*, vol. 21, pp. 383–392, Wiley Online Library, 2002.

[10] G.-P. Paillé, N. Ray, P. Poulin, A. Sheffer, and B. Lévy, "Dihedral angle-based maps of tetrahedral meshes," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 54, 2015.

[11] J. Stam, "A simple fluid solver based on the fft," *J. Graph. Tools*, vol. 6, pp. 43–52, Sept. 2002.

[12] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pp. 421–430, ACM Press/Addison-Wesley Publishing Co., 1997.

[13] T. L. Kunii, G. V. Nosovskij, and T. Hayashi, "A diffusion model for computer animation of diffuse ink painting," in *Computer Animation'95., Proceedings.*, pp. 98–102, IEEE, 1995.

[14] T. L. Kunii, G. V. Nosovskij, and V. L. Vecherinin, "Two-dimensional diffusion model for diffuse ink painting," *International Journal of Shape Modeling*, vol. 7, no. 01, pp. 45–58, 2001.

[15] N. S.-H. Chu and C.-L. Tai, "Moxi: real-time ink dispersion in absorbent paper," in *ACM Transactions on Graphics (SIGGRAPH)*, vol. 24, pp. 504–511, ACM, 2005.

[16] T. Van Laerhoven, J. Liesenborgs, and F. Van Reeth, "Real-time watercolor painting on a distributed paper model," in *Computer Graphics International, 2004. Proceedings*, pp. 640–643, IEEE, 2004.

[17] T. Van Laerhoven and F. Van Reeth, "Real-time simulation of watery paint," *Computer Animation and Virtual Worlds*, vol. 16, no. 3-4, pp. 429–439, 2005.

[18] H. T. Wong and H. H. Ip, "Virtual brush: a model-based synthesis of chinese calligraphy," *Computers & Graphics*, vol. 24, no. 1, pp. 99–113, 2000.

[19] J. Lee, "Diffusion rendering of black ink paintings using new paper and ink models," *Computers & Graphics*, vol. 25, no. 2, pp. 295–308, 2001.

[20] D.-L. Way and Z.-C. Shih, "The synthesis of rock textures in chinese landscape painting," in *Computer Graphics Forum*, vol. 20, pp. 123–131, Wiley Online Library, 2001.

[21] Y. J. Yu, Y. B. Lee, H. G. Cho, and D. H. Lee, "A model based technique for realistic oriental painting," in *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pp. 452–453, IEEE, 2002.

[22] S.-W. Huang, D.-L. Way, and Z.-C. Shih, "Physical-based model of ink diffusion in chinese paintings," 2003.

[23] Y. Morimoto, M. Tanaka, R. Tsuruno, and K. Tomimatsu, "Visualization of dyeing based on diffusion and adsorption theories," in *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pp. 57–64, IEEE, 2007.

[24] S. Liu, G. Chen, P. Yang, J. Zhang, and J. Sun, "Realistic simulation of stains on cloth," *Journal of CAD&CG*, vol. 20, no. 9, pp. 1110–1116, 2008.

[25] J. Stam, "Flows on surfaces of arbitrary topology," *ACM Transactions On Graphics (TOG)*, vol. 22, no. 3, pp. 724–731, 2003.

[26] L. Shi and Y. Yu, "Inviscid and incompressible fluid simulation on triangle meshes," *Computer Animation and Virtual Worlds*, vol. 15, no. 3-4, pp. 173–181, 2004.

[27] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, circulation-preserving, simplicial fluids," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 1, p. 4, 2007.

[28] O. Azencot, S. Weißmann, M. Ovsjanikov, M. Wardetzky, and M. Ben-Chen, "Functional fluids on surfaces," in *Computer Graphics Forum*, vol. 33, pp. 237–246, Wiley Online Library, 2014.

[29] B. Liu, G. Mason, J. Hodgson, Y. Tong, and M. Desbrun, "Model-reduced variational fluid simulation," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 244, 2015.

[30] S. Auer, C. B. Macdonald, M. Treib, J. Schneider, and R. Westermann, "Real-time fluid effects on surfaces using the closest point method," in *Computer Graphics Forum*, vol. 31, pp. 1909–1923, Wiley Online Library, 2012.

[31] H. Wang, G. Miller, and G. Turk, "Solving general shallow wave equations on surfaces," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 229–238, Eurographics Association, 2007.

[32] H. Wang, P. J. Mucha, and G. Turk, "Water drops on surfaces," in *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 921–929, ACM, 2005.

[33] Y. Zhang, H. Wang, S. Wang, Y. Tong, and K. Zhou, "A deformable surface model for real-time water drop animation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 8, pp. 1281–1289, 2012.

[34] Y. Jung and J. Behr, "Gpu-based real-time on-surface droplet flow in x3d," in *Proceedings of the 14th international conference on 3D web technology*, pp. 51–54, ACM, 2009.

[35] K. Djado, R. Egli, and F. Granger, "Particle-based drop animation on meshes in real time," *Computer Animation and Virtual Worlds*, vol. 23, no. 3-4, pp. 301–309, 2012.

[36] R. Angst, N. Thuerey, M. Botsch, and M. Gross, "Robust and efficient wave simulations on deforming meshes," in *Computer Graphics Forum*, vol. 27, pp. 1895–1900, Wiley Online Library, 2008.

[37] P. Neill, *Fluid flow on interacting, deformable surfaces*. PhD thesis, 2008.

[38] K. Hegeman, M. Ashikhmin, H. Wang, H. Qin, X. Gu, *et al.*, "Gpu-based conformal flow on surfaces," *Communications in Information & Systems*, vol. 9, no. 2, pp. 197–212, 2009.

[39] S. Jeong and C.-H. Kim, "Combustion waves on the point set surface," in *Computer Graphics Forum*, vol. 32, pp. 225–234, Wiley Online Library, 2013.

[40] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 121–128, ACM Press/Addison-Wesley Publishing Co., 1999.

[41] D. Lukkassen, L.-E. Persson, and P. Wall, "Some engineering and mathematical aspects on the homogenization method," *Composites Engineering*, vol. 5, no. 5, pp. 519–531, 1995.

[42] S. J. Hollister and N. Kikuchi, "A comparison of homogenization and standard mechanics analyses for periodic porous composites," *Computational Mechanics*, vol. 10, no. 2, pp. 73–95, 1992.

[43] S.-H. Bae, H. Motomura, and Z. Morita, "Diffusion/adsorption behaviour of reactive dyes in cellulose," *Dyes and pigments*, vol. 34, no. 4, pp. 321–340, 1997.

[44] D. Cohen-Steiner and J.-M. Morvan, "Restricted delaunay triangulations and normal cycle," in *Proceedings of the nineteenth annual symposium on Computational geometry*, pp. 312–321, ACM, 2003.

[45] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. CRC press, 2010.

[46] M. P. Do Carmo, "Differential geometry of surfaces," in *Differential forms and applications*, pp. 77–98, Springer, 1994.

[47] M. Eigensatz, R. W. Sumner, and M. Pauly, "Curvature-domain shape processing," in *Computer Graphics Forum*, vol. 27, pp. 241–250, Wiley Online Library, 2008.

[48] S. Fröhlich and M. Botsch, "Example-driven deformations based on discrete shells," in *Computer graphics forum*, vol. 30, pp. 2246–2257, Wiley Online Library, 2011.

[49] Y. Wang, B. Liu, and Y. Tong, "Linear surface reconstruction from discrete fundamental forms on triangle meshes," in *Computer Graphics Forum*, vol. 31, pp. 2277–2287, Wiley Online Library, 2012.

[50] E. Corman, J. Solomon, M. Ben-Chen, L. Guibas, and M. Ovsjanikov, "Functional characterization of intrinsic and extrinsic geometry," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 2, p. 14, 2017.

[51] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[52] F. d. Goes, P. Memari, P. Mullen, and M. Desbrun, "Weighted triangulations for geometry processing," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 3, p. 28, 2014.

[53] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot, "3d mesh compression: Survey, comparisons, and emerging trends," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 44, 2015.

[54] M. Deering, "Geometry compression," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 13–20, ACM, 1995.

[55] P. Mullen, Y. Tong, P. Alliez, and M. Desbrun, "Spectral conformal parameterization," in *Computer Graphics Forum*, vol. 27, pp. 1487–1494, Wiley Online Library, 2008.

[56] G. Rong, Y. Cao, and X. Guo, "Spectral mesh deformation," *The Visual Computer*, vol. 24, no. 7, pp. 787–796, 2008.

[57] H. Zhang, O. Van Kaick, and R. Dyer, "Spectral mesh processing," in *Computer graphics forum*, vol. 29, pp. 1865–1894, Wiley Online Library, 2010.

[58] B. Liu, Y. Tong, F. D. Goes, and M. Desbrun, "Discrete connection and covariant derivative for vector field analysis and design," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 3, p. 23, 2016.

[59] S. Kircher and M. Garland, "Free-form motion processing," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 2, p. 12, 2008.

[60] R. Tamstorf and E. Grinspun, "Discrete bending forces and their jacobians," *Graphical Models*, vol. 75, no. 6, pp. 362–370, 2013.

[61] H. Fu, O. Kin-Chung Au, and C.-L. Tai, "Effective derivation of similarity transformations for implicit laplacian mesh editing," in *Computer Graphics Forum*, vol. 26, pp. 34–45, Wiley Online Library, 2007.

[62] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 34, no. 5, pp. 827–828, 1978.

[63] R. Lucas, "Rate of capillary ascension of liquids," *Kolloid Z*, vol. 23, no. 15, pp. 15–22, 1918.

[64] I. Langmuir, "the constitution and fundamental properties of solids and liquids. part i. solids.," *Journal of the American Chemical Society*, vol. 38, no. 11, pp. 2221–2295, 1916.

[65] G. Haller, "Lagrangian coherent structures," *Annual Review of Fluid Mechanics*, vol. 47, pp. 137–162, 2015.

[66] M. Farazmand and G. Haller, "Polar rotation angle identifies elliptic islands in unsteady dynamical systems," *Physica D: Nonlinear Phenomena*, vol. 315, pp. 1–12, 2016.

[67] G. Haller and G. Yuan, "Lagrangian coherent structures and mixing in two-dimensional turbulence," *Physica D: Nonlinear Phenomena*, vol. 147, no. 3, pp. 352–370, 2000.

[68] F. Lekien, S. C. Shadden, and J. E. Marsden, "Lagrangian coherent structures in n-dimensional systems," *Journal of Mathematical Physics*, vol. 48, no. 6, p. 065404, 2007.

[69] S. C. Shadden, F. Lekien, and J. E. Marsden, "Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows," *Physica D: Nonlinear Phenomena*, vol. 212, no. 3, pp. 271–304, 2005.

[70] J. Kasten, C. Petz, I. Hotz, H.-C. Hege, B. R. Noack, and G. Tadmor, "Lagrangian feature extraction of the cylinder wake," *Physics of Fluids (1994-present)*, vol. 22, no. 9, p. 091108, 2010.

[71] S. Ali and M. Shah, "A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–6, IEEE, 2007.

[72] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, pp. 163–169, ACM, 1987.

[73] J. D. Furst and S. M. Pizer, "Marching ridges.," in *SIP*, pp. 22–26, 2001.

[74] R. Peikert and F. Sadlo, "Height ridge computation and filtering for visualization," in *2008 IEEE Pacific Visualization Symposium*, pp. 119–126, IEEE, 2008.

[75] F. Sadlo, A. Rigazzi, and R. Peikert, "Time-dependent visualization of lagrangian coherent structures by grid advection," in *Topological Methods in Data Analysis and Visualization*, pp. 151–165, Springer, 2011.

[76] M. H. Sulman, H. S. Huntley, B. Lipphardt, and A. Kirwan, "Leaving flatland: Diagnostics for lagrangian coherent structures in three-dimensional flows," *Physica D: Nonlinear Phenomena*, vol. 258, pp. 77–92, 2013.

[77] S. G. Raben, S. D. Ross, and P. P. Vlachos, "Computation of finite-time lyapunov exponents from time-resolved particle image velocimetry data," *Experiments in fluids*, vol. 55, no. 1, pp. 1–14, 2014.

[78] F. Sadlo and R. Peikert, "Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1456–1463, 2007.

[79] Z. Yuan, F. Chen, and Y. Zhao, "Pattern-guided smoke animation with Lagrangian Coherent Structure," in *ACM Trans. Graph.*, vol. 30, p. 136, 2011.

[80] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun, "Voronoi-based variational reconstruction of unoriented point sets," in *Symposium on Geometry processing*, vol. 7, pp. 39–48, 2007.

[81] K. L. Schlueter-Kuck and J. O. Dabiri, "Coherent structure colouring: identification of coherent structures from sparse data using graph theory," *Journal of Fluid Mechanics*, vol. 811, pp. 468–486, 2017.