Stony Brook University

Shape-Based Analysis

A Dissertation Presented

by

Krishna Chaitanya Gurijala

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

December 2014

Stony Brook University

The Graduate School

Krishna Chaitanya Gurijala

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Arie Kaufman – Dissertation Advisor
Distinguished Professor and Chairman, Computer Science Department

Xianfeng Gu – Chairperson of Defense
Associate Professor, Computer Science Department

Allen Tannenbaum
Professor, Computer Science Department

David G. Ebin
Professor, Mathematics Department

Feng Qiu
Research Scientist
Siemens Corporate Research

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Shape-Based Analysis

by

Krishna Chaitanya Gurijala

Doctor of Philosophy

in

Computer Science

Stony Brook University

2014

Shape analysis plays a critical role in many fields, especially in medical analysis. There has been substantial research performed for shape analysis in manifolds. On the contrary, shape-based analysis has not received much attention for volumetric data. It is not feasible to directly extend the successful manifold shape analysis methods, such as heat diffusion, to volumes due to the huge computational cost. The work presented herein seeks to address this problem by presenting two approaches for shape analysis in volumes that not only capture the shape information efficiently but also reduce the computational time drastically.

The first approach is a cumulative approach and is called the *Cumulative Heat Diffusion*, where the heat diffusion is carried out by simultaneously considering all the voxels as sources. The cumulative heat diffusion is monitored by a novel operator called the *Volume Gradient Operator*, which is a combination of the well-known Laplace-Beltrami operator and a data-driven operator. The cumulative heat diffusion is computed by considering all the voxels and hence is inherently dependent on the resolution of the data. Therefore, we propose a second approach which is a stochastic approach for shape analysis. In this approach the diffusion process is carried out by using tiny massless particles termed shapetons. An appropriate distance value is chosen as new definition of time step. The shapetons are diffused in a Monte Carlo fashion across the voxels until the pre-defined distance value (serves as single time step) is reached. The direction of propagation for the shapetons is

determined by the volume gradient operator. The shapeton diffusion is a novel diffusion approach and is independent of the resolution of the data. These approaches robustly extract features and objects based on shape.

Both shape analysis approaches are used in several medical applications such as segmentation, feature extraction, registration, transfer function design and tumor detection. This work majorly focuses on the diagnosis of colon cancer. Virtual colonoscopy is a viable non-invasive screening method, whereby a radiologist can explore a colon surface to locate and remove the precancerous polyps (protrusions/bumps on the colon wall). To facilitate an efficient colon exploration, a robust and shape-preserving colon flattening algorithm is presented using the heat diffusion metric which is insensitive to topological noise. The flattened colon surface provides effective colon exploration, navigation, polyp visualization, detection, and verification. In addition, the flattened colon surface is used to consistently register the supine and prone colon surfaces. Anatomical landmarks such as the taeniae coli, flexures and the surface feature points are used in the colon registration pipeline and this work presents techniques using heat diffusion to automatically identify them.

Shape analysis in graphs is vital to represent and visualize relationships between data items. Graph embedding methods play an important role in visualizing the data items and their relationships by providing an automatic and clutter free layout. A novel graph embedding approach is presented that will compute the global characteristics of a graph, such as hyperbolic or parabolic type, and also the Ricci curvature in the local neighborhood, which can analyze the structure of the graph. The method has three stages. In the first stage, the graph is embedded on a topological surface. In the second stage it is embedded on a Riemann surface by computing the Ricci flow and finally in the third stage it is embedded onto a surface in three dimensional Euclidean space. The approach is general, practical, and theoretically rigorous.

Contents

List of Figures

List of Tables

List of Algorithms

Publications

- K. C. Gurijala, R. Shi, W. Zeng, X. Gu, and A. E. Kaufman. Colon flattening using heat diffusion Riemannian metric. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2848–2857, 2013

- K. C. Gurijala, L. Wang, and A. E. Kaufman. Cumulative heat diffusion using volume gradient operator for volume analysis. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2069–2077, Dec 2012

- K. C. Gurijala, A. E. Kaufman, W. Zeng, and X. Gu. Extraction of landmarks and features from virtual colon models. *Proceedings of MICCAI Workshop on Virtual Colonoscopy and Abdominal Imaging*, 6668:105–112, 2010

- W. Zeng, J. Marino, K. C. Gurijala, X. Gu, and A. E. Kaufman. Supine and prone colon registration using quasi-conformal mapping. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1348–1357, Nov. 2010

- K. C. Gurijala, L. Wang, and A. E. Kaufman. Monte-Carlo based real-time shape analysis in volumes. *Submitted to IEEE Transactions on Visualization and Computer Graphics*, 2014

- K. C. Gurijala, H. Peng, X. Yu, W. Zeng, X. Gu, and A. E. Kaufman. Graph embedding on surface using Ricci flow. *Submitted to IEEE Transactions on Visualization and Computer Graphics*, 2014

Chapter 1

Introduction

1.1 Preface

There has been substantial research on volume data analysis using various parameters, such as voxel intensity, gradient, curvature, and size. However, not much attention has been given for shape-based volume analysis and hence incorporating shape information for volume analysis still remains a challenge. This is not the scenario in the case of manifolds, where diffusion based techniques, such as heat diffusion have become popular for manifold shape analysis. However, the heat diffusion theory has been solely applied to manifolds and no previous attempts have been made to exploit this lucrative heat diffusion process for volumes. Directly extending the manifold heat diffusion idea to volumes is not trivial, mainly due to the extremely high computational cost. Even in the case of manifolds, where the heat diffusion has been successful, the shape analysis is limited to 3D models of modest resolution due to the high complexity of the heat kernel computation. Therefore, when dealing with large volumetric data, there is a need for a novel diffusion approach which not only allows an efficient shape analysis but also reduces the computational overhead. Hence, in this work we present two approaches for shape-based analysis, the first one being a cumulative approach and the second one being a stochastic approach.

The first approach presented is a cumulative approach which is a modified heat diffusion process, called the *Cumulative Heat Diffusion*. Unlike the conventional heat diffusion process, where the diffusion is carried out by considering each node separately as the source, we simultaneously consider all the voxels as sources and carry out the diffusion, hence the term cumulative heat diffusion. Accordingly, the initial heat is assigned to all the voxels at once. The manifold heat diffusion process is governed by the heat kernel evaluation which is expressed using the Laplace-Beltrami Operator. However, just using the Laplace-Beltrami operator would not suffice for volumetric data which also possess the intensity on the voxels. Hence,

we propose to also use the voxel intensity by introducing a new operator to evaluate the cumulative heat diffusion process for volumes, called the *Volume Gradient Operator* and the heat diffusion is carried out using this operator. The volume gradient operator is defined as a combination of Laplace-Beltrami operator and a data driven operator, which is a function of the half gradient. The half gradient is the absolute value of the difference between the voxel intensities. The volume gradient operator is used to determine the initial heat values of the voxels and also find the edge weights that monitor the heat diffusion process. The cumulative heat diffusion method is conceptually sound and simple to implement, which makes it an attractive approach for shape-based volume analysis.

The cumulative heat diffusion method, however, cannot be adopted for real-time shape analysis due to the dependency on the size of the volume. In addition, the heat diffusion is carried out only between the voxels and their corresponding 1-ring neighboring voxels per time step. Hence, the number of time steps required to capture the shape information increases with the increasing number of voxels. In other words, the rate of heat flow is influenced by the resolution of the data. Therefore, we present a second approach which is a Monte Carlo based real-time diffusion process for shape-based analysis of volumetric data. This diffusion process is a stochastic approach and is carried out by using tiny massless particles termed *shapetons*. These shapetons are used to capture the shape information. Initially, the shapetons are randomly distributed inside the voxels of the volume data. The shapetons are then diffused in a Monte Carlo fashion across the voxels to obtain the shape information. Owing to its success in capturing the shape information, the volume gradient operator is used to monitor the direction of propagation for the shapetons. The shapetons are diffused for a pre-defined distance which accounts for one time step. As a result, the diffusion of the shapeton particles is independent of the resolution of the data and only depends on the distance value chosen. The distance value is a user defined parameter and hence can be chosen based on the application. All the shapetons are diffused simultaneously and all the computations are evaluated in parallel using a GPU, thereby facilitating a real-time monitoring of the final result.

One major area where the shape analysis and visualization can be immensely helpful is medical analysis. Both the shape analysis approaches are used in several medical applications such as segmentation, shape-based transfer function design and polyp detection inside the colon surface. In several medical diagnosis cases, the shape information plays a vital role. For example, structures on the colon wall have shapes that are indication of malignancy or pre-malignancy. Flattening the twisted 3D colon surface to a 2D domain is proven to be a successful approach for effective colon exploration and polyp detection. We present a novel shape-preserving flattening algorithm for the colon surface. By using the heat diffusion metric [33, 94], we

proposed a new colon flattening algorithm that is efficient, shape-preserving, and robust to topological noise. Unlike previous approaches, which require a mandatory topological denoising to remove fake handles, our algorithm directly flattens the colon surface without any denoising. In our method, we replace the original Euclidean metric of the colon surface with a heat diffusion metric that is insensitive to topological noise. Using this heat diffusion metric, we then solve a Laplacian equation followed by an integration step to compute the final flattening. The presented flattening method is conformal (shape-preserving) and the shape of the polyps are well preserved. We further show how the supine-prone colon registration pipeline is made robust by using our flattening approach. The registration of supine and prone colon surfaces using the flattened colon surfaces needs anatomical landmarks and reliable feature points to ensure a consistent registration. The anatomical colon landmarks include taeniae coli and the flexures. We present techniques to automatically identify these anatomical landmarks for the colon surfaces. In addition, the end points of the haustral folds (i.e., colon folds) serve as reliable feature points. We detect these points using a max-flow min-cut approach which serve as constraints in registering the supine and prone surfaces using a quasi-conformal approach.

Another area where understanding shape characteristics and topology plays a vital role is in graph analysis. Graphs are used to represent and visualize relationships between data items. Graph embedding methods play an important role in visualizing the data items and their relationships by providing an automatic and clutter free layout. Graphs play fundamental roles in many engineering fields. For example, they have been used to model social relations of communities, traffic between telecommunication switches, networking with wireless sensor nodes and airline routes among cities. We present a novel graph embedding approach that will compute the global characteristics of a graph, such as hyperbolic or parabolic type, and also the Ricci curvature in the local neighborhood, which can analyze the structure of the graph. The pipeline of the algorithm has three stages. In the first stage, the graph is embedded onto a *topological surface*. In the second stage it is embedded onto a *Riemann surface* and in the last stage it is finally embedded onto a *surface in* $\mathbb{R}^3$. This method removes the edge crossings completely, and preserves the topological structure of the graph. This approach is general, *practical* for visualizing large scale graphs, and *rigorous* with solid theoretic foundations. By virtue of our graph embedding approach, local changes in the graph do not affect the overall embedding and thus finds applications in visual graph comparison and visualizing changes in dynamic graphs.

1.2 Volume Rendering

In a traditional computer graphics rendering pipeline, pre-defined surfaces are rendered [47]. These surfaces can be represented as discrete meshes, with triangular meshes often being a popular choice due to the guarantee that the vertices of each mesh will lie within a plane. It is also possible to use continuous representations, such as non-uniform rational B-spline (NURBS) surfaces [121]. Volumes can be rendered in a similar manner by first extracting an isosurface [105]. On the other hand, direct volume rendering (DVR) allows for the volume tiself to be directly rendered without the need to extract surfaces which would be typically rendered in a computer graphics pipeline. Maximum intensity projection (MIP) rendering of volumes can also be accomplished, which yields a 2D image that appears similar to an x-ray [111]. However, this technique is not applicable for a great many applications due to its visual limitations.

In order to perform DVR, an optical model must be established, and there are several possibilities [107]. The most common model is the emission-absorption model, in which the elements of the volume are considered as particles within a cloud which are able to both emit their own light and absorb incident light. This leads to the following so-called volume rendering integral:

$$I(D) = I_0 T(D) + \int_0^D g(s) T'(s) ds. \tag{1.1}$$

In this equation, the result $I(D)$ represents the radiance reaching the camera from D. The first term represents the background illumination I_0 multiplied by the transparency of the cloud $T(D)$. The second term represents the integration over all sample positions s, multiplying each sample's source value $g(s)$ by the transparency between s and the eye $T'(s)$.

This volume rendering integral in the continuous domain can be discretized into compositing for use with discretely sampled data. In this case, composition is possible using both back-to-front and front-to-back methods. For back-to-front compositing, the following is performed at each sample step:

$$C_{dst} \leftarrow (1 - \alpha_{src}) C_{dst} + C_{src}. \tag{1.2}$$

For front-to-back compositing, the following is performed at each sample step:

$$\begin{aligned} C_{dst} &\leftarrow C_{dst} + (1 - \alpha_{dst}) C_{src}, \\ \alpha_{dst} &\leftarrow \alpha_{dst} + (1 - \alpha_{dst}) \alpha_{src}. \end{aligned} \tag{1.3}$$

In both instances, C_{dst} and C_{src} are the destination and source colors, while α_{dst} and α_{src} are the destination and source opacities. Note that while front-to-back

compositing requires extra maintenance of the opacity term through the integration, it also allows for early termination of the composition when the opacity reaches a sufficient level (often $\alpha_{dst} = 0.95$).

There are a number of methods for performing the actual volume rendering. A technique has been introduced where the voxels are *splatted* onto the screen space and rendered as disks [154]. A method known as *shear warp* was also proposed where the viewing transformation would be factored into a 3D shear parallel to the volume slices, a projection would create a distorted image, and a 2D warp would then be used to undistort the final image [93]. The volume as a whole can also be decomposed into individual slices which are then rendered and composited with typical 2D texture rendering in the graphics pipeline. The method known as ray casting involves shooting rays through the volume data and sampling at regular points along the rays [101]. Ray casting typically provides the best image quality and is the preferred method of performing DVR [135].

When sampling through the volume, interpolation is needed to obtain sample values not lying at an exact voxel. Nearest neighbor interpolation is the quickest and easiest method, though it leads to undesirable quality for the rendered image. Trilinear interpolation provides significantly better results for moderately more work, and is the most commonly used interpolation method for volume rendering. More advanced methods have also been used to achieve better results, such as tricubic interpolation or the use of Voronoi splines [109], though the tradeoff between speed and quality is not advantageous.

When rendering volumes, which often consist of singular scalar density values for each voxel, it is desirable to map these scalar values to optical properties, such as color and opacity. A 1D transfer function is used to provide such a mapping [120]. Multi-dimensional transfer functions can also be used to give more control over the mapping. Most commonly, a 2D transfer function is utilized with density and gradient magnitude as the two axes. The rendering of isosurfaces within volumes can be improved by utilizing peak finding [88].

To convey greater realism and recognition of the objects being rendered, illumination and shading are used. Based on the gradient information at each sampling point, shading calculations can be performed using a typical illumination model, such as Blinn shading. Gradients are often estimated as a simple central difference in each of the three axes. These gradients can be calcualted on-the-fly or precomputed, based on the needs of the renderer. When more precision is required, better methods of gradient estimation have been proposed [71].

There have been a number of improvements to increase the quality of rendering, both in visual quality and speed. Stochastic jittering can be used to slightly offset the positions of rays so that wood-grain effects are reduced [41]. Pre-integrated transfer functions have been introduced to account for values between two dis-

cretely sampled values [42]. Empty space skipping has been introduced to allow a ray caster to not sample from regions which do not contain displayable voxels [103]. Direct interval volume visualization has also been presented as a way of representing sharp isosurfaces so that they are visualized accurately [4].

Although specialized hardware for volume rendering was developed [118, 119], it has become common today to utilize commodity graphics hardware for this task [41]. The graphics processing unit (GPU) used for gaming provides a high level of SIMD performance at a relatively low cost and can allow for real-time ray casting [137]. More recently, general processing on the GPU has become popular, with NVidia C-like CUDA language being widely adopted [35].

1.3 Shape Analysis

1.3.1 Deterministic Methods

Spectral and diffusion geometry methods have been popularly used for shape analysis in mesh surfaces. The Laplace-Beltrami Operator and the heat diffusion methods have become attractive due to their properties specifically in shape analysis. The Laplace-Beltrami operator is used to express the heat kernel [102]. A discrete Laplace-Beltrami operator has been defined and its convergence to the continuous case has been discussed [19, 129]. For triangular meshes, one of the most common discretizations of the LBO is by using the cotangent weight scheme which was originally introduced [39, 122]. The heat diffusion process is governed by the heat kernel. The heat kernel has been used to define a shape signature called the heat kernel signatures and used for applications such as shape comparisons and feature extractions in a most effective manner [140]. Bronstein et al. [24] have used the heat kernel signature for non-rigid shape recognition. Ovsjanikov et al. [116] have used the heat kernel to map isometries between a pair of shapes. Zobel et al. [162] have generalized the heat kernel signature from the functional space to the differential form space. Raviv et al. [127] have generalized the surface heat kernel signature to volumes for shape retrieval. Vaxman et al. [147] have proposed a multi-resolution approach to evaluate the heat diffusion process with a reduced running time. Recently, the physical phenomenon of waves governed by the Scrodinger wave equation has been used for shape analysis by Aubry et al. [11].

Shape has been used for volume classification previously. Sato et al. [132] proposed a volume classification based on shapes where they detect pre-defined shapes such as edge lines and blobs by measuring the multi-scale responses to 3D filters. Skeleton based approaches were extensively used to study shapes and for shape based volume visualization. Hilaga et al. [65] used skeletons for shape matching and volume visualization. Pizer et al. [123] proposed a framework of stable me-

dial representation for segmentation of objects, registration and statistical 3D shape analysis. Several other attempts using skeletons for shape-based volume classification were done by Correa et al. [34] and Reniers et al. [128]. Motivated by these ideas, Praßni et al. [125] presented a shape-based transfer function using the curve-skeleton of the volumetric structure. Bruckner and Moller [25] introduced similarity maps for volume analysis in which they represent the similarity of isosurfaces for different isovalues. Later Haidacher and Bruckner [60] extended the idea of the similarity maps to multimodal data and used it for analysis, fusion and classification of multimodal data.

1.3.2 Stochastic Methods

The Monte Carlo simulation technique has formally existed since the early 1940s, where it had applications in research into nuclear fusion. However, with the increase in computer technology and power this technique become more widely used. This is because computers are now able to perform millions of simulations much more efficiently and quickly than before. This is an important factor because it means that the technique can provide an approximate answer quickly and to a higher level of accuracy. Monte-Carlo methods, in visualiztion for volumes, have been largely used for photorealistic rendering by a photon mapping technique which was developed by Jensen [75, 76, 77]. Visualization of global illumination involves the evaluation of light scattering on different surfaces. This scattering is performed by Monte Carlo sampling against the surfaces. The main reason to use Monte Carlo methods is due to their ability to approximate an answer qucikly, that would be very time-consuming to find out the answer to, if we were using other methods to determine the exact answer. In simple words, Monte Carlo methods are used to simulate problems that are too difficult and time-consuming to use other methods for.

Probabilistic methods are used to simulate problems that are too difficult and time-consuming to use other methods for. Brownian motion and random walks are known to address problems in a probabilistic way. The successful approach of shape analysis using heat kernel is also related to Brownian motion. Bass [17] has showed that Brownian motion, which is a significant concept in particle theory, and the heat kernel are related. In general, any diffusion process is connected with the study of Brownian motion [27, 145, 151] in probability theory. Kac [84] has discussed the relation between the random walks and the theory of Brownian motion. Hence, diffusion phenomenon, Brownian motion and random walks are all related in probability theory.

Probabilisitic and stochastic methods have been used on mesh surfaces for various other applications. Methods such as random walks and markov random fields are used in applications such as mesh smoothing and segmentation. Lai et al. [95]

have used the random walk to develop an interactive and automatic mesh segmentation method. Meila and Shi [108] have re-interpretated the spectral methods for clustering and segmentation with probabilistic foundation. Castellani et al. [26] have modeled local areas of a surface using a Hidden Markov Model which they used for object recognition on cluttered scenes. Fattal et al. [45] have presented a blue-noise sampling using a statistical mechanics interacting particle model.

1.4 Surface Mapping

Colon flattening, a method in which the entire inner surface of the colon is displayed as a succinct 2D image, has been used successfully for several medical imaging applications. Initial attempts to flatten the colon surface include iterative methods based on electrical field lines [14, 148, 149, 150], cartographic projection [117], and some others [15, 16, 104]. However, most of these methods deform the colon surface or do not preserve the local shapes well. Conformal geometry, an approach where the local angles are preserved, has been well established in the field of computer graphics, especially in the creation of texture maps [40, 62, 136, 159]. Discrete Ricci flow is a more recent method of computing conformal maps of structures [81, 85] and is very useful in the construction of geometric structures [79] and to obtain optimal surface parameterizations using inverse curvature maps [155]. Colon flattening techniques have been proposed using conformal mapping [61, 158] and holomorphic 1-form parameterization [68, 134]. The conformally flattened colon was used in the detection of colonic polyps [69] and supine-prone colon registration [158]. Further, surface parameterization using harmonic functions has also been used in graphics [143] and medical imaging of the brain [6, 7, 53, 63, 106, 138] and blood vessels [161].

All the above methods require a mandatory pre-process of topological denoising because of the topological noise (fake handles), without which the flattening would be unsuccessful. While some of these methods use colon surfaces that have been manually denoised, others utilize a topological denoising algorithm [59, 83, 160]. A fast topological denoising algorithm for conformal colon flattening has also been proposed [68].

1.5 Graph Embedding

Generation of good graph layouts have received much attention in graph visualization. The force-directed method is a popular graph layout approach where the nodes are modeled as rigid bodies, and edges are modeled as elastic springs. Several energy models [36, 48, 114, 146] have been proposed based on different

aesthetics. Force-directed algorithms for graph layouts have been generalized to calculate the layout of a graph in an arbitrary Riemannian geometry [90] where the Euclidean notions of distance, angle, and force-interactions are extended to smooth non-Euclidean geometries via projections to and from appropriately chosen tangent spaces. Hyperbolic and spherical layouts are demonstrated. However, force-directed methods perform well only for relatively small graphs and encounter difficulties for large scale graphs, due to the local optimality of the energy and the time complexity of the global optimization. To improve computational efficiency, fast multilevel algorithms [8], GPU-accelerated force-based models [49] and simplified energy functions [91] have been proposed to generate high quality layouts for large graphs. Space filling curves are used to compute graph layouts for visualizing network data [112]. This method is very fast and guarantees that there will be no nodes that are collocated. However, it focuses only on the node location, and ignores the edge crossings. Compared to all the above graph layout methods, our Ricci flow graph embedding approach ensures no edge crossings, and has good scalability since the convergence of the Ricci flow is exponentially fast [31].

A range of visual graph comparison techniques have been proposed using side-by-side view [5, 66, 98], superimposed views [2, 43] and animations [38, 156]. Dynamic graph visualization is closely related to visual graph comparison and several approaches have been proposed using small multiples [44, 46, 124], difference graphs [3, 64], by animating the transitions between time steps [12, 21, 50, 52, 115] and 3-D cubes [13]. A main challenge for such methods is to provide stability to the layout [23, 50, 72, 92]. Visualizing the changes in graphs requires a trade-off between the layout and the stability. The key factor influencing stability is how the graph is laid out when some changes occur. If a stable layout is used, then regardless of the changes the graph can be visualized well [9, 10, 51]. Stability, alternately can be defined as preserving the mental map of the users. The mental map is the image users have of the information and preserving it implies minimizing changes in the visual representation. Our graph layout based on Ricci flow energy optimization provides a stable graph layout that helps in identifying the changes in the graph easily. It not only provides a stabilized layout across all the time steps but also provides an optimized layout for each of the time steps. In addition, our approach achieves high mental map preservation. We employ an approach similar to that of small multiples where the graphs for all time steps are displayed side-by-side and the corresponding structural changes are highlighted between time steps.

Chapter 2

Cumulative Shape Analysis

In this chapter, a simple, yet powerful method called the cumulative heat diffusion is introduced. Through the cumulative heat diffusion, one can obtain the shape information in different scales and perform a shape-based volume analysis. The cumulative heat diffusion reduces the computational cost drastically compared to conventional heat diffusion and hence makes the shape analysis for volumetric datasets feasible. Unlike the conventional heat diffusion process, where the diffusion is carried out by considering each node separately as the source, the cumulative heat diffusion simultaneously considers all the voxels as sources and carries out the diffusion. Accordingly, the initial heat is assigned to all the voxels at once. Moreover, a new operator, called the volume gradient operator, is introduced for the evaluation of cumulative heat diffusion. The volume gradient operator is a combination of the Laplace-Beltrami operator and a data-driven operator which is a function of the half gradient. The half gradient is the absolute value of the difference between the voxel intensities. The volume gradient operator by its definition captures the local shape information. It is used as the weighting parameter for the heat diffusion process and to assign the initial heat values.

This chapter is organized as follows. Section 2.1 presents the motivation. Section 2.2 describes the algorithm with a detailed description of the volume gradient operator and the cumulative heat diffusion process for volume analysis. Various properties of the method are discussed in Section 2.3. Section 2.4 discusses the influence of the parameter p and Section 2.5 provides some results of the method.

2.1 Motivation

Much research has been undertaken on incorporating information for volume data analysis from various parameters such as voxel intensity, gradient, curvature, and size. However, incorporating shape information for volume analysis still remains a challenge. Some studies have been done using pre-defined shapes [34,

65, 123, 125, 128, 132]. However, in most volumetric datasets, especially medical datasets, the shape is complex and cannot be perfectly pre-defined. Shape analysis has become sophisticated for manifolds, thanks to heat diffusion theory and related techniques [102, 130, 131]. The heat diffusion process, governed by a partial differential equation on the shape, captures the geometric information. It has many useful properties; namely it is stable, informative, isometric invariant, robust to noise and independent of the initial conditions [140]. According to the heat theory, the heat flow or diffusion is successfully able to differentiate the shapes, irrespective of orientation and scaling, while enforcing no shape restrictions (i.e., there is no need to pre-define the shapes). These merits form the motivation of our work in this chapter.

Nevertheless, in order to evaluate the heat diffusion on the entire mesh surface, the heat flow on all the paths of the mesh surface has to be considered. Due to its extremely high computational cost, directly extending the manifold heat diffusion idea to volumes is not trivial. A multi-resolution approach to evaluate the heat diffusion process was proposed to reduce the running time [147]. Though theoretically it is possible, no attempt has been made to use the heat diffusion for shape-based volume analysis. For the first time, we apply the heat diffusion theory directly to volumetric data by introducing a cumulative heat diffusion process. Unlike the mesh surfaces, the volumetric datasets have the intensity values stored on a regular Cartesian 3D grid and hence need an additional operator to monitor the heat diffusion process. Therefore, in this cumulative heat diffusion process for volumes, we propose to use the voxel intensity along with the LBO, by introducing a new operator called the volume gradient operator.

2.2 Algorithm

The cumulative heat diffusion is a modified heat diffusion on volumes used for shape-based volume analysis. The diffusion process is carried out by considering all the voxels as sources simultaneously. A new operator, volume gradient operator (VGO) is used to assign the initial heat values and determine the edge weights that monitor the heat diffusion process. At the end of the cumulative heat diffusion, for a certain value of time t, all the voxels will have final heat values. By virtue of the heat diffusion theory, when t is very large, all the voxels belonging to a single object will have the same heat values. Similarly, all the objects with similar shapes will also have the same heat values, independent of their position, scale and orientation (as will be discussed in the Section 2.3.4). These heat values of all the voxels at each time step are stored and displayed in a histogram for visualization purposes. This histogram created by using the heat values is used in volume analysis, such as for

exploration and classification. The evaluation of the histogram, the interpretation of the information depicted in the histogram and some of the properties of our cumulative heat diffusion are discussed in the Section 2.3. This cumulative heat diffusion process is simple to implement on volume data whereas the conventional heat diffusion is not feasible in practice. Figure 2.1 shows the result of applying the cumulative heat diffusion approach for object classification based on shape. Figures 2.1(a), (b) and (c) show the results on MRI head, abdominal stent and foot datasets. Figure 2.1(d) shows the corresponding heat histogram of the foot data.

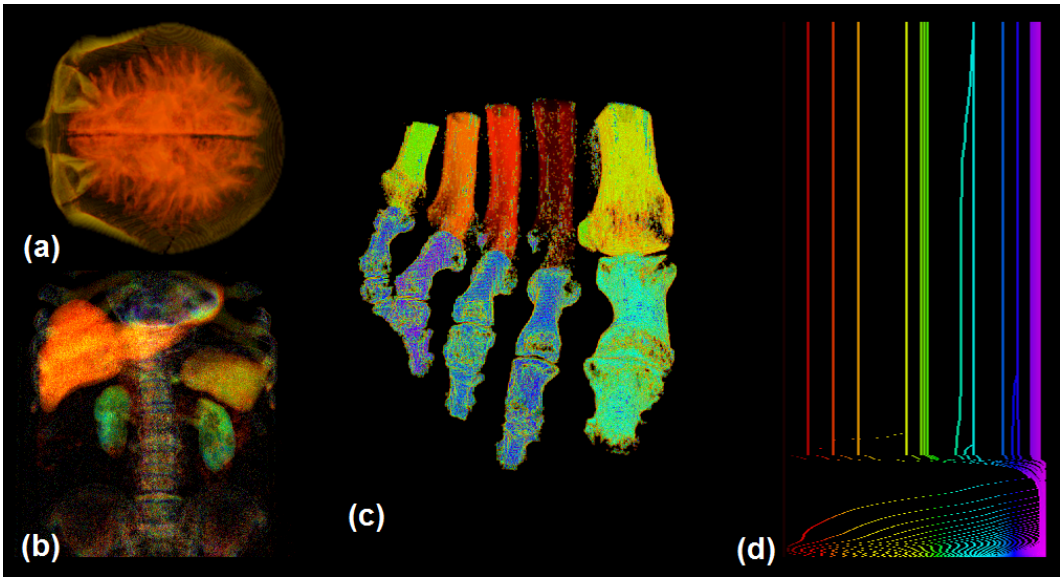


Figure 2.1: Classifying objects based on their shape using our cumulative heat diffusion on (a) MRI head, (b) abdominal stent, and (c) foot volumetric datasets. The shape-based volume exploration reveals various organs such as kidneys, liver, pancreas and spinal cord (for the abdominal stent), brain, eye sockets and cranium (for the MRI head) and different shaped bones (for the foot). (d) The corresponding heat histogram of the foot data where each vertical line represents a bone of the foot.

2.2.1 Cumulative Heat Diffusion

We will now introduce the cumulative heat diffusion on volume data where voxels are the data elements. The difference between the normal heat diffusion process and the cumulative heat diffusion is the way the diffusion process is carried out and the way the initial conditions are defined. In the conventional heat diffusion process, the initial heat is assigned by considering one voxel as the source at a time. The initial heat assigned to the source voxel is a fixed constant value not exceeding

1 (it can be any random value less than 1) while the initial heat of all the other voxels is assigned to zero. The heat is diffused from this source voxel for a given number of time steps t and at the end, the heat remaining on the source is recorded as the final heat value of that voxel after a time t. This entire process of assigning the initial heat value and carrying out the diffusion is repeated by considering each of the voxels separately as a source. In each step, all the voxels are taken into account to determine the diffusion process. This explains the large time complexity of $t \times n^2$ in the conventional heat diffusion process. To reduce the computational cost, in our cumulative heat diffusion approach, instead of considering each voxel as a source separately, we consider all the voxels as sources simultaneously. The way we do the heat initialization makes it possible for our cumulative heat diffusion to produce results identical to those of the conventional process.

Let V be a volume, then the cumulative heat diffusion process over V is governed by the following heat equation:

$$\Gamma_V k(x,t) = -\frac{\partial k(x,t)}{\partial t} \tag{2.1}$$

where Γ_V is the edge weight which is determined by using the volume gradient operator of V and $k(x,t)$ is the temperature (heat) at location (voxel) x at time t. Γ_V is used to determine the amount of heat flow between voxels in the cumulative heat diffusion process. It is evaluated using Equations 2.4 and 2.5 explained in the Section 2.2.2. Finally on the discrete volume data, the cumulative heat diffusion is estimated using the discrete version of Γ_V which is evaluated as follows:

$$\Gamma_V f_i = -\sum_{j \in N(i)} VGO(i,j)(f_i - f_j) \tag{2.2}$$

where $N(i)$ is the set of all the 1-ring neighboring voxels j of voxel i, f is the corresponding heat of the voxels and $VGO(i,j)$ is the weighting factor of the edge between the voxels i and j.

We virtually extend the boundary of the volume data so that the heat will go over the boundary and after a large time step the heat of the background voxels will be zero. This virtual extension of the boundary also gets rid of the problem of dealing with the complicated boundary conditions. The solution $k(x,t)$ of Equation 2.1 with the initial conditions $k(x,0) = InitH(x)$ determines the amount of heat at voxel x at time t. $InitH(x)$ is the initial heat on voxel x defined by Equation 2.3 and is pre-computed using the volume gradient operator. The respective initial heat values are assigned to all the voxels simultaneously. Thus, the initial heat value assigned to the voxels is not a random fixed constant number (as it was in the case of the conventional heat diffusion), rather each voxel has a different initial heat value

decided by using the volume gradient operator. The initial heat value $InitH(v_0)$ assigned to the source voxel v_0 is determined by using the sum of the weights of the six edges around the voxel so that the initial heat describes the local shapes and is defined as follows:

$$InitH(v_0) = 1 - \frac{\sum_{v \in N(v_0)}(VGO(v_0, v))}{\max_{\forall i \in N_0}(\sum_{v' \in N(i)}(VGO(i, v')))} \tag{2.3}$$

where N_0 is the set of all voxels. Equation 2.3 is defined such that all the initial heat values lie between 0 and 1 to start with.

Finally, for a given time step t, the cumulative heat diffusion process is carried out by allowing the heat assigned using Equation 2.3 to flow based on Equation 2.2. As all the voxels are considered as sources simultaneously, the total computational time for the cumulative heat diffusion for a given large time t is of the order $t \cdot n$ where n is the number of voxels in the volume data. Thus, we have reduced the computational time by a factor of n when compared to the conventional heat diffusion. As n is a large number, we have achieved an order of magnitude speed up through the use of our cumulative heat diffusion process in obtaining the shape information of the volume data.

2.2.2 Volume Gradient Operator

Much information about the structure of the volume can be obtained from the intensity of the voxels and from comparing the intensity of the neighboring voxels. A lot of research has gone into using the intensity and the gradient (difference of the intensities with the neighboring voxels) of the voxels for volume analysis [86, 87]. Motivated by this idea, we propose to use the intensity of the voxels to define an operator which monitors the heat diffusion process in obtaining the shape information. This will be the first time that an operator based on voxel intensities (along with the LBO) is being used in the theory of heat diffusion. In addition, the intensity of the voxels would provide a better initial framework to start the diffusion process rather than starting with a random initial condition. Hence, we propose an operator called volume gradient operator that helps in the estimation of the cumulative heat diffusion process. We tried different possibilities such as the method by Levin et al. [100] but from our experience our approach gave the best results.

The volume gradient operator is a modification of the Laplace-Beltrami operator. It also considers the intensities on the volume grid points. The volume gradient operator is defined as a combination of LBO and the intensity information, as shown in Equation 2.4. The Laplace-Beltrami operator term, denoted by Δ, in Equation 2.4 captures the geometric information of the grid and the F_v term considers the intensity information, which is defined by Equation 2.5 as a function of the half gradient.

In the case of regular grids (which is the most common representation of a volumetric dataset), Δ can be ignored, which leaves only the F_v term and hence results in $VGO = F_v$. We have used regular volume grids (voxelized volumes) and hence just consider the F_v evaluation. Though we have just used the regular volume grids, in general, the volume gradient operator definition can be applied to irregular volume grids too (in this case Δ cannot be ignored). This shows the benefit of our method over the discrete Laplacian, which does not work for irregular grids.

$$VGO(v_0, v) = \Delta(v_0, v) + F_v(v_0, v) \tag{2.4}$$

where Δ is the LBO and F_v is the data-driven operator.

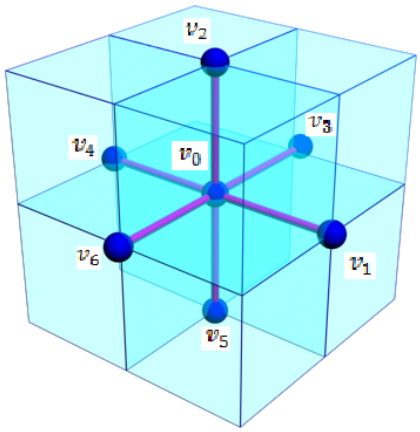


Figure 2.2: v_0 at the center is the source voxel, $v_1, v_2, v_3, v_4, v_5, v_6$ are the 1-ring neighboring voxels of v_0.

We use Figure 2.2 to explain F_v and the half gradient. Suppose v_0 is the voxel under consideration (source) and v_1, v_2, v_3, v_4, v_5 and v_6 are the 1-ring neighboring voxels in positive x, positive y, positive z, negative x, negative y and negative z directions, respectively. Then, F_v is defined by the following equation:

$$F_v(v_0, v) = 1 - p * h_g(v_0, v) \quad \text{where } v \in \{v_1, v_2, v_3, v_4, v_5, v_6\} \tag{2.5}$$

where h_g is the half gradient and p is a user defined value. The effect of p on the result is explained in detail in Section 5.1. The half gradient is defined as the absolute value of the difference of the voxel intensities. As each voxel has six 1-ring neighboring voxels, six half gradients are obtained per voxel. The half gradients h_g of the voxel v_0 are given by:

$$h_g(v_0, v) = \left|\frac{I(v) - I(v_0)}{res}\right| \quad \text{where } v \in \{v_1, v_2, v_3, v_4, v_5, v_6\} \tag{2.6}$$

where I gives the intensity of the corresponding voxel, *res* is the size of the voxel which will account for the distance between the two voxels under consideration. By taking the absolute value in the half gradient computation, we are ensuring that the edge weight between two neighboring voxels (for example say $VGO(v_0, v_1)$ and $VGO(v_1, v_0)$) will be the same value. In other words, we are avoiding any conflicts while assigning the weights to the edges between the voxels. We add an additional constraint where the value of volume gradient operator is equated to 0 if it is less than 0 and equated to 1 if it exceeds 1. Thus, the volume gradient operator is defined in such a way that all the edge weights lie in the range of [0, 1].

2.3 Properties

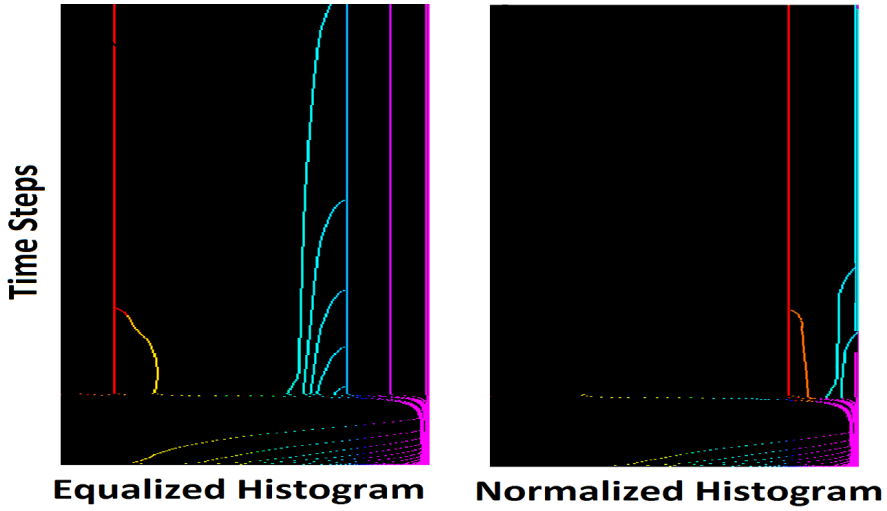
2.3.1 Equalized Histogram

When there is a low number of distinct shapes in the given volume data, a histogram of the normalized heat values would help to clearly distinguish between the different shapes and features. However, volume data is often complicated, having many objects and features with subtle differences in terms of shape, especially in medical datasets. It is impossible to capture the difference between these features using normalized histograms as the variation of the heat values is not uniform. Hence, we use an equalized histogram instead of the normalized histogram for visualization. Histogram equalization increases the global contrast inside the histogram by effectively spreading out the most frequent intensity values. The intensities are better distributed in the histogram through equalization.

Figures 2.3(a) and (b) show the difference between using the equalized histogram and the normalized histogram, respectively. Figure 2.3(a) shows many lines, each line indicating an object of specific shape and all the lines are well-distributed. However, Figure 2.3(b), which is the normalized histogram of the same heat values, shows fewer lines than Figure 2.3(a). This is because the normalized histogram does not distribute the heat values well and hence most of the features cannot be discerned using the histogram. Thus, we use the equalized histogram instead of the normalized histogram.

2.3.2 Histogram Analysis

The equalized heat histogram is useful to analyze the volume data at different scales. The horizontal axis of the histogram denotes the equalized heat values and the vertical axis denotes the time steps as shown in Figure 2.3(a) and these labels hold good for all the figures showing equalized histograms. The shape information obtained at different time steps is synonymous with the shape information obtained



(a) (b)

Figure 2.3: The histogram (a) with equalization and (b) with normalization. The horizontal axis in (a) denotes the equalized heat values, while the horizontal axis in (b) denotes the normalized heat values. The vertical axis in both (a) and (b) denotes the time steps.

at different scales. The heat values at a particular time step incorporate the shape information of the data at that time step, or in other words, at that scale. Thus, by choosing a desired time step from the histogram the shape information at different scales is analyzed. At small number of time steps, local shape features are detected and for large number of time steps, global shape information is captured.

Figure 2.4 shows the result of cumulative heat diffusion at different time steps on a synthetic cube volume data. The figure shows the result for 6 different time steps, namely 20, 100, 750, 900, 1000 and 1100. For a small time step, say 20, local features such as corners and edges of the cube are detected. With increasing time steps, the heat flows gradually and captures larger features. The heat initially resides on the edges and starts moving towards the corners of the cube. This is correct as the initial heat values are determined by using the volume gradient operator which in turn involves the half gradients. As the gradient captures the local shape information, very local features are obtained initially. The heat keeps accumulating at the corners of the cube and eventually covers the entire volume. Thus, when t = 1100 the entire volume of the cube is obtained. From the results in Figure 2.4, it is interesting to observe that the rate of diffusion is slower at sharp corners or edges and is faster at flatter areas. It is clear that the rate of cumulative heat diffusion from t = 100 to t = 750 is less than the rate of cumulative heat diffusion from t = 900 to t = 1100. It has only taken around 200 time steps to capture all the faces of the cube

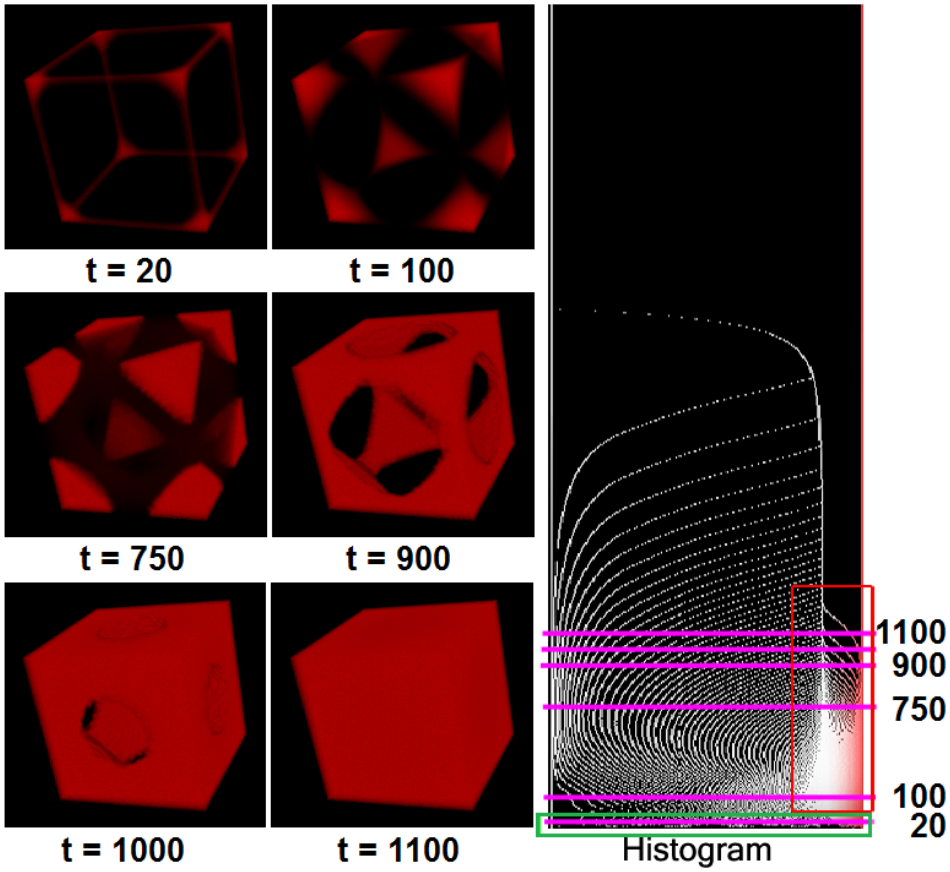


Figure 2.4: The progress of different features being captured at different time steps, which in turn depicts different scales.

while it has almost taken 800 time steps to capture the corners, thus emphasizing the point that the heat flow is slower at sharper areas compared to the flatter areas. This observation is very useful in determining the number of time steps required for the diffusion process in order to obtain specific features. Hence, based on the complexity, shape structure and scale of the desired features of the given data the number of time steps required to obtain the entire feature would vary. For example, pointed features would require more time steps compared to flat shaped features.

Figure 2.4 also shows the corresponding heat histogram obtained by carrying out the cumulative heat diffusion process for 4000 time steps on the cube data set. The respective locations of the time steps corresponding to the results shown are marked in pink lines. The area shown by the green box at the bottom of the histogram shows the initial heat values of all the voxels. Initially, all the voxels that are close to the boundary of an object (both inside and outside) have the same heat val-

ues and so all the curves merge as marked by the red box. With the increasing time, the heat flows and the heat values of all the background voxels move to the value of zero while the voxels belonging to an object have different heat values. This phenomenon can be seen on the histogram when some of the curves diverge. The background voxels eventually join the white line shown on the left border of the histogram while the voxels inside the objects converge to different lines based on their shape information. With the increasing time, we can observe from the histogram that the heat diffuses inside the volume, eventually stabilising and converging to different lines which represent different shapes identified in the dataset. For example, in Figure 2.4, we obtain a single red line shown on the right border of the histogram at t = 1100 when the entire volume of the cube is obtained. In this case, as there is only a single shaped structure, which is the cube, only one single line is obtained in the histogram. If there are several different shaped structures present in the volume, several different lines can be seen in the histogram where each line represents individual shapes as shown in Figure 2.6. The time step where the heat values for all the voxels are stabilized is chosen as an optimum time step where all the shape information can be obtained. Though the background is assigned the color white, it is rendered as transparent in the images shown in the Figure 2.4.

We have used a simple synthetic cube data set of size 32x32x32 to compare the results obtained by using the conventional heat diffusion and our cumulative heat diffusion. Theoretically, the results obtained by considering all the voxels as sources simultaneously and using a single voxel as a source separately will produce the same results at steady state (i.e., for large number of time steps). The steady state result is not affected by the initialization or the evolution of the heat diffusion. The steady state result will smooth out the differences between the initialization conditions. On top of that the initial heat condition in our cumulative heat diffusion is similar to the result obtained after the first time step of the conventional heat diffusion. Figure 2.5 shows the visual comparison of the identical results obtained by using the cumulative heat diffusion (bottom) and conventional heat diffusion (top) for different time steps. A noteworthy observation here is that our cumulative heat diffusion method requires fewer time steps than the conventional heat diffusion method to obtain identical results. Figure 2.5 shows that to obtain a similar result to our method using 30 time steps, the conventional method requires 100 time steps. This shows that the cumulative heat diffusion converges faster than the conventional method.

2.3.3 Object Classification

The cumulative heat diffusion, with the help of the heat histogram, is able to detect different objects present in the given dataset based on their shape. After a

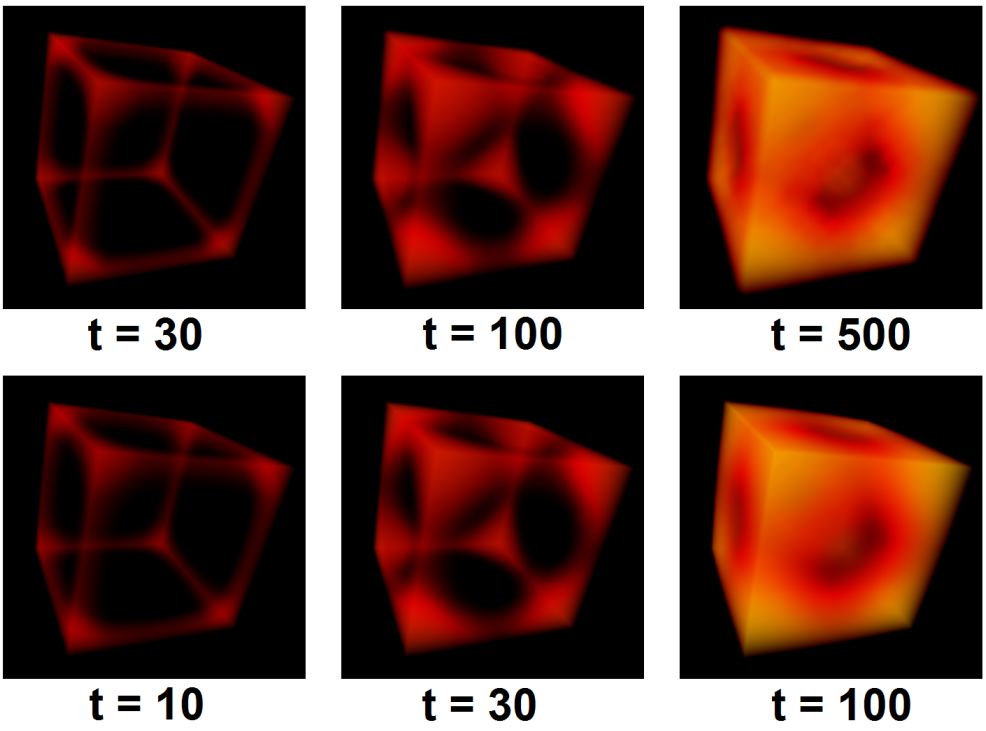


Figure 2.5: Visual comparison of the identical results obtained by using the conventional heat diffusion (top) and the cumulative heat diffusion (bottom). Fewer number of time steps is required by the cumulative heat diffusion than the conventional heat diffusion to obtain identical results.

certain time, all the voxels from a single object will have similar heat values and hence will converge to a single line in the histogram. Hence, by using the histogram, the cumulative heat diffusion is used in shape-based object classification in volume data.

We use the foot dataset to show the object classification using the cumulative heat diffusion. Figure 2.6 shows the result of our cumulative heat diffusion on the foot data for 10^6 time steps. The rendering in Figure 2.6 is based on the heat values after the last time step. The diffusion process and the heat values of the voxels for all the time steps can be visualized in the histogram. Each line in the histogram represents a shape in the foot data. Objects with similar shape have their corresponding histogram lines also close to each other. The color range is varied such that all similar shaped objects are assigned similar colors. The color of the line in the histogram is synonymous with the color of the object in the data.

The distance between the lines in the heat histogram is an indication as to how similar or dissimilar each object is compared to others in terms of shape, as the

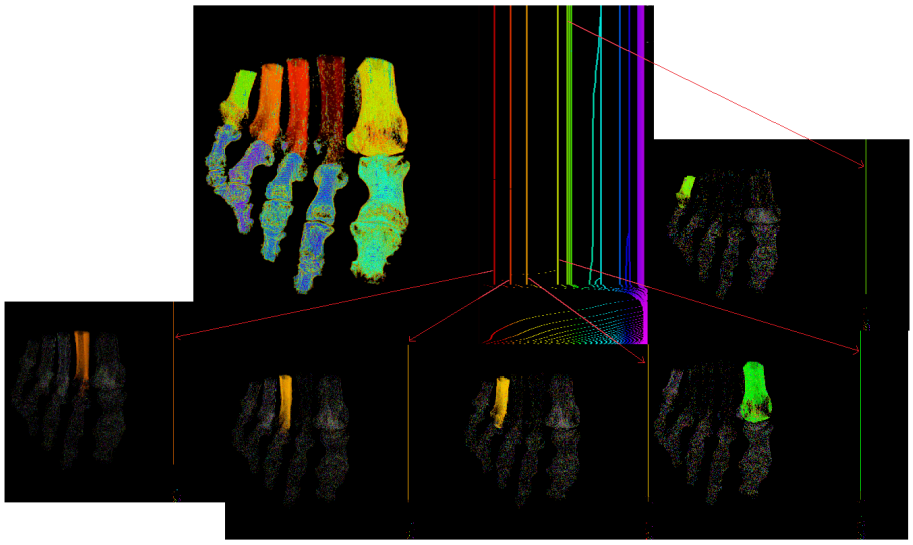


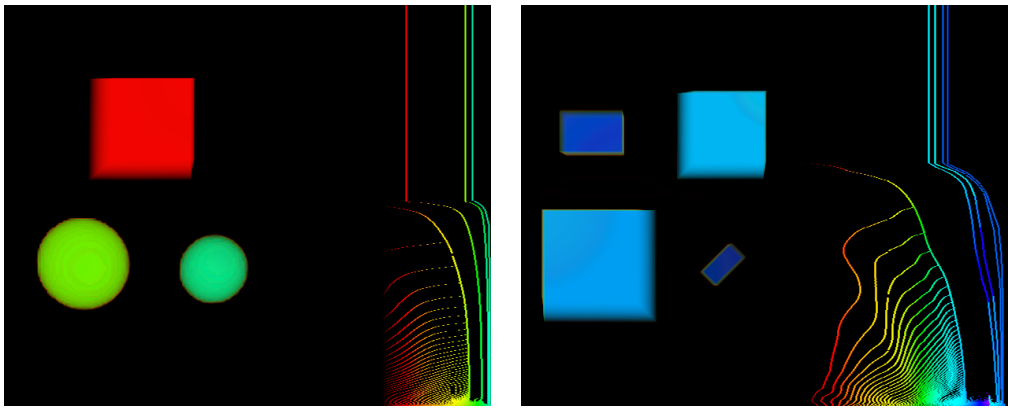
Figure 2.6: The classification of different bones with different shape in the foot dataset. Each of the lines in the histogram represents a certain bone with a certain shape. All the bones with similar shape are depicted by lines that are very close to each other. The top half of the three toe bones in the middle are identified by the first three lines in the histogram, which are relatively close to each other, thus showing that the bones are almost similar in shape. The results, by separately focusing on the individual lines, are also displayed showing each of the toe bones separately.

difference between the heat values of similar shaped objects is very small. The lines of objects with similar shape are closer when compared to the lines of the objects that do not have similar shape. For example, in Figure 2.6 we can see that the first three lines in the histogram represent three toe bones (in the middle) at the top of the foot. All three bones are more similar in shape to each other than the other bones and hence the corresponding lines in the histogram are closer. In this way, the similar objects in a volume dataset can be classified from other shapes with the aid of the heat histogram.

2.3.4 Shape Analysis

Cumulative heat diffusion displays advantageous properties such as invariance to scaling and orientation. These have been proved both theoretically and experimentally in several previous works [1, 24, 130, 131]. We generated two synthetic data sets to support the scale and orientation invariance of our method. The first synthetic data consists of a cube and two spheres of different sizes. The second synthetic data consists of two cubes and two cuboids with different sizes and ori-

entation. Figures 2.7(a) and (b) show the result of our cumulative heat diffusion on the first and second synthetic data respectively for 40000 time steps. In both Figures 2.7(a) and (b) the rendering is based on the heat values of the voxels after the last time step. The diffusion process and the heat values of the voxels for all the time steps can be visualized in the histogram.



(a) (b)

Figure 2.7: Invariance to scale and orientation of our approach using synthetic data sets. In (a) and (b) objects with different shape are identified. (a) The lines in the histogram representing the two spheres (different sizes) are close to each other identifying them as similar shape, establishing invariance of scale; (b) The two cubes are of different size, while the two cuboids are of different size and orientation. The corresponding lines in the histogram are close to each other identifying them as similar shapes, showing invariance to scale and orientation.

The histogram in Figure 2.7(a) shows that the final heat of all the voxels in the volume converges to one of the three lines indicating that there are three different objects in the given data. Of the three lines, two lines which pertain to the two green spheres are very close to each other. The line pertaining to the cube is far away from the two lines showing that the two spheres, though of different sizes, are classified as similar shape and the cube as a different shape. This shows the scale invariance of our approach. Similarly, the histogram in Figure 2.7(b) shows that the final heat of all the voxels in the volume converges to one of the four lines indicating that there are four different objects in the given data. All the four lines are close to each other showing that the objects in the data are almost of similar shape. In addition, for the four lines, the distance between the left two lines (pertaining to the two blue cubes) and the distance between the right two lines (pertaining to the two purple cuboids) is smaller than the distance between the two lines in the middle. This in turn indicates that there are two distinct pairs of objects, namely a cube and

a cuboid with similar shape, which is true for the given data. Despite the different dimensions and orientation, the final result correctly recognises all the shapes, thus showing that the cumulative heat diffusion is invariant to both orientation and scale.

As discussed earlier, we use the normalized volume gradient operator to assign the initial heat value to all the voxels. In addition, the edge weights are also determined by the volume gradient operator. The calculation of volume gradient operator itself involves the use of the absolute value of the half gradient. This use of the absolute values in all the calculations makes sure that the cumulative heat diffusion is not affected by change in the orientation and scaling of the objects in the data. This property of our method is very useful to detect all objects with similar shape at different scales and orientations in the given volume simultaneously. All the objects with similar shape irrespective of their scale and orientation would converge to very close lines in the heat histogram.

2.4 Parameter

The shape information obtained does not entirely depend on the time step being considered. It also depends on how clearly the boundary is defined between different objects so that they can be clearly identified as different shapes. If the boundary between two objects or shapes is not clear then our approach will see both the objects as a single object of certain shape and the heat flow will work its way accordingly. In addition, we have also seen that sharper features take more time steps to be recognised when compared to flat surfaces as the rate of diffusion is slower at sharper areas than on the flatter areas. Choosing the boundary will also decide the amount of sharpness and smoothness in the volume data desired by the user. Choice of clear boundaries between objects and the time step is dependent on the application and is entirely in the hands of the user. Using the histogram the user can select the time step of his choice to understand the different shape-based features present in the volume at different time steps or scales. We discussed the different results obtained by choosing different time steps in the previous sections. We will now discuss an application of our approach where the user can decide the boundary between the objects based on his requirement.

In our volume gradient operator definition, we used a user defined parameter, p. This parameter is used to decide the boundaries of the objects by the user for the given data. The volume gradient operator is a function of the half gradient and gradients work well in identifying the object boundaries and edges [110]. Using the parameter p along with the half gradient in the volume gradient operator definition will give the user the extra flexibility to make a choice of the object boundaries. Hence, by changing the value of p, the user can draw a clear boundary between

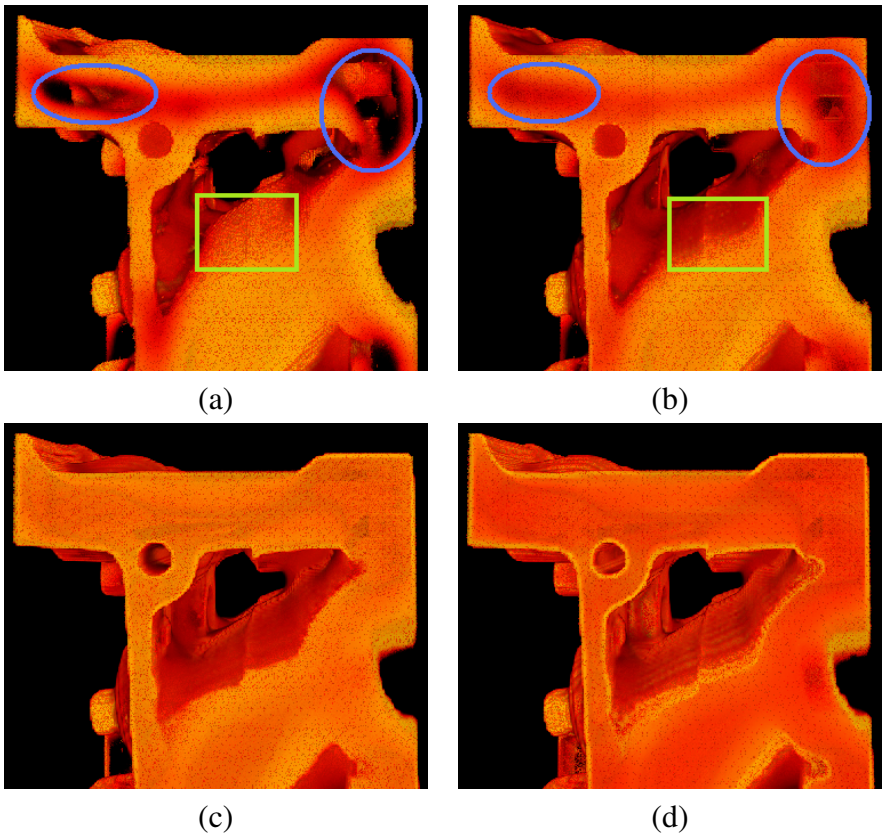


Figure 2.8: Comparison of choosing different values for p, which is a user determined quantity. (a), (b), (c) and (d) are the results obtained by choosing p = 2, 5, 10 and 15, respectively on the engine dataset for 100 time steps. (a) and (b) show a clear difference in features obtained (enclosed in blue ellipses and green boxes). Though the features obtained in (c) and (d) are almost similar, the features are sharper in (c) than in (d).

different objects and also decide on the level of smoothness or sharpness in the data. We leave it as a user dependent parameter so the user can choose the value based on his/her requirement. This choice of p made by the user will also affect the amount of heat flow during the cumulative heat diffusion process. Thus the results of cumulative heat diffusion might vary locally or globally for different values of p based on how well the objects are distinguished and how sharp the features are.

Figure 2.8 shows the result of choosing different values of p on the engine dataset. Figures 2.8(a)-(d) show the result when $p = 2$, 5, 10 and 15 respectively for 100 time steps. In Figures 2.8(a) and (b), there is a lot of difference in the features obtained such as those marked in blue and green. In Figures 2.8(c) and (d), though the features present are almost the same, the features in Figure 2.8(c) are clearer

and sharper than those in Figure 2.8(d). The features in Figures 2.8(c) and (d) are better than those in Figures 2.8(a) and (b). The small round hole on the engine is clear and the edges of the big hole are sharper. Thus, based on what features the user wishes to focus on and what features the user wants to analyze, the user can choose different values of p.

2.5 Results

The heat values obtained at the end of a time step will incorporate the shape information at that time step. This information can be used to design a shape-based transfer function. By choosing different time steps, the shape information obtained from the volume data at different scales can be visualized at once in the histogram, thereby aiding a volume exploration based on shape. By choosing a small number of time steps, local shape can be obtained, and by choosing a large number of time steps, global shape information is obtained. Thus, the heat histogram is used to design the transfer function to visualize both the local and global shape of the data. We provide a tool for the user to design a transfer function using the heat histogram. After the cumulative heat diffusion process is carried out for a large number of time steps, the corresponding heat histogram is generated using the equalized heat values obtained. The user can then choose a desired time step (vertical axis), which provides all the shape information (heat values) obtained until that time step. The user can then assign different colors and opacities to these heat values (horizontal axis), which forms a 1-D transfer function for exploring the data based on the shape information.

Figure 2.9 shows the result of our shape-based transfer function on MRI head data for $t = 4 \times 10^5$. We were able to segment different parts of the data such as cranium (outer part), brain and eye sockets based on the shape information. Figure 2.9(a) shows all the segmented parts of the MRI head data by using our transfer function. Figure 2.9(b) shows just the outer cranium that covers the brain while Figure 2.9(c) shows the brain. The cerebral fluid around the brain is captured in Figure 2.9(d).

The results of the transfer function can also be varied by varying the value of p. As the cumulative heat diffusion process is not real time, if the value of p is changed, the cumulative heat diffusion is repeated on the entire data for the required number of time steps and a new heat histogram is obtained which can then be used for transfer function design.

Figure 2.10 shows the result of our approach on aneurysm data. The aneurysm is correctly classified into an isolated blob (yellow), the thin vessel regions (green) and a large vessel (red). All the thin vessels are tube like structures and have similar

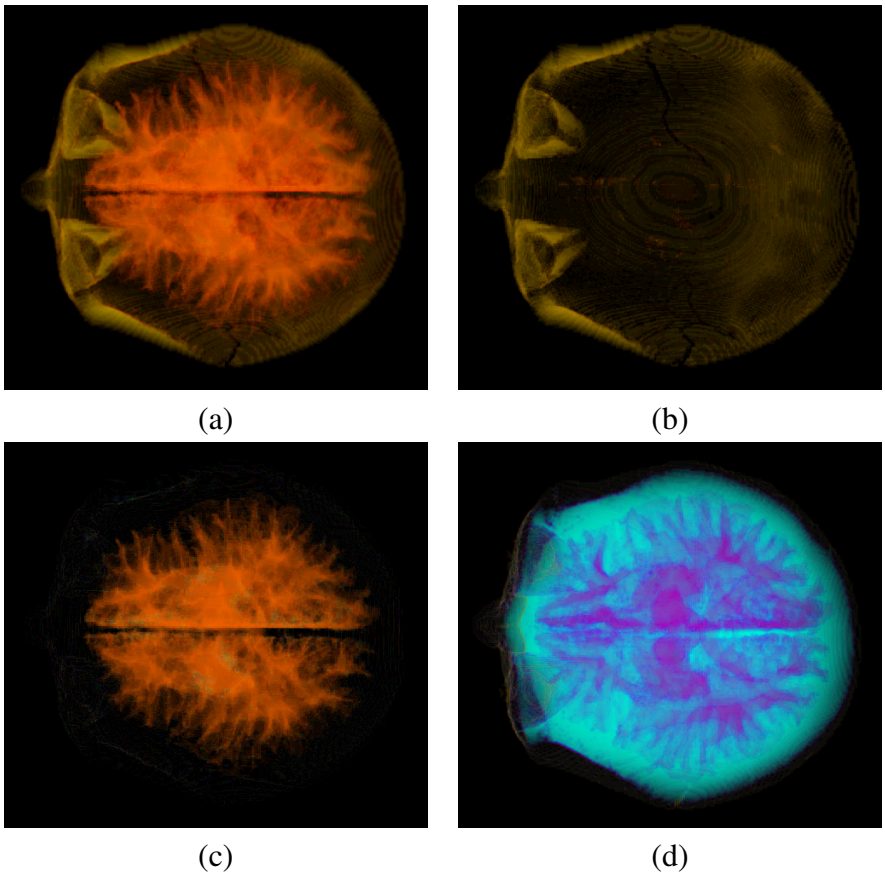


Figure 2.9: Results of the shape-based tranfer function designed using the heat histogram on the MRI head dataset. (a) The successfully classified cranium, brain and eye sockets; (b) The segmented cranium; (c) The segmented brain; (d) The cerebral fluid around the brain.

shape. Hence, the cumulative heat diffusion method classifies all the thin vessels as the same shape (tube like structures) and hence have similar color. The aneurysm (blob) in the data has a completely different shape and hence it was clearly recognised as shown in Figure 2.10. It is important to note that the intention of our method is to provide a shape-based volume visualization and not to provide a reliable segmentation technique. Generally in visualization, segmentation errors do not pose a big problem and can be tolerated. Though Figure 2.10 provides a "weak" segmentation (the yellow color seen in the red area and the green area) all the shapes can still be clearly discerned, providing very good shape-based volume visualization. We do not propose to use our method as a reliable segmentation method or use it as a sole segmentation method. Instead, we demonstrate that our cumulative heat diffusion based volume analysis provides information about shapes, which can be

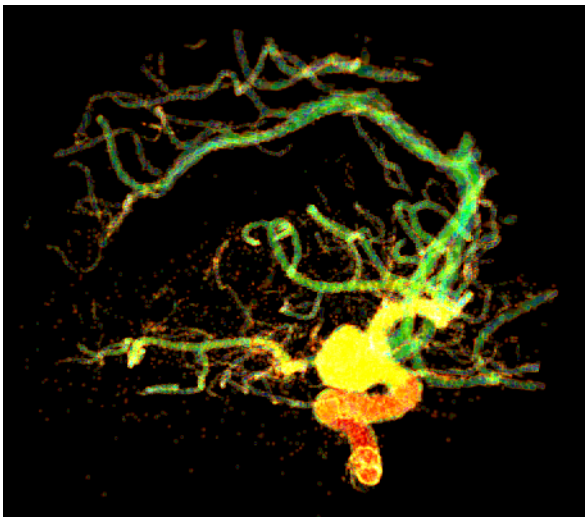
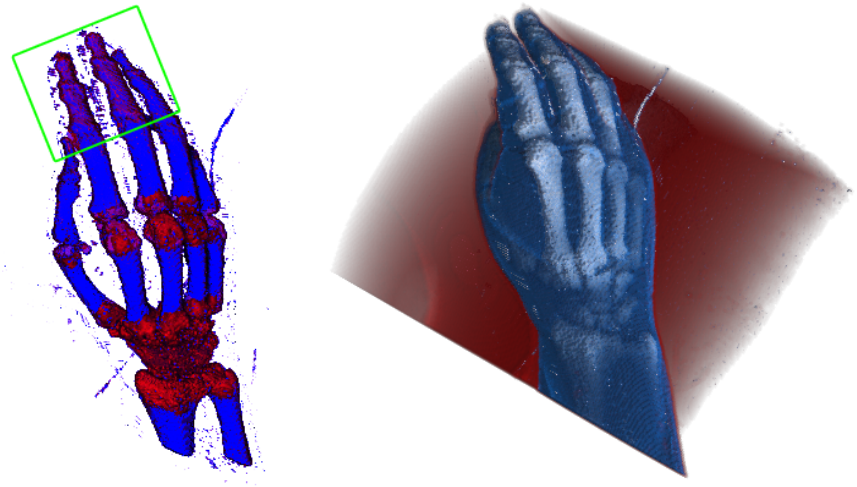


Figure 2.10: Objects with different shape in the aneurysm volume data. The aneurysm blob (yellow), the thin vessels (green), and the large vessel (red) are correctly distinguished based on their shape.

exploited to build improved tools for shape-based volume visualization. The data was run for a sufficiently long time in order to capture all the shape information in the dataset. We ran the cumulative heat diffusion on the data for 10^5 time steps.

In Figure 2.11, we show the result of our diffusion process on the visible female right hand dataset. Figure 2.11(a) shows the result of considering a small number of time steps of t = 1000. Hence, we can see local shape information, which in this case provides simple yet useful information. We can observe from Figure 2.11(a) that all the joints between the finger bones and the joint at the wrist have the same shape and thus the same color (red). Similarly, all the finger bones have similar stem-like shape and thus have the same color (blue). This can also be confirmed visually from the figure. The joint present at the wrist is also identified to be of similar shape as that of other joints though it is of larger size, which shows that our method successfully detects similar shaped features irrespective of size and orientation. However, we can see mixed colors of red and blue in the green box. The region inside the green box consists of two finger joints in a very small space and thus the entire region has been classified as a shape that is in between the shape of the stem-like bones and the shape of the joints. As the difference between the different shapes is very subtle in this region, the colors have been mixed. Figure 2.11(b) shows the result of considering a large (steady) number of time steps of t = 10^6. At this steady state time, all the global shape information is obtained. In this case, we can observe that it has been classified into three parts, namely the bone (white),



(a) (b)

Figure 2.11: The results of classifying the visible female right hand for (a) small (t = 1000), and (b) large (t = 10^6) number of time steps. (a) All the joints between the finger bones and the joint at the wrist have the same shape and hence the same color (red). All the fingers have similar stem-like shape and hence the same color (blue). (b) The hand has been classified based on the global shape information into bone (white), skin (blue) and the area on which the hand rests (red).

the skin (blue) and the area on which the hand rests (red). These results show the power of our method in utilising both the local and the global shape information for visualizing similar features and objects in otherwise complex volumes.

A comparison in terms of the running times by using our approach and directly using the conventional heat diffusion on different volume datasets is given in Table 2.1. The third column of the table shows the number of iterations (time steps) for which the cumulative diffusion process was carried out. The fourth column shows the value of p used for the datasets. The fifth column gives the time taken per iteration by the conventional heat diffusion (HD in Table 2.1) process on volumes (we run the HD for only 1 iteration to get the time) while the last column gives the time taken per iteration by our approach of using cumulative heat diffusion (CHD in Table 2.1) on volumes. We can clearly see that in all the cases, for the same number of iterations, the time taken for the cumulative heat diffusion process is smaller than the conventional heat diffusion by a very large factor, thus showing that our method reduces the computational time drastically. All tests were conducted on a system equipped with an Intel Xeon E5620 CPU and NVIDIA GeForce GTX 480 graphics board.

Table 2.1: Comparison of the running times of our cumulative heat diffusion with the conventional heat diffusion for the same number of time steps.

Dataset	Resolution	# Iterations (time steps)	p	Time (sec) per HD iteration	Time (msec) per CHD iteration
Foot	$256 \times 256 \times 256$	$1x10^5$	7.0	80,531	4.8
Engine	$256 \times 256 \times 128$	$5.2x10^4$	10.1	11,744	1.4
Aneurysm	$256 \times 256 \times 256$	$1x10^5$	7.5	85,564	5.1
Abdominal Stent	$512 \times 512 \times 174$	$1x10^5$	8.0	2,098,201	46.1
MRI Head	$256 \times 256 \times 256$	$4x10^5$	6.0	352,322	21.3
Visible Female Hand	$176 \times 187 \times 190$	$1x10^6$	10.0	112,559	18.2
Bonsai	$256 \times 256 \times 256$	$1x10^6$	8.5	845,572	50.4

Chapter 3

Stochastic Shape Analysis

In this chapter, we introduce a Monte Carlo based real-time diffusion process for shape-based analysis of volumetric data. The diffusion process is carried out by using tiny massless particles termed shapetons, that are used to capture the shape information. Initially, these shapetons are randomly distributed inside the voxels of the volume data. The shapetons are then diffused in a Monte Carlo fashion to obtain the shape information. The direction of propagation for the shapetons is monitored by the volume gradient operator. This operator is known to successfully capture the shape information and thus the shape information is well captured by the shapeton diffusion method. All the shapetons are diffused simultaneously and all the results can be monitored in real-time. We demonstrate several important applications of our approach including colon cancer detection and design of shape-based transfer function. We also present supporting results for the applications and show that this method works well for volumes.

The rest of this chapter is organized as follows. Section 3.1 provides the motivation. Section 3.2 describes the algorithm with a detailed description of the shapeton diffusion process. We discuss how our shapeton diffusion method can robustly extract the features based on their shape information in the volume data. Various properties of the method are discussed in Section 3.3. The influence of different parameters on the result is analyzed in Section 3.4. Section 3.5 discusses applications of our method to colon cancer detection and transfer function design and Section 3.6 presents the results of the method.

3.1 Motivation

The cumulative heat diffusion approach is a cumulative shape analysis approach and is computed by considering all the voxels as the sources for diffusion. As a result, the cumulative heat diffusion process is dependent on the resolution of the data. Higher resolution of the data leads to higher computational time. In

addition, the diffusion is carried out only between voxels and 1-ring neighboring voxels per time step and hence the number of time steps required to capture the shape information increases with the increasing number of voxels. In other words, the rate of heat flow is influenced by the resolution of the data. As a result, the diffusion based methods suffer from the problem of long running times. Monte Carlo methods are popular due to their ability to quickly approximate an answer that otherwise would be very time-consuming to determine. Such probabilistic methods are used to simulate problems that are too difficult and time-consuming as compared to other methods. The motivation of this chapter is to address these challenges using a Monte Carlo approach. We introduce a Monte Carlo based shape analysis method for volumes which not only obtains efficient results but also provides a means of real-time shape analysis.

The new diffusion based shape analysis method uses new particles, termed as shapetons. This is the first time the diffusion particles (in our case, the shapetons) are diffused across the voxels separated by some distance, rather than just between the adjacent voxels. The method is independent of the size of the volume; it only depends on the number of shapetons. This independence on the resolution (size) of the data is an important contribution. In addition, using probabilistic methods for shape analysis is in itself a contribution. Furthermore, to the best of our knowledge, this is the first time volume analysis based on shape with real-time monitoring of the result is being carried out. Consequently, we achieve orders of magnitude improvement in the computational cost compared to other diffusion based shape analysis methods for volumes.

3.2 Algorithm

The shapeton diffusion process efficiently captures the shape information in volumes and in addition facilitates a real time monitoring of this information. The diffusion particles, the shapetons, are able to capture the majority of the shape information and hence the name shapetons. Initially, these shapetons are randomly distributed inside the data. The primary idea of our approach is that each shapeton is diffused based on the local shape information in a probabilistic manner. The probability that a shapeton moves in a particular direction is based on how much the region in that direction contributes to the shape information. In continuous space, generally the shape information around each shapeton can be represented in the form of an uneven distribution. This is because the local shape information around the shapeton is not uniform (varies depending on the data). The area of this shape distribution would give a measure of the shape information obtained around the shapeton. A random number is used to select a fraction of the area. This fractional

area indicates the shape information obtained in that direction and in turn the probability for the shapeton to move in that direction. In other words, the probability of shapeton diffusion is based on the ratio of the area of a sub-region to the area of the total shape distribution.

For the sake of simplicity, an illustrative example of shape distribution in the 2D case is shown in Figure 3.1. In Figure 3.1, s is the shapeton and the boundary around it indicates the shape distribution around that shapeton. Assume the area of the total shape distribution to be A. A random number r is chosen such that $r \in (0,1]$. If we assume the interval $(0,1]$ to be a straight line of length 1, then r will represent a position which is at a distance of r from the starting point of 0 (length of the blue line segment) as shown in Figure 3.1. If we map the total area, A of the shape distribution to this line, then choosing r will be analogous to choosing a sub-region of the shape distribution whose area will be $A1 = r \times A$ (shown by the blue region in Figure 3.1). This $A1$ gives the measure of the shape information obtained from that particular region and the probability for the shapeton to propagate in that direction, as indicated by the arrow. Hence, the probability of the shapeton to propagate in a particular direction is given by $r = \frac{A1}{A}$. By evaluating the angle enclosed by this randomly chosen region, the direction of the shapeton propagation is estimated. As can be seen, the region to the left of the shapeton covers more area which indicates that relatively more shape information is obtained in that region. Consequently, the shapeton also has higher probability to move in that direction. Hence, it is essential to first estimate the shape distribution around each shapeton in order to determine its direction of propagation.

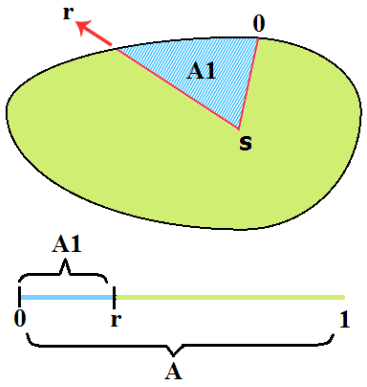


Figure 3.1: The shape distribution around the shapeton s. The probability of the shapeton diffusion to propagate in the direction shown by the arrow is based on the ratio of the areas of the region shown in blue to the total area of the shape distribution.

We deal with voxel-based volumes and hence only deal with discrete space of shape information. In discrete space, the shape information is measured in terms

of the edge weights (around each voxel) evaluated using the volume gradient operator. These edge weights are distributed linearly based on the angle between the weights (linear interpolation of the edge weights). The shape distribution around the shapeton is estimated by evaluating the edge weights and the areas between them in the discrete case. Thus, the area between the edge weights would give a measure of the shape information contributed by that particular region. In the discrete case, the red lines in Figure 3.1 are nothing but the edge weights and the area between the weights is calculated by using the angle between them. This information captures the local shape information and is used to decide the direction of the shapeton diffusion. The probability evaluation and the estimation of shapeton propagation direction is same as explained above for the continuous case. The final accumulated density of shapetons after the entire diffusion process indicates the amount of shape information obtained.

3.2.1 Volume Data

The difference between the shapeton diffusion method and previous diffusion based methods such as the cumulative heat diffusion is that, this is a particle-based (shapetons) diffusion process while the latter is not. In cumulative heat diffusion, the medium of diffusion is heat which is distributed on the voxels and during the diffusion process the heat flows only between the 1-ring neighboring voxels along the edges joining them in each time step. Therefore, conventionally the diffusion process is highly dependent on the resolution of the data, that is, finer data (higher resolution) would require more time steps while a coarser data (lower resolution) would need fewer time steps to capture the same amount of shape information. On the contrary, in this approach, during the diffusion process the shapetons are moved inside the volume, across the voxels, for a pre-defined distance in each time step. As a result, the shapetons have the freedom to move anywhere inside the volume and not just between the 1-ring neighboring voxels. Therefore, the rate of shapeton diffusion is not affected by the resolution of the data and is independent of the size of the data. Please note that all the pre-defined distance values are chosen by considering a $[0, 1]$ normalized space of the volume data. Hence, the distance value will always lie in the interval $[0, 1]$.

Initially, all the shapetons are randomly distributed inside the voxels. The shapetons accumulate most of the shape information along their path and consequently the shape information is obtained at a much faster rate. In order to describe the direction along which the shapetons travel in each time step, two angles are used, namely the longitudinal angle and the latitudinal angle. We now describe the elaborate process of shapeton diffusion in detail. In general, any voxel is surrounded by six adjacent voxels in a volume. Thus, for any shapeton s inside a voxel (say

v_0), there are six adjacent voxels (say v_1, v_2, v_3, v_4, v_5 and v_6). The edge weights

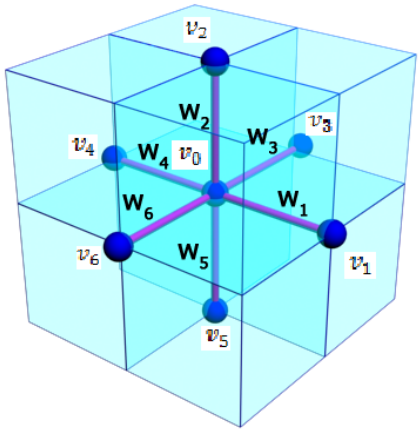


Figure 3.2: $v_1, v_2, v_3, v_4, v_5, v_6$ are the 1-ring neighboring voxels of the source voxel v_0 and w_1, w_2, w_3, w_4, w_5 and w_6 are the corresponding edge weights, respectively.

w_1, w_2, w_3, w_4, w_5 and w_6 between the voxel v_0 and its adjacent voxels, as shown in the Figure 3.2, are determined using the volume gradient operator, defined by Equation 3.1. This volume gradient operator captures the local shape information of the volume. There is a parameter p in the volume gradient operator definition that influences the final result. We discuss the effect of the parameter p in Section 3.4.3. For $i \in \{1,2,3,4,5,6\}$:

$$w_i = VGO(v_0, v_i) = \Delta(v_0, v_i) + F_v(v_0, v_i) \tag{3.1}$$

where Δ is the Laplace-Beltrami Operator and F_v is a data-driven operator:

$$F_v(v_0, v_i) = 1 - p \cdot h_g(v_0, v_i) \tag{3.2}$$

where h_g is the half gradient and p is a user defined value. The half gradient h_g of the voxel v_0 is given by:

$$h_g(v_0, v_i) = |\frac{I(v_i) - I(v_0)}{res}| \tag{3.3}$$

where I gives the intensity of the corresponding voxel, *res* is the size of the voxel which accounts for the distance between the two voxels under consideration.

We use these six edge weights to create a shape distribution diagram around the shapeton, as shown in Figure 3.3. This shape distribution accounts for the shape information around the voxel v_0 (the shapeton is inside this voxel) and is used to determine the direction of shapeton diffusion in a probabilistic manner. The six

weights form eight regions where each region represents an octant of a sphere. We call this octant of the sphere octavusphere (derived from Latin). Therefore, the six weights form eight octavuspherical regions where sets of three weights form a single octavuspherical region, as shown in Figure 3.3. In spherical coordinates we normally need two angles (say θ and ϕ) to describe the direction of shapeton propagation. The angle θ is measured with respect to the x-axis on the $x-y$ plane and the angle ϕ is measured with respect to the y-axis on the $y-z$ plane. In geographical terms, we refer to the angle θ as the longitudinal angle and the angle ϕ as the latitudinal angle.

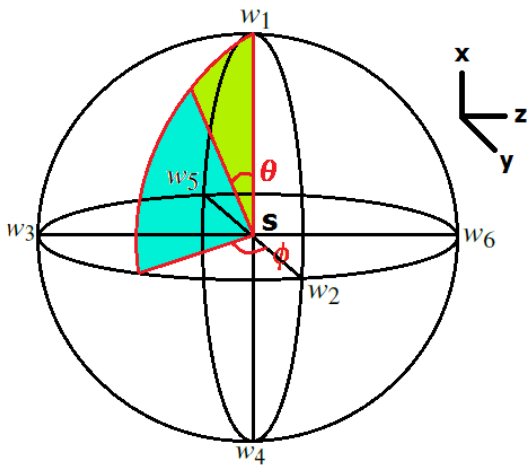


Figure 3.3: The shape distribution around the shapeton s shown using the edge weights. The probabilistically estimated angles ϕ and θ define the direction of the shapeton propagation.

Since we have to determine two angles probabilistically, namely θ (longitude) and ϕ (latitude), two random numbers are drawn, one for each of them. We do it in a step-by-step manner. Firstly, the value of the angle ϕ is determined by employing the first random number. Fixing this value of ϕ, the value of the angle θ is then estimated by employing the second random number. In a given octavuspherical region both ϕ and θ vary between 0 and $\frac{\pi}{2}$. As explained earlier, the probability of the shapeton diffusion should take into account the shape information around it, which is indicated by the volume of the octavuspherical region enclosed by the edge weights. In other words, the probability of the shapeton to move in a certain octavusphere is based on the ratios of the volumes of the octavuspheres to the whole volume. The volumes are evaluated by using the integral in Equation 3.4 involving the edge weights and the angles between them. By employing the first random number over the volumes of the octavuspherical regions, a particular octavusphere region is selected and the corresponding value of ϕ is estimated. This angle ϕ

splits the selected octavuspherical region into two sub-regions, which are separated by a sector shown by the green and blue regions in Figure 3.3. By employing a second random number over the area of this sector the final value of θ is estimated. More details about the steps involved in calculating the longitude and latitude for shapeton propagation are given below.

- We evaluate the volume of the eight octavuspheres (say V_1, V_2, V_3, V_4, V_5, V_6, V_7 and V_8). To be specific, the volume of region i is V_i where $i \in \{1,2,...,8\}$. The formula to calculate the volume of the octavusphere formed by the weights (say w_1, w_2 and w_3) is given by the integral shown in Equation 3.4. A more detailed explanation and derivation of the formula is provided in Chapter 7.1.

$$V = \int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{2}} \frac{f^3}{3} \cdot \sin(\theta)\, d\theta\, d\phi \tag{3.4}$$

  where the angles θ and ϕ define the longitude and latitude angles, respectively, as mentioned earlier. Here, f denotes the shape information in a given direction. It is a function of angles θ and ϕ and is determined by the edge weights of that octavuspherical region. We assume that the shape information is distributed linearly based on the angles θ and ϕ. Thus, f is defined as follows:

$$f = \frac{(w_1 \cdot (\frac{\pi}{2} - \theta) + W \cdot \theta)}{\frac{\pi}{2}} \tag{3.5}$$

  W is given by:

$$W = \frac{(w_2 \cdot (\frac{\pi}{2} - \phi) + w_3 \cdot \phi)}{\frac{\pi}{2}} \tag{3.6}$$

  Finally, the sum of the volumes of the eight octavuspherical regions $V = V_1 + V_2 + V_3 + V_4 + V_5 + V_6 + V_7 + V_8$ is computed.

- One of these octavuspherical regions is selected in a probabilistic manner by using a Russian roulette. For this, initially a random value is chosen in the interval (0,1]. This value is multiplied by the sum of the volumes V to obtain a fraction of the volume, say V'. Based on the value of V' one of the octavuspherical region is selected, using the condition shown in Equation 3.7.

$$\begin{aligned} &\text{If, } 0 < V' \leq V_1, \text{ then region 1} \\ &\text{If, } \sum_{i=1}^{k-1} V_i < V' \leq \sum_{i=1}^{k} V_i, \text{ and } k > 1, \text{ then region } k \end{aligned} \tag{3.7}$$

  This selection of the octavuspherical region based on the value of V' also results in the selected region being split into two sub-regions, with one of the

regions forming an angle ϕ, as shown in Figure 3.3. This angle ϕ is what has to be determined. The volume of the sub-region enclosed by the angle ϕ is given by $\sum_{i=1}^{k} V_i - V'$ where k is the region selected. Using this volume, the angle ϕ is then estimated by solving the equation of the volume shown in Equation 3.4 for ϕ (taking θ to be between 0 and $\frac{\pi}{2}$). Since the evaluation of the volume involves a complex fourth degree equation in ϕ (refer to Chapter 7.1), we do not solve this equation to determine ϕ. Alternatively, we use a binary search approach to estimate the angle ϕ. We perform a binary search on the volume of the selected octavusphere to estimate the value of ϕ. On average, the binary search required four iterations to compute the ϕ value. More details about how we employ the binary search are described in Chapter 7.3.

- Now that the angle ϕ is decided, we have to determine the angle θ to completely define the final direction of shapeton propagation. As described earlier, the angle ϕ divides the selected octavusphere into two sub-regions. The two sub-regions are separated by a sector between them. The area of this sector is used to probabilistically detemine the value of θ. In Figure 3.3, this sector is shown by the green and blue regions.

- The area A of the sector is computed by solving the integral shown in Equation 8.7. A more detailed explanation and derivation of the forumla is provided in Chapter 7.2. Thus, the area A is given by:

$$A = \frac{1}{2} \cdot \left(\int_0^{\pi/2} r^2 \, d\theta' \right) \tag{3.8}$$

for some arbitrary angle θ' between 0 and $\pi/2$. r is defined using linear interpolation of weights and the angles between them (which is $\pi/2$ in this case) as follows:

$$r = \frac{(w_1 \cdot (\frac{\pi}{2} - \theta') + W' \cdot \theta')}{\frac{\pi}{2}} \tag{3.9}$$

where W' is obtained using Equation 3.6 by replacing ϕ with the already estimated ϕ value.

- After the area of the sector is computed, a random number is chosen in the interval (0,1]. This value is multiplied with the area A to get a fraction of the area (say A'). In Figure 3.3, A is given by the sum of the areas of the green and blue regions and A' is given by the area of the green region. The angle enclosed by the sector whose area is A' (green area in Figure 3.3) is θ which is to be determined. By using the binary search approach (Chapter 7.3), we

find the value of θ for which the area of the sector is A'. On average, the binary search needed four iterations.

Thus, the angles of ϕ and θ are estimated by employing two random numbers in a Monte Carlo manner. Now that we have evaluated both ϕ and θ, we have the final direction for the shapeton to move. This entire process of evaluating the direction of propagation is carried out simultaneously for all the shapetons.

Once the direction of propagation for the shapeton is determined, the shapeton is moved in that direction for a pre-defined distance. This accounts for one time step of the shapeton. After each time step, all the steps described above are repeated to calculate the new direction for the shapeton to diffuse. Similarly, all the shapetons are diffused inside the volume for the required number of time steps. After each time step, the number of shapetons inside each voxel is summed up to get the accumulated density of shapetons. That is, the number of shapetons inside each voxel after each time step is added to the accumulated shapeton count obtained in that voxel through the previous time step. The accumulated number of shapetons in each voxel indicates the probability of the shapetons to appear at that location. For all the voxels corresponding to objects of similar shape, the shapetons have a similar probability to visit them. Hence, the number of shapetons within each voxel would be the same for all voxels corresponding to objects of similar shape.

The diffusion of shapetons in volumes is influenced by the volume gradient operator which incorporates the local shape information. Thus, the shapetons capture the shape information along their path of diffusion and the accumulated number of the shapetons inside the voxels quantifies the shape information obtained. Therefore, the final position of the shapetons after a certain number of time steps would accumulate all the shape information until that time step and can be used for analyzing the volume data based on shape. When the shapeton diffusion process is carried out for a large number of time steps, a stable state or convergence is attained, after which the result does not change much. Stable state indicates that all of the global shape information has been obtained. However, en route to the stable state, additional shape information such as the local features or smaller objects is obtained which is useful for volume analysis based on shape. The next section describes how different parameters can be manipulated to analyze both local and global shape information inside the volume data.

3.3 Properties

3.3.1 Shape Classification

We now show that the shapeton diffusion method is indeed successful in classifying different shapes. The shapetons are diffused based on the volume gradient operator in volumes, which captures the shape information. Hence, the probability of a shapeton to go in a particular path is influenced by the shape information. The accumulated number of shapetons per voxel in a shape such as a cube would be different from a shape such as a sphere since both of them have different shape and thus bear different probabilities for the shapetons to capture them.

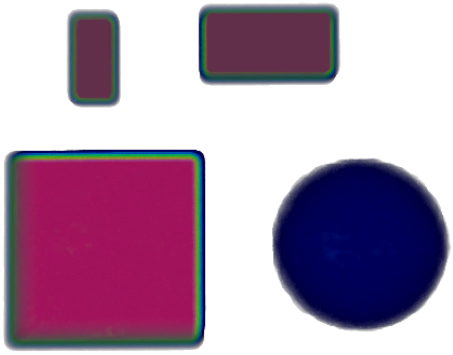


Figure 3.4: Shape classification capability of the shapeton diffusion approach shown using a synthetic data consisting of a cube, two cuboids of different size and orientation and a sphere.

We use a synthetic data consisting of a cube, two cuboids of different size and orientation and a sphere to confirm this. Figure 3.4 shows the result of using the shapeton diffusion method on the synthetic data. We consider a large number of 2,500 time steps to make sure that a stable state is reached. In each time step, the shapeton was moved by a distance of 0.01. We can clearly see from Figure 3.4 that all the shapes have been identified and distinguished successfully (shown by the different colors). The colors are assigned based on the number of shapetons accumulated. The shapeton propagation is based on the shape information (volume gradient operator) and hence the number of shapetons accumulated per voxel is the same in similar shaped objects. This fact can be observed in Figure 3.4 where both cuboids have the same color. In addition, the color of the cube is almost similar to that of cuboids indicating that they have almost similar shape. The cube and the sphere have also been classified as different shapes, thus asserting that the shapeton diffusion method serves as a powerful tool in finding objects with similar shape and distinguishing them from objects with other shape. An important observation

that can be made from Figure 3.4 is that both cuboids have been identified as similar shape irrespective of their size and orientation. This further confirms that the shapeton diffusion method can recognize different shapes independent of their size and orientation.

3.3.2 Invariance to Deformations

The shapeton diffusion method displays some lucrative properties such as invariance to deformations. We generated a synthetic data to establish this property. The synthetic data consists of a cuboid, a deformed cuboid, a sphere and a deformed sphere. Figure 3.5 shows the result of the shapeton diffusion on this synthetic data. Again the diffusion process is carried out for a large number of time steps to ensure a stable state is reached and all the objects in the volume data are obtained.

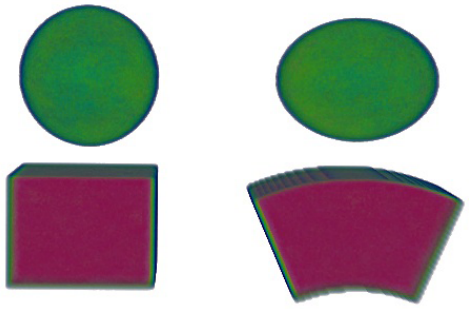


Figure 3.5: Objects of similar shape identified successfully irrespective of their deformations.

You can observe that although the cuboid has been deformed, the number of shapetons accumulated per voxel in both the cuboid and its deformed version are the same and hence both have similar color. Likewise, the sphere and its deformed version have similar color. The sphere and the cuboid have also been distinguished from each other. We used 1000 shapetons for 2500 time steps to obtain the results. The results in Figure 3.5 show that the shapeton diffusion method is successful in identifying objects of similar shape though they have been deformed, thus showing that it is invariant to deformation.

3.4 Parameters

The shapeton diffusion process is an efficient method in classifying different objects based on their shape. The shape information is obtained irrespective of the size and deformation of the objects. However, the amount of shape information

obtained is influenced by a number of parameters such as the number of shapetons, the value of the pre-defined distance, and the value of p. We provide a detailed analysis of the effect of each of the parameters on the shapeton diffusion process. In the remainder of the paper, for all the results, the rendering is based on the accumulated number of shapetons in the voxels for a given number of time steps and the colors are assigned such that a higher shapeton count is shown in red and the color changes from red to blue with the decrease in the shapeton count.

3.4.1 Steady State and Number of Shapetons

For any Monte Carlo method the higher the number of samples, the better the results. The same is true with our method as well. As we increase the number of shapetons, the probability of the shapetons to take a particular path increases and hence the rate of accumulation of shapetons at a particular feature increases. Thus, the shape information is obtained much faster in terms of the number of iterations with the increase in the number of shapetons. If the number of shapetons is reduced, it takes more iterations to capture a specific feature, which otherwise would have taken fewer iterations using more shapetons. However, there is a tradeoff. Though the number of iterations decreases, the time taken for each iteration (time step) increases with the increase in the number of shapetons.

We say that the shapeton diffusion process has reached a steady state if the rate of change of the accumulated shapeton density on all the voxels is uniform. For this, we check if the rate of change of the accumulated shapeton density on all the voxels after every time step ($\Delta t = 1$) is below a threshold value as follows:

$$\Delta s(t) = \sum_{i \in V} (s_t(i) - s_{t-1}(i))^2 \leq \varepsilon \tag{3.10}$$

where $\Delta s(t)$ denotes the rate of change in the accumulated shapeton density for all the voxels after t time steps, V denotes the number of voxels in the volume, $s_t(i)$ and $s_{t-1}(i)$ are the accumulated shapeton densities on voxel i after t and $t-1$ time steps respectively and ε is the threshold value. In all our datasets, we choose the threshold value to be 0.05. This threshold value chosen is not an accurate estimation and is chosen experimentally by observing the shapeton diffusion process on several datasets. As future work, we plan on finding a way to provide a more accurate estimate of the threshold value that will be dependent on the dataset. However, for now the threshold value works well for all our experiments. We check if the condition in Equation 3.10 is satisfied continuously in atleast 90% of the last 50 time steps. The 10% leverage is given to account for some unexpected changes caused due to the probabilistic movement of the shapetons. The number of time steps after which all these requirements are satisfied is chosen to be the point where

a steady state is reached. For example, assume for a particular data, the condition in Equation 3.10 is satisfied at $t = 100$ and furthermore the condition remains to be valid in 46 of the next 50 time steps (i.e., until $t = 150$), then the time steps needed to reach the steady state is taken to be 150. Once the steady state is reached, there would not be much change in the amount of shape information obtained.

Table 3.1: Comparison of the timings for different number of shapetons using a synthetic cube data.

# Shapetons	Time (msec) per time step	# Time steps (iterations)	Total time (sec) for convergence
4,000	0.057	7,100	0.40
8,000	0.087	4,400	0.38
15,000	0.129	2,800	0.35
30,000	0.215	1,600	0.34
65,000	0.236	1,500	0.35
150,000	0.310	1,200	0.37

Due to the probabilistic nature, the convergence to a steady state by using our approach is slow. This limitation is overcome by utilizing the parallel aspect of our approach and as a result achieving remarkable reduction in the computational time. The number of shapetons used for carrying out the shapeton diffusion should not be too small, otherwise it is possible that a steady state is never attained despite using any number of time steps. It is a case when the number of shapetons used are not adequate to diffuse over the entire mesh model to capture the shape information. Hence, sufficient number of shapetons are needed for the shapeton diffusion to be carried out. We will now discuss how an optimal value for the number of shapetons is chosen experimentally for each dataset.

A comparison of the time taken per iteration and the total time taken for convergence (reach a stable state) using different number of shapetons on a synthetic cube data is given in Table 3.1. The first column of the table shows the number of shapetons used. The second column of the table shows the corresponding time taken (in msec) per time step (iteration). The third column shows the total number of time steps required to reach the convergence. By convergence we mean that the entire global shape information of the cube is obtained, that is, the shapetons cover the entire cube volume (similar to the result shown in Figure 3.6(d)). The last column of the table shows the total time (in sec) taken for convergence by using

the number of shapetons shown in the first column. From the second column of Table 3.1, we can observe that the time taken per time step increases with increasing number of shapetons, which is expected. However, the total number of time steps needed to obtain the shape of the cube in each of the cases is not the same and keeps decreasing. This tradeoff between the time taken per time step and the total number of time steps gives an optimum value for the number of shapetons to be used for a particular dataset. In the synthetic cube case, we can observe that the total time taken decreases until the number of shapetons used is $30,000$ and later the total time increases to obtain the same result inspite of an increase in the number of shapetons. Thus, without any loss of generality, we can say that $30,000$ is the optimal value of the number of shapetons to be used for this cube data to obtain the result with minimal time. Similarly, different datasets have different optimal number of shapetons to be used.

We use the same synthetic cube dataset to show a comparison of the results obtained with the change in the number of shapetons keeping the number of time steps to be a constant. Figure 3.6 shows the results of the synthetic cube data using different number of shapetons for 1,600 time steps. Note that Figure 3.6 only shows a comparison of the results using the same number of time steps and does not consider the total time taken for these iterations as it differs based on the time taken per iteration. Figure 3.6(a) shows the result using $4,000$ shapetons; Figure 3.6(b) shows it using $8,000$ shapetons; Figure 3.6(c) shows it using $15,000$ shapetons, and finally Figure 3.6(d) shows the result using $30,000$ shapetons. The entire shape of the cube volume data is captured in Figure 3.6(d) in 1,600 time steps. However, using the same number of time steps some of the shape information is still missing by using fewer shapetons in Figures 3.6(a), (b) and (c). Moreover, the amount of uncovered regions increases with the decreasing number of shapetons. This comparison demonstrates that a decrease in the number of shapetons would require more iterations to capture the same amount of shape information.

3.4.2 Effect of the Distance Value

When a shapeton travels a pre-defined distance (defined by the user), it is said to complete one time step or iteration of the diffusion process. This distance value also affects the diffusion process of the shapetons and the shape information captured. When we use a large distance value, the shapetons travel a larger distance in one time step. Thus, if we increase the distance value the diffusion process converges faster in terms of the number of time steps when compared to a lower distance value. The smaller distance values cause the shapetons to move slowly, thereby resulting in more time steps needed to capture the global shape. However, the catch here is that we cannot obtain the local features using a large distance value because most of the

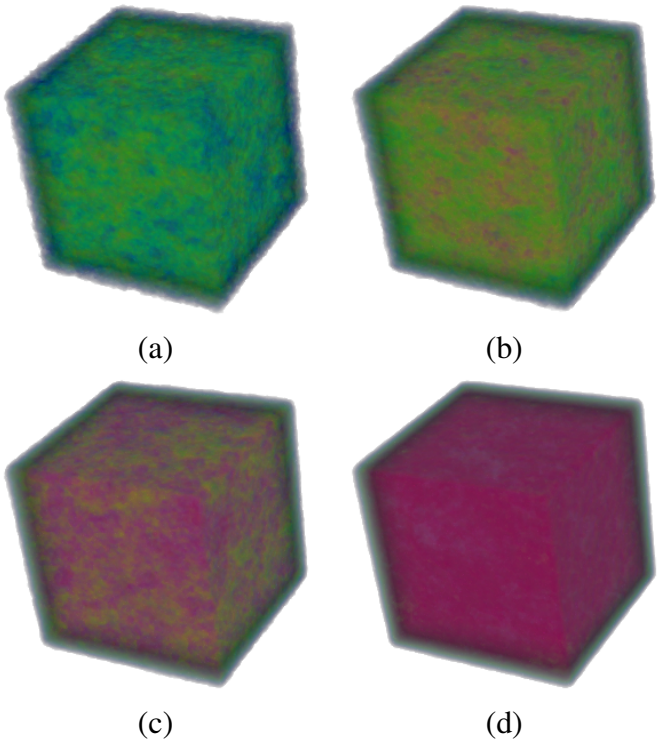


Figure 3.6: Comparison of the results obtained on a synthetic cube data using 1600 time steps and (a) $4,000$ shapetons (b) $8,000$ shapetons (c) $15,000$ shapetons, and (d) $30,000$ shapetons.

shapetons will travel over the smaller features missing them completely. Smaller distance values are useful in obtaining and analyzing local features. Therefore, the distance value is an indication of the shape information obtained at different scales of the data. Intricate local shape details are obtained by using a smaller distance value, while global shape information is obtained using a higher distance value (with a smaller number of time steps). It is not that the smaller distance value is unable to capture the global shape information, it is just that it takes more time steps to obtain the global shape information using a smaller distance value. On the contrary, a higher distance value is unable to obtain the local features despite using more time steps.

Figure 3.7 shows the results of the shapeton diffusion on the aneurysm dataset using different distance values for the same number of 400 time steps. Figure 3.7(a) shows the result for a distance value of 0.001; Figure 3.7(b) shows it for a distance value of 0.005; Figure 3.7(c) shows it for a distance value of 0.01, and Figure 3.7(d) shows the result for a distance value of 0.05. For small distance values even the

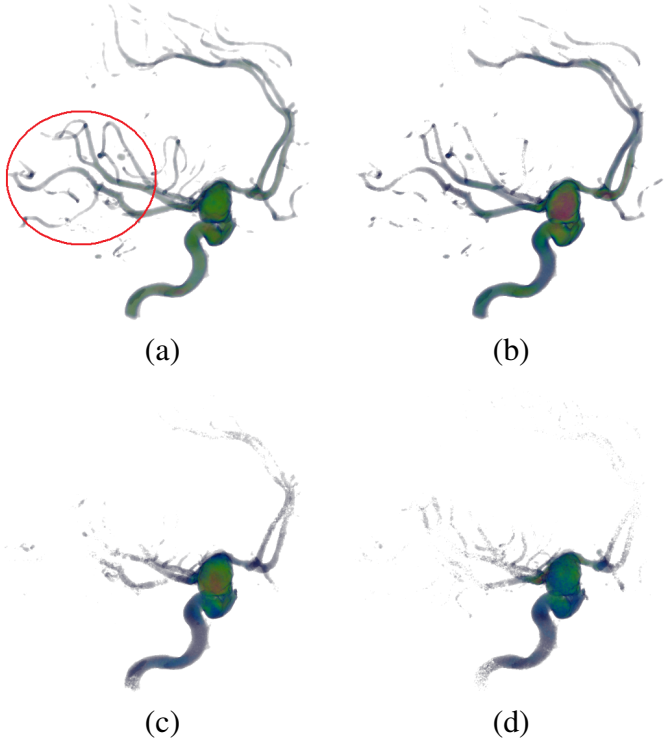


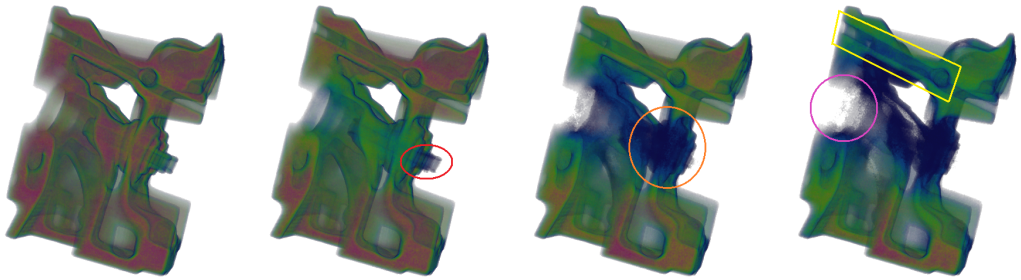
Figure 3.7: Effect of the different distance values on the aneurysm volume data using 400 time steps. Smaller features such as the narrow blood vessels (shown in the red circle) are captured using small distance values of 0.001 in (a) and 0.005 in (b) which are absent when larger distance values of 0.01 in (c) and 0.05 in (d) are used.

smaller features such as the narrow blood vessels (shown in the red circle) are captured. As the distance value is increased, only the relatively larger features such as the aneurysm blob are captured by the shapetons. The smaller features such as the narrow vessels are missing in Figures 3.7(c) and (d), where a higher distance value is used.

3.4.3 Effect of p

The direction of shapeton propagation is guided by the volume gradient operator. The volume gradient operator has a parameter p which influences the result obtained. p is a user defined parameter that is used to decide the boundaries of the objects in a given volume data. The clarity of the boundary determines how clearly the different shapes are identified. The parameter p gives the user extra flexibility in deciding the object boundaries. A large p value would enhance the local shape differences within an object and hence result in more sub-objects. Therefore, by

increasing the value of p the local internal objects within an object can be obtained. However, we tend to lose some of the global shape information for larger values of p. Thus, the final results obtained might vary both locally and globally for different values of p based on how well the objects are distinguished and how sharp the features are. All these effects of p are shown experimentally using the engine data in Figure 3.8.



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

Figure 3.8: Comparison of choosing different values for p. (a), (b), (c) and (d) are the results obtained by choosing p = 4, 9, 15 and 20, respectively, on the engine dataset for 1,300 time steps. Internal parts such as the pipe (shown in the red ellipse), the outer rim around the pipe (shown in the orange circle) and the beam (shown in the yellow box) are captured in (b), (c) and (d), respectively. For large values of p in (d) some of the global shape information is missing (shown in the pink circle).

Figure 3.8 shows the result of choosing different values of p on the engine dataset. Figures 3.8(a), (b), (c) and (d) show the result when p = 4, 9, 15 and 20, respectively for 1,300 time steps with a pre-defined distance of 0.05. We can observe that different parts of the engine are captured by using different values of p. In Figure 3.8(b) where p = 9, the internal pipe (shown in the red ellipse) is separated which was not when p = 4 in Figure 3.8(a). Similarly, when p = 15 in Figure 3.8(c) the outer rim around the pipe (shown in the orange circle) is captured. Finally, when p = 20 in Figure 3.8(d) the beam of the engine (shown in the yellow box) is captured. We can see that by increasing the value of p more internal parts of the engine are captured as the local shape differences between these parts are enhanced. However, some of the global shape information is missing (shown in the pink circle) in Figure 3.8(d). This is because a high value of p divides the same object into much smaller sub-parts and because the pre-defined distance used was relatively high, these smaller sub-parts are not captured. The same is the reason why the internal pipe from Figure 3.8(b) is missing in Figures 3.8(c) and (d). In this way, different parts of the engine based on their shape can be obtained and analyzed using different values of p. This facilitates a better analysis and understanding of

the data. As the convergence of shapetons can be monitored in real time, even though the value of p is changed, the new result can be obtained very fast. This was not the case in cumulative heat diffusion. Thus, based on what features the user wishes to focus on and what features the user wants to analyze, different values of p can be selected.

3.4.4 Convergence

When optimal values for all the parameters (number of shapetons, distance value) are used, the shapeton diffusion process is bound to converge to a steady state. This can be theoretically explained as follows. If the number of shapetons used for the diffusion process in our method tends to a very large value (say infinite), our method would be similar to the conventional heat diffusion process. The only difference would be that the heat can then diffuse in any direction instead of only between voxels as was the case in Gurijala et al. [57]. The process of heat diffusion is evaluated by using the VGO in case of volumes, which is a partial differential equation. The well-known solution for this partial differential equation is obtained by using random walks which is nothing but a Monte Carlo approach. Since our method is also based on a Monte Carlo approach, it will serve as a solution of the partial differential equation and hence it must converge.

3.5 Applications

3.5.1 Transfer Function Design in Volumes

The shapetons accumulate all the shape information over different time steps while diffusing inside the volume. This information can be used to design a shape-based transfer function. The user can assign different colors and opacities to the final accumulated shapeton count, which forms a 1-D transfer function based on the shape information.

Figure 3.9 shows a volume rendered image of a CT chest dataset with a transfer function designed using the shape information obtained by the shapeton diffusion method. We were able to classify different parts of the data, such as the rib bones (shown in red), the sternum (shown in dark green), the clavicle bones (shown in magenta), the soapula (shown in fluorescent green), and small bones of the spinal cord (shown in blue) based on the shape information. Figure 3.9 shows all the segmented parts of the CT chest data by using the transfer function designed using the shape information obtained by the shapeton diffusion process. All the ribs have similar curved shape and hence have been classified as the same shape indicated by the same color. Even the small but important part named xiphoid (greyish blue shown

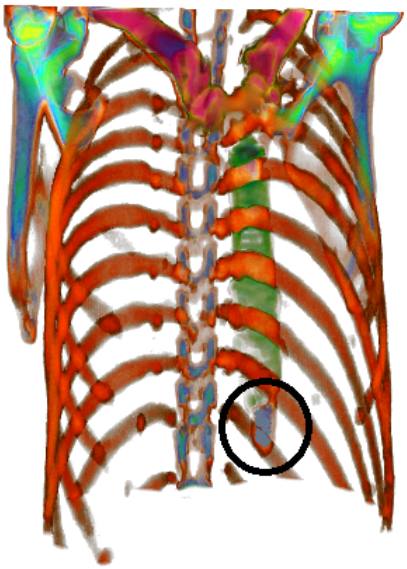
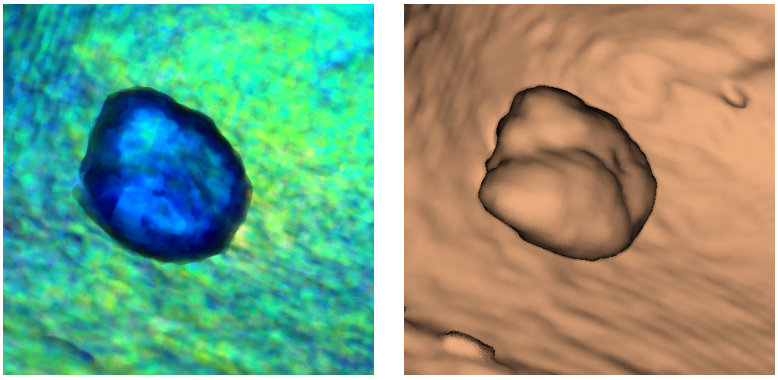


Figure 3.9: Volume rendering with the shape-based transfer function on the CT chest dataset. The rib bones (red), the sternum (dark green), the clavicle bones (magenta) and the soapula (fluorescent green) are obtained. The small bones of the spinal cord (blue), xiphoid (greyish blue in the black circle) - a small part present at the tip of the sternum are also classified.

in the black circle), which is present at the tip of the sternum has been classified by the shape-based transfer function. The number of shapetons used was $65,000$ with a distance value of 0.05. The diffusion process was carried out for 1,600 time steps, for a total time of 3.10 sec.

3.5.2 Colon Cancer Detection

We used the shapeton diffusion approach to detect the polyps (precursors of colorectal cancer) on the colon surface, obtained from a CT scan of the patient's abdomen for virtual colonoscopy (VC) [67]. We used real volumetric colon data from VC to show the effectiveness of the shapeton diffusion process in polyp detection. The volumetric colon is electronically cleansed CT data. Figure 3.10 shows the result of the polyp detection using the shapeton diffusion method on the real colon data. Figure 3.10(a) shows the result obtained by the shapeton diffusion method and Figure 3.10(b) shows the volume rendering result of the corresponding location of the polyp inside the colon. Since polyps have a blob-like shape, different from the shape of the colon walls, we were able to successfully detect the polyps using the shapeton diffusion method. Figure 3.10(a) shows one such polyp (shown in blue)



(a) (b)

Figure 3.10: Polyp detection inside the colon using the shapeton diffusion method. (a) Polyp (shown in blue) detected using the shapeton diffusion approach; (b) Volume rendering of the corresponding location inside the colon confirming the presence of the polyp.

detected. We confirmed the position of the polyp by examining the corresponding location inside the colon volume data. This result can be seen in Figure 3.10(b). It took just 4,000 time steps using $65,000$ shapetons to achieve this result. The p value was chosen to be 15. The reason to choose a high value for p is to get a clear boundary of the polyps. Since a smaller scale is needed for the polyp detection, a low distance value of 0.005 was chosen. The total time taken was 5.44 sec.

3.6 Results

We used several datasets to demonstrate the efficiency of the shapeton diffusion method. Figures 3.11(a)-(e) show the object classificaiton capability of the shapeton diffusion approach based on the shape information for hydrogen atom, visible female hand, CT abdomen, MRI brain and visible female feet volumetric datasets, respectively. In Figure 3.11(a) both the orbitals of similar shape are clearly distinguished from the nucleus (center) and the orbit (around the nucleus) in a hydrogen atom as inidicated by different colors. In Figure 3.11(c), the shape-based volume exploration of the CT abdomen reveals various organs such as the kidneys, liver, pancreas, and vital parts such as the aortic vessel, spinal cord and pelvic bones using 260,000 shapetons and a pre-defined distance of 0.01. All the internal organs have different shapes and by virtue of the shapeton diffusion method, they have been identified successfully. Furthermore, it has just taken only 2.13 sec using 2,600 time steps to obtain this result. In Figure 3.11(d), we are able to separate the brain from the cranium and eye sockets in the MRI head data, using the shape-based transfer

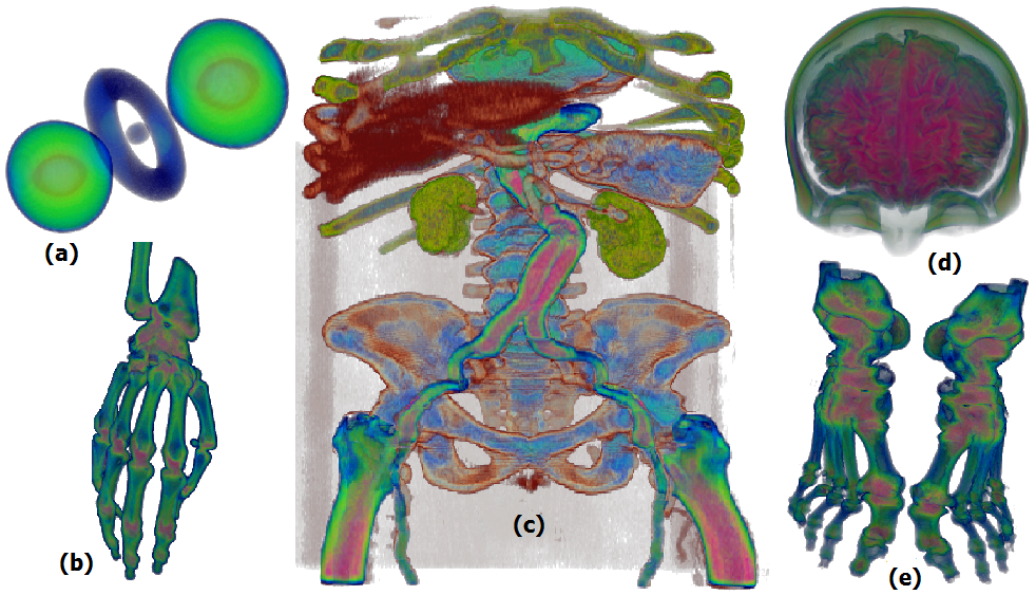


Figure 3.11: Classifying objects based on their shape using our shapeton diffusion approach on (a) hydrogen atom, (b) visible female hand, (c) CT abdomen, (d) MRI head, and (e) visible female feet volumetric datasets.

function designed by the shapeton diffusion approach. We used 65,000 shapetons for a pre-defined distance value of 0.01 and 4,160 time steps which accounted for a total time of 2.82 sec. Figures 3.11(b) and (e) show that the bones and the joints between the bones are identified in the visible female hand and feet data, respectively. While we used 65,000 shapetons and a pre-defined distance of 0.01 in both the cases, the number of time steps were 460 and 420 with a total time of 0.34 sec and 0.27 sec for the visible female hand and feet, respectively.

We compared the shapeton diffusion approach with the cumulative heat diffusion (CHD) approach, in terms of the running time per iteration and the number of time steps required to obtain visually similar or even better results. Table 3.2 shows the comparison results using different volume datasets. The third column shows the value of p used for each dataset. The fourth column shows the number of iterations (time steps) for which the cumulative heat diffusion was carried out. The fifth column gives the time taken (in msec) per iteration by the cumulative heat diffusion on volumes. The sixth column shows the number of iterations (time steps) for which the shapeton diffusion was carried out to obtain a visually similar or better result compared to that of the cumulative heat diffusion. The seventh column gives the time taken (in msec) per iteration by the shapeton diffusion approach. Finally, the eighth and ninth columns give the total time (in sec) taken by the cumulative heat

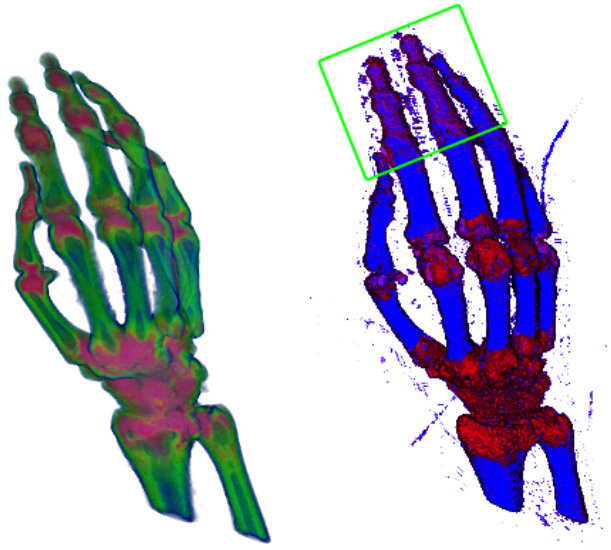
Table 3.2: Comparison of the running times of the shapeton diffusion approach with the cumulative heat diffusion (CHD) for different volumes.

Dataset	Resolution	p	# Iterations CHD	# Iterations shapeton diffusion
Foot	256x256x256	7.0	100,000	2,800
Engine	256x256x128	10.0	52,000	1,100
Aneurysm	256x256x256	7.5	100,000	3,100
MRI Head	256x256x256	6.0	400,000	4,160
CT Abdomen	512x512x174	8.0	100,000	2,600
CT Chest	384x384x240	6.0	100,000	1,600
Visible Female Hand	176x187x190	10.0	1000,000	4,600
Visible Female Foot	176x187x190	9.0	500,000	4,200

Dataset	Time (msec) per CHD iteration	Time (msec) per shapeton diffusion iteration	Total time (sec) for CHD	Total time (sec) for shapeton diffusion
Foot	4.8	0.69	480	1.90
Engine	1.4	0.74	73	0.81
Aneurysm	5.1	0.69	510	2.10
MRI Head	21.0	0.68	8400	2.82
CT Abdomen	46.0	0.82	4600	2.13
CT Chest	32.0	1.93	3200	3.10
Visible Female Hand	18.0	0.73	18 000	3.36
Visible Female Foot	13.0	0.63	6500	2.65

diffusion and the shapeton diffusion approach, respectively. For the sake of comparison we have used the same value of 0.01 for the distance (that defines the iteration) in the shapeton diffusion approach in all the cases. For all datasets, the shapeton dif-

fusion method took far fewer time steps and less time per iteration when compared to the cumulative heat diffusion method to obtain similar results. Furthermore, the total time taken for the shapeton diffusion method is orders of magnitude faster than the previous method. All tests were conducted on a system equipped with an Intel Xeon E5620 CPU and NVIDIA GeForce GTX 480 graphics board.



(a) (b)

Figure 3.12: Visual comparison of the results obtained for the visible female hand dataset using (a) the shapeton diffusion method and (b) the cumulative heat diffusion method.

For the sake of completion, we also provide a visual comparison of the results obtained by using the shapeton diffusion method with that of the results obtained using the cumulative heat diffusion method using the visible female hand dataset (see Figure 3.12). Figure 3.12(a) shows the results obtained using the shapeton diffusion approach, while Figure 3.12(b) shows the result obtained by using the cumulative heat diffusion method. In Figure 3.12(b) 1,000 time steps were considered while in Figure 3.12(a) only 460 time steps were considered for a distance value of 0.01. Since we wanted to capture the local features, a smaller distance value was used. The same p value of 10 was used in both the cases. We can clearly observe that visually better results were obtained using the shapeton diffusion method compared to the cumulative heat diffusion method. We can also see that a better distinction of shapes was obtained using the shapeton diffusion method even in very local regions, as indicated by the region in the green box in Figure 3.12(b). The joints have been clearly distinguished from the hand bones. Furthermore, the result was

obtained in much less time compared to the cumulative heat diffusion approach, further emphasizing the superiority of the shapeton diffusion method.

Chapter 4

Colon Landmark Detection

The colon is a very complicated structure with a large number of folds and bends. Colonic landmarks and features serve as tools to assist in the study of the colon surface segment by segment. In addition, identification of landmarks and feature points plays a vital role in the registration of supine and prone colon surfaces. Nearly accurate colon registration is achieved by flattening the colon surfaces and using the colon feature points as constraints. We present methods to identify the locations of the taeniae coli and the four major flexures which form the prominent anatomical landmarks on the colon surface. The colon surface is cut open along these landmarks and the obtained segments can be used to study the surface of the colon. We define new feature points on the flattened colon surfaces and use well established graph based algorithms for their detection.

This chapter is organized as follows. Section 4.1 presents the motivation. Section 4.2 describes the methods to identify the anatomical landmarks of taeniae coli and flexures. Section 4.3 describes the detection of feature points on the colon surface.

4.1 Motivation

Virtual colonoscopy (VC) has been developed as a non-invasive, comfortable, accurate and low cost alternative to the conventional optical colonoscopy for the early detection of colorectal cancer. In VC, CT scans are typically acquired with the patient in both supine (facing up) and prone (facing down) positions to improve the detection rate. However, the shape of the colon is flexible and changes very easily with the change in position of the patient. Thus, to understand the surface of the colon and to help the user know the current position inside the colon during navigation, some landmarks and feature points are necessary. These landmarks and features can be used for applications in the VC system such as virtual navigation, virtual dissection, registration of the colon surfaces, polyp matching, and polyp

bookmarking. The landmarks and feature points also help to toggle between the positions in supine and prone colons to confirm a polyp location. The motivation of this chapter is the detection of these landmarks and feature points on the colon surface.

Taeniae coli are the significant anatomical landmarks stretching along the entire length of the colon. We extend previous ideas to extract the taeniae coli on the colon surface. In addition, there are four major flexures (bends) which also serve as good anatomical landmarks. We present a method to identify the locations of the four major flexures, which are the prominent flexures in the colon, using the CT colon data. The colon surface is cut along the taenia coli and the flexures to obtain precisely five flat colon segments. These segments are used for segment wise comparison of supine and prone colon surfaces, to know about the haustral folds and to understand the intricacies of the surface of the colon. Furthermore, we detect feature points which assist in the colon registration.

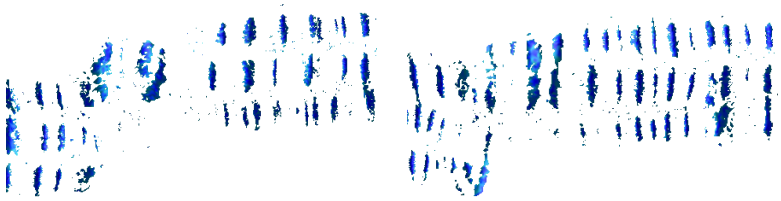
4.2 Anatomical Landmarks

The colon anatomical landmarks are extracted which are used in splitting the colon for later processing. The important anatomical landmarks include the taenia coli and flexures, which do not change despite the change in position of the patient. Corresponding anatomical landmarks in the supine and prone colon models are extracted and then used to cut each colon into its anatomical segments, as well as slicing the colon open for flattening. The main taenia coli is identified and used as a consistent cutting line between the supine and prone colons. This consistent cutting line is important for the robustness of the registration as the boundaries should be the same between the two colon models. The flexures are used to split the colon into five consistent sections which can be processed individually. This splitting reduces the computational burden and allows for better results by having multiple aligned boundaries along the length of the colon.

4.2.1 Taeniae Coli

Taeniae coli are three bands of longitudinal muscle on the surface of the colon which run from the appendix to the sigmoid colon and are ideal references for virtual navigation. The three taeniae coli are named taenia omentalis, taenia mesolica, and taenia libera according to the position on the transverse colon. Taeniae coli are located where the haustral folds meet and hence can be regarded as ridge breakers for the haustral folds [32, 96, 153]. It is relatively easy to extract the taenia omentalis as it is clearly visible on the transverse and ascending colons. The taeniae coli

detection is based on the detection of haustral folds. Using the haustral folds, the taenia omentalis is initially extracted from which the taenia mesolica, and taenia libera are later extracted as straight lines approximated at one third and two thirds of the circumference of the colon.

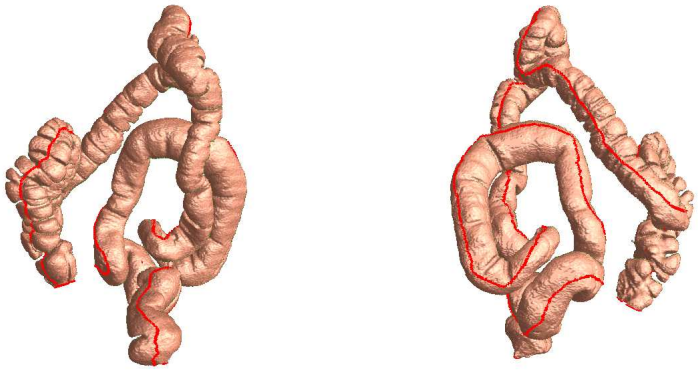


(a) Prone (b) Supine

Figure 4.1: Haustral folds (blue) on the (a) prone and (b) supine colon surfaces.

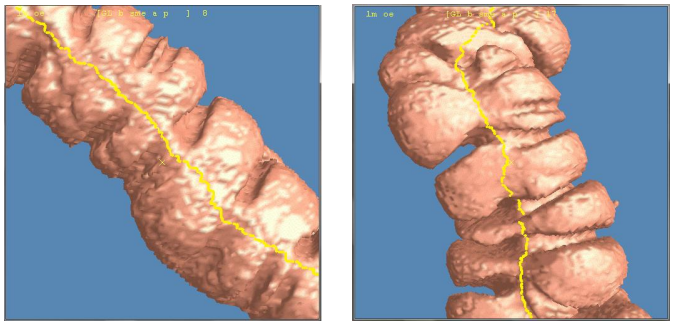
The sense of direction along the colon surface is defined by using the centerline of the colon. The haustral folds are detected by using the characteristic hyperbolic curvature of the folds. By using heat diffusion, curvature-based filter [74], and connected components, the haustral folds are detected. The curvature filter helps to obtain a rough set of all points which form potential candidates for the folds and these folds are assigned a different color to identify the taeniae coli quickly. These points are obtained using an experimentally determined threshold value. For the datasets we used, the points belonging to the haustral folds were chosen such that they lie in the threshold range of [-4.5,-0.5]. Finally, by finding the connected components and performing a certain amount of geometrical processing, we obtain the haustral folds. Figure 4.1 shows the haustral folds detected in prone and supine colons. Using these haustral folds, the taeniae coli are extracted by using the fuzzy C-means clustering algorithm iteratively [73]. Figure 4.2 shows the front and back views of the taeniae coli on the prone colon surface. The detected taeniae coli are used in colon flattening.

Figure 4.3 shows the extracted taenia coli on the colon surface. Automatic extraction of the taeniae coli through the entire colon is often possible. In some datasets, where automatic extraction of taeniae coli is not possible, manually placed markers could be used to improve the reliability. Out of the 9 datasets that we tested the algorithm, manual markers were needed on two datasets. The taeniae coli is extracted to use as a guide to virtually cut open the colon so that it can later be used for feature points detection. Considering this requirement, detection of the taeniae coli using our algorithm is reliable and very accurate. The haustral fold detection and the taeniae coli detection is performed on the original colon surface directly.



(a) Front view (b) Back view

Figure 4.2: Taenia coli (in red) on the prone colon surface.



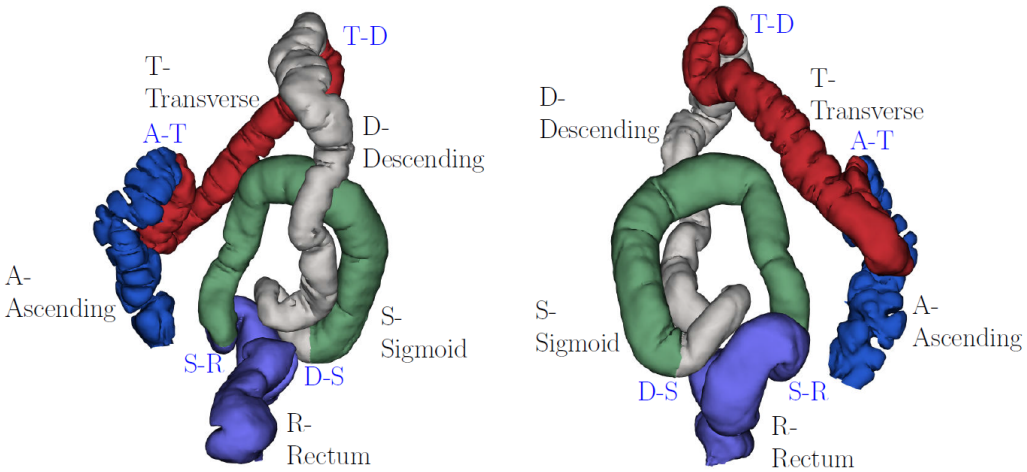
(a) Transverse colon (b) Ascending colon

Figure 4.3: Taenia coli (yellow) shown in the (a) transverse and (b) ascending segments.

4.2.2 Flexures

The five segments of the colon, starting from the cecum, are the ascending colon (A), the transverse colon (T), the descending colon (D), the sigmoid colon (S), and the rectum (R). A method is needed to identify the locations of the four major flexures between these five segments in the colon. These flexures are further anatomical landmarks which help in virtual navigation, supine-prone alignment and splitting. The first major flexure occurs between the ascending colon and the transverse colon (A-T flexure). This is the flexure close to the liver and is called the hepatic flexure. The second major flexure occurs between the transverse colon and the descending colon (T-D flexure). This flexure is close to the spleen and is named the splenic flexure. The third flexure occurs between the descending colon and the sigmoid (D-S flexure), and the final flexure is between the sigmoid and the rectum (S-R

flexure). All of these flexures form very sharp bends and are distinguishable from other smaller bends. Theoretically, the A-T flexure forms the topmost point of the ascending colon and the T-D flexure forms the topmost point of the descending colon.

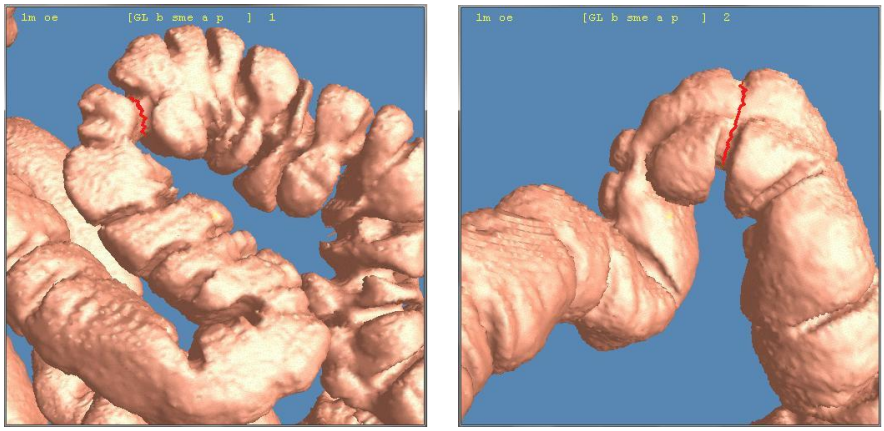


(a) Front view (b) Back view

Figure 4.4: The four flexures on the prone colon, which divide the colon surface into five segments, depicted in various colors.

For the detection of these flexures, the 3D centerline through the colon is utilized [20, 78]. This centerline is projected onto a 2D coordinate system in the positive $z-x$ and positive $y-z$ planes. The bends in the centerline are identified by iteratively evaluating the slopes along the projected curves in the two planes. Not all of the bends are important, so small bends are discarded based on a threshold and only the major bends are retained. All of these detected bends are sorted based on their z-coordinate (up direction). The T-D flexure is identified as the bend with the highest z-coordinate and the A-T flexure is identified as the bend with second highest z-coordinate. In the event that the A-T flexure has a higher z-coordinate than the T-D flexure, the order of the two points along the centerline can be used to confirm the correct identifications. The S-R flexure is the bend with the lowest z-coordinate. The next bend in the sorted order after the T-D flexure whose y-coordinate is comparable to that of the T-D flexure forms the D-S flexure. These four positions are then mapped back to the 3D coordinate system and the corresponding 3D coordinates of the centerline are obtained.

However, the points that are obtained are those on the centerline and not on the surface. Using these points, the areas on the surface where the flexures are present need to be marked. For each point, a plane is defined othogonal to that point on



(a) Hepatic flexure (b) Splenic flexure

Figure 4.5: (a) Hepatic and (b) splenic flexures (marked in red band) on the colon surface

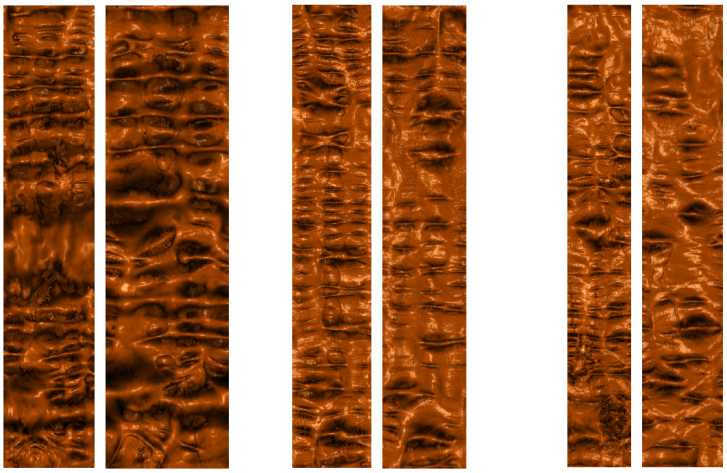
the centerline. The intersection of the plane with the mesh surface is computed and a polyline is marked by joining the intersection points using the well-known Dijkstra's shortest path algorithm. Consequently, the flexures on the surface are extracted. The four extracted flexures between the five segments of the colon are illustrated in Figure 4.4 on the prone colon surface. Figure 4.5 shows the hepatic and the splenic flexures marked on the colon surface.

4.3 Surface Features

In order to align the two colon surfaces, it is necessary to identify and correlate feature points on the two surfaces. For this task, the problem of feature detection and matching in 3D is changed into a 2D image matching problem. This is accomplished through the use of conformal mapping. Each colon segment is mapped conformally to a planar rectangle, allowing for all work to then be accomplished within the 2D domain. The results for corresponding segments are shown in Figure 4.6. Since the conformal modules of corresponding segments are not equal, the segments are not conformally equivalent, and thus the requirement for additional feature points to align the two segments.

4.3.1 Feature Detection and Matching

Given a mapping of each colon segment onto a planar rectangle, the mean curvature is color encoded to generate color images. The mean curvature on v_i, $H(v_i)$,



(a) Segment A Mod = 5.41089; 3.85803 (b) Segment T Mod = 6.20148; 5.87936 (c) Segment D Mod = 7.77978; 6.15164

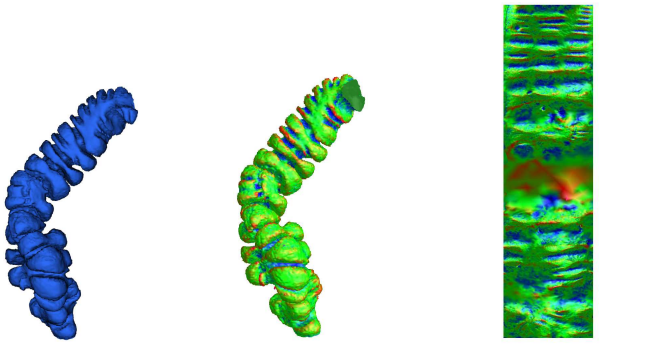
Figure 4.6: Conformal modulus comparison between three different segments of the supine and the corresponding segments of the prone (Ascending, Transverse, and Descending). In each column, the left image shows the flattened rendering of the supine and the right image shows the flattened rendering result of the prone. The conformal module for each segment is defined as the ratio of height to width for the flat rectangle map, $Mod = height/width$. Each of the segment pairs is not conformally equivalent.

is approximated by

$$H(v_i) = \sum_{[v_i,v_j]\in E} w_{ij} < \mathbf{v_j} - \mathbf{v_i}, \mathbf{n_i} >,$$

where $\mathbf{n_i}$ is the normal at the vertex v_i, $<,>$ is the inner product in $\mathbb{R}^3$, and w_{ij} is the cotangent weight. An example of the color encoded colon surfaces in 3D and 2D are shown in Figure 4.7.

These color coded flattened colon segments are analyzed to obtain specific feature points for the registration. The well known graph cut approach from computer vision is used to find the regions of interest. In the case of the colon, the desired regions of interest on the flattened supine and prone images are the folds. A graph is constructed using the pixels of the image as the nodes. Assuming a virtual sink and source, edges are constructed with appropriately assigned weights. Using the max-flow min-cut method to solve the energy minimization problem [22], all of the pixels belonging to the folds are obtained. The appropriate assignment of weights ensures efficient detection of the folds.



(a) Supine (b) Supine curvature (c) Flat supine

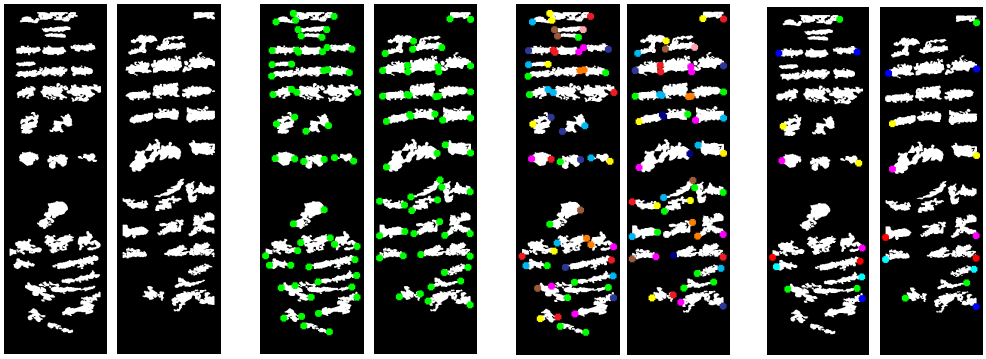
(d) Prone (e) Prone curvature (f) Flat prone

Figure 4.7: Color encoding mean curvature on corresponding supine and prone segments: (a) and (d) original colon surface segments; (b) and (e) color encoded mean curvatures; (c) and (f) color encoded mean curvature on the flattened surfaces.

Due to the drastic change in position of the patient in supine and prone, the colon tends to stretch, contract, and move, resulting in the distortion of the folds. In addition, some areas might be densely populated with folds, while other areas are sparsely populated. To prevent any ambiguities, only the significant folds are chosen which have a lower chance of distortion. Therefore, all of the folds whose length and size are below a certain threshold are discarded. For this, a pre-processing step is performed using a breadth-first search to find all of the connected components among the extracted folds. The prominent folds form larger connected components and thus by defining an experimental threshold value, all the smaller components are discarded. By varying the threshold, it is possible to modify the number of detected folds based on the requirements. Figure 4.8(a) shows the segmentation results for both the supine and prone surfaces.

After applying the threshold all of the significant folds are now detected. These

folds can be approximated as ellipses and hence the axial end points form good feature points for registration. These axial points of the detected folds are extracted and form the feature point set. Figure 4.8(b) shows the detected features (in green) on both supine and prone surfaces.



(a) Segmentation (b) Features (c) Matching results (d) Constraint points

Figure 4.8: Feature detection and matching for the flat images between supine and prone (see Figures 4.7(d) and (g)). In each frame, supine is on the left and prone is on the right: (a) segmentation results using graph cut; (b) feature detection results (in green); (c) feature matching results using graph matching; (d) matching features used to constrain the registration. Two corresponding feature points are encoded in the same color on the supine and prone flat images in (c) and (d).

Feature matching is performed on each segment of the colon separately. Finding a correspondence between the extracted feature points in supine and prone is again formulated as an energy minimization problem by defining an objective function. The dual decomposition approach is employed for energy optimization, which is solved as an instance of the well-known graph matching problem [144].

Suppose the feature point sets are S_1 and S_2 on supine and prone respectively, and the matching is $\phi : S_1 \rightarrow S_2$, then the objective function is defined as

$$E(\phi) = \lambda \sum_{p \in S_1} |p - \phi(p)|^2 + (1-\lambda) \sum_{q \in S_2 - \phi(S_1)} |q|^2,$$

where λ is determined experimentally.

The objective function is chosen depending on the geometrical compatibility of feature correspondences and the spatial coherence of the matched features. It is defined by considering the weighted sum of two energy terms. The first term includes the geometric distance between the feature points within a pre-defined neighborhood. The second term evaluates the effect of including the distance of the un-

matched features in the objective function. Minimization of this function using the dual decomposition approach will result in a unique set of feature correspondences.

Due to the large deformation between supine and prone, their folding patterns are highly inconsistent. Figure 4.8(a) shows that the number of significant folds is different for supine and prone. Therefore, it is impossible to find the correspondence for all feature points on supine and prone, as shown in Figure 4.8(b) and (c). In the graph matching algorithm, it is attempted to try and utilize all of the matched features, and ignore those that are unmatched. In practice, it was found that the feature points near the taenia coli (that is, the border of the rectangle) are more reliable and accurate than those in the middle. Therefore, the algorithm emphasizes the feature points near the borders more than those in the middle. Table 4.1 shows the number of correctly and incorrectly matched feature points in different segments of supine and prone colons.

Table 4.1: Number of feature correspondences in supine and prone colon segments.

Colon	Segment	# border feature points	# Correctly matched feature points	# Incorrectly matched feature points
Prone	Whole	125	114	11
Prone	Ascending	29	28	1
Prone	Transverse	37	34	3
Prone	Descending	27	25	2
Prone	Sigmoid	17	13	4
Prone	Rectum	15	14	1
Supine	Whole	134	114	20
Supine	Ascending	30	28	2
Supine	Transverse	42	34	8
Supine	Descending	31	25	6
Supine	Sigmoid	16	13	3
Supine	Rectum	15	14	1

The experimental results demonstrate that the colon registration result is satisfactory even without the unreliable middle feature points. The points which are used as constraints for the alignment step of the registration algorithm are shown in Figure 4.8(d).

Chapter 5

Colon Flattening

In this chapter, we propose a new colon flattening algorithm that is efficient, shape-preserving, and robust to topological noise. Unlike previous approaches, which require a mandatory topological denoising to remove fake handles, our algorithm directly flattens the colon surface without any denoising. In our method, we replace the original Euclidean metric of the colon surface with a heat diffusion metric that is insensitive to topological noise. Using this heat diffusion metric, we then solve a Laplacian equation followed by an integration step to compute the final flattening. We demonstrate that our method is shape-preserving and the shape of the polyps are well preserved. The flattened colon also provides an efficient way to enhance the navigation and inspection in virtual colonoscopy. We further show how the existing colon registration pipeline is made more robust by using our colon flattening. We have tested our method on several colon wall surfaces and the experimental results demonstrate the robustness and the efficiency of our method.

This chapter is organized as follows. Section 5.1 provides the motivation. Section 5.2 describes the flattening algorithm using the heat diffusion metric. We discuss how our flattening algorithm is used to flatten the colon surface. We compare our flattening algorithm to several existing techniques in Section 5.3. Section 5.4 discusses applications of our method to polyp visualization, colon registration and handle detection and Section 5.5 presents the results of the method.

5.1 Motivation

Colorectal cancer is the second leading cause of cancer related mortality in the United States [70]. Optical colonoscopy (OC), whereby precancerous polyps (protrusions/bumps on the colon wall) can be located and removed, has been recommended for screening and has greatly reduced the mortality from colorectal cancer [28]. Virtual colonoscopy (VC) has been developed as a non-invasive, comfortable, accurate and low cost alternative to the conventional OC for the early detection

of colorectal cancer. [67, 82]. VC is a non-invasive screening method, whereby a radiologist can explore a colon surface in a way similar to that of a gastroenterologist performing an OC. The radiologist is mainly interested in visualizing the inner surface of the colon where polyps might be detected. VC uses computed tomographic (CT) scan of a patient's abdomen and provides a virtual fly-through visualization system through the virtual colon reconstructed from the CT scan [67]. VC has many advantages, including non-invasiveness, cheaper, faster, and higher patient tolerability [67]. However, VC has a fundamental problem, which it shares with OC. Due to the twisted nature of the colon and the numerous colonic haustral folds, the navigation using the inner endoluminal view is very challenging and sizable sections of the colon are not inspected, resulting in an incomplete examination. As a result, polyps hidden behind folds and sharp bends are missed. An efficient supplemental approach is flattening, where the colon is cut open and flattened onto a 2D plane. This not only facilitates comprehensive inspection of the colon but also reduces the time for inspection.

Several flattening techniques have been proposed for the colon surface, whereby the entire colon can be mapped from the 3D domain to a 2D rectangular domain. The colon mesh surface serves as the input. This mesh surface is extracted from the CT images by performing electronic cleaning, segmentation, and mesh extraction. However, the major problem here is that the extracted colon surface includes topological noise, such as handles, as shown in Figure 5.1(a). A close-up view of a handle is also shown in Figure 5.1(a). This topological noise is due to two reasons. The first is due to artifacts present in the CT scan. The second is due to the colon surface reconstruction method. Although many surface reconstruction methods are capable of generating water-tight surfaces from the CT data, the resulting models may still exhibit topological errors in the form of small handles. These high-frequency topological features unnecessarily increase the complexity of the colon model and make it unsuitable for subsequent processing tasks, such as colon flattening, 3D navigation, and polyp detection. Hence, in all previous methods for colon flattening, these unnecessary noisy features were either removed manually or by some topological denoising techniques. However, owing to the large surface of the colon, the use of these denoising methods is time-consuming and incurs high computational overhead.

The motivation of this chapter is to address this problem. We have introduced a new colon flattening algorithm using the heat diffusion metric, which is efficient, robust (insensitive to any topological noise) and shape-preserving. In this method, we use the extracted colon mesh directly without performing any topological denoising. To start, we compute the heat diffusion distances (HDDs) for the entire colon mesh (with noise). This HDD is used as a metric for the flattening. We compute this metric for an appropriate time step and replace the original Euclidean metric

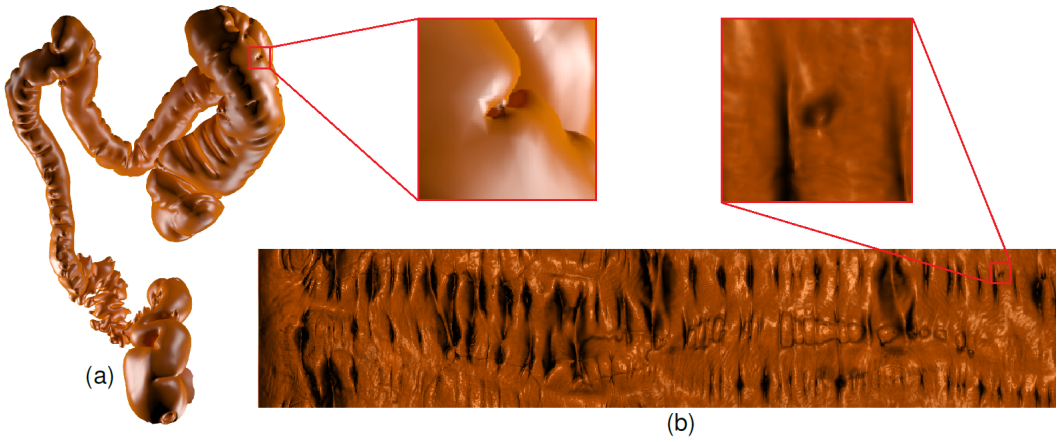


Figure 5.1: (a) A 3D colon model with topological noise, such as handles. A handle is shown in a close-up view. (b) The flattening of the 3D colon in (a) to a 2D rectangle using our method with heat diffusion Riemannian metric (flattening of only the transverse segment of the colon is shown). A colonic polyp (protrusion on colon wall) that is adjacent to a fold is shown in a close-up view.

of the colon surface with this new metric. In the next step, we solve a Laplacian equation on the colon surface to obtain a harmonic form. By applying the Hodge star operator on this harmonic form, we obtain another form that is perpendicular to the harmonic form. Finally, we integrate these two metric forms on the colon surface to obtain the flattened colon. We render the flattened colon image using direct volume rendering to provide a view similar to that of the endoluminal view, as can be seen in Figure 5.1(b). We demonstrate how the existing colon registration pipeline is turned more robust by using this new method for colon flattening. Furthermore, we show that the method enhances the colon navigation by preserving the important features, such as the polyps and folds. In addition, we also present an efficient handle detection and removal approach on the flattened colon using the flattening approach.

The novel flattening method has the following advantages:

1. **Robustness:** The method of using the heat diffusion metric for flattening is insensitive to topological noise, such as fake handles. As a result, by using this flattening algorithm, the entire global shape is preserved after flattening in spite of having many handles. On the contrary, other methods, such as the Ricci flow, produce a highly distorted flattening result of the colon in the presence of noise.

2. **Efficiency:** The method is very efficient since we need to solve only one Laplacian equation for the entire colon surface. On the other hand, other

methods, such as the holomorphic 1-form [68], require solving of $2g$ Poisson equations, where g is the number of handles on the colon surface. In case of a large number of handles (which is typical), this method performs far better than the other methods.

3. **Shape preserving:** The method is shape preserving. Even under the constraint of preserving the global shape, the local shape distortion is still under control. The final result obtained by using this method on a colon surface with noise is very similar to the one obtained by using other conformal based methods on the same colon surface with noise removed. Therefore, the method exhibits a good trade off between conformality and robustness.

5.2 Algorithm

Let the original input surface with handles be approximated by a triangular mesh M. This mesh M has a number of handles varying from tens to hundreds. Let γ_0 be the outer boundary and $\gamma_k, 1 \leq k \leq n$ be n inner boundaries or holes (holes are not handles) of M with topological noise. A larger value of n (greater number of holes) indicates a more complicated topology of M. Let V be the vertex set and E be the edge set of the mesh M. We denote v_i as the i^{th} vertex, $[v_i, v_j]$ as the edge, $[v_i, v_j, v_k]$ as the face, and θ_i as the corner angle at vertex v_i. Let the functions on M be approximated by the piecewise linear functions defined on the vertices, $f(v_i)$.

5.2.1 Heat Diffusion Metric

The *heat diffusion* process on the surface is governed by a partial differential equation defined as follows:

$$\frac{\partial u(p,t)}{\partial t} = -\Delta_{\mathbf{g}} u(p,t), \tag{5.1}$$

where $u(p,t)$ is the temperature or heat at a point $p \in S$ at time t and $\Delta_{\mathbf{g}}$ is the *Laplace-Beltrami Operator* (LBO) defined as:

$$\Delta_{\mathbf{g}} = e^{-2\varphi(x,y)}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right). \tag{5.2}$$

where $e^{-2\varphi(x,y)}$ represents the area term. The solution to Equation 5.1 with the initial condition, $u_0(p) = \delta(p-q)$, is called the *heat kernel* and is denoted as $K_{\mathbf{g},t}(p,q)$. Intuitively, the heat kernel can be interpreted as the amount of heat

transferred from a point p to a point q in a given time t. The heat kernel is isometric invariant and hence any definition using the heat kernel is also isometric invariant.

Theorem 5.2.1. *Let $\phi : (S_1, g_1) \rightarrow (S_2, g_2)$ be a diffeomorphism. Then, the heat kernels K_1, K_2, such that $K_1(p,q,t) = K_2(\phi(p), \phi(q), t)$, for all p, q, t, then ϕ is an isometry.*

The LBO has an eigendecomposition of the form: $\Delta_{\mathbf{g}}\Phi_k = \lambda_k \Phi_k$, $k = 0, 1, 2, \cdots$, where $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \cdots$ are the eigenvalues and $\Phi_k : S \rightarrow \mathbb{R}$ are the corresponding eigen functions. Then, the heat kernel is defined as:

$$K_{\mathbf{g},t}(p,q) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \Phi_i(p) \Phi_i(q) \tag{5.3}$$

Using the definition of the heat kernel, the *heat diffusion distance* (HDD) at a time t is then given by:

$$d_{\mathbf{g},t}(p,q) = K_{\mathbf{g},t}(p,p) + K_{\mathbf{g},t}(q,q) - 2K_{\mathbf{g},t}(p,q) \tag{5.4}$$

An equivalent spectral expression of the heat diffusion distance is:

$$d^2_{\mathbf{g},t}(p,q) = \int_S |K_{\mathbf{g},t}(p,r) - K_{\mathbf{g},t}(q,r)|^2 dr = \sum_{i=1}^{\infty} e^{-2\lambda_i t} (\Phi_i(p) - \Phi_i(q))^2 \tag{5.5}$$

where r is some point on the surface S. Equation 5.5 shows that the diffusion distance is the L^2 distance between two probability distributions of Brownian motions (random walks), $K_{\mathbf{g},t}(p,r)$ and $K_{\mathbf{g},t}(q,r)$. Essentially, the mapping, $p \rightarrow (e^{-\lambda_1 t}\Phi_1(p), e^{-\lambda_2 t}\Phi_2(p), \cdots)$ embeds the surface to an infinite dimensional functional space. The definition in Equation 5.5 shows that the heat diffusion distance is the same as the classical Euclidean metric in this functional space. Therefore, the heat diffusion distance is a Riemannian metric.

The heat diffusion distance (HDD) given by Equation 5.5, introduced by Lafon et al. [33, 94], is computed as an average of all paths connecting two points on the surface. The HDD (diffusion process) can be obtained by the convolution of the signal and its heat kernel. Consequently, the Laplacian surface smoothing is achieved by performing diffusion on the surface that filters out the high frequency components. However, as the surface changes during smoothing, the heat kernels are also evolving. The evaluation of HDD smooths out the small perturbations (e.g., handles) on the surface and hence makes the HDD insensitive to the topological noise. As a result, the flattening algorithm using the HDD is more robust.

Using the vertex positions, we compute the edge lengths directly as $l_{ij} = |v_i - v_j|$. We call this set of edge lengths the induced Euclidean metric. We compute the heat diffusion metric $\mathbf{g}$ by using the induced Euclidean metric. For every edge in the mesh we evaluate a new edge length value (HDD) and we call this set of new edge lengths the heat diffusion metric. For every face, $[v_i, v_j, v_k]$ of the mesh M, the corner angles are computed by using the Euclidean cosine law as $\theta_i = \cos^{-1}((l_{ij}^2 + l_{ki}^2 - l_{jk}^2)/(2l_{ij}l_{ki}))$. We evaluate the cotangent edge weight for every edge in M by using this corner angle to obtain a weighted adjacency matrix $W := (w_{ij})$, where w_{ij} is the cotangent edge weight of the edge $[v_i, v_j]$. The cotangent edge weight is defined as follows [122]:

Definition 5.2.1 (Cotangent Edge Weight)**.** *Suppose edge $[v_i, v_j]$ is adjacent to two faces $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$, then the weight of the edge is given by: $w_{ij} = (\cot\theta_k + \cot\theta_l)/2$.*

In the next step, using W and the function values at the vertices, the LBO at each vertex is computed to obtain a Laplace-Beltrami matrix, L. In the discrete case, the LBO on a vertex is defined as follows:

Definition 5.2.2 (Discrete Laplace-Beltrami Operator)**.** *Suppose edge $[v_i, v_j]$ is adjacent to two faces $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$, then the Laplace-Beltrami Operator, $\Delta_{\mathbf{g}}$ on vertex v_i is given by $\Delta_{\mathbf{g}} f(v_i) = \sum_{[v_i,v_j]\in E} w_{ij}(f(v_j) - f(v_i))$.*

We perform eigendecomposition of L and compute the eigenvalues and their corresponding eigen functions. Using these eigenvalues and eigen functions, we finally compute the HDD lengths of every edge in M using Equation 5.5. The set of all new edge lengths forms the heat diffusion metric $\mathbf{g}$.

The number of eigenvalues and the number of time steps used have an effect on the heat diffusion metric. In our case, we have used the first 50 eigenvalues and the corresponding eigen functions to calculate the HDD for the edges. Also, we choose the value of time step t in Equation 5.5 to be 8 for all our cases. These respective values have been chosen experimentally by assessing the quality of the mesh that is obtained by replacing the edge lengths with the HDD values. The quality of the mesh is assessed by measuring the "closeness" of each of its faces to an equilateral triangle. For each face of the mesh we compute the ratio of its circumradius to two times its inradius. For an equilateral triangle, this ratio is equal to 1. The faces with ratio > 0.6 are considered as good quality faces and < 0.6 are considered as bad quality faces. We obtain the histogram of the faces of the mesh based on their ratio values. Two meshes are said to be of similar quality if they have a comparable number of bad quality faces and also comparable histograms (using L^2-norm comparison). We have observed that by increasing the number of

eigenvalues the quality of the obtained mesh improves. However, the quality did not change much after 50 eigenvalues.

The value of the time step represents the length of the path over which the heat diffusion metric is averaged. Each handle has two loops, the handle loop and the tunnel loop [37]. In general, their lengths are about 8 edges. Hence, we chose the value of t to be 8 so that only the handles are filtered out. If the value of t is too small, the heat diffusion metric becomes close to the induced Euclidean metric. On the other hand, if the value of t is too large, the heat diffusion metric will smooth out all the features on the surface that would make two different surfaces yield the same result.

We articulate the difference between the HDD and the geodesic distance by means of a simple experiment. For this, we consider a hand model as our input where the index finger is touching the thumb as shown in Figure 5.2(a). In Figure 5.2(b) we detached the two fingers, by manually cutting both the fingers at the location indicated by the red arrow, so that the topology in the two hand models is totally different. In both models, we choose two points p and q such that p is a point on the tip of the pinky finger and q is a point on the thumb nail. By using these two points as epicenters, for any point x on the surface, we compute the following distance function, f:

$$f(x) = d(p,x) + d(q,x) - d(p,q) \tag{5.6}$$

Here, d can represent either the HDD or the geodesic distance. We evaluate f using both HDD and geodesic distance values for both hand models and color encode f. The color changes from blue to red with increasing f values. Figures. 5.2(c) and (d) show the color encoded distance function using geodesic distance values for the hand models in Figures 5.2(a) and (b), respectively. It can be clearly seen that the distance function changes dramatically when the topology changes. We can also see that the geodesic distance between the points p and q, shown by a white path, has changed when the topology changed. On the other hand, Figures 5.2(e) and (f) show the color encoded distance function using HDD values for the hand models in Figures 5.2(a) and (b), respectively. It can be seen that the results are more consistent, showing that the topological changes do not affect the HDD function. This experiment shows that HDD is more robust to topological noises, and hence a better choice as compared to geodesic distance.

In fact, the HDD of an edge $[p,q]$ and the original edge length are completely different. Theoretically, the HDD $d_{\mathbf{g},t}(p,q)$ is related to all paths starting from p and ending at q with length t. When t is large, the random paths cover large regions of the surface. Thus, HDD is a global concept. In contrast, the edge length of $[p,q]$ is local. In other words, the HDD is inversely related to the connectivity of points p and q, by paths of length t, hence insensitive to topological noises.

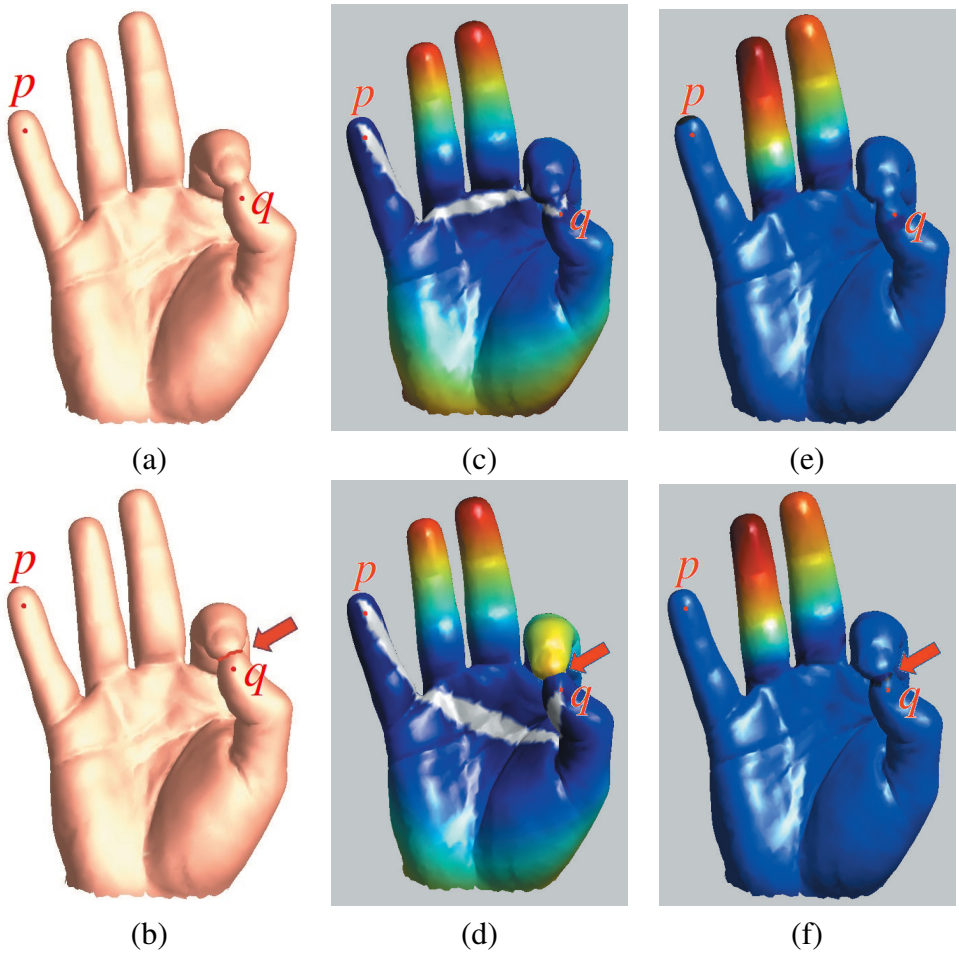


Figure 5.2: Comparison between geodesic distance and HDD using a hand model with (a) thumb and index finger touching, and (b) thumb and index finger detached by manually cutting at the location indicated by the red arrow. With points p and q as epicenters, color encoded (c) geodesic distance function of (a); (d) geodesic distance function of (b); (e) HDD function of (a); (f) HDD function of (b). When the topology changes in (b), (d) changes drastically (the geodesic path between p and q in white also changes), while (f) is not affected and is consistent.

5.2.2 Conformal Flattening Algorithm Based on HDD

We first present brief theoretical fundamentals required to explain our colon flattening algorithm and to compare our method with the existing methods for colon flattening.

Conformal Mapping

Let S be a surface embedded in $\mathbb{R}^3$ and $\mathbf{g}$ be a Riemannian metric tensor that defines the inner products on tangent planes at every point on S. Let $\bar{\mathbf{g}}$ be another metric of S, such that

$$\bar{\mathbf{g}} = e^{2\varphi}\mathbf{g}, \tag{5.7}$$

where $\varphi : S \to \mathbb{R}$ is a scalar function defined on S. Then, by means of direct computation, it is easy to verify that $\bar{\mathbf{g}}$ preserves angles. Hence $\bar{\mathbf{g}}$ is said to be *conformal* (angle-preserving) to the original metric $\mathbf{g}$.

Given two surfaces with Riemannian metrics $(S_1, \mathbf{g_1})$ and $(S_2, \mathbf{g_2})$, consider a mapping $\phi : (S_1, \mathbf{g_1}) \to (S_2, \mathbf{g_2})$ between them. ϕ is said to be a *conformal mapping* if the pull back metric induced by ϕ satisfies the following relation:

$$\phi^* \mathbf{g_2} = e^{2\varphi}\mathbf{g_1}. \tag{5.8}$$

If a Riemann surface $(S, \mathbf{g})$ is orientable, then for every point p on the surface, there exists a neighborhood $U(p)$ and a local coordinate system (u, v) on $U(p)$, such that the metric $\mathbf{g}$ can be represented as: $\mathbf{g} = e^{2\varphi(u,v)}(du^2 + dv^2)$. Here the coordinates, (u, v) are called as *isothermal parameters* or *isothermal coordinates*. According to [29], we can cover the whole surface by a collection of isothermal coordinate charts. All isothermal coordinate charts form a *conformal structure* of the surface. The surface with a conformal structure is called a Riemann surface.

Theorem 5.2.2. *All oriented metric surfaces are Riemann surfaces.*

Conformal mapping, by definition is angle preserving. For example, if any two intersecting curves γ_1 and γ_2 are mapped to $f(\gamma_1)$ and $f(\gamma_2)$ by a conformal map f, then the intersection angle between γ_1 and γ_2 equals to the intersection angle between $f(\gamma_1)$ and $f(\gamma_2)$.

Hodge Theory

Let (x, y) be the isothermal (local) coordinates, then a differential 1-form denoted by ω has the local representation: $\omega = f(x,y)dx + g(x,y)dy$. Let d denote the *exterior differential operator*. If $f : S \to \mathbb{R}$ is a function defined on S, then the gradient of f, called the *exact 1-form*, is given by: $df = f_x dx + f_y dy$. The exterior differential operator acting on ω is given by:

$$d\omega = (g_x - f_y)dx \wedge dy. \tag{5.9}$$

If $d\omega = 0$, then ω is called a *closed 1-form*. Exact 1-forms must be closed. The space of all closed 1-forms is denoted as $Ker\ d^1$ and the space of all exact 1-forms

is denoted by $Img\ d^0$. Then, the first dimensional cohomology group $H^1(S,\mathbb{R})$ is given by:

$$H^1(S,\mathbb{R}) = \frac{Ker\ d^1}{Img\ d^0} \tag{5.10}$$

Each element in $H^1(S,\mathbb{R})$ is a cohomological class. Two closed 1-forms are said to be in the same class if they differ by an exact 1-form.

Under isothermal parameters, the *Hodge star* operator is defined as:

$$^*(fdx+gdy) = -gdx+fdy, ^*f = fdx\wedge dy, ^*(gdx\wedge dy) = g. \tag{5.11}$$

The co-differential operator δ is defined as $\delta = *d*$. A differential 1-form ω is said to be a *harmonic 1-form*, if $d\omega = 0$ and $\delta\omega = 0$. All harmonic 1-forms form a group which is isomorphic to the first cohomology group based on the following Hodge theorem:

Theorem 5.2.3 (Hodge)**.** *Each cohomological class has a unique harmonic 1-form.*

Current work focuses on genus zero surfaces with mutiple boundaries, $\partial S = \gamma_0, \gamma_1 \cdots \gamma_n$. Then, the first cohomology is n dimensional. One can choose n harmonic 1-forms $\{\omega_k\}, 1 \leq k \leq n$, forming the basis of $H^1(S,\mathbb{R})$ by: $\int_{\gamma_j} \omega_i = \delta_i^j$, where δ_i^j is the Kronecker symbol. Intuitively, a closed 1-form can be interpreted as a curl free vector field; a harmonic 1-form is both curl free and divergence free. Hodge theorem claims fixing the topological condition and the cohomological class. It further claims that the harmonic form exists and is unique.

Holomorphic Differentials

Suppose ω is a harmonic 1-form, then its Hodge dual $^*\omega$ is also a harmonic 1-form. The pair, $\eta = \omega + i^*\omega$ is called a *holomorphic 1-form*. On a Riemann surface, all the holomorphic 1-form form a group, which is isomorphic to the first cohomology group.

The holomorphic 1-form can be treated as the complex derivative of a conformal map $\phi : S \to \mathbb{C}$, and the conformal map ϕ can be recovered by integrating the holomorphic 1-form $\phi = \int \eta$. In practice, from the harmonic 1-form basis $\{\omega_k\}$, we can construct the holomorphic 1-form basis $\{\eta_k = \omega_k + i^*\omega_k\}, k = 1, 2, \cdots, n$. Then we can construct any holomorphic 1-form by linearly combining the basis, and obtain the conformal mapping by integration.

Flattening Algorithm

We map the surface M with the heat diffusion Riemannian metric $\mathbf{g}$ conformally onto $\mathbb{C}$ which represents a (u,v)-planar domain. For this, we first replace all the edge lengths in M with the edge lengths from the heat diffusion metric and denote the new mesh also by M. Since M has n inner boundaries, the cohomology group and therefore its harmonic 1-form basis are n dimensional (Theorem 3.2).

The first step of our algorithm is to compute the basis for exact harmonic 1-forms of M. To compute the exact harmonic 1-forms, we first compute the harmonic functions, $f_k : M \to \mathbb{R}$ by solving the following Dirichlet problem on M for each inner boundary component γ_k:

$$\begin{cases} \Delta_{\mathbf{g}} f_k &\equiv 0 \\ f_k|_{\gamma_k} &= 1 \\ f_k|_{\gamma_i} &= 0, 0 \leq i \leq n, i \neq k \end{cases} \tag{5.12}$$

The Laplace matrix $\Delta_{\mathbf{g}}$ in Equation 5.12 is positive definite and thus, non-degenerate. The stability of the Laplace matrix, measured by its condition number [89], depends on the mesh triangulation quality. In our case, the triangles in the mesh are close to Delaunay (Laplace matrix has a good condition number) and thus the linear system is stable. In the discrete case, the harmonic function is defined as follows:

$$\Delta_{\mathbf{g}} f(v_i) = \sum_{[v_i,v_j] \in E} w_{ij}(f(v_j) - f(v_i)) = 0. \tag{5.13}$$

Equation 5.13 is the discrete Laplacian equation. The discrete harmonic function satisfies the *mean value* property where a function value at a vertex $f(v_i)$ is equal to the mean of the function values of the neighboring vertices, $f(v_j)$'s. Thus, Equation 5.13 is equivalent to:

$$f(v_i) = \sum_{[v_j,v_i] \in E} \frac{w_{ij}}{\sum_k w_{ik}} f(v_j), \tag{5.14}$$

The discrete harmonic function f_k on M is thus computed by using Equation 5.14 and by solving the linear system in Equation 5.12. For solving the linear system we use the publicly available UMFPACK library. Once the harmonic function f_k is obtained, the exact harmonic 1-form df_k is computed as the gradient of the harmonic function as follows:

$$df_k([v_i,v_j]) = f_k \partial [v_i,v_j] = f(v_j) - f(v_i). \tag{5.15}$$

The n-dimensional exact harmonic 1-form basis is denoted by $\{df_1, df_2, \cdots, df_n\}$.

The next step of our algorithm is to compute the holomorphic 1-form basis. For this we compute the Hodge star of the exact harmonic 1-form $^*(df_k)$ using Equation 5.11, which is called the conjugate harmonic 1-form. The conjugate harmonic 1-form is also harmonic (Section 3.3). Intuitively, the exact harmonic 1-form and its conjugate harmonic 1-form are orthogonal everywhere. While using the Hodge star operator, in general, a regularization step is performed based on Hodge decomposition to decompose $^*(df_k)$ into exact, coexact and harmonic components [141]. The exact and coexact components are discarded and only the harmonic component is retained. However, in our method, since we use a reasonably good resolution mesh the Hodge star operator is accurate enough to avoid the regularization step. Finally, by pairing each base exact harmonic 1-form with its conjugate, we obtain the set of basis for the holomorphic 1-form on M as $\eta_k = df_k + i^*(df_k)$. Then, the holomorphic 1-form basis is represented by $\{\eta_1, \eta_2, \cdots, \eta_n\}$.

In the final step, using the holomorphic 1-form basis, we compute the induced conformal mapping $\phi : M \to \mathbb{C}$ by integration that maps the surface M to a planar domain $\mathbb{C}$. To start with, for each of the inner boundaries γ_k, we find the corresponding shortest paths τ_k connecting γ_k to the outer boundary γ_0. We cut M along one of these τ_k's to obtain a simply connected mesh $\bar{M}$. Then, we compute a unique holomorphic 1-form $\eta = \sum_{k=1}^{n} \lambda_k \eta_k$, $\lambda_k \in \mathbb{R}$, as a linear combination of the holomorphic 1-form basis, such that it satisfies the following topological condition:

$$Img \int_{\gamma_0} \eta = 2\pi, Img \int_{\gamma_k} \eta = -2\pi, Img \int_{\gamma_i} \eta = 0, i \neq 0, k \tag{5.16}$$

where Img denotes the imaginary part. Here, λ_k's are a set of unknowns which are obtained by solving the linear system in Equation 5.16. We then choose a base vertex $p \in \gamma_0$. For any vertex $q \in \bar{M}$, we choose an arbitrary integration path connecting p and q in $\bar{M}$. Finally, the conformal mapping is obtained by integrating η over this path as follows:

$$\phi(q) = exp(\int_p^q \eta) \tag{5.17}$$

Equation 5.17 maps the surface M to a 2D annulus. Note that several conformal maps are possible by changing the topological conditions in Equation 5.16. If we just compute the integral in Equation 5.17 without the exponential, then the mapping computed will map M onto a rectangle.

Since the mesh $\bar{M}$ is simply connected and η is holomorphic, the integration result is independent of the choice of the path and hence any arbitrary path can be chosen for integration to obtain the conformal mapping. In fact, the results after integration by choosing different paths are the same. In Equation 5.17, we are effectively integrating df_k and $^*(df_k)$ to obtain the conformal mapping of the mesh.

Algorithm 5.1: Heat diffusion metric based discrete conformal mapping.

Input: Surfaces M, heat diffusion metric $\mathbf{g}$.
Output: The conformal parameterization of M.
1. Replace the edge lengths in M with edge lengths from $\mathbf{g}$.
2. Compute the harmonic function by solving the linear system 5.12.
3. Compute exact harmonic 1-form, $df([v_i, v_j]) = f(v_j) - f(v_i)$ for each edge.
4. Compute the Hodge star of df by using Equation 5.11.
5. Compute special holomorphic 1-form η satisfying the topological condition in Eq. 5.16.
5. Integrate η to get the final 2D coordinate for each vertex, using Eq. 5.17.

Both the fields df_k and $^*(df_k)$ are harmonic 1-forms and since any harmonic 1-form is closed (Section 3.2), it implies that both these fields are locally integrable. In our case, we assume that the fake handles on the surface are tiny (which is typical), which implies that the corresponding tunnel and the handle loops are short [37]. Hence, the integration of the conjugate harmonic 1-form $^*(df_k)$ (Hodge star of exact harmonic 1-form df_k) is very close to zero. In practice, the harmonic 1-forms corresponding to the tiny handles can be neglected. Since the complex derivative (η) in Equation 5.17 is a holomorphic 1-form, it is equivalent to saying that the mapping obtained by integration satisfies the Cauchy-Riemann equation [99, 113] (Section 3.3). This proves that our mapping is indeed conformal with respect to the given heat diffusion metric.

We summarize our conformal mapping algorithm in Algorithm 5.1. Our flattening algorithm requires the input mesh to be a two dimensional manifold. In the entirety of our algorithm, we assume that the handles on the mesh surface are tiny. If the handles are large, then the harmonic 1-forms corresponding to the handles can no longer be ignored and it in turn affects the integration for computing the conformal maps. Moreover, we assume that the input mesh surface has good resolution so that the Hodge star regularization is unnecessary.

5.2.3 Generality of our Algorithm

As described above, our flattening algorithm is general and can handle flattening problems with complicated topologies containing any number of inner boundaries or holes. We now illustrate the different steps of our flattening algorithm using a genus zero surface with an arbitrary number of holes. For this, we use a complicated topology of a human face surface S with four boundary components, namely γ_0 is the outer boundary and γ_1, γ_2 and γ_3 are the inner boundaries. Thus, $n = 3$ in this case. We first evaluate the heat diffusion Riemannian metric $\mathbf{g}$ for the face surface. Then, we compute the corresponding exact harmonic 1-forms (df_1, df_2, df_3), conjugate harmonic 1-forms $(^*(df_1), ^*(df_2), ^*(df_3))$ using Hodge star, and holomor-

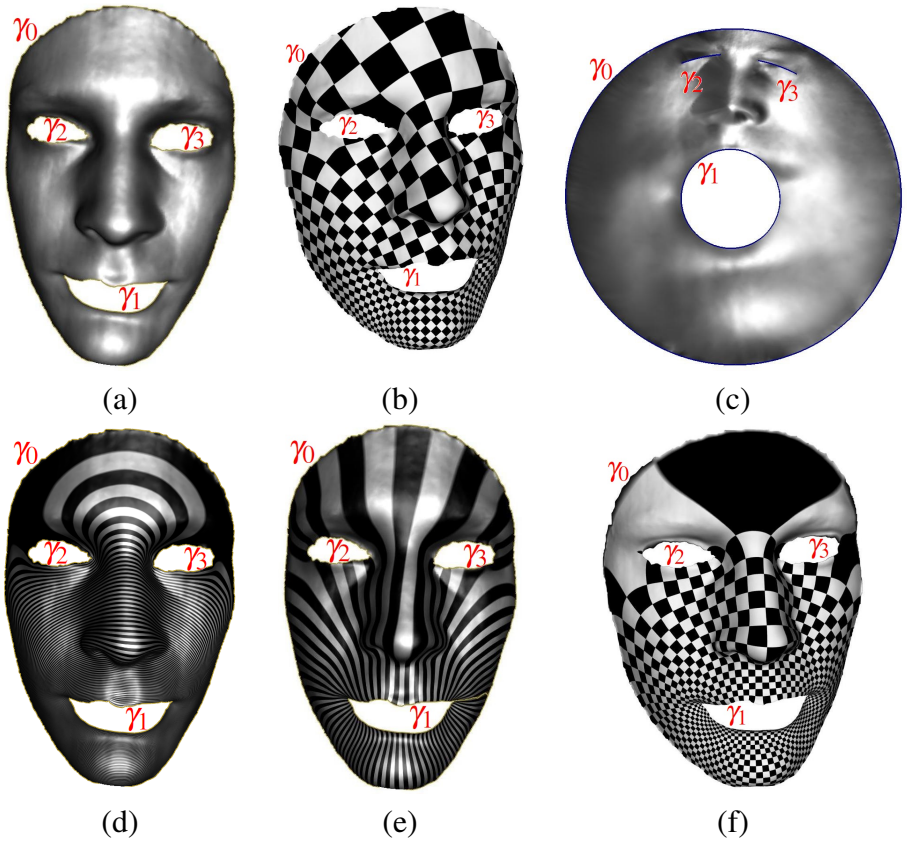


Figure 5.3: The flattening of (a) human face surface with outer boundary γ_0 and inner boundaries $\gamma_1, \gamma_2, \gamma_3$ using our algorithm. (b) Checker board mapping of (a), showing that angles are well preserved. (c) Slit map showing the flattening of (a). Level set visualization of: (d) exact harmonic 1-form, df_1 with respect to γ_1; (e) Hodge star of (d), $^*(df_1)$; (f) holomorphic 1-form, η_1 by combining (d) and (e).

phic 1-forms (η_1, η_2, η_3), as described earlier. All the 1-forms can be visualized by using level sets. Figures 5.3(d), (e) and (f) show the level set visualization of the exact harmonic 1-form (df_1), the conjugate harmonic 1-form $(^*(df_1))$, and the holomorphic 1-form (η_1), respectively, with respect to the inner boundary γ_1, that is $k = 1$ in Equation 5.12. Similar results can be obtained with respect to the inner boundaries γ_2 and γ_3 as well.

Now, using the holomorphic 1-form basis, we construct a conformal mapping from the input surface $(S, \mathbf{g})$ to a planar annulus domain by integration using Equation 5.17. This integration is carried out along an arbitrary path using the special holomorphic 1-form (linear combination of η_1, η_2 and η_3) satisfying Equation 5.16. This method is popularly known as *slit mapping* [152, 157] where one of the inner

boundaries of S is mapped to a circle and all the remaining boundaries are mapped to the concentric circular slits. Figure 5.3(c) shows the flattening result of the face where γ_0 is mapped to the outer radius, γ_1 is mapped to the inner radius, and γ_2 and γ_3 are mapped to the slits using our algorithm. To obtain the flattening in Figure 5.3(c), we cut the face along the shortest path connecting γ_1 and γ_0 and solve Equation 5.16 also over γ_1. Two additional slit map results are possible with respect to the other two inner boundaries, γ_2 and γ_3. If we map a checker board pattern onto the flat annulus in Figure 5.3(c) and have it correspondingly mapped back onto Figure 5.3(a), the angles and shapes inside the checker board are preserved since our algorithm is conformal. This fact is confirmed in Figure 5.3(b), illustrating that our algorithm is indeed angle (shape) preserving.

5.2.4 Colon Flattening

The colon surface is a special case with a simple topology (cylindrical) having only two boundary components, namely γ_0 and γ_1 ($n = 1$). Thus, the colon surface has only one holomorphic 1-form base, η_1. Then, the unique holomorphic 1-form η for integration is obtained as $\eta = \lambda_1 \eta_1$, satisfying the condition: $Img \int_{\gamma_0} \eta = 2\pi, Img \int_{\gamma_1} \eta = -2\pi$, which is a simpler form of Equation 5.16. Finally, a conformal mapping of the colon surface is obtained by integration using Equation 5.17. However, in the case of colon flattening, the common practice is to map the colon onto a rectangle since it is the more intuitive way to visualize the colon surface. Hence, we compute the logarithm of the integral in Equation 5.17 which now maps the colon to a rectangle instead of an annulus. In other words, we are effectively only computing the integral part of Equation 5.17 without the exponential. Thus, the colon surface is conformally mapped to a planar rectangle to obtain a flattened rectangular map of the colon surface. We chose conformal mapping for the colon flattening due to the following main reason: In VC, after the colon surface is flattened, it is mandatory to preserve the local shapes. Conformal maps, by definition, are angle preserving (local shape preserving) (Section 3.1). Consequently, polyps can still be identified based on their circular shape on the flattened colon image.

5.2.5 Euclidean versus Heat Diffusion Flattening

Our conformal flattening algorithm can be used either with the heat diffusion metric or with the original Euclidean metric. We now show that our flattening algorithm is more robust using the heat diffusion metric (obtained using the HDD) when compared to using the original Euclidean metric. For this, we consider a patch of a teapot surface with the handle and p_0, p_1, p_2, p_3 being four corners on

the boundary as shown in Figure 5.4(a). We now manually cut the handle along a loop indicated by the red line in Figure 5.4(b). We then use our flattening algorithm to map both the teapots in Figures 5.4(a) and (b) to a planar rectangle by using the heat diffusion metric as well as the original Euclidean metric. The teapot handle is collapsed onto the plane after flattening. Figures 5.4(c) and (d) show the conformal flattening results by using the original metric for Figures 5.4(a) and (b), respectively. Figures 5.4(e) and (f) show the conformal flattening results by using the heat

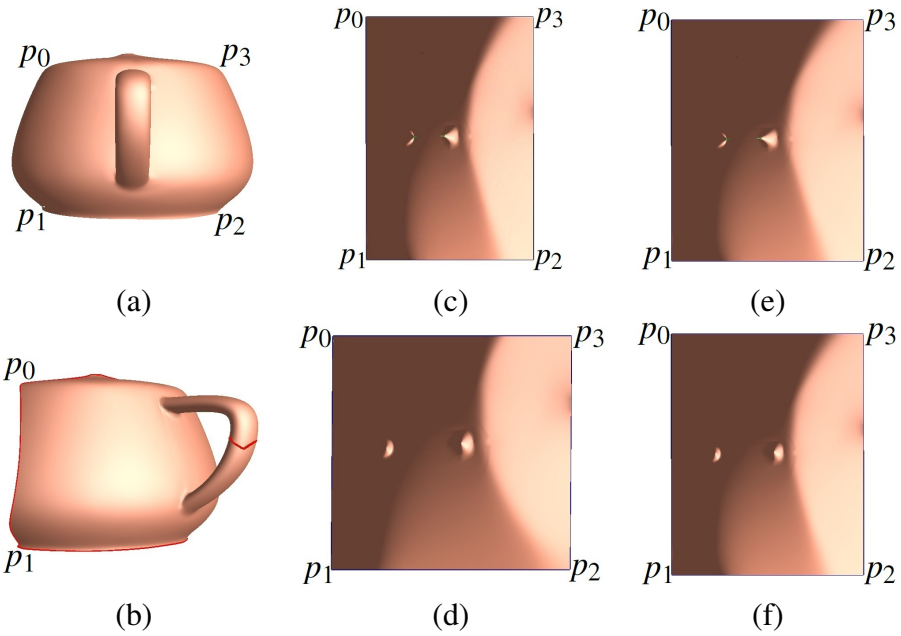


Figure 5.4: Comparison of our flattening algorithm using the original Euclidean metric and heat diffusion metric. (a) Teapot patch with handle; (b) Teapot patch with handle cut along a loop, shown in red; (c) Conformal module of (a) using Euclidean metric; (d) Conformal module of (b) using Euclidean metric; (e) Conformal module of (a) using heat diffusion metric; (f) Conformal module of (b) using heat diffusion metric.

diffusion metric for Figures 5.4(a) and (b), respectively. When a surface is conformally mapped to a planar rectangle, the ratio between the height and the width of the rectangle is called the *conformal module* of the surface. It can be clearly seen that the conformal modules based on the original metric changed much, due to the cutting of the handle (change in topology) whereas the conformal modules based on the heat diffusion metric are similar. These experimental results confirm that the heat diffusion metric is preferrable for our flattening algorithm than the original metric, due to its insensitivity to topological noise.

5.3 Comparison

The current approach is much more efficient and robust to topological noise when compared to previous methods. Here, we compare our method to the well known conventional holomorphic 1-form method, the conventional topological denoising method, and the Ricci flow method and show that ours fairs better for colon surfaces with noise.

5.3.1 Conventional Holomorphic 1-form Method

In the conventional holomorphic 1-form method, the most time consuming part is solving the Poisson equations [68]. Every handle on the mesh corresponds to two cohomological classes. Since the harmonic 1-forms are computed for all the cohomological classes, if the colon surface has g handles, then the Poisson equations in the conventional holomorphic 1-form method need to be solved $2g$ times [68]. In practice, there may be more than a hundred fake handles and therefore the computation is very time consuming. Hence, though theoretically this method can also be used for flattening noisy colons, it is not efficient. On the contrary, in our method, we need to solve only a single Laplace equation. This is because we only consider the cohomological class corresponding to a cylinder and ignore all the other classes (the colon is roughly cylindrical). Hence we compute only one harmonic 1-form, by solving only one Laplace equation given by Equation 5.12. Thus, our method is virtually independent of the number of handles, thereby increasing the efficiency.

5.3.2 Conventional Topological Denoising

In the conventional topological denoising method, initially, for each handle, two loops, namely the handle loop and the tunnel loop are computed [37]. Then, the handle is cut along one of these loops, which are then filled with two small disks to ultimately remove the handles. In this method, it is extremely challenging and time consuming to compute the two loops for each handle since there might be hundreds of handles. Moreover, the topological surgery on the meshes is also complicated. In our method, by virtue of the metric used, all the fake handles present on the colon surface are collapsed and flattened on the destination plane. Consequently, all the vertices near the handles have non-zero curvatures. Thus, by simply evaluating the Gaussian curvature of the vertices the handles can be easily located (handles occur at vertices with non-zero curvatures). The handles can then be removed by removing the neighborhood of these vertices with non-zero curvature and filling small disks into the holes. This provides a simpler and more efficient way to remove handles compared to the traditional denoising method. We provide more elaborate

details of the handle detection and removal using our method in Section 6.

5.3.3 Ricci Flow Method

The Ricci flow method deforms the Riemannian metric to the constant curvature uniformization metric [81], which induces constant curvatures everywhere. This constant is solely determined by the topology of the surface. The colon surface is a topological cylinder. Hence, the uniformization metric induces zero Gaussian curvature for the interior points, and a zero geodesic curvature along the boundaries. However, for colon surfaces with topological noise (fake handles), the Ricci flow converges to a negative constant curvature metric and hence they cannot be flattened onto the Euclidean plane. Therefore, the Ricci flow method is intrinsically vulnerable to topological noise. On the contrary, our method is insensitive to topological noise and a robust flattening is achieved.

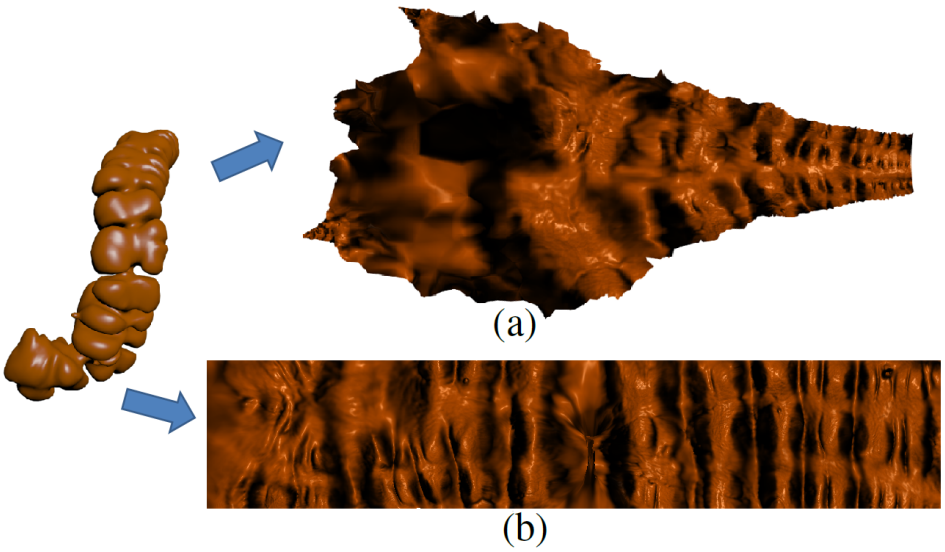


Figure 5.5: The flattening of the ascending segment of a colon using (a) Ricci flow, and (b) our method.

For comparison, we show in Figure 5.5 a flattened result of the ascending colon segment, using both ours and the Ricci flow approach. Figure 5.5(a) shows the flattening result of the colon segment using the Ricci flow method and Figure 5.5(b) shows the flattening result of the colon segment using our method. It can be clearly seen that the Ricci flow flattening is non-regular and the global shape is lost. This is because the Ricci flow method is sensitive to noise, such as handles. On the other hand, the colon flattening using our approach is regular and the entire global shape is preserved.

Table 5.1: Comparison of the running times of our colon flattening approach with the colon flattening using the Ricci Flow method.

Model Num.	#Faces	Time (sec) Ricci Flow	Time (sec) Our Method
192 (Supine)	56,800	125.7	11.2
192 (Supine A)	12,500	35.6	3.4
192 (Supine T)	23,100	62.2	5.2
192 (Prone)	52,900	112.5	12.3
192 (Prone A)	12,800	33.6	3.3
192 (Prone T)	22,700	57.2	5.4
241 (Supine)	55,900	121.5	11.4
241 (Prone)	53,100	110.8	11.3

We compared our method with the Ricci flow approach, in terms of the colon flattening running times. Table 5.1 shows the comparison results using different colon models and segments. The first column shows the number and segment of the colon model used. 192 and 241 are model numbers of the colon, *A* represents the ascending segment of the colon and *T* represents the transverse segment. The second column shows the number of faces in the colon model. The third column gives the time taken (in sec) for flattening the colon models using the Ricci flow method. Finally, the fourth column gives the time taken (in sec) for flattening using our method. Table 5.1 shows that for all the colon models, our method is an order of magnitude faster than the Ricci flow method. Note that the timing details provided in Table 5.1 only show the time taken for flattening and do not include the time taken for the metric evaluation. Since both methods use different metrics, for the sake of comparison we only use the time taken for flattening. All the experiments were conducted on a system equipped with an Intel(R) Xeon(R) CPU with a 1.87 GHz processor.

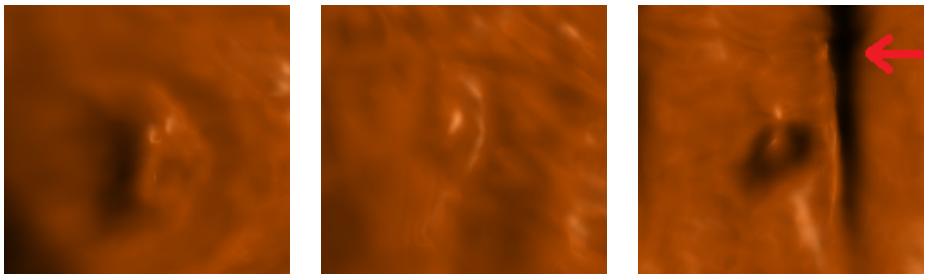
5.4 Applications

Our major focus is to achieve a robust colon flattening, thereby avoiding the time-consuming topological denoising step. Since the shape information is preserved in the flattened colon, we show its application for polyp visualization, espe-

cially in hidden regions such as behind the folds. We also show how the supine and prone colon registration pipeline becomes more robust by using our method.

5.4.1 Polyp Visualization and Detection

One of the important applications of colon flattening is to provide a better way for the physicians to visualize and even detect the polyps. Specifically, polyps behind haustral folds are hidden and missed during the VC fly through of a 3D model. The polyps are small protrusions or bumps on the colonic wall. By using our method, all the shapes on the colon surface are preserved even after flattening. Therefore, the polyps can be clearly seen as bumps and hence form an effective means of polyp visualization. Furthermore, the volume rendering of the flattened colon provides a realistic rendering of polyps. In addition, the physician can zoom-in at the suspicious regions to confirm the location of the polyps. Thus, even relatively small polyps can be seen, as shown by few examples in Figure 5.6.



(a) (b) (c)

Figure 5.6: Close up view of the polyps (bumps on the colon wall). (a) Polyp 1 in Figure 5.9(a); (b) Polyp 2 in Figure 5.9(a); (c) Polyp 3 in Figure 5.9(c), which is hidden behind a colonic fold indicated by the red arrow.

Figure 5.6 shows a close up view of some of the polyps (protrusions) observed by navigating along the flattened colon surface. Figure 5.6(a) shows a close up view of polyp 1 in Figure 5.9(a), which is a large polyp. Figure 5.6(b) shows a close up view of polyp 2 in Figure 5.9(a) which is a relatively small polyp. Figure 5.6(c) shows a close up view of polyp 3 in Figure 5.9(c) that is hidden behind a fold (pointed to by the red arrow). It is difficult to find such polyps during navigation using a conventional VC system. However, when a flattened colon is available, even polyps hidden behind colonic folds can be observed, as seen in Figure 5.6(c). Another example of a close up view of a polyp is shown in Figure 5.1(b). These close up views help the physician to verify that the suspicious regions detected are indeed polyps and not some leftover stool. The shape of polyps can be identified and

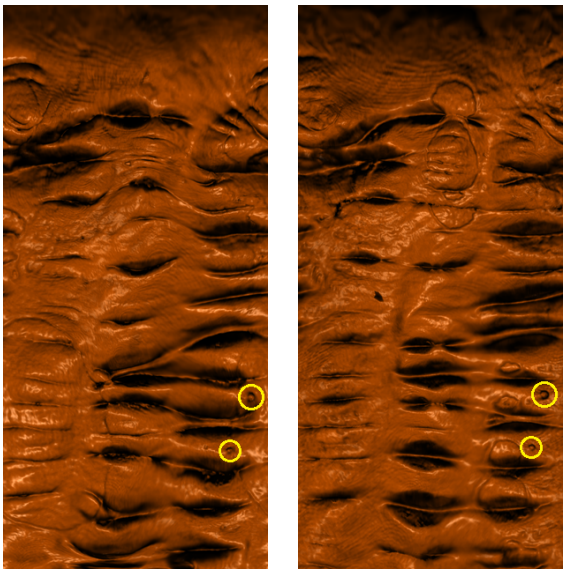
their sizes can be measured. Note that our technique does not provide an automatic method of polyp detection but rather makes it easy to locate, confirm, and visualize polyps, especially in areas which are otherwise hard to navigate. By performing size and shape analysis on the flat colon, we can detect the polyps automatically, which we plan to do in the future.

5.4.2 Colon Registration

Shape registration is very fundamental for shape analysis problems, especially for abnormality detection in medical applications. The colon deformation and the diverse shapes of polyps make it difficult to distinguish polyps from other non-threatening objects in the colon. Hence, for a VC procedure, CT scans of the abdomen are commonly acquired with the patient in both supine (facing up) and prone (facing down) positions to improve the visualization of the colon wall, reduce false positives, and improve sensitivity. Comparisons between the supine and prone colon surfaces can be facilitated by computerized registration between these scans.

Registration of supine and prone colon surfaces using quasi-conformal mapping has been described by Zeng et al. [54, 158]. In their approach, a costly topological denoising step is performed on both supine and prone colon surfaces before the start of the registration pipeline. Using the same registration approach, we show the registration of noisy supine and prone colon surfaces. We use our colon flattening approach to obtain the rectangular maps of supine and prone colon surfaces with handles. By virtue of our method, the flattening is not affected by the handles. We then obtain the surface feature points on these supine and prone flattened maps. Finally, using these feature points as constraints, we register the supine to prone using quasi-conformal mapping steps [158]. Therefore, we improved the robustness of the supine and prone colon registration algorithm [158] by directly registering the supine and prone colon surfaces with handles, which otherwise was impossible without denoising.

We have analytically evaluated the quality of the registration by a distance measurement between corresponding features located on the registered colon surfaces [158]. Since our registration is in the 2D space using the flattened colon surfaces, a point in $\mathbb{R}^3$ on the original colon surface corresponds to a point in $\mathbb{R}^2$ on the registered surface. For two corresponding points (polyps or feature points used as constraints) on the supine and prone flattened colons, we compute the L^2 norm of their 2D coordinates with the width of the flattened images fixed to a unit length of 1. We also compute the 3D distance error by measuring the distance between the same two corresponding feature points on the supine and prone original colon surfaces (see [158] for details of the evaluation procedure). We have evaluated the



(a) (b)

Figure 5.7: Registered flattened views of the ascending colon segments with handles of (a) supine and (b) prone colon surfaces. Two polyps found on (a) (shown in yellow circles) can be located on (b) (shown in yellow circles) at nearly the same position.

registration using a total of 6 pairs on 2 datasets by considering 16 pairs of feature points. The average distance error is 0.0385 in $\mathbb{R}^2$ and 8.14 mm in $\mathbb{R}^3$, which is comparable to the error values of Zeng et al. [158]. Figure 5.7 provides a visual verification for supine-prone colon registration. Figures 5.7(a) and (b) show the registered flattened views of the ascending colon segments of supine and prone colon surfaces with handles, respectively. Two polyps found on the flattened supine surface (yellow circles in Figure 5.7(a)) can be located on the flattened prone surface (yellow circles in Figure 5.7(b)) at nearly the same position. Moreover, the images of the registered segments in Figure 5.7 show very good alignment of the supine and prone colon structures and a good correspondence between their features, such as folds and polyps.

5.4.3 Handle Detection and Removal

Our method of colon flattening using the heat diffusion metric is insensitive to topological noise and hence the flattening process does not require any denoising as a mandatory pre-processing step. Nonetheless, it is important to remove these topological artifacts such as the tiny handles, so that they do not obstruct any further processing of the data, that is, if the user wants to perform any further simplifica-

tion of the colon data for other applications. In all the previous methods, the handles were detected and removed directly on the 3D model. Hence, only a limited visualization of the location of the handles was obtained. On the contrary, in our method, we detect the handles on the flattened colon surface and thus obtain a clear visualization of the handles. This helps us to know the location of the handles better.

By using the heat diffusion metric, the region around the handles is smoothed while flattening. As a result, a small overlapped region is formed at the location of the handles. In other words, a non-zero curvature is formed for all the vertices in the regions with handles. Thus, the problem of handle detection would simply boil down to the problem of finding all the vertices with non-zero curvature. Therefore, we calculate the Gaussian curvature for each vertex of the colon mesh. We then mark all the vertices which have a non-zero Gaussian curvature. These vertices are nothing but the vertices around the handles. In this way, all the handles can be detected in a fast and effective way.

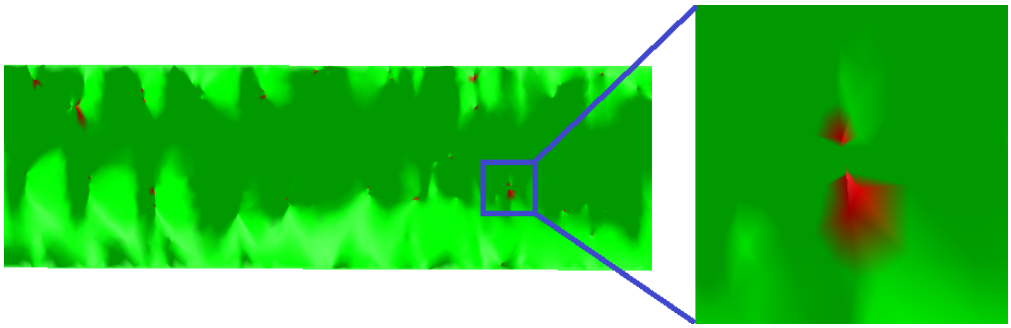


Figure 5.8: Handles detected by computing the Gaussian curvature for each vertex. Red areas indicate handles detected (one shown in close-up view) and the green area shows the zero Gaussian curvature region.

Figure 5.8 shows the result obtained by using our approach to detect handles for one third segment of a flattened colon surface (starting from the cecum on the left). The green color represents the regions with zero Gaussian curvature and the red color represents the regions with non-zero Gaussian curvature. Hence, all the red regions on the colon surface in Figure 5.8 show the handles detected using our approach. Figure 5.8 also shows the close up view of a red-colored region, which confirms that it indeed is a handle (flipped triangles). Once all the handles have been detected, they can be easily removed. For this, we delete the faces around the vertices with non-zero Gaussian curvature (red regions representing handles) and subsequently fill the resulting holes with small disks. Thus, all the handles are removed to obtain a flattened colon surface free of topological noise.

We compared the speed of our approach of handle detection and removal with a

previous approach [68]. Using our approach, the average time for the entire process of handle detection and removal on the colon surfaces with around 55,000 faces was 2.8 secs. On the other hand, the previous approach took an average of 800 secs to detect and remove all the handles using the same colon surface. Thus, we have obtained more than two orders of magnitude gain in speed using our approach of noise removal. In total, there were 43 handles on the colon surface shown in Figure 5.8. We tried our approach on four other colon surfaces and all the handles in all cases were detected proving the accuracy and efficiency of our approach. We also verified the handle count obtained by our method with that obtained by the previous method [68] and the value matched in all cases.

5.5 Results

5.5.1 Flat Colon Rendering

The result of our method is a colon surface flattened onto a 2D rectangle, which also results in polyps being flattened. The shape of the polyps is a good clue for polyp detection and hence the rendering of the flattened colon image is crucial for their detection. Volume rendering of flattened colons has been presented and suggested for use in VC navigation [68]. We perform the volume rendering of the flattened colon in the same way and obtain a high-quality image of the flattened colon. The volume rendered flattening result of a colon mesh model using our method is shown in Figure 5.9. Since the colon is very long, we show the flattening result in three parts. The colon stretches from the rectum which is on the left of Figure 5.9(a) to the cecum which is on the right of Figure 5.9(c). We can clearly see how easy it is to examine the whole interior colon region using our method. Moreover, important features such as the polyps and the haustral folds are clearly visible and well preserved. We show the location of three polyps (1, 2 and 3) on the flattened colon surface, marked in yellow circles in Figures 5.9(a) and (c). The resolution of the rendered image in Figure 5.9 is 3000×200. The rendering was performed on a system equipped with an Intel Xeon E5620 CPU and NVIDIA GeForce GTX 480 graphics board.

Polyp 1 is a large polyp of size 6.1×9.6 mm and can directly be inspected in Figure 5.9(a). However, polyp 2 of size 3.1×3.7 mm and polyp 3 of size 3.8×2.5 mm are relatively small and hard to recognize. Therefore, in a clinical application, the resolution should be at least four times higher than the one used in Figure 5.9, such as shown in Figure 5.6. The rendering using the GPU provides a real-time high-quality zoom-in functionality, which allows the physicians to interactively inspect suspicious regions. By having a flattened colon visualization, we are providing a better means of navigation so that no area of the colon is missed.

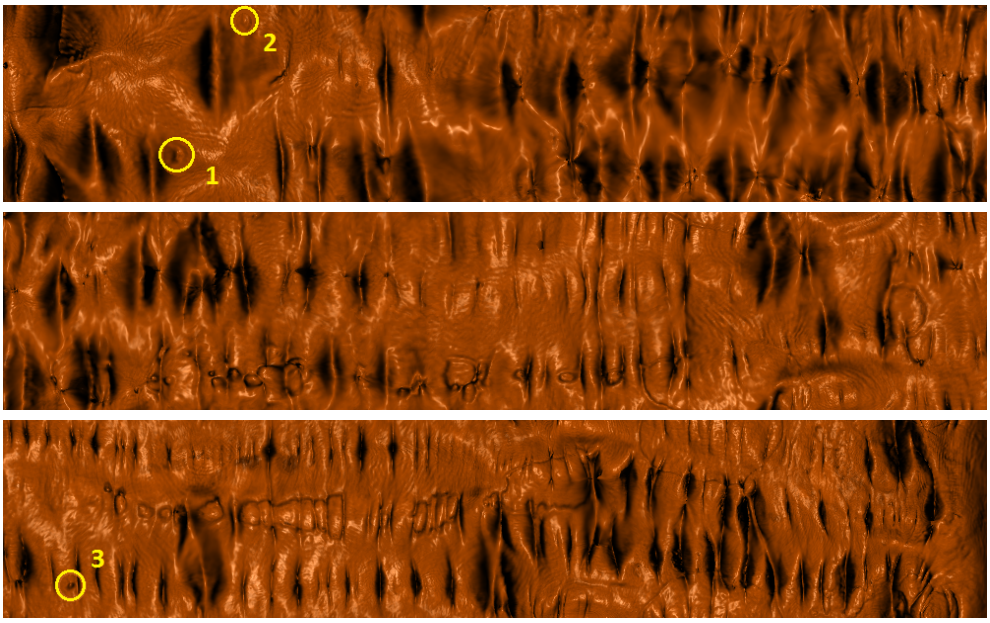


Figure 5.9: A flattened image for a whole colon dataset is shown in three images. The rectum of the colon is on the left of (a) and the colon stretches to the cecum, which is on the right of (c). The colonic polyps and the haustral folds are well preserved. Three polyps, 1 and 2 in (a) and 3 in (c) are shown within the yellow circles.

This flattened colon visualization opens an additional option for the physician to improve the colon surface exploration.

We have also shown our results to a radiologist who was involved with the early conception of VC and has over ten years of experience in reading them. He noted that the flat rendering was realistic, and that the anatomical features such as folds, and especially significant polyps ($\geq$6 mm in diameter) are well preserved and easily noticeable (he had not been exposed to flattened rendering prior to viewing this work). In addition, we have cross-verified the location of the polyps on the flattened colon by checking with the VC and OC reports provided by the Walter Reed Army Medical Center. We plan to conduct in the near future a more comprehensive study with radiologists.

5.5.2 Implementation

The pipeline was implemented in C++ in a Windows environment. The volumetric rendering was performed on the GPU using OpenGL and Cg. The colon data used in this work come from volumes with an approximate size of $512 \times 512 \times 400$ voxels. Preprocessing of the volumes includes electronic cleansing, segmentation,

triangular mesh extraction, and skeleton extraction. The meshes obtained are very large (typically over 1.5 million faces) and are simplified to approximately 5% of their original size. We have successfully obtained flattening results for six different colon models. Apart from the flattening of the whole colon, we have also segmented the colon into three segments, namely ascending, transverse and descending segments. Thus, we have tested our colon flattening approach successfully with a total of 24 cases, including the segments and the whole colon surfaces. The whole process of colon flattening for an entire colon model with around 55,000 faces took an average of 23.93 sec, not including the time taken for the pre-processing of the colon. It took on average 12.38 sec to evaluate the heat diffusion metric using 50 eigenvalues and eigen functions and an average time of 11.55 sec to obtain the flattening using this heat diffusion metric.

Chapter 6

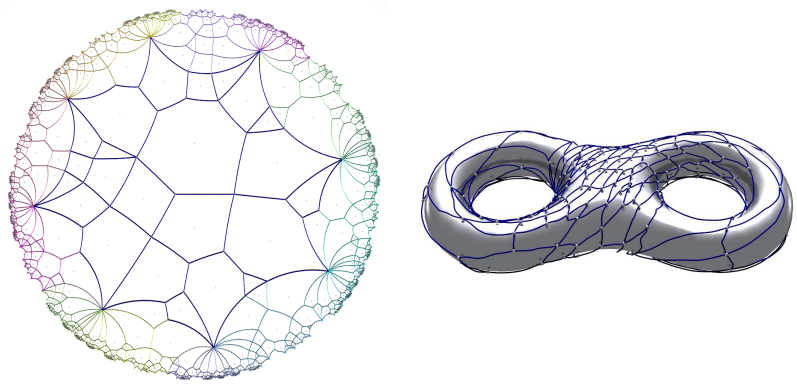
Graph Embedding

In this chapter, we propose a novel, and general approach for embedding planar graphs onto a sphere based on Ricci flow obtained by optimizing a convex energy. The embedding has no edge-crossings, and is more intuitive to visualize graphs. The method has three stages, first the graph is embedded on a topological surface, then on a Riemann surface, and finally on a surface in three dimensional Euclidean space. We employ an efficient Newton's method to optimize Ricci flow. The proposed method produces a consistent and topologically stable graph layout. Any local structural changes of the graph do not affect the overall embedding. As a result, we apply our method for efficient visual graph comparison and in identifying, understanding and visualizing dynamic graphs. Our experimental results demonstrate the efficacy, efficiency and robustness of our graph embedding method.

This chapter is organized as follows. Section 6.1 provides the motivation. Section 6.2 provides the necessary theoretical background and Section 6.3 describes the algorithm. Section 6.4 and Section 6.5 describe the application of our approach to visual graph comparison and dynamic graph visualization respectively. Using our method to visualize sensor network data is shown in Section 6.6.

6.1 Motivation

Graphs are used to represent and visualize relationships between data items. Graph embedding methods play an important role in visualizing the data items and their relationships by providing an automatic and clutter free layout. A graph $G(V,E)$ is an abstract structure, modeling a relation E over a set V of nodes. Graphs play fundamental roles in many engineering fields. For example, they have been used to model social relation of communities, traffic between telecommunication switches, networking with wireless sensor nodes and airline routes among cities. Graphs are abstract and ubiquitous and it is high desirable to develop general, practical and rigorous algorithms to visualize graphs.



(a) (b)

Figure 6.1: Embedding a graph onto (a) Riemann surface $\mathbb{H}^2/\Gamma$; (b) surface in $\mathbb{R}^3$ (torus in this case).

Conventionally, graphs are mapped into low dimensional Euclidean spaces $\mathbb{R}^n$ ($n \leq 3$), the unit sphere $\mathbb{S}^2$ or the hyperbolic plane $\mathbb{H}^2$. Planar graphs can be easily embedded onto the unit sphere. But for most non-planar graphs, such kind of mappings do not preserve the topological structures (connectivity) of the graph. Moreover, edge crossings are unavoidable. The visual clutter (edge crossing) poses a great obstacle for analyzing graphs. Many methods have been proposed to improve the visualization of graphs by reducing edge crossings. However, finding layouts of general graphs without edge crossings remains one of the most challenging problems in graph visualization.

In this work, we propose a novel graph embedding (layout) approach using Ricci flow energy optimization. This method removes the edge crossings completely, and preserves the topological structure of the graph. This approach is general, *practical* for visualizing large scale graphs, and *rigorous* with solid theoretic foundations.

Several criteria that define a good graph visualization (aesthetics) have been determined [18, 126, 139]. We believe that the following two criteria are essential for any good graph embedding approach: (1) the embedding should have no overlapping vertices and no edge-crossings and (2) the embedding ensures that the resulting layout conveys the correct information to the user, that is, the node connectivity is preserved. Our proposed graph embedding approach using Ricci flow fulfills both these criteria. Our approach provides connectivity preserving, topologically stable and consistent graph layout.

Our method is based on the fact that all graphs can be embedded onto a two dimensional surface. Our algorithm pipeline has three stages. In the first stage, the

graph is embedded onto a *topological surface*. In the second stage it is embedded onto a *Riemann surface* and in the last stage it is finally embedded onto a *surface in* $\mathbb{R}^3$. Figure 6.1 shows the results illustrating the embedding of an example graph onto a Riemann surface and onto a surface in $\mathbb{R}^3$ (torus). Figure 6.1(a) shows the embedding of the graph onto a Riemann surface (stage two) and Figure 6.1(b) shows the final embedding of the graph onto a torus (stage three).

1. Embedding on Topological Surface A rotation system of graph G with vertex set $\{v_1, v_2, ..., v_n\}$ is a collection $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$ such that π_i is a cyclic permutation of the edges incident with v_i. An embedded graph is denoted by G_Π where G is a connected graph and Π is a rotation system of G. π_i is called the $\Pi - clockwise$ orientation around v_i. If the valence of v_i is k, then the edges' orientation around v_i is $\{e_{\pi_i(1)}, e_{\pi_i(2)}, ..., e_{\pi_i(k)}\}$. A rotation system Π determines an embedding surface of G, denoted as G_Π.

2. Embedding on Riemann Surface In this step, we assign a geometric structure (Riemannian metric) to the topological surface R using Ricci flow [31, 80]. Ricci flow deforms a Riemannian metric proportional to its Gaussian curvature, such that the curvature evolves according to a heat diffusion process. Eventually, Ricci flow leads to a Riemannian metric, such that the curvature is constant everywhere.

The graph G and its dual graph together tessellate the surface R such that each cell is a quadrilateral. By adding the diagonals to the quadrilaterals, the topological surface of G_Π becomes triangulated. By running Ricci flow, a constant curvature Riemannian metric is produced on G_Π. A topological surface with a constant curvature Riemannian metric is called a *Riemann surface*. A Riemann surface has canonical representations. Genus zero Riemann surface can be represented as the unit sphere $\mathbb{S}^2$. Genus one Riemann surface is represented as the quotient space $\mathbb{R}^2/\Gamma$, where Γ is a translation graph acting on $\mathbb{R}^2$, generated by two translations. High genus Riemann surfaces can be represented as the quotient space $\mathbb{H}^2/\Gamma$, where Γ is generated by $2g$ hyperbolic rigid motions. Our method will give such Riemann surface representations and embed the graph on these Riemann surfaces to make the navigation much easier.

3. Embedding on Surface in $\mathbb{R}^3$ A Riemman surface with non-zero genus can be cut open along a special set of geodesics. For genus one surfaces, the open Riemman surface is a parallelogram. For high genus surfaces, the open Riemann surfaces are hyperbolic polygons. These Riemann surfaces (may to be parallelogram or hyperbolic polygon) are called as *fundamental polygons*. For high genus surfaces,

the fundamental polygons are Euclidean convex polygons on the Hyperbolic disk model of $\mathbb{H}^2$.

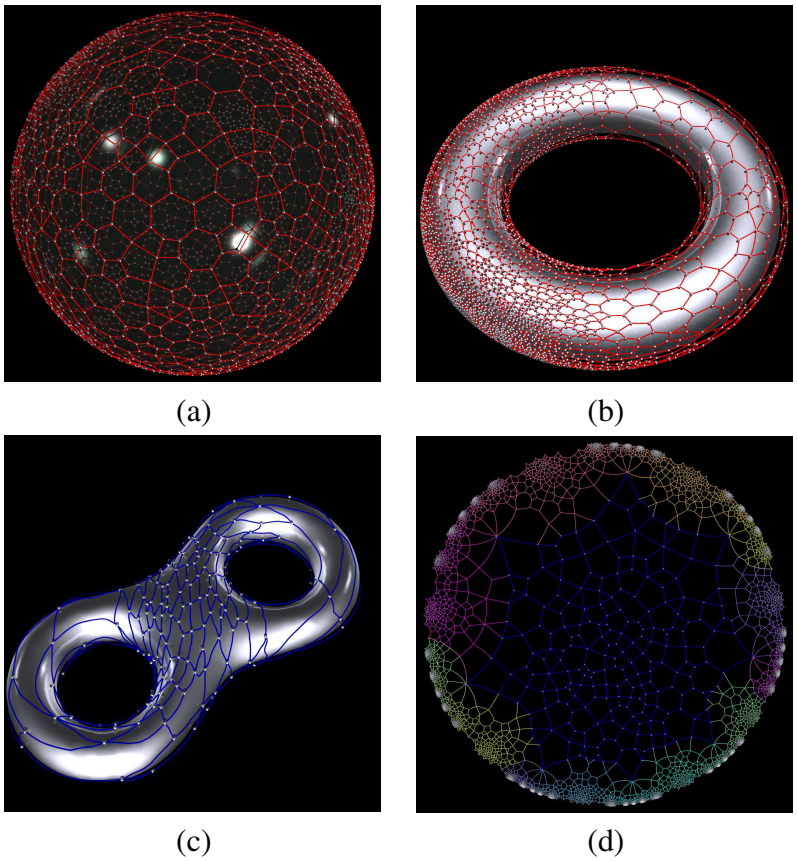


Figure 6.2: Embedding of different graphs onto a surface in three dimensional Euclidean space by using our three stage Ricci flow based embedding approach. (a) Genus zero graph embedded onto a sphere; (b) Genus one graph embedded onto a torus; (c) Genus two graph embedded onto a torus; (d) High genus graph embedded onto a Riemann surface.

Figure 6.2 shows the embedding of different graphs onto a surface in three dimensional Euclidean space by using our three stage Ricci flow based embedding approach. Figure 6.2(a) shows the embedding result of a genus zero (planar) graph onto a sphere. Figure 6.2(b) shows the embedding result of a genus one graph onto a torus. Figure 6.2(c) shows the embedding result of a genus two graph onto a torus. Finally, Figure 6.2(d) shows the embedding result of a high genus graph onto a Riemann surface.

Our Ricci flow graph embedding results in a consistent layout that helps to efficiently deal with problems that pertain to identifying and tracking the structural

changes in graphs. We show how our embedding method successfully addresses two of the important problems - visual graph comparison and dynamic graph visualization. Visual graph comparison is a challenging task which not only involves matching graphs but also provides a means that allows the users to visually confirm it. Generally, two isomorphic graphs differ in topology and node positions making it a non-trivial task to visually know if they are similar. Our method matches the graphs in terms of connectivity and facilitates visual graph comparison. Dynamic graph visualization is a more recent topic of research and it addresses the problem of graphs changing over time where vertices and edges are added and removed over time. Most of these changes occur at local sub-regions of the graph. Irrespective of the structural changes over time our embedding approach provides a stable graph layout that preserves the mental map, making it easy to understand the changes of the graph with minimum user effort.

This work proposes a novel method to embed undirected graphs on surface for visualization purposes. The main contributions of this paper are:

1. Develop an algorithm to embed a graph onto a topological surface based on rotation system G_Π, and reduce the genus of G_Π to a relative small genus $G_{\Pi'}$.

2. Generalize discrete Ricci flow from triangular mesh setting to general graph setting.

3. Develop a method to embed a graph onto a Riemann surface.

4. Application to dynamic graph visualization and visual graph comparison

The method is general, practical, rigorous, and the embedding results have no edge crossings. Moreover, our method uses Newton's method which is computationally more efficient compared to the conventional gradient descent method. Our method provides a stable graph layout that is not affected by local structural changes which forms the motivation to address the challenges of graph comparison and dynamic graph visualization.

6.2 Theory

We present here brief theoretical fundamentals required to explain our Ricci Flow graph layout.

Algebraic Topology Let S denote a manifold.

A curve is defined as a continuous mapping $\gamma : [0,1] \to S$

A closed curve (loop) through a point p is a curve, such that $\gamma(0) = \gamma(1) = p$

Let $\gamma_1, \gamma_2 : [0,1] \to S$ be two curves. A homotopy connecting γ_1 and γ_2 is a continuous mapping $F : [0,1]x[0,1] \to S$, such that $F(0,1) = \gamma_1(t), F(1,t) = \gamma_2(t)$

We say γ_1 is homotopic to γ_2 if there exists a homotopy between them.

Let q be a base point $\in S$, then all the oriented closed curves (loops) through q can be classified by the homotopy. All the homotopy classes form the so-called *fundamental group* of S, denoted as $\pi_1(S,q)$.

For a genus g closed surface, one can find *canonical homotopy group generators* $\{a_1, b_1, a_2, b_2, \cdots, a_g, b_g\}$, such that $a_i \cdot a_j = 0$, $b_i \cdot b_j = 0$, $a_i \cdot b_j = \delta_{ij}$, where the operator $r_1 \cdot r_2$ represents the algebraic intersection number between two loops γ_1 and γ_2, and δ_{ij} is the Kronecker symbol.

Let $p : \tilde{S} \to S$ be a continuous surjective map, such that for each point $q \in S$ with a neighborhood U, its corresponding preimage $p^{-1}(U) = \cup_i \tilde{U}_i$ is a disjoint union of open sets $\tilde{U}_i$, and the restriction of p on each $\tilde{U}_i$ is a local homeomorphism. Then $(\tilde{S}, p)$ is a *covering space* of S and p is called a *projection map*. The automorphisms of $\tilde{S}$, $\tau : \tilde{S} \to \tilde{S}$, are called *deck transformations*, if they satisfy $p \circ \tau = p$. All the deck transformations form a group, termed as *covering group*, and denoted as $Deck(\tilde{S})$.

If a covering space $\tilde{S}$ is simply connected (i.e., $\pi_1(\tilde{S}) = \{e\}$), then $\tilde{S}$ is called a *universal covering space* of S.

Combinatorial Topology of Graph A graph is represented as a vertex set and an edge set denoted by $\{v_1, v_2, ..., v_n\}$ and $\{e_1, e_2, ..., e_m\}$, respectively. Each edge e_i has two vertices as end points. The valence of a vertex v_i is defined as the number of edges incident with v_i as an end point.

We treat embedding of a graph G to be purely combinatorially. A rotation system of graph G with vertex set $\{v_1, v_2, ..., v_n\}$ is denoted as a collection $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$ such that π_i is a cyclic permutation of the edges incident with v_i. The embedded graph is denoted by G_Π where G is a connected graph and Π is a rotation system of G. π_i is called the $\Pi - clockwise$ orientation around v_i. If the valence of v_i is k, then the edges' orientation around v_i is $\{e_{\pi_i(1)}, e_{\pi_i(2)}, ..., e_{\pi_i(k)}\}$.

Let v_1 and v_2 be the two end vertices of an edge e_1. Then the triplets (v_1, e_1, v_2) and (v_2, e_1, v_1) are defined as two edge sides of the edge e_1. Similarly, the triplet $(e_{\pi_2(i+1)}, v_2, e_{\pi_2(i)})$ is defined as a corner of a vertex v_2. This corner is denoted as $\hat{v}_2^i$ ($i = 0, 1, ..., k-1$ where k is the valence of v_2).

A chain C of a graph G is an alternating sequence of vertices and edges such that any two consecutive elements are related. We call the chain is closed if the first and the last elements are vertices. Then, the first and the last vertex of the closed

chain are termed as its extremities. Thus, it can be seen that a chain is made up of edge sides and corners. A closed chain with the same extremities is a circle.

Consider a chain sequence $x_0, y_0, x_1, y_1, ..., x_{r-1}, y_{r-1}, x_r$ (x_i is vertex, y_i is edge and r is an interger greater than 1) of vertices and edges, where the indices are expressed as modulo of r and y_i is an edge joining x_i and x_{i+1} for $i = 0, 1, ..., r-1$. Since the indices are expressed as a modulo of r, we have $x_0 = x_r$ and thus this chain is a circle. Suppose $x_i = v_j$ and if $y_{i-1} = \pi_j(a+1)$ and $y_i = \pi_j(a)$, then the chain sequence is called a facial walk of G_Π.

Suppose G_Π has a set of facial walk $\{f_1, f_2, ..., f_s\}$ and if a corner $\hat{v}_a^i$ appears in the facial walk f_j sequence, we denote as $\hat{v}_a^i \in f_j$. The genus g of G_Π is defined by Euler's formula

$$V - E + F = 2 - 2g(G_\Pi)$$

where V, E and F are the number of vertices, edges and facial walks of G_Π respectively. The genus $g(G)$ is the minimum genus over all the embeddings of G.

In graph G_Π, a vertex v_a may appear in a facial walk f_j more than once. The maximum times v_a appears in a facial walk f_j is k, where k is the valence of v_a. In other words $\hat{v}_a^i \in f_j, (i = 0, 1, ..., k-1)$. An embedded graph G_Π is quasiplanar if no vertex appears more than once in any facial walk of G_Π.

Theorem 6.2.1 (Graph Embedding)**.** *A connected graph G can be embedded on a topological surface* Σ *without edge crossings.*

We will now describe a simple method to construct the embedding surface Σ for any connected graph G. In fact, the vertex set of Σ is exactly the same as the vertex set, V of G, and the edge set of Σ is the same as the edge set, E of G. Σ is well defined, if and only if we can construct its face set F. We use the rotation system of the graph to determine the embedding and the steps to construct the embedding surface are as follows:

1. For each edge e connecting vertices v_i, v_j, we associate two *halfedges* (oriented edges), one from v_i to v_j, denoted as $[v_i, v_j]$; the other from v_j to v_i denoted as $[v_j, v_i]$.

2. We define the *next* pointer for each halfedge. Suppose v_j is a vertex, its neighboring vertices are sorted clockwisely $\{v_{i_0}, v_{i_1}, ..., v_{i_{k-1}}\}$. Then

$$[v_{i_l}, v_j].next := [v_j, v_{i_{l+1}}].$$

3. For each halfedge h_0, we trace along its next pointer, to get a consecutive sequence of halfedges, $\{h_0, h_1, h_2, \cdots, h_m\}$, where $h_k.next = h_{k+1}$ and $h_0 = h_m$. Then this loop of halfedges form a face. By repeating this halfedge tracing procedure, we can exhaust all faces of the embedding surface.

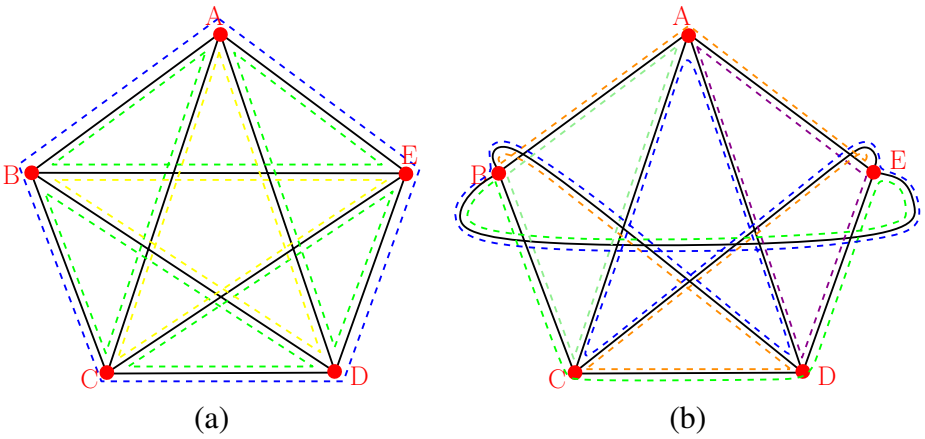


Figure 6.3: Two different embeddings using different rotation systems of K_5 graph. (a) Initial embedding; (b) Embedding with minimum genus of K_5 graph.

Hence, the rotation system Π is used to determine an embedding surface of G, denoted as G_Π. Note that once a rotation system Π is fixed, then it induces the face set uniquely.

The above method gives a local algorithm to compute one embedding surface of a connected graph G. However, the embedding may have high genus. The genus of a connected, orientable surface is an integer representing the maximum number of cuttings along non-intersecting closed simple curves without rendering the resultant manifold disconnected. In other words, it is equal to the number of handles on it. It is a NP-hard problem to find the embedding surface with the minimal genus [142].

Definition 6.2.2 (Graph Genus)**.** *The genus of a graph G is the minimal genus of its embedding surfaces,*

$$genus(G) := \min_{\Pi} genus(G_\Pi).$$

In this current work, we introduce some heuristic ideas to decrease the genus of the embedding surface. Figure 6.3 shows two different embeddings of a K_5 graph. Figure 6.3(a) shows an initial embedding of K_5 while Figure 6.3(b) shows the embedding surface of the same K_5 graph with minimum genus. The minimum genus is equal to 1.

6.2.1 Riemann Surface

Intuitively, a Riemann surface R is a topological surface with an atlas, such that all transitions are complex analytic functions. Equivalently, a closed Riemann surface can be treated as a surface with a special Riemannian metric $\mathbf{g}$, which produces

constant Gaussian curvature everywhere. If the genus of surface equals to 0, 1 or other, the corresponding constant curvature is equal to $+1$, 0, -1 respectively. The existence of such kind of Riemanian metric is guaranteed by the uniformization theory [133].

6.2.2 Topological Surface

If the uniformization metric $\mathbf{g}$ of a Riemann surface R can be assigned to the universal covering space $\tilde{R}$ of R, then $\tilde{R}$ can be isometrically embedded onto any of the following three domains: the sphere $\mathbb{S}^2$, the plane $\mathbb{R}^2$ or the hyperbolic plane $\mathbb{H}^2$. For example, if R is of genus one, then its deck transformations $Deck(\tilde{R})$ are translations onto a plane and if R is of high genus, its deck transformations are hyperbolic rigid motions. The deck transformation group is represented by Γ. Then the original Riemann surface can be represented as the quotient space: $\mathbb{R}^2/\Gamma$ for torus, $\mathbb{H}^2/\Gamma$ for high genus case.

Let q be one point on R, such that its pre-images $\tilde{q}_k$'s form an orbit. A simply connected domain $\Omega(R)$ in $\tilde{R}$ is called a *fundamental domain*, if it intersects each orbit in one point. Furthermore, assume that q is the base point, and a set of canonical fundamental group generators $\{a_1, b_1, \cdots, a_g, b_g\}$ are geodesics. A fundamental domain bounded by the preimages of the generators is called a *canonical fundamental domain*, and its boundary is a $4g$-side geodesic polygon given by,

$$\partial\Omega(R) = a_1 b_1 a_1^{-1} b_1^{-1} a_2 b_2 a_2^{-1} b_2^{-1} \cdots a_g b_g a_g^{-1} b_g^{-1}$$

which is called a *canonical fundamental polygon*.

There are two models for the hyperbolic plane $\mathbb{H}^2$. The first model is the *Poincaré disk* and the second model is the *Klein model*. Both of them are unit disks on the complex plane. For the Poincaré model, the metric is given by

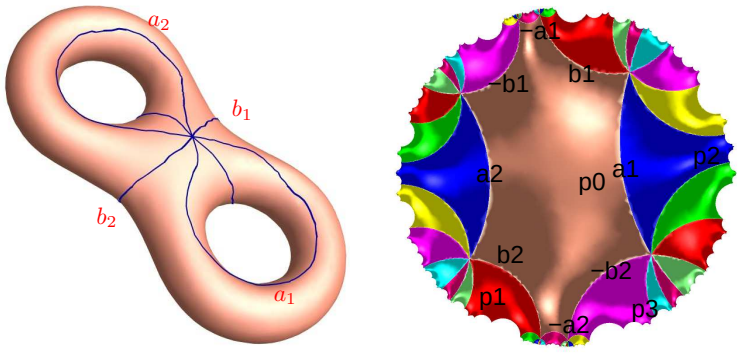
$$ds^2 = \frac{dz d\bar{z}}{(1 - z\bar{z})^2}.$$

The rigid motions are Möbius transformations given by,

$$z \to \frac{z - z_0}{1 - \bar{z}_0 z}.$$

The geodesics are circular arcs, which are orthogonal to the unit circle. On the other hand, in case of the Klein's model, the rigid motions are real projective transformations, and the the geodesics are straight lines.

Figure 6.4 illustrates all the concepts. The Figure 6.4(a) shows a genus two surface with a set of canonical fundamental group generators. The Figure 6.4(b) shows



(a) (b)

Figure 6.4: Embedding a finite portion of the universal covering space of a genus two graph onto the hyperbolic plane using (a) canonical fundamental group generators to obtain (b) canonical fundamental polygon.

a finite portion of its universal covering space that is embedded onto the Poincaré disk. Fundamental domains are color encoded, whose boundaries are canonical fundamental polygons.

6.2.3 Surface Ricci Flow

Let S be a topological surface with a Riemannian metric $\mathbf{g}$, then the isothermal coordinates (u,v) locally satisfy:

$$\mathbf{g} = e^{2\lambda(u,v)}(du^2 + dv^2).$$

The Gaussian curvature of the surface is given by:

$$K(u,v) = \Delta_{\mathbf{g}}\lambda, \tag{6.1}$$

where $\Delta_{\mathbf{g}} = e^{-2\lambda(u,v)}(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2})$ is the Laplace-Beltrami operator induced by $\mathbf{g}$. Although the Gaussian curvature is intrinsic to the Riemannian metric, Gauss-Bonnet theorem [133] claims that the total Gaussian curvature is a topological invariant:

$$\int_S KdA = 2\pi\chi(S),$$

where $\chi(S)$ is the Euler number of the surface.

Let $\mathbf{g}_1$ and $\mathbf{g}_2$ be two Riemannian metrics on the topological surface S. If there is a differential function $\lambda : S \to \mathbb{R}$, such that

$$\mathbf{g}_2 = e^{2\lambda}\mathbf{g}_1,$$

then the two metrics are *conformal equivalent*. The Gaussian curvature determined by $\mathbf{g}_1$ and $\mathbf{g}_2$ are denoted by K_1 and K_2 respectively and they satisfy the following *Yamabe equation*

$$K_2 = \frac{1}{e^{2\lambda}}(K_1 - \Delta_{\mathbf{g}_1}\lambda).$$

Let $\mathbf{g} = (g_{ij})$. Then the surface Ricci flow is defined as follows:

$$\frac{dg_{ij}}{dt} = -Kg_{ij}.$$

During the flow, the Gaussian curvature will evolve according to a heat diffusion process. If the total area is preserved, the surface Ricci flow will converge to a special metric whose Gaussian curvature is constant everywhere. [30]. This provides an alternate argument to prove the Poincaré uniformization theorem: all closed surfaces can be conformally deformed to one of the three canonical spaces, the unit sphere $\mathbb{S}^2$, the plane $\mathbb{E}^2$ or the hyperbolic space $\mathbb{H}^2$.

6.2.4 Discrete Surface Ricci Flow

Surface Ricci flow is a powerful tool to construct conformal Riemannian metrics by prescribed Gaussian curvatures. Discrete surface Ricci flow generalizes the curvature flow method from smooth surface to discrete triangular meshes.

Suppose Σ is a triangular mesh, we say the mesh exhibits a Euclidean (or hyperbolic) background geometry, if each face is a Euclidean (or hyperbolic) triangle. A discrete Riemannian metric on a mesh Σ is a piecewise constant metric with cone singularities at the vertices. The edge lengths are sufficient to define a discrete Riemannian metric, as long as the edge lengths satisfy the triangle inequality for every face.

The angles of each triangle are determined by the edge lengths. According to different background geometries, there are different cosine laws. For simplicity, in each face $[v_i, v_j, v_k]$, we use e_i to denote the edge against the vertex v_i, and l_i to denote the edge length of e_i. The cosine laws are given as:

$$\begin{array}{lcll} l_k^2 & = & l_i^2 + l_j^2 - 2l_i l_j \cos\theta_k & \mathbb{E}^2 \\ \cosh l_k & = & \cosh l_i \cosh l_j - \sinh l_i \sinh l_j \cos\theta_k & \mathbb{H}^2 \end{array}$$

The discrete Gaussian curvature K_i on a vertex $v_i \in \Sigma$ can be computed as the angle deficit as follows:

$$K_i = \begin{cases} 2\pi - \sum_{[v_i,v_j,v_k]\in\Sigma} \theta_i^{jk}, & v_i \notin \partial\Sigma \\ \pi - \sum_{[v_i,v_j,v_k]\in\Sigma} \theta_i^{jk}, & v_i \in \partial\Sigma \end{cases} \tag{6.2}$$

where θ_i^{jk} represents the corner angle attached to vertex v_i in the face $[v_i, v_j, v_k]$, and $\partial\Sigma$ represents the boundary of the mesh.

The Gauss-Bonnet theorem still holds on meshes as follows:

$$\sum_{v_i \in V} K_i + \lambda \sum_{f_i \in F} A_i = 2\pi\chi(M), \tag{6.3}$$

where A_i denotes the area of face f_i, and λ represents the constant curvature for the background geometry with values of 0 for the Euclidean geometry, and -1 for the hyperbolic geometry.

Generally, we associate each vertex v_i with a circle (v_i, γ_i) centered at v_i with radius γ_i. On an edge $[v_i, v_j]$, two circles intersect at an angle Θ_{ij}. During the conformal deformation, the radii of circles can be modified, but the intersection angles are preserved.

Let $u : V \to \mathbb{R}$ be the *discrete conformal factor*, which measures the local area distortion. If the vertex circles are with finite radii, then u_i can be formulated as:

$$u_i = \begin{cases} \log \gamma_i & \mathbb{E}^2 \\ \log \tanh \frac{\gamma_i}{2} & \mathbb{H}^2 \end{cases} \tag{6.4}$$

In the Euclidean case, the edge length is given by:

$$l_{ij} = \sqrt{\gamma_i^2 + \gamma_j^2 + 2\gamma_i\gamma_j \cos\Theta_{ij}}$$

In hyperbolic geometry, the edge length is given by [31] and [80]

$$l_{ij} = \cosh^{-1}(\cosh\gamma_i \cosh\gamma_j + \sinh\gamma_i \sinh\gamma_j \cos\Theta_{ij})$$

In all configurations, the discrete Ricci flow is defined as follows:

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i), \tag{6.5}$$

where $\bar{K}_i$ is the user defined target curvature, K_i is the curvature induced by the current metric. The discrete Ricci flow has exactly the same form as the smooth Ricci flow, which conformally deforms the discrete metric according to the Gaussian curvature.

The discrete Ricci flow can also be formulated in the variational setting as a negative gradient flow of a special energy form. This energy is called *entropy energy* and is given by:

$$f(\mathbf{u}) = \int_{\mathbf{u_0}}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i, \tag{6.6}$$

where $\mathbf{u_0}$ is an arbitrary initial metric.

Computing the desired metric with user-defined curvature $\{\bar{K}_i\}$ is equivalent to minimizing the discrete entropy energy. The existence, uniqueness of the solution, and the exponential convergence rate of discrete Ricci flow have been proved in [31].

6.3 Algorithm

Based on the theory, this section explains the algorithmic pipeline of our Ricci flow embedding of a graph onto a surface. We call the embedding surface as the *background surface*. The topology of the background surface is determined by the graph. There are three major stages, namely:

1. *Embedding onto a topological surface*
   (a) Given an abstract graph G, we get initial G_{Π^0} ($\Pi^0 = \{\pi_1^0, \pi_2^0, ..., \pi_n^0\}$) by randomly assigning a cyclic permutation of the edges incident with v_i.
   (b) Using our genus reduction algorithm, we can reduce the genus of G_{Π^0} monotone and get the relative small genus G_{Π^k}.
   (c) On the topological surface G_{Π^k}, the dual graph $\tilde{G_{\Pi^k}}$ of G_{Π^k} is computed. Both G_{Π^k} and $\tilde{G_{\Pi^k}}$ form a quadrilateral tessellation of the topological surface Σ.
2. *Embedding onto a Riemann surface*
   (a) By using Ricci flow on Σ, a constant curvature Riemannian metric is obtained.
   (b) Depends on the topology of the Σ, the surface is embedded onto one of the three canonical spaces using the constant curvature metric, namely the sphere $\mathbb{S}^2$ (genus 0), the plane $\mathbb{R}^2$ (genus 1), or the hyperbolic space $\mathbb{H}^2$ (high genus). Finally, the canonical fundamental polygon is computed.
3. *Embedding onto a surface in* $\mathbb{R}^3$
   (a) The fundamental polygon obtained is used to embed the graph onto the corresponding surface in $\mathbb{R}^3$, namely genus zero graph to a unit sphere, genus one graph to a torus and high genus graphs to various surfaces based on their genus number.

Figure 6.5 illustrates the different steps of the pipeline. We will now elaborate each of these stages in more detail.

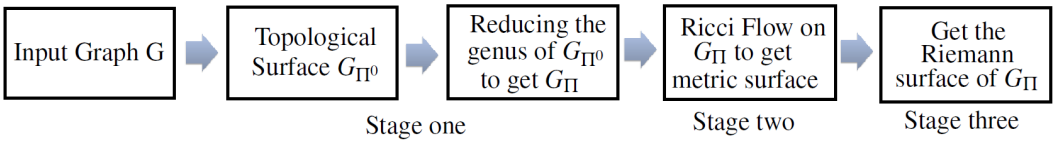


Figure 6.5: Three major stages of the algorithm pipeline for graph embedding: stage one - the embedding of a graph G onto a topological surface; stage two - the embedding of the graph onto a Riemann surface in $\mathbb{R}^2$; stage three - the embedding of the graph onto a surface in $\mathbb{R}^3$.

6.3.1 Embedding on Topological Surface

In this stage, we compute the topological embedding surface of the input graph G. There are two steps in the algorithm:

1. Initial Embedding Surface: We randomly choose a rotation system Π. We then trace the edges to form the faces and obtain the initial embedding surface G_Π.

2. Optimizing the genus : We apply a heuristic method to reduce the genus of the embedding surface.

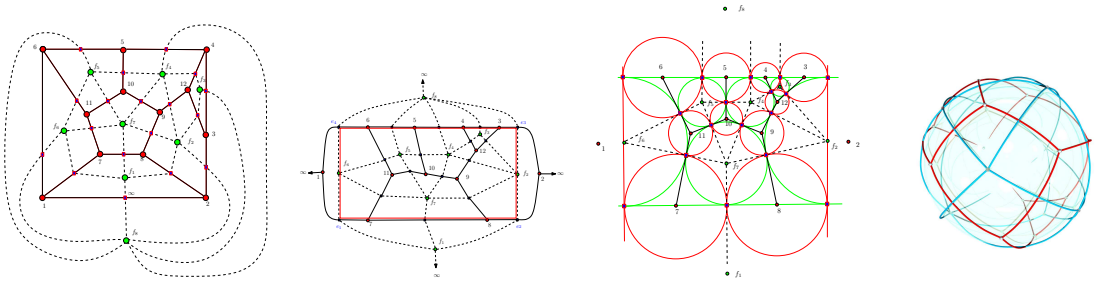
The key to minimizing the genus of the embedding surface is to maximize the number of faces. For this, we first find a special subset of vertices Ω, such that $v_i \in \Omega$ if and only if that there exists a face f of G_Π, such that v_i appears in f for multiple times. Then we apply a genetic algorithm to change the cyclic order of all vertices in Ω. In practice, by applying the genetic algorithm, the genus can be reduced to be one third of the initial value.

The graph G_Π defines a tessellation of the topological surface Σ. Each simply connected component in the complement of G_Π in Σ is called a face. Using these faces, the dual graph $\tilde{G_\Pi}$ can be constructed directly as follows: Each face of G_Π corresponds to a vertex of $\tilde{G_\Pi}$ and each vertex of G_Π corresponds to a face of $\tilde{G_\Pi}$. Each edge of G_Π separates two faces, that is, there exists an edge connecting the two face vertices in $\tilde{G_\Pi}$.

We call the union $D := G_\Pi \cup \tilde{G_\Pi}$ as the *overlapping graph*. As shown in Figure 6.6, on the overlapping graph D, there are three types of nodes:

1. Vertex node v_i corresponding to a vertex in G_Π, is represented as a red round dot.

2. Face node f_j corresponding to a face in G_Π, is represented as a green round dot.



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

Figure 6.6: Different steps illustrating the embedding a planar graph (genus 0) onto the sphere. (a) Input graph and its dual; (b) The reduced graph; (c) Ricci flow result; (d) Spherical embedding.

3. Edge node e_{ij}^{kl} corresponding to an intersection of an edge $[v_i, v_j]$ in G_Π, and an edge $[f_k, f_l]$ in the dual graph $\tilde{G_\Pi}$, where $[v_i, v_j]$ is the common edge of the faces f_k and f_l.

Each facet on the overlapped graph D is a topological quadrilateral, with two edge nodes, one vertex node v_i and one face node f_j, we denote the quadrilateral as $\square(v_i, f_j)$. By adding the diagonal connecting v_i and f_j, each quadrilateral is divided to 2 triangles, thus obtaining a triangulated graph. Let us denote this triangulated graph by R. All the following steps are carried out on this triangulated graph.

Figure 6.6(a) shows an example of the overlapping graph. The k-th vertex is labeled as k, and the j-th face is denoted as f_j. Note that, face f_8 represents the infinite face.

6.3.2 Embedding onto Riemann Surface

In this stage, Ricci flow is run on the triangulated graph, R, that is obtained from the previous stage. Each triangle has one vertex node v_i, one face node f_j and one edge node e_k, $[v_i, f_j, e_k]$, as shown in Figure 6.7.

Ricci Flow

Circle Packing Metric For each vertex node v_i, we associate a circle $C(v_i, \gamma_i)$, centered at v_i with radius γ_i. For each face node f_j, we associate a circle $C(f_j, \gamma_j)$, centered at f_j with radius γ_j. For each edge node e_k, we associate a circle with zero radius all the time, denoted by $C(e_k, 0)$. The circles associated with the vertex node and the face node intersect each other at a right angle. The intersection angle between the circles associated with the edge node and all other circles is zero.

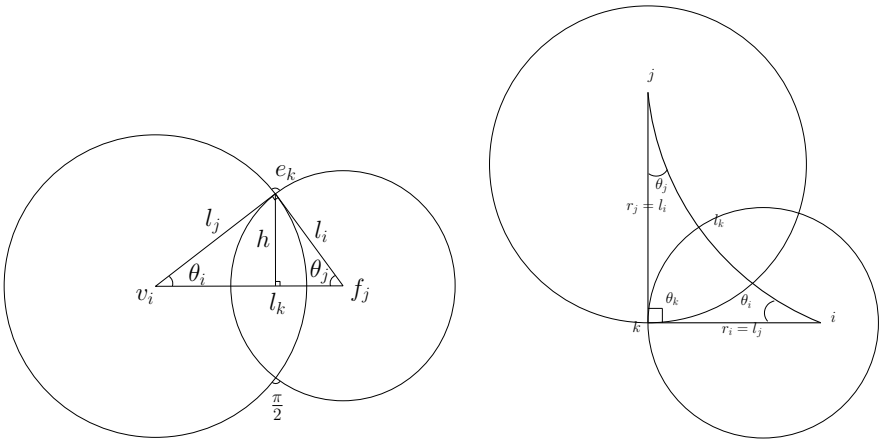


Figure 6.7: Circle packing metric for Euclidean and Hyperbolic Ricci flow.

Hessian Matrix The Ricci flow is computed by optimizing the corresponding Ricci energy. For this optimization, Newton's method is employed. In order to use Newton's method to optimize the Ricci energy, we need to compute its Hessian matrix.

For Euclidean Ricci flow, the elements of the Hessian matrix are given by:

$$\frac{\partial \theta_i}{\partial u_j} = \frac{h}{l_k}, \frac{\partial \theta_i}{\partial u_i} = -\frac{h}{l_k},$$

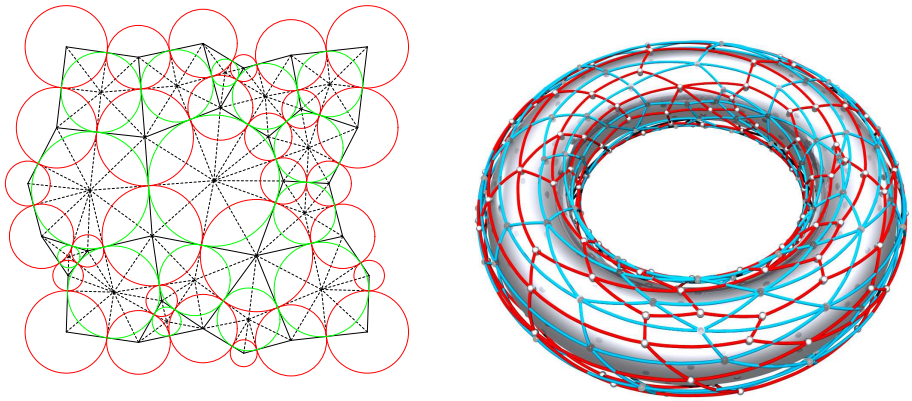
For hyperbolic Ricci flow, the elements of the Hessian matrix are given by:

$$\frac{\partial \theta_i}{\partial u_j} = -\frac{\sinh r_i \sinh r_j}{\cosh^2 r_i \cosh^2 r_j - 1} \tag{6.7}$$

$$\frac{\partial \theta_i}{\partial u_i} = \frac{\cosh r_i \cosh r_j \sinh r_i \sinh r_j}{\cosh^2 r_i \cosh^2 r_j - 1}. \tag{6.8}$$

Boundary Condition If the triangulated graph, R is of genus one, then we use Euclidean Ricci flow, such that the target curve is zero for all nodes. Similarly, if the triangulated graph, R is with high genus, then we use hyperbolic Ricci flow with zero target curvature everywhere.

If the input graph is a planar graph, then the triangulated graph, R is of genus zero. In this case, some preprocessing is needed as follows. Figure 6.6 shows the results of different steps for embedding a genus zero (planar) graph onto a spherical surface ($\mathbb{R}^3$). We select one edge node to be mapped to the infinity point. We call it the *infinity edge node* and denote it as e_∞. The choice of the edge node could be arbitrary. Figure 6.6(b) shows an example when the edge node $e_\infty = [v_1, v_2] \cap [f_1, f_8]$



(a) (b)

Figure 6.8: Embedding a genus one graph onto a torus. (a) Shows the flattening of the genus one graph onto $\mathbb{R}^2$; (b) Shows the final embedding of the genus one graph onto a torus (surface in $\mathbb{R}^3$)

is selected as the infinity edge node. Suppose the infinity edge node is given by $e_\infty = [v_i, v_j] \cap [f_k, f_l]$, then we remove all the quadrilateral facets adjacent to v_i, v_j or f_k, f_l in the overlapping graph D, to get the reduced graph $\bar{G}$. Now, there are four quadrilaterals adjacent to e_∞, each quadrilateral having a unique edge node other than e_∞. Let these edge nodes be denoted as $\{e_1, e_2, e_3, e_4\}$. The target curvature is set to be $\frac{\pi}{2}$ for these four nodes, and zero everywhere else, including the boundary nodes.

Embedding in Constant Curvature Space Once the boundary conditions of the triangulated graph R are decided for different genus cases (0, 1 and greater than 1), we map R to a plane $\mathbb{R}^2$ or a hyperbolic plane $\mathbf{H}^2$, consequently obtaining the canonical fundamental polygon.

Genus Zero For genus zero graph, the graph R is flattened onto a plane. We choose a root face randomly, and embed it onto the plane isometrically. The neighbors of this face are then enqueued into a queue. When the queue is not empty, we pop the first face, say $[v_i, v_j, v_k]$. Two out of the three vertices of this face must have been embedded already. Let these two vertices be v_i and v_j. Then v_k is the intersection of two circles (v_i, l_{ik}) and (v_j, l_{jk}) and the normal of v_i, v_j, v_k is consistent with that of the plane. After embedding this head face, we check all the faces sharing an edge with it. If they have not yet been embedded, then they are enqueued into the

queue. By repeating this procedure, we flatten face by face, and eventually flatten the whole mesh onto the plane. The planar image, thus obtained is a rectangle.

Genus One If the graph R is of genus one, then the corresponding cut graph Γ is used to find the embedding. We compute the cut graph Γ of R as follows. Essentially, if two faces share a common edge, then one can go from one face to the other. We first choose a root face. Starting from this face, we then perform a breadth first search to transverse all the faces of the triangulated graph R. Finally, all those edges which have not been covered during the search form the cut graph. In the next step, we prune the cut graph to remove all the dangling edges. The vertices in graph R whose valence in Γ is greater than 2 are defined as the nodes of the cut graph Γ. These nodes separate the cut graph into segments. Let these segments be denoted by $\{s_1, s_2, \cdots, s_n\}$.

We then slice open the graph R along the cut graph to obtain a simply connected graph $\bar{R}$. $\bar{R}$ is then isometrically flattened onto a plane in the way similar to genus zero case. The resulting planar image of $\bar{R}$ is still denoted as $\bar{R}$.

Each segment $s_k \in \Gamma$ has two images on the boundary of $\bar{R}$, denoted as s_k^+ and s_k^-. There exists a rigid motion η_k which maps s_k^- to s_k^+. All such rigid motions combinedly form a group called the Deck transformation group. By acting the group dynamics on $\bar{R}$, we can cover the whole plane.

We apply all η_k's and η_k^{-1}'s to cover a finite portion of the universal covering space of R, denoted by $\tilde{R}$. We then choose a base vertex v_0 on R arbitrarily and compute a set of canonical fundamental group generators using the method by Lazarus et al. [97]. These generators are lifted to the covering space to obtain the fundamental domains, whose corner points are preimages of the base vertex. Finally, we use straight lines to connect the corners to get the canonical fundamental polygon, which is a parallelogram.

High Genus If the graph R is of high genus (greater than 1), the process of embedding a finite portion of graph R onto the hyperbolic space and subsequently computing a canonical fundamental polygon is very similar to the genus one case.

We compute a cut graph Γ of R and slice R along the cut graph to obtain a simply connected mesh $\bar{R}$. We then isometrically map $\bar{R}$ to the hyperbolic space $\mathbb{H}^2$. Later, the generators of the deck transformation group are computed by using the segments of Γ. By acting the deck transformations on $\bar{R}$ a finite portion of the universal covering space of R is computed. We then choose a base vertex v_0 on R arbitrarily and compute a set of canonical fundamental group generators [97]. These generators are then lifted to the covering space to obtain the fundamental domains. Finally, we use the hyperbolic geodesics to straighten the boundary of the

fundamental domain, to get canonical fundamental polygon.

All the computations in this case are carried out using hyperbolic geometry. By using the Poincaré disk model for hyperbolic plane, all hyperbolic ruler and compass constructions can be converted to Euclidean ones. For detailed explanations see [31, 80].

6.3.3 Embedding on a surface in $\mathbb{R}^3$

The canonical fundamental polygons obtained from the previous stage belong to a genus zero, genus one or higher genus category. For a genus zero graph, the background surface is the unit sphere. For a genus one graph, the background surface is a torus $T : [0,2\pi] \to [0,2\pi] \to \mathbb{R}^3$,

$$T(u,v) = (r\cos(u)\cos(v) + R\cos(u), r\sin(u)\cos(v) + R\sin(u), r\sin(v))$$

where r, R are the two shape parameters. We represent the torus using a triangular mesh.

Finally, if for a high genus graph, say a genus g surface ($g > 1$), we compute the union of g tori to construct the background surface S. Hence, in this case the embedding is similar to the embedding in genus one case onto a torus but with a union of g tori. We will now discuss embedding of genus zero and genus one graphs onto the respective background surfaces, namely the sphere and torus. The background surface for any high genus surface (genus greater than 1) differs based on the genus number. The embedding itself is similar to the case of the genus one graphs.

Genus Zero

Using the result from the previous step, for a genus zero graph G, the reduced triangulated graph R is flattened onto a place which is a planar rectangle. We use a stereographic projection to map the whole plane to the unit sphere as follows: $\mathbb{S}^2$, $\tau : (u,v) \to (x,y,z)$

$$\tau(u,v) = \left(\frac{2u}{1+u^2+v^2}, \frac{2v}{1+u^2+v^2}, \frac{-1+u^2+v^2}{1+u^2+v^2} \right).$$

where (u,v) is the coordinate system in the planar rectangle domain while (x,y,z) is the coordinate system in the spherical domain. Stereographic projection maps planar circles and lines to spherical circles. In Figure 6.6, let $e_\infty = [v_i, v_j] \cap [f_k, f_l]$, then the vertical lines of the rectangle are mapped to the vertex node circles of v_i and v_j and the horizontal lines correspond to the face node circles of f_k and f_l. Thus, the graph is embedded on a unit sphere as shown in Figure 6.6(d).

Genus One

In case of genus one graph, the embedding onto a background surface S is obtained in a different manner. Suppose we were to construct a background surface S. We then apply Ricci flow to this background surface to find its flat metric and flatten a portion of its universal covering space onto a plane. Thus, we obtain a fundamental polygon which is a parallelogram. Now we have a fundamental polygon of the background surface S and fundamental polygons for the reduced graph R of a genus one graph G. Both the fundamental polygons of the graph R and the background surface S are parallelograms, denoted as $\Omega(R)$ and $\Omega(S)$ individually. Then we can find a linear map $\tau : \Omega(R) \to \Omega(S)$, which matches the two parallelograms.

We denote the mapping from the surface to its fundamental domain as $\phi_R : R \to \Omega(R)$ and $\phi_S : S \to \Omega(S)$. Then, the mapping from graph R to the background surface S is given by: $\phi : R \to S$,

$$\phi = \phi_S^{-1} \circ \tau \circ \phi_R.$$

Thus, ϕ embeds the input graph G onto the surface S in $\mathbb{R}^3$.

6.4 Results

In this section, we report our experimental results, which shows the efficacy and efficiency of our method. The algorithms were implemented using $C++$ on Windows platform. All the experiments were conducted on a CPU1.2GHz, 3GB RAM laptop.

Genus Zero Examples Figure 6.9 shows the embedding result for a genus zero graph. The graph has $3k$ nodes and $6k$ edges. Figure 6.9(a) shows the embedding result onto a complex sphere using Ricci flow. Figure 6.9(b) shows the spherical embedding result of the graph while Figure 6.9(c) shows the spherical embedding of its corresponding overlapped graph. Figure 6.9(d) shows the spherical embedding result with circle packing. The time cost taken for all the computations is 28 seconds. Out of this, the majority of the time was taken in the computation of Ricci flow.

Genus One Examples Figure 6.10 shows the embedding results of a genus one graph. The graph in 6.10 has $10K$ nodes and $18k$ edges. The Figure 6.10(a) shows the embedding of the fundamental domains onto the plane. Figure 6.10(b) shows the embedding of the graphs onto the torus. The total running time is around one minute.

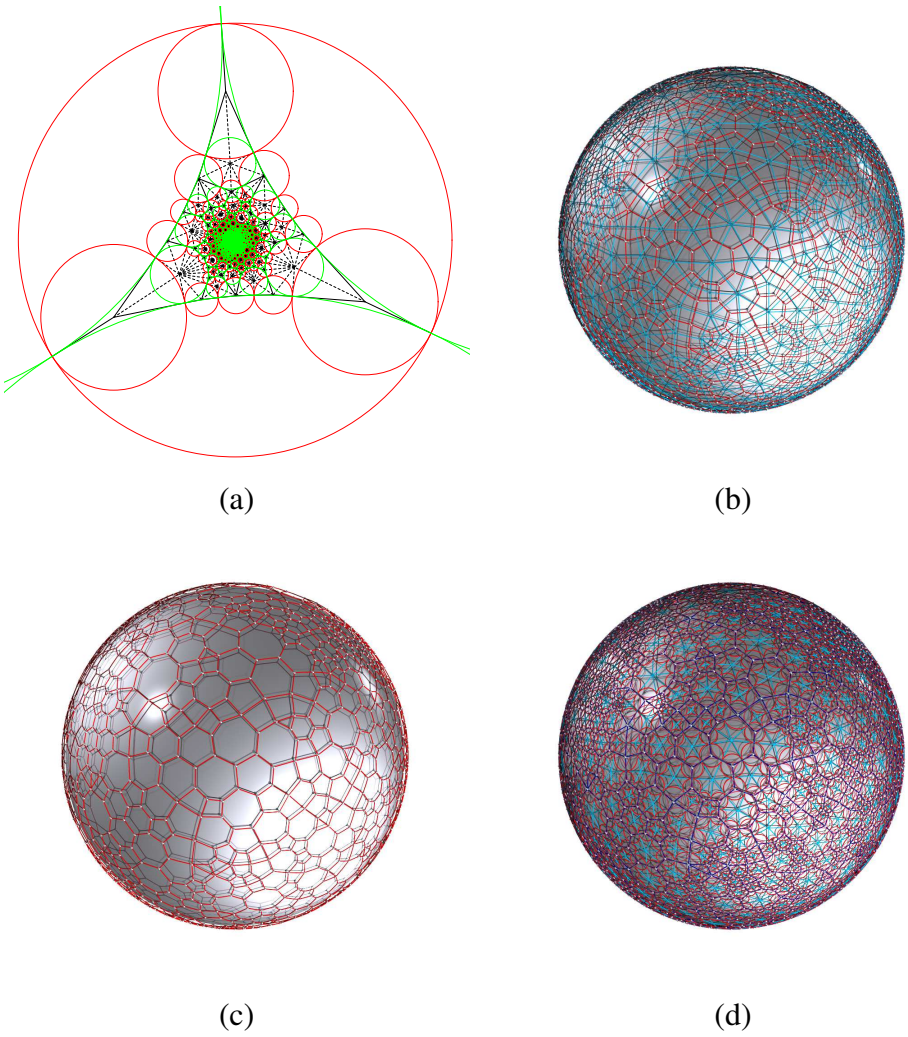
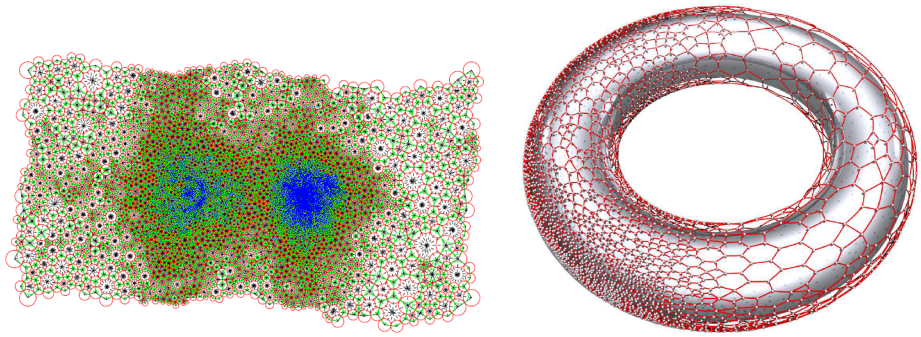


Figure 6.9: Embedding of a genus zero graph showing (a) embedding onto a complex sphere using Ricci flow, (b) spherical embedding of the graph itself, (c) spherical embedding of its overlapped graph, (d) spherical embedding with circle packing.

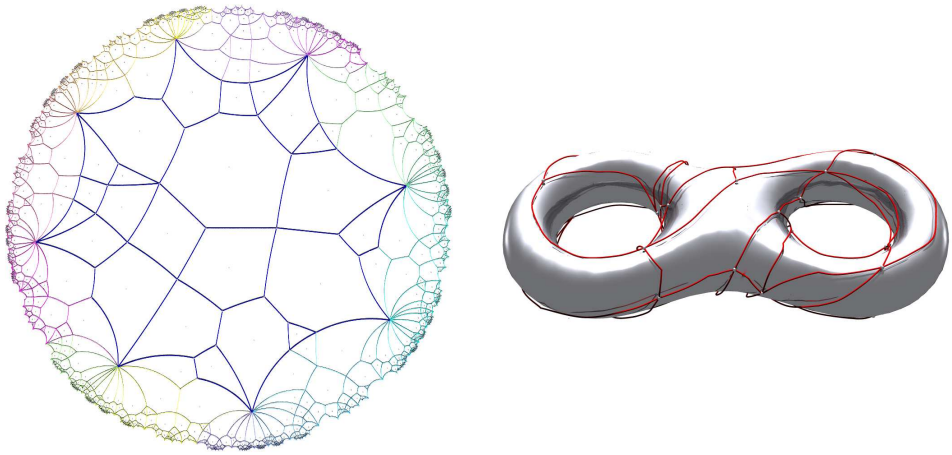
High genus Figure 6.11 shows the embedding results of a genus two graph. Figure 6.11(a) shows the embedding result onto a Riemann surface $\mathbb{H}^2/\Gamma$, where different fundamental domains are color-encoded differently. The corresponding Deck transformation group generators of Γ are shown in Table 6.1. Figure 6.11(b) shows the final embedding of the graph onto a surface in $\mathbb{R}^3$.

Similar results are shown for more high genus graphs in Figure 6.12. Figures 6.12(a) and (b) show the embedding result of two genus two graphs onto Riemann surfaces. The corresponding Deck transformation group generators for graphs in



(a) (b)

Figure 6.10: Embedding of a genus one graph showing (a) embedding onto a plane, (b) embedding in R^3.



(a) (b)

Figure 6.11: Genus 2 graph embedding on (a) Riemann surface $\mathbb{H}^2$ and (b) surface in R^3 (torus)

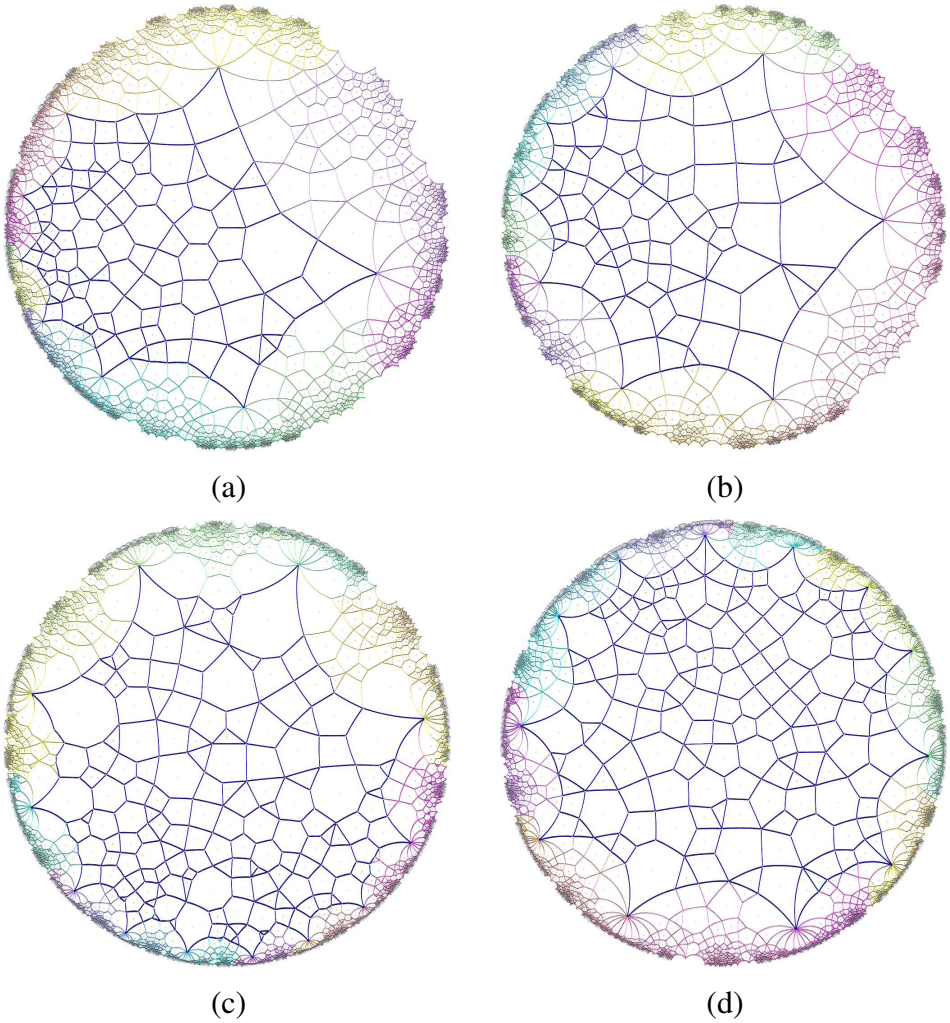


Figure 6.12: Embedding results of two genus 2 graphs (a) and (b) onto Riemann surfaces. Embedding results of two genus 3 graphs (c) and (d) onto Riemann surfaces.

Table 6.1: Deck transformation group generators of the graph in Figure 6.11

Group	real(z)	img(z)	θ
0	0.554278	0.784424	2.13226
1	-0.595666	0.638663	1.16896
2	0.959107	-0.0515265	-2.13223
3	0.820739	0.298469	-1.16883
4	0.763219	-0.536063	1.67342
5	0.92414	-0.274074	2.21251
6	-0.455089	-0.814097	-1.67339
7	0.333543	-0.904376	-2.21247

Table 6.2: Deck transformation group generators of the graph in Figure 6.12(a)

Group	real(z)	img(z)	θ
0	0.473141	0.793373	1.68923
1	-0.0444407	0.926901	1.77595
2	0.843724	-0.376077	-1.68923
3	0.89841	0.232338	-1.77595
4	-0.511646	-0.718428	1.47841
5	0.241374	-0.875041	1.4262
6	-0.668161	0.575745	-1.47841
7	-0.90069	-0.112767	-1.4262

Figures 6.12(a) and (b) are shown in Tables 6.2 and 6.3, respectively. Similarly, Figures 6.12(c) and (d) show the embedding result of two genus three graphs onto Riemann surfaces. Their corresponding Deck transformation group generators are shown in Tables 6.4 and 6.5, respectively.

Our graph embedding approach finds useful applications for visual graph comparison and dynamic graph visualization, particularly for genus zero graphs which are embedded onto a sphere. The spherical embedding provides a novel way to represent the nodes of the graph with circles on a sphere. In the next two sections, we will show how our approach efficiently helps in visual graph comparison and to easily identify regions of change in dynamically changing graphs. Note that all the graphs used in the later sections are genus zero graphs (planar graphs). To clearly illustrate the efficiency of our approach to visual graph comparison and dynamic

Table 6.3: Deck transformation group generators of the graph in Figure 6.12(b)

Group	real(z)	img(z)	θ
0	-0.423878	0.767216	1.16253
1	-0.867863	0.127443	1.39665
2	0.872451	0.0844032	-1.16252
3	0.275935	0.832636	-1.39664
4	0.140488	-0.936178	1.9649
5	0.636244	-0.688274	1.90233
6	-0.810489	-0.489155	-1.9649
7	-0.443586	-0.825681	-1.90229

Table 6.4: Deck transformation group generators of the graph in Figure 6.12(c)

Group	real(z)	img(z)	θ
0	-0.483681	-0.846774	2.31836
1	-0.228992	-0.959415	2.51752
2	-0.949812	-0.220979	-2.31836
3	-0.746462	-0.644755	-2.51752
4	0.574592	-0.803201	2.53001
5	0.837867	-0.511424	2.44972
6	0.00927105	-0.987523	-2.53001
7	0.31892	-0.928367	-2.44972
8	0.0897692	0.93249	1.59591
9	-0.705865	0.527996	1.34124
10	0.93445	-0.0663243	-1.59591
11	0.674762	0.567205	-1.34124

graph visualization, we only use planar graphs and not high-genus graphs.

6.5 Visual Graph Comparison

Visual graph comparison is a challenging task which allows the users to visually answer the question whether two given graphs are isomorphic. In several situations, researchers are confronted with the task of graph comparison to find out

Table 6.5: Deck transformation group generators of the graph in Figure 6.12(d)

Group	real(z)	img(z)	θ
0	0.684803	-0.664529	1.69622
1	0.911636	-0.204362	1.81809
2	-0.573648	-0.762549	-1.69622
3	0.0250182	-0.933926	-1.8181
4	0.277177	0.927289	2.24515
5	-0.13187	0.967995	2.29782
6	0.897412	0.362412	-2.24513
7	0.635588	0.741912	-2.29781
8	-0.954146	0.186617	2.30815
9	-0.919973	-0.315278	2.16587
10	-0.503332	0.831792	-2.30815
11	-0.776821	0.585064	-2.16585

if two graphs are indeed the same. Two graphs are said to be isomorphic or similar if there is a one-to-one structural correspondence between the nodes of the two graphs. Generally, two similar graphs differ in terms of topology, node positions, and ordering of the nodes, making it a non-trivial task to know if two given graphs are similar. Graph matching methods provide an answer to this question algorithmically but do not provide any visual confirmation. It is always desirable to present the matched graphs to the end users in a way that allows easy comparison. Our method is a graph invariant deterministic approach which implies that applying our method to the same graph any number of times will yield exactly the same graph layout, regardless of the node ordering. Moreover, two similar graphs when embedded onto a sphere using Ricci flow differ only by a Möbius transformation. Irrespective of the order of the nodes our method results in the same node positions (represented as circles on the sphere) for the same graphs, that is, two isomorphic graphs yield the exact same visual layout. By a mere glance, users can immediately deduce that the two graphs are exactly similar. This ability of our method also lets the users to visually identify the regions that change in the graph, thereby facilitating dynamic graph visualization (discussed in detail in the next section).

We illustrate the visual graph comparison ability of our method by using a simple graph shown in Figures 6.13(a) and (b). Figures 6.13(a) and (b) show two different layouts of the same graph. Both the graphs are isomorphic and structurally similar, that is, all the nodes have the similar connectivity, yet they differ topo-

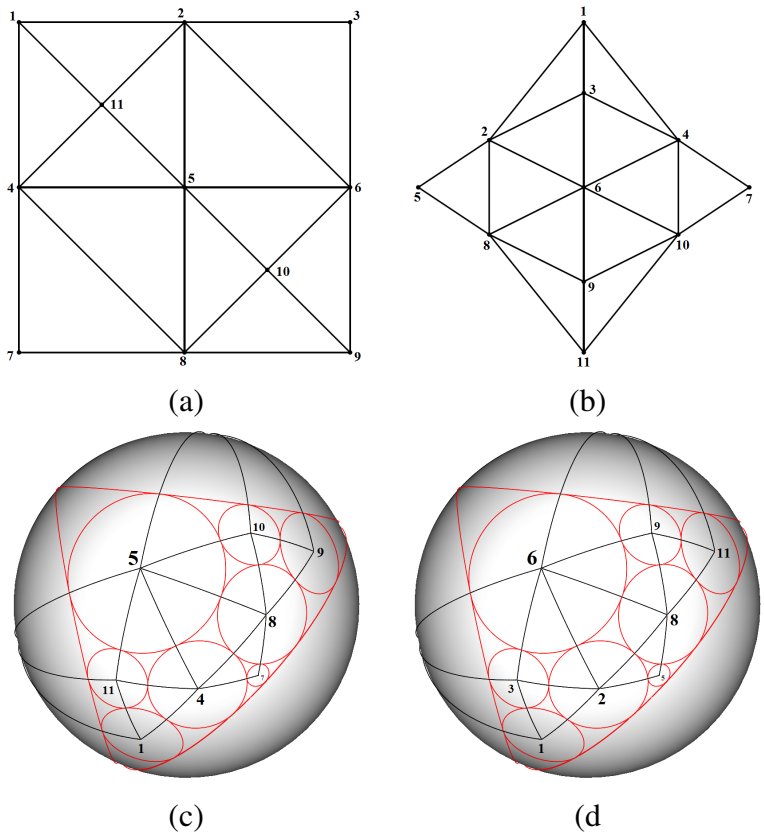


Figure 6.13: Illustration of visual graph comparison ability of our approach using two very simple isomorphic graphs (a) and (b). Ricci flow circle packing layout (c) and (d) for (a) and (b) respectively clearly showing the similarity between graphs.

logically. The similarity of the two graphs is not quite obvious by simple visual observation of the two graphs. In addition, the nodes have been ordered differently as shown by their labels. Figures 6.13(c) and (d) show the Ricci flow embedding for the graphs in Figures 6.13(a) and (b), respectively. It can clearly be seen that both the results look exactly similar visually, thus confirming that both the graphs are indeed the same. The edges (connectivity), the nodes and their corresponding labels have also been shown to provide further confirmation. One can notice that the two graph embeddings are similar not only in terms of structural connectivity of nodes but also have equal radius for circles representing the same nodes in both the graphs. Moreover, the user can rotate, zoom, apply Möbius transformation to the embedding on the sphere to visually compare and deduce the similarity between two graphs.

Figures 6.14(a) and (b) show two different layouts of a same graph representing

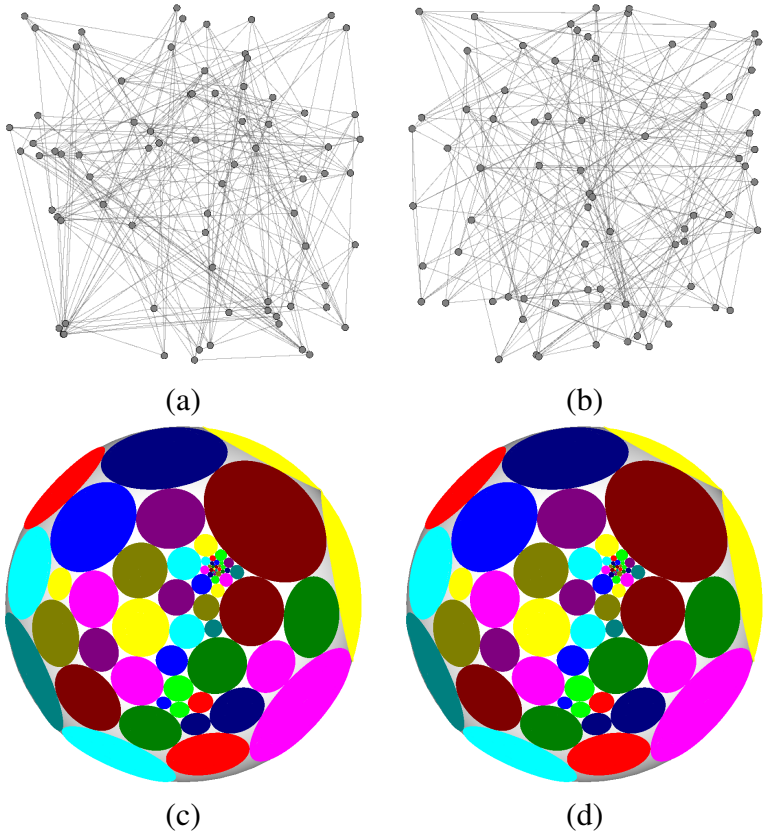


Figure 6.14: Ricci flow graph layout (c) and (d) of two isomorphic graphs (a) and (b) respectively. It is difficult to see the similarity between (a) and (b) but by using our approach the similarity is clearly captured by simple visual observation.

real-world data. It is a sub-graph of a graph representing email transactions. The node labels are also removed to increase the difficulty. This illustrates that finding a correspondence between the nodes is extremely difficult by simple observation, especially with larger graph size and no node labels. Figures 6.14(c) and (d) show the Ricci flow layouts for the graphs in Figures 6.14(a) and (b), respectively. The node positions are consistently placed on the sphere, thereby aiding a very good visual graph matching and comparison. Colors have also been assigned to the circles to further aid the visual graph comparison. Two corresponding nodes in the two graphs have the same color and same circular radius, thus providing a visual evidence of successful graph matching. The added color information for the nodes helps to establish correspondences between the nodes of the two graph even without any additional information such as the node labels or attributes.

6.6 Dynamic Graph Visualization

Dynamic graph visualization is a more recent topic of research and it addresses the problem of graphs changing over time where vertices and edges are added and removed over time. Our Ricci flow based embedding provides a consistent and stable graph layout for all the time steps which helps in efficiently capturing the evolution of dynamic graphs. The overall layout is not affected by the local structural changes and the mental map of the graph is preserved. As a result, the users can easily estimate the regions of the graph and how much of the graph changes by simple visual observation.

The two important criteria to be considered for visualizing dynamic graphs are readability and mental map preservation. Readability refers to graph embedding for individual time steps such that there are no edge crossings and no node overlaps. The mental map refers to the cognitive model of the graph that the user creates internally. Preservation of mental map for dynamic graphs is commonly defined as minimising the movement of nodes between time steps. It can be achieved by ensuring that nodes that appear in consecutive graphs in the sequence remain in more or less the same positions, so that they can easily be identified as the same nodes over time. Research has shown that it is a difficult task to get an acceptable consensus between these two criteria. If each of the individual time steps is embedded without regard to the changes, the position of the nodes change drastically and the mental map is disturbed. On the other hand if the position of nodes is fixed, individual graph embedding will involve lot of edge crossings. Several studies have also showed that it is important to have a consistent mental map over all the time steps of the dynamic graphs [9, 10]. Not having a consistent mental map can cause confusion and make it difficult to track and observe changes in dynamic graphs. Our Ricci flow based graph embedding method achieves a stable graph layout with no edge crossings and preserves the mental map over all time steps. As a result, the differences between the original and modified graphs are easily comprehended by the user.

The stability of our embedding approach helps to visualize the following in dynamic graphs: (1) the region of change in the graph between time steps; (2) mental map preservation; (3) track the progress of a node or a set of nodes over time. We use a simple graph to illustrate all the above shown in Figure 6.15(a). For the sake of illustration, we manually made minimal changes to the graph by inserting a node or two at each time step to obtain three modified versions of the graph as shown in Figures 6.15(b), (c) and (d), thus forming a sequence of graphs representing three progressive time steps. We use color to highlight the regions of change between two graphs from different time steps. Archambault et al.[30] and Zamen et al.[45] confirmed that changes between two graphs are best highlighted by

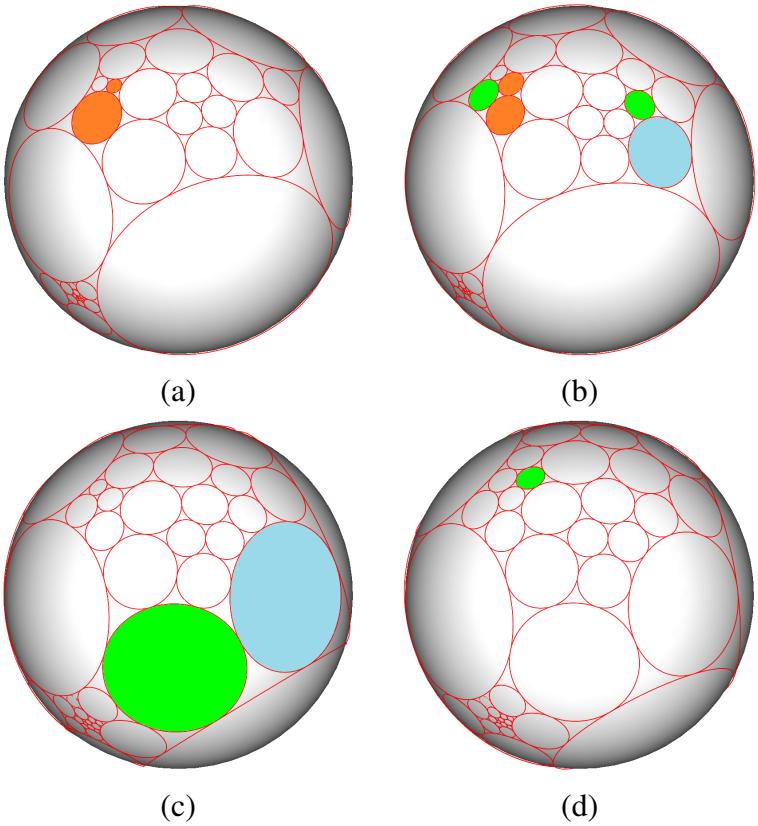


Figure 6.15: Dynamic graph visualization using manually generated sequence of graphs depicting four different time steps, (a), (b), (c) and (d). The regions of change are highlighted in green. In order to see the local adjustments made by our approach some of the nodes are highlighted in orange and light blue.

using color. All the regions of change between adjacent time steps are highlighted using green color. It should be noted that the green highlighting only indicates the regions of the graph that have been affected and do not indicate what exactly the changes are. The green highlighting serves more as bookmarks for the users to detect the structural changes in the graph. Since the graph layout remains stable at all time steps, highlighting only the regions of change easily draws the users' attention to regions of the graph that are changing. The green regions typically indicate how many regions have been affected by the changes, approximately which portion of the graph is being affected, and the nodes whose connectivity (structural information) has changed. Red or grey color is used to indicate the rest of the graph that is not affected by the structural changes between time steps.

The users can focus on the regions highlighted in green and by virtue of our

method can easily identify the changes by simple visual observation as shown by the graphs in Figures 6.15(b), (c) and (d). This simplicity of being able to observe the changes in dynamic graphs, is in itself a contribution on its own. We mimic the approach of small multiples to show all the graphs in the sequence side-by-side with all the regions of change indicated in green. When nodes are inserted or removed, our method makes local adjustments to maintain the overall layout. Figures 6.15(b), (c) and (d) show that the radius of some of the nodes change, some examples of which are highlighted by the circles in orange and light blue color. The key feature of our approach is the preservation of mental map of the graph and our algorithm ensures this by locally altering the radius of some of the nodes. However, there is no or minimal change in the position of the nodes which helps in tracking the nodes over all the time steps easily. Figure 6.15 also illustrates that overall graph layout remains intact regardless of the changes, thus preserving the mental map.

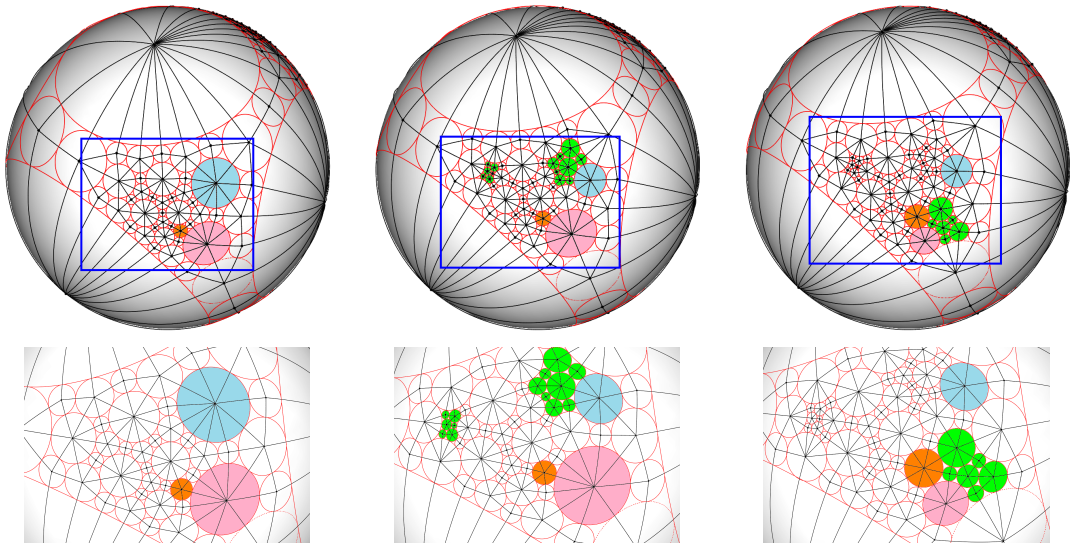


Figure 6.16: Ricci flow based embedding of a dynamic graph for three time steps in sequential order (a), (b) and (c). The changes in the graph between time steps are highlighted using green, as shown in (b) and (c). The zoomed part of the regions enclosed by the blue box is shown in the bottom row. Three nodes highlighted using light blue, pink and orange colors in (a), (b), (c) are used as query nodes and are tracked over all time steps. The mental map of the graph across time steps is preserved, despite the changes.

Figures 6.16(a), (b), and (c) illustrate the dynamic graph visualization of a real-world graph for three progressive time steps. The zoomed regions enclosed in blue boxes are shown so that the changes can be noticed clearly. All the regions of

change between adjacent time steps are highlighted in green color. Figure 6.16 also shows that the overall layout of the graph is not disturbed much due to the changes. Our method is not just adept in identifying the regions of change between adjacent time steps but it can identify the changes between any two time steps. Any two time steps can be chosen and the corresponding regions of change between them are highlighted.

Graphs consist of some nodes that are not affected over time and remain in the graph from beginning to end. Sometimes it is desirable to track such nodes. By virtue of our method, the layout of the graph is unaffected by the local structural changes and hence a particular node or a set of nodes can be tracked over time. The user can highlight a node or a specific set of nodes by selecting the corresponding circles. The user needs to highlight the nodes of his interest in one of the time steps and the same nodes are highlighted in all the time steps automatically. The position of these nodes, the change in their radii, and the change in their connectivity can be tracked over all the time steps. An illustration of this can be seen in Figure 6.16. Three nodes have been chosen for tracking by highlighting them using orange, pink and light blue colors as shown in Figures 6.16(a), (b) and (c). There is no change in the pink and orange nodes between Figures 6.16(a) and (b) but there is a noteworthy change for the light blue node. Both the radius and the connectivity of the light blue node have slightly varied due to the changes in the graph around it. For the same reason, there are changes in the orange and pink nodes between Figures 6.16(b) and (c).

6.7 Sensor Networks

In recent years, there has been a fantastic growth in the use of mobile devices with wireless sensor capabilities. As a result, participatory sensing applications have gained lot of interest. The participatory sensing has applications in monitoring health and wellness, increase public awareness on civic issues, capture road traffic and transportation data (which is vital for monitoring the traffic flow and development of projects such as google car) etc. In addition, interest has also been developed on studying and monitoring the change in the connectivity between the sensor networks caused due to various factors such as mobility, environmental factors and power issues. All the aforementioned applications deal with large range of wireless sensor networks. By using our layout and visualization approach we are able to explore, observe and study the mobility, link dynamics and connectivity changes for sensor networks over time.

The graph data is obtained by simulating a sensor network scenario under special constraints. Each node in the graph indicates a specific device. All these de-

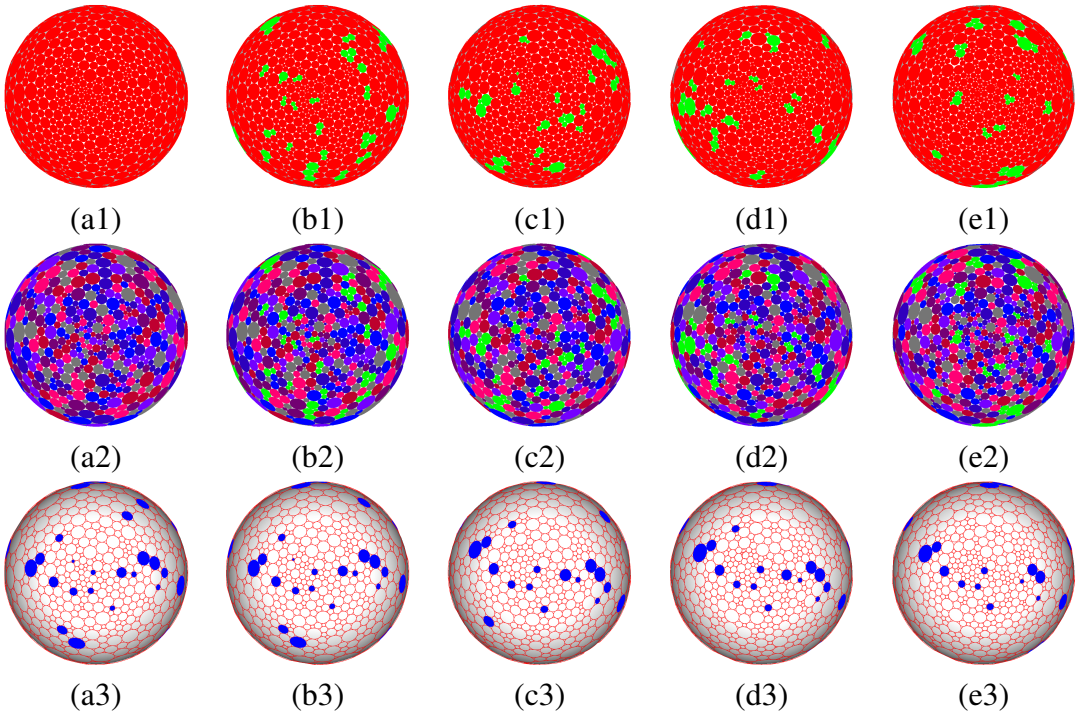


Figure 6.17: Hourly snapshots of a sensor network data for five hours. (a1)-(e1) show the regions of change between adjacent time steps highlighted in green. All the unaffected regions are shown in red. (a2)-(e2) show the regions of change (in green) between adjacent time steps by additionally assigning colors to other nodes. Colors help to visually notice the correspondence between the nodes across all the time steps. (a3)-(e3) track the progress of some query nodes, highlighted in blue, over all time steps.

vices keep moving and have variable power levels. We used a defined geographical boundary for our simulation and any device that goes out of this boundary is said to have left the network. All the devices which are in 1-hop connectivity range of other devices are connected by an edge. If any device leaves the network, changes its position or loses power, the connectivity of the neighboring devices is changed and is updated accordingly.

Our approach yields, what we term as a navigable snapshot of the sensor network. The graph layout helps the users to navigate and gain knowledge about the topology of the network and hence the term navigable snapshot. Figure 6.17 shows the hourly snapshots of the network for five hours using our method. Figures 6.17 (a1)-(e1) show the regions of change between adjacent time steps highlighted in green and the rest of the nodes in red. This provides information regarding which regions of the graph are being affected by the change and how much of the region

is affected. Figures 6.17(a2)-(e2) provide the same information about the region of changes but this time all the nodes are assigned color. Assigning color to the nodes provides a visual correspondence between the nodes across all time steps. Moreover, information about the changes in 1-hop connectivity of different nodes, particularly around the regions of change, can be easily visualized. In addition, it helps in tracking the progress of certain nodes of interest. Such tracking results are shown in Figures 6.17(a3)-(e3). We have chosen a set of nodes highlighted in blue as the set of query nodes in order to track the progress of these nodes. While some of the nodes are not affected across time steps, some of the nodes disappear after few time steps. Being able to visualize all the changes over all the time steps simultaneously helps to know the regions of the network where the changes occur more frequently, thereby differentiating more problematic regions of the network from stable regions of the network.

We also collected a short, informal user feedback of our embedding approach from 9 users (6 male and 3 female), out of which, 5 users were researchers in the field of wireless sensor networks. We let them use our method to visualize different kinds of data including the sensor network data. We asked them some general questions such as: How do you like the visualization system? Does it help in visualizing dynamic graph data? Does assigning colors to the nodes help to track them over time? etc. The overall feedback was that our method provided a more refreshing and natural way to visualize the data. Our approach was well liked by the users. 3 of the users particularly mentioned that they were impressed by the spherical interface for graph visualization. The researchers from sensor network liked the representation of nodes as circles. They mentioned that it now allows them to have additional information about the nodes such as the number of neighbors, frequency values, bar charts etc. on the graph itself without having to make separate tables.

Chapter 7

Conclusions and Future Work

7.1 Summary

A large amount of research has been done on manifold shape analysis using spectral methods and diffusion based methods, such as heat diffusion and Laplace-Beltrami operator. The heat diffusion method using the heat kernel have proven to be successful shape analysis technique for 3D mesh models with modest resolution. In contrast, there has been little research on shape analysis for volumetric data, and hence the research here was performed to provide computationally efficient shape-based volume analysis methods.

Presented herein were two methods for effective shape analysis using volumetric data. The first method was a cumulative approach which is a modified heat diffusion approach, called the cumulative heat diffusion. The diffusion of heat is carried out by considering all the voxels as sources and the accumulated heat values after each time step is recorded. The final heat values of the voxels encode the shape information and can be used to create a shape-based voxel classification. The second method was a stochastic approach using a new set of diffusion particles, called the shapetons. The shapetons are diffused in a Monte Carlo manner across the voxels. The shapeton propagation is guided by taking the local shape information into account (volume gradient operator) and hence the final accumulated shapeton count of the voxels encode the shape information. The shape analysis plays a crucial role in medical applications. The two shape analysis methods presented successfully demonstrate their application to medical analysis in the areas of segmentation, shape-based transfer function design and tumor (polyp) detection in colon.

Using a 3D mesh model of a colon presents problems during navigation, as occlusions can cause questions to be raised as to one's precise location. To mitigate this problem, a map with no occlusions would be useful, and a 2D map would be preferred so that the entire colon surface can be viewed at once. For this, a robust

conformal flattening method was presented which will create an efficient 2D flattening of the colon surface. This shape-preserving flattened colon provides an effective means of polyp visualization, detection and verification. Furthermore, the supine and prone colon registration pipeline is made robust by utilizing this flattening algorithm. The colon registration pipeline requires anatomical landmarks and feature points to ensure consistent registration. Herein were presented techniques to automatically identify prominent anatomical landmarks on the colon surface, namely the taeniae coli and the flexures. Also presented was a graph based approach to identify a set of feature points on the colon surface using the haustral folds of the colon.

Finally, a novel graph embedding approach was presented using Ricci flow that helps to efficiently compute the global shape characteristics and topology of the graphs. The approach is general, practical and theoretically rigorous. Furthermore, the consistency of the approach successfully facilitated graph comparison by simple visual observation and to easily track the local changes in dynamic graphs, particularly for genus zero (planar) graphs.

7.2 Future Work

7.2.1 Short Term Plan

Controlled Stochastic Shape Analysis

In the stochastic shape analysis approach, the result obtained at the stable state is the one with actual relevance than the results obtained at intermediate time steps. This approach is not efficient in analyzing the shape information at different scales (time steps). Figures 7.1(a) and (b) show the progressive results for the synthetic cube data, obtained with increasing number of times steps using the cumulative heat diffusion and the shapeton diffusion respectively. It can clearly be seen that meaningful shape information is obtained in the intermediate stages of the cumulative heat diffusion approach, whereas not much useful local shape information is obtained in the stochastic shapeton diffusion approach. While the cumulative heat diffusion captures the local shape information such as the corners and edges of the synthetic cube data, no such information is captured using the shapeton diffusion approach. Consequently, while different time steps are synonymous to different scales in the cumulative approach, there is no such relationship between the time step and the scale in the stochastic approach.

In the shapeton diffusion approach, the shapetons are randomly distributed initially. Though the diffusion of shapetons is guided by the volume gradient operator, the initial random distribution of the shapetons causes random results in the

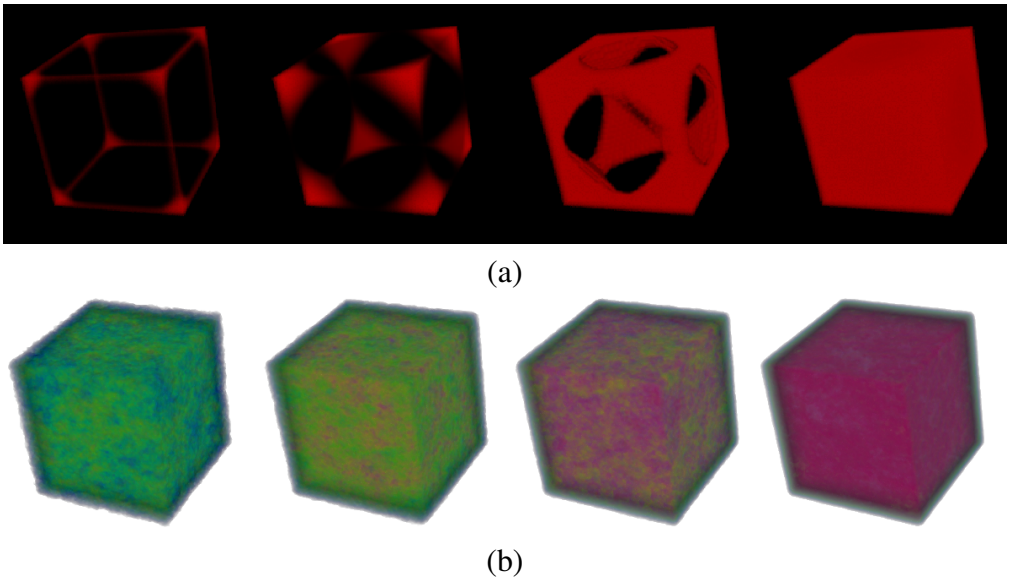


Figure 7.1: Comparison of progressive shape information obtained with increasing number of time steps using (a) cumulative heat diffusion and (b) shapeton diffusion approach.

intermediate stages of the diffusion process. Moreover, the shapetons are not diffused between the voxels and moved for a pre-defined distance per single time step. One has less control of the diffusion process with this new definition of the time step. However, the shape information obtained at the stable state after large number of time steps is similar in both the cumulative and stochastic approaches. Hence, as part of the future work, we plan to develop a method that allows a controlled stochastic shape analysis and establishes a relationship between the shape information at different scales and the time step.

Heat Visualization of Complex Volume Data

The cumulative heat diffusion captures the hierarchical shape information. However, when the level of complexity of the volume data increases, the number of objects with different shapes also increases. In that case, the subtle variations of shapes between different objects are all captured by the heat diffusion, which makes the histogram difficult to analyze, as shown in Figure 7.2. We can see that the histogram is cluttered (circled in white) with many lines. Each of the lines corresponds to one or many of the leaves of the bonsai tree. Due to this clutter, the leaves of the bonsai cannot be classified perfectly.

For the future work, we are planning to explore alternate ways (for example conformal welding) to visualize the heat values in the cumulative heat diffusion process

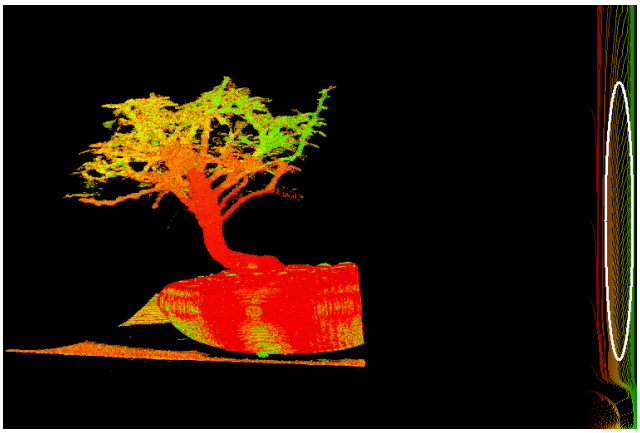


Figure 7.2: Bonsai dataset showing the leaves, trunk and the pot. The region circled in white shows the cluttered area with the lines pertaining to the leaves of the bonsai tree.

so that shape information for complex volumetric data can be better visualized.

7.2.2 Long Term Plan

Automatic Parameter Selection

In the computation of the shape information, there is a parameter p, which decides the boundary of an object, in the cumulative heat diffusion method. Whenever, p is changed, the whole heat diffusion process needs to be re-run. The parameter p should be selected based on the shape of the given data. It might be possible to automatically determine the most proper p based on the boundary conditions in the data. Furthermore, we also intend to focus on finding the optimum time step t. A small time step t only depicts the local shapes. For a certain time step t (called optimum time step), the voxels belonging to each object have the same heat. For any time step bigger than the optimum one, the heat of the voxels will not change and the extra iterations are a waste of computational time. The time step t is also based on the shapes in a data. Proper estimations of p and t will be helpful in the cumulative heat diffusion method and they are the future directions.

In the long term, we would like to analyze the shape information of each voxel in synthetic datasets, such as a sphere, to map the shape information to particular shapes. Based on the difference of the shape information and the shapes, we want to construct a shape database. Using the shape database, it might be possible to automatically identify a shape.

Augmented Heat Diffusion

Heat diffusion is a powerful method of shape analysis and it can be used in conjunction with other methods. Statistical methods, conformal welding, template based methods and knowledge based methods have proven to be successful approaches for shape analysis by making use of already known shape information. By augmenting the topological and geometrical shape information obtained with these methods to heat diffusion, we can increase the accuracy of our results. As of now heat diffusion is a successful classification method and only supports weak segmentation. However, if the heat diffusion method is augmented with statistical shape analysis methods such as principal component analysis or with prior knowledge about the data, then it can also be used as a powerful segmentation tool.

In the long term, we would like to augment the previously successful methods with heat diffusion to not only obtain perceptually and semantically relevant shape information but also achieve accurate segmentation and classification based on shape.

Graph Embedding

It is an NP-hard problem to find a minimal genus embedding surface for a given graph. In this work, we introduced some heuristic ideas to decrease the genus of the embedding surface. However, the problem still remains challenging and requires more research, particularly for large complicated graphs. In the long term, we would like to explore better methods to smooth the embedding of graphs on surfaces, and to reduce the genus of the embedding surfaces.

Chapter 8

Appendix

8.1 Volume of the Region Formed by the Edge Weights

Let w_1, w_2 and w_3 be the three weights which form a octavuspherical region. Let the weight w_1 be aligned in the direction of $x-axis$, the weight w_2 be aligned in the direction of $y-axis$ and the weight w_3 be aligned in the direction of $z-axis$. Let ϕ be the angle measured along the latitudinal direction, that is, along the $y-z$ plane and θ be the angle measure along the longitudinal direction, that is, along the $x-y$ plane. The angle between all these weights is $\frac{\pi}{2}$ since it is a octavusphere. Though the angle between all the weights in $\frac{\pi}{2}$, we show the volume computation for a general case by considering an arbitrary angle $\Phi \in [0, \frac{\pi}{2}]$ so that the equation can be used to calculate the volume of any sub-region in the octavusphere. This in turn is helpful in the binary search approach explained in Section 7.3. Hence, in the general case, the angle ϕ will vary from 0 to Φ and the angle θ will vary from 0 to $\frac{\pi}{2}$. Using these limits, the volume of the region is given by the integral defined as follows:

$$V = \int_0^{\Phi} \int_0^{\frac{\pi}{2}} \int_0^{R} r^2 \cdot \sin(\theta)\, dr\, d\theta\, d\phi \tag{8.1}$$

where r is the radius of an arbitrary point inside the sub-region of the octavusphere. As described above, θ will be the angle formed by this radius r with w_1 and ϕ will be the angle between the projection of r on to the $y-z$ plane and the weight w_2. We get rid of the variable r by integrating Equation 8.1 with respect to r in the limits 0 and R which would arrive us at the following equation:

$$V = \int_0^{\Phi} \int_0^{\frac{\pi}{2}} \frac{R^3}{3} \cdot \sin(\theta)\, d\theta\, d\phi \tag{8.2}$$

where R is a point on the curve joining the three weights and hence is defined in terms of weights w_1, w_2 and w_3 as follows:

$$R = \frac{w_1 \cdot (\frac{\pi}{2} - \theta) + A \cdot \theta}{\frac{\pi}{2}} \tag{8.3}$$

where A is given by:

$$A = \frac{w_2 \cdot (\frac{\pi}{2} - \phi) + w_3 \cdot \phi}{\frac{\pi}{2}} \tag{8.4}$$

By solving the integral in Equation 8.2, we get the volume of the region as:

$$\begin{aligned}
V \;=\; & -\frac{1}{12} \cdot (384 \cdot w_3^3 + 1152 \cdot w_2^2 \cdot w_3 - 144 \cdot w_2^2 \cdot \pi^2 \cdot w_3 + 144 \cdot w_2 \cdot \pi^2 \cdot w_3^2 \\
& + 48 \cdot w_2^3 \cdot \pi^2 - 48 \cdot w_3^3 \cdot \pi^2 - 1152 \cdot w_2 \cdot w_3^2 - 384 \cdot w_2^3) / \pi^6 \cdot \Phi^4 \\
& - \frac{1}{9} \cdot (576 \cdot w_2 \cdot \pi \cdot w_3^2 - 192 \cdot w_1 \cdot \pi^2 \cdot w_2 \cdot w_3 + 144 \cdot w_2^2 \cdot \pi^3 \cdot w_3 \\
& - 576 \cdot w_1 \cdot \pi \cdot w_3^2 - 48 \cdot w_1 \cdot \pi^3 \cdot w_2 \cdot w_3 - 72 \cdot w_2^3 \cdot \pi^3 + 24 \cdot w_1 \cdot \pi^3 \cdot w_3^2 \\
& + 96 \cdot w_1 \cdot \pi^2 \cdot w_2^2 - 1152 \cdot w_2^2 \cdot \pi \cdot w_3 - 576 \cdot w_1 \cdot \pi \cdot w_2^2 + 24 \cdot w_1 \cdot \pi^3 \cdot w_2^2 \\
& + 576 \cdot w_2^3 \cdot \pi - 72 \cdot w_2 \cdot \pi^3 \cdot w_3^2 + 96 \cdot w_1 \cdot \pi^2 \cdot w_3^2 \\
& + 1152 \cdot w_1 \cdot \pi \cdot w_2 \cdot w_3) / \pi^6 \cdot \Phi^3 - \frac{1}{6} \cdot (96 \cdot w_1 \cdot \pi^3 \cdot w_2 \cdot w_3 \\
& - 96 \cdot w_1 \cdot \pi^3 \cdot w_2^2 + 24 \cdot w_1 \cdot \pi^4 \cdot w_2 \cdot w_3 + 96 \cdot w_1^2 \cdot \pi^3 \cdot w_2 \\
& - 576 \cdot w_1 \cdot \pi^2 \cdot w_2 \cdot w_3 + 36 \cdot w_2^3 \cdot \pi^4 - 288 \cdot w_1^2 \cdot \pi^2 \cdot w_2 - 36 \cdot w_2^2 \cdot \pi^4 \cdot w_3 \\
& - 24 \cdot w_1 \cdot \pi^4 \cdot w_2^2 - 288 \cdot w_2^3 \cdot \pi^2 - 96 \cdot w_1^2 \cdot \pi^3 \cdot w_3 + 288 \cdot w_2^2 \cdot \pi^2 \cdot w_3 \\
& + 576 \cdot w_1 \cdot \pi^2 \cdot w_2^2 + 288 \cdot w_1^2 \cdot \pi^2 \cdot w_3) / \pi^6 \cdot \Phi^2 - \frac{1}{3} \cdot (144 \cdot w_1^2 \cdot \pi^3 \cdot w_2 \\
& + 48 \cdot w_2^3 \cdot \pi^3 + 24 \cdot w_1^3 \cdot \pi^4 - 48 \cdot w_1^3 \cdot \pi^3 - w_1^3 \cdot \pi^6 - 48 \cdot w_1^2 \cdot \pi^4 \cdot w_2 \\
& - 6 \cdot w_2^3 \cdot \pi^5 + 6 \cdot w_1 \cdot \pi^5 \cdot w_2^2 + 24 \cdot w_1 \cdot \pi^4 \cdot w_2^2 \\
& - 144 \cdot w_1 \cdot \pi^3 \cdot w_2^2) / \pi^6 \cdot \Phi
\end{aligned} \tag{8.5}$$

Equation 8.5 is a general equation that can be used to evaluate the volume of any sub-region in the octavusphere. This equation is used in the angle estimation using the binary search approach explained in Section 7.3. However, if you want to compute the volume of the entire octavusphere, it can simply be obtained by replacing the value of Φ in Equation 8.5 with $\frac{\pi}{2}$. Thus, the volume of the entire

octavusphere is given by:

$$\begin{aligned} V &= \frac{1}{12}\cdot(2\cdot w_1^3\cdot\pi^3 - 16\cdot w_1\cdot\pi\cdot w_2^2 + 48\cdot w_1^2\cdot\pi\cdot w_2 \\ &\quad -16\cdot w_1\cdot\pi\cdot w_3^2 - 48\cdot w_1^3\cdot\pi - 16\cdot w_1\cdot\pi\cdot w_2\cdot w_3 \\ &\quad +48\cdot w_1^2\cdot\pi\cdot w_3 - 24\cdot w_2\cdot w_3^2 - 4\cdot w_1\cdot\pi^2\cdot w_2^2 \\ &\quad +3\cdot w_2^2\cdot w_3\cdot\pi^2 + 96\cdot w_1^3 + 96\cdot w_1\cdot w_2^2 + 3\cdot w_3^3\cdot\pi^2 \\ &\quad -144\cdot w_1^2\cdot w_2 - 24\cdot w_3^3 + 3\cdot w_2^3\cdot\pi^2 + 3\cdot w_2\cdot w_3^2\cdot\pi^2 \\ &\quad -144\cdot w_3\cdot w_1^2 - 24\cdot w_2^2\cdot w_3 + 96\cdot w_1\cdot w_3^2 \\ &\quad -4\cdot w_3\cdot w_1\cdot\pi^2\cdot w_2 - 24\cdot w_2^3 + 96\cdot w_1\cdot w_2\cdot w_3 \\ &\quad -4\cdot w_3^2\cdot w_1\cdot\pi^2)/(\pi^2) \end{aligned} \tag{8.6}$$

8.2 Area of the region formed by the edge weights

Let w_1 and w_2 be the edge weights. These edge weights along with the arc connecting them enclose the region whose area is to be evaluated. The angle between these weights is $\frac{\pi}{2}$. However, we evaluate the area by considering an arbitrary angle, Θ between the weights to obtain a generalized formula for the area. Using this formula, the area of any sub-region between the weights can be evaluated, which in turn is used in the binary search approach explained in Section 7.3.

The weights are assumed to be linearly distributed based on the angles between them. To find the area of the region formed by the weights and the arc joining them, we consider an arbitrary position between them on the arc which indicates the weight of an arbitrary direction between w_1 and w_2. Let us denote this by w'. Assume the angle between w' and w_1 be θ. Since the weights are linearly distributed, the value of w' is given in terms of w_1, w_2 as follows:

$$\begin{aligned} w' &= \frac{w_1\cdot(\frac{\pi}{2}-\theta)+w_2\cdot\theta}{\frac{\pi}{2}} \\ &= w_1\cdot(1-(\frac{2\cdot\theta}{\pi}))+w_2\cdot(\frac{2\cdot\theta}{\pi}) \end{aligned}$$

By varying θ between 0 and Θ, the area of the region is given by the following integral:

$$\begin{aligned} A &= \frac{1}{2}\cdot\int_0^{\Theta} w'^2\,d\theta \\ &= \frac{1}{2}\cdot\int_0^{\Theta}(w_1\cdot(1-(\frac{2\cdot\theta}{\pi}))+w_2\cdot(\frac{2\cdot\theta}{\pi}))^2\,d\theta \\ &= \frac{1}{2}\cdot\int_0^{\Theta}[w_1^2\cdot(1-\frac{2\cdot\theta}{\pi})^2+\frac{4\cdot w_2^2}{\pi^2}\cdot\theta^2+4\cdot w_1\cdot w_2\cdot\frac{\theta}{\pi}\cdot(1-\frac{2\cdot\theta}{\pi})]\,d\theta \end{aligned}$$

Finally, solving this integral in the limits we obtain the area of the region as:

$$A = \frac{2}{3 \cdot \pi^2} \cdot (w_2 - w_1)^2 \cdot \Theta^3 + \frac{1}{\pi} \cdot w_1 \cdot (w_2 - w_1) \cdot \Theta^2 + \frac{1}{2} \cdot w_1^2 \cdot \Theta \quad (8.7)$$

As mentioned earlier, Equation 8.7 is a general equation used to find the area of the region formed by the edge weights when the angle between them is any arbitrary angle Θ. Equation 8.7 is used in area computation for angle estimation by using the binary search approach explained in Section 7.3. However, for the whole sector, the angle between the weights enclosing the region will be $\frac{\pi}{2}$. Then, the area of the whole sector is obtained by replacing Θ of Equation 8.7 with $\frac{\pi}{2}$, which is given by:

$$\begin{aligned} A &= \frac{1}{48} \cdot (\frac{2 \cdot w_1}{\pi} - \frac{2 \cdot w_2}{\pi})^2 \cdot \pi^3 + \frac{1}{8} \cdot w_1 \cdot (\frac{2 \cdot w_2}{\pi} - \frac{2 \cdot w_1}{\pi}) \cdot \pi^2 + \frac{1}{4} \cdot w_1^2 \cdot \pi \\ &= \frac{\pi}{12} \cdot (w_1^2 + w_2^2 + w_1 \cdot w_2) \end{aligned}$$

In the algorithm in Section 3.2, w_2 should be replaced with W' to obtain the area of the sector (Refer to Equation 3.9 of Section 3.2).

8.3 Binary search

In general, binary search is used to find the position of a specific value in a sorted array of values. This is achieved by recursive comparison of the value with the middle element of the array and discarding a half of the array in each iteration. In the approach, we employ this technique with a slight modification to estimate the value of the angle that evalutes an area or volume. Let θ be the angle to be computed, A denote the total area or total volume of the region under consideration, α be the angle enclosing the total region whose area or volume is A and A' be the area or the volume of the sub-region defined by θ. In all the evaluations the value of α is $\frac{\pi}{2}$. Let $\alpha\_i_{1/2}$ represent the half angle at i^{th} iteration and $A\_i_{1/2}$ represent the corresponding area or volume of the region defined by $\alpha\_i_{1/2}$. The intial value of half angle, which is given by $\alpha\_0_{1/2}$ is equal to $\frac{\alpha}{2}$. Hence, the initial value of $A\_i_{1/2}$, which is given by $A\_0_{1/2}$ denotes the area or volume of the region defined by $\alpha\_0_{1/2} = \frac{\alpha}{2}$. Here note that $A\_0_{1/2}$ is not equal to $\frac{A}{2}$, rather it is the area or the volume of the region defined by half of the angle $\frac{\alpha}{2}$. The value of $A\_0_{1/2}$ or in general the value of any $A\_i_{1/2}$ is computed by using Equations 8.5 (for volume) or 8.7 (for area) in Sections 7.1 and 7.2, respectively.

In the first iteration, the value A' is compared with the value $A\_0_{1/2}$. In the subsequent iterations, the value A' is compared with the value $A\_i_{1/2}$. If A' is smaller

than $A\_i_{1/2}$, then the values of $\alpha\_i_{1/2}$ and $A\_i_{1/2}$ are updated as follows.

$$\begin{aligned} \alpha\_i_{1/2} &= \alpha\_(i-1)_{1/2} - \frac{\alpha}{(2^{i+1})} \\ A\_i_{1/2} &= \text{Area/Volume of region defined by the} \\ &\quad \text{updated } \alpha\_i_{1/2} \end{aligned} \tag{8.8}$$

If A' is greater than $A\_i_{1/2}$, then the values of $\alpha\_i_{1/2}$ and $A\_i_{1/2}$ are updated as follows.

$$\begin{aligned} \alpha\_i_{1/2} &= \alpha\_(i-1)_{1/2} + \frac{\alpha}{(2^{i+1})} \\ A\_i_{1/2} &= \text{Area/Volume of region defined by the} \\ &\quad \text{updated } \alpha\_i_{1/2} \end{aligned} \tag{8.9}$$

Based on these conditions, this process is repeated recursively until $|A' - A\_i_{1/2}| < \varepsilon$ where ε is a threshold value. In the calculation, we choose the value of ε to be 0.01. Hence, in this approach we get an approximate estimate of the angle θ which evaluates the area or volume A'. Since the area or volume of the region defined by a larger θ value is greater than the area or volume of the region defined by a smaller θ value, this approach of using the binary search to estimate the required angle is justified.

Bibliography

[1] Y. Aflalo, R. Kimmel, and D. Raviv. Scale invariant geometry for nonrigid shapes. *SIAM Journal of Imaging Sciences*, 6(3):1579–1597, 2013.

[2] C. Ahlberg. Spotfire: An information exploration environment. *SIGMOD Record*, 25(4):25–29, 1996.

[3] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 483–492, 2013.

[4] M. Ament, D. Weiskopf, and H. Carr. Direct interval volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1505–1514, Nov. 2010.

[5] K. Andrews, M. Wohlfahrt, and G. Wurzinger. Visual graph comparison. In *IV*, pages 62–67. IEEE Computer Society, 2009.

[6] S. Angenent, S. Haker, A. Tannenbaum, and R. Kikinis. Conformal geometry and brain flattening. *Medical Image Computing and Computer-Assisted Intervention MICCAI*, 1679:271–278, 1999.

[7] S. Angenent, S. Haker, A. Tannenbaum, and R. Kikinis. On the Laplace-Beltrami operator and brain surface flattening. *IEEE Transactions on Medical Imaging*, 18(8):700–711, 1999.

[8] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[9] D. Archambault and H. C. Purchase. Mental map preservation helps user orientation in dynamic graphs. *Graph Drawing*, 7704:475–486, 2012.

[10] D. Archambault, H. C. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552, 2011.

[11] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. *ICCV Workshops*, pages 1626–1633, 2011.

[12] B. Bach, E. Pietriga, and J.-D. Fekete. GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks. *Transactions on Visualization and Computer Graphics*, 20(5):740–754, Nov. 2014.

[13] B. Bach, E. Pietriga, and J.-D. Fekete. Visualizing dynamic networks with matrix cubes. *SIGCHI Conference on Human Factors in Computing Systems*, pages 877–886, 2014.

[14] E. Balogh, E. Sorantin, L. G. Nyul, K. Palagyi, A. Kuba, G. Werkgartner, and E. Spuller. Colon unraveling based on electronic field: Recent progress and future work. *Proceedings SPIE*, 4681:713–721, 2002.

[15] A. V. Bartrol, R. Wegenkittl, A. Knig, E. Grller, and E. Sorantin. Virtual colon flattening. *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 127–136, May 2001.

[16] A. V. Bartrolí, R. Wegenkittl, A. König, and E. Gröller. Nonlinear virtual colon unfolding. *IEEE Visualization*, pages 411–420, 2001.

[17] R. F. Bass. Brownian motion, heat kernel and harmonic functions. *Proceedings of International Congress of Mathematicians*, 2:980–985, 1995.

[18] C. Batini, L. Furlani, and E. Nardelli. What is a good diagram? a pragmatic approach. *Proceedings of the Fourth International Conference on Entity-Relationship Approach*, pages 312–319, 1985.

[19] M. Belkin, J. Sun, and Y. Wang. Discrete laplace operator on meshed surfaces. *Proceedings of the Symposium on Computational Geometry*, pages 278–287, 2008.

[20] I. Bitter, A. E. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, July 2001.

[21] K. Boitmanis, U. Brandes, and C. Pich. Visualizing internet evolution on the autonomous systems level. *Graph Drawing*, 4875:365–376, 2007.

[22] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept. 2004.

[23] U. Brandes and M. Mader. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. *Graph Drawing*, 7034:99–110, 2011.

[24] M. M. Bronstein and I. Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1704–1711, 2010.

[25] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, June 2010.

[26] U. Castellani, M. Cristani, and V. Murino. Statistical 3D shape analysis by local generative descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2555–2560, 2011.

[27] F. Cecconi, M. Cencini, M. Falcioni, and A. Vulpiani. Brownian motion and diffusion: From stochastic processes to chaos and beyond. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 15(2):026102, 2005.

[28] M. M. Center, A. Jemal, R. A. Smith, and E. Ward. Worldwide variations in colorectal cancer. *CA: A Cancer Journal for Clinicians*, 59(6):366–378, November/December 2009.

[29] S.-s. Chern. An elementary proof of the existence of isothermal parameters on a surface. *Proceedings of American Mathematical Society*, 6(5):771–782, 1955.

[30] B. Chow, P. Lu, and L. Ni. *Hamilton's Ricci Flow*. American Mathematical Society, 2006.

[31] B. Chow and F. Luo. Combinatorial ricci flows on surfaces. *Journal Differential Geometry*, 63(1):97–129, 2003.

[32] A. S. Chowdhury, J. Yao, R. L. V. U. Jr., M. G. Linguraru, and R. M. Summers. Detection of anatomical landmarks in human colon from computed tomographic colonography images. *International Conference on Pattern Recognition*, pages 1–4, 2008.

[33] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, pages 7426–7431, 2005.

[34] C. D. Correa and D. Silver. Dataset traversal with motion-controlled transfer functions. *IEEE Visualization*, pages 359 – 366, Oct. 2005.

[35] CUDA. http://www.nvidia.com/object/cuda_home_new.html.

[36] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[37] T. K. Dey, K. Li, J. Sun, and D. Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3D models. *ACM Transactions on Graphics*, 27(3):45:1–45:9, Aug. 2008.

[38] S. Diehl and C. Grg. Graphs, they are changing - dynamic graph drawing for a sequence of graphs. *Graph Drawing*, 2528:23–31, 2002.

[39] J. Dodziuk. Finite-difference approach to the Hodge theory of harmonic forms. *American Journal of Mathematics*, 98(1):79–104, 1976.

[40] A. Dominitz and A. Tannenbaum. Texture mapping via optimal mass transport. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):419–433, 2010.

[41] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. AK Peters, 2006.

[42] K. Engel, M. Kraus, and T. Ertl. Interactive high-quality pre-integrated volume rendering using hardware-accelerated graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 9–16, 2001.

[43] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. V. Yee. Graphael: Graph animations with evolving layouts. *Graph Drawing*, 2912:98–110, 2003.

[44] M. Farrugia, N. Hurley, and A. Quigley. Exploring temporal ego networks using small multiples and tree-ring layouts. *International Conference on Advances in Human Computer Interfaces*, pages 23–28, 2011.

[45] R. Fattal. Blue-noise point sampling using kernel density model. *SIGGRAPH*, 28(3):1–10, 2011.

[46] P. Federico, W. Aigner, S. Miksch, F. Windhager, and L. Zenk. A visual analytics approach to dynamic social networks. *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, pages 47:1–47:8, 2011.

[47] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, 1995.

[48] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. *Proceedings of the DIMACS International Workshop on Graph Drawing*, pages 388–403, 1994.

[49] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1310–1319, 2007.

[50] Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008.

[51] S. Ghani, N. Elmqvist, and J. S. Yi. Perception of animated node-link diagrams for dynamic graphs. *Computer Graphics Forum*, 31(3pt3):1205–1214, June 2012.

[52] C. Grg, P. Birke, M. Pohl, and S. Diehl. Dynamic graph drawing of sequences of orthogonal and hierarchical graphs. *Graph Drawing*, 3383:228–238, 2004.

[53] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S. tung Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transactions on Medical Imaging*, 23:949–958, 2004.

[54] K. C. Gurijala, A. E. Kaufman, W. Zeng, and X. Gu. Extraction of landmarks and features from virtual colon models. *Proceedings of MICCAI Workshop on Virtual Colonoscopy and Abdominal Imaging*, 6668:105–112, 2010.

[55] K. C. Gurijala, H. Peng, X. Yu, W. Zeng, X. Gu, and A. E. Kaufman. Graph embedding on surface using Ricci flow. *Submitted to IEEE Transactions on Visualization and Computer Graphics*, 2014.

[56] K. C. Gurijala, R. Shi, W. Zeng, X. Gu, and A. E. Kaufman. Colon flattening using heat diffusion Riemannian metric. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2848–2857, 2013.

[57] K. C. Gurijala, L. Wang, and A. E. Kaufman. Cumulative heat diffusion using volume gradient operator for volume analysis. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2069–2077, Dec 2012.

[58] K. C. Gurijala, L. Wang, and A. E. Kaufman. Monte-Carlo based real-time shape analysis in volumes. *Submitted to IEEE Transactions on Visualization and Computer Graphics*, 2014.

[59] I. Guskov and Z. J. Wood. Topological noise removal. *Graphics Interface*, pages 19–26, 2001.

[60] M. Haidacher, S. Bruckner, and M. E. Gröller. Volume analysis using multimodal surface similarity. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1969–1978, Oct. 2011.

[61] S. Haker, S. Angenent, A. Tannenbaum, and R. Kikinis. Nondistorting flattening maps and the 3D visualization of colon CT images. *IEEE Transactions on Medical Imaging*, 19:665–670, July 2000.

[62] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, Apr. 2000.

[63] S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent. Optimal mass transport for registration and warping. *International Journal on Computer Vision*, 60:225–240, 2004.

[64] M. Hascoët and P. Dragicevic. Interactive graph matching and visual comparison of graphs and clustered graphs. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 522–529, 2012.

[65] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. *SIGGRAPH*, pages 203–212, 2001.

[66] D. Holten and J. J. van Wijk. Visual comparison of hierarchically organized data. *Computer Graphics Forum*, 27(3):759–766, 2008.

[67] L. Hong, S. Muraki, A. E. Kaufman, D. Bartz, and T. He. Virtual voyage: interactive navigation in the human colon. *SIGGRAPH*, pages 27–34, 1997.

[68] W. Hong, X. Gu, F. Qiu, M. Jin, and A. Kaufman. Conformal virtual colon flattening. *ACM Symposium on Solid and Physical Modeling*, pages 85–93, 2006.

[69] W. Hong, F. Qiu, and A. Kaufman. A pipeline for computer aided polyp detection. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):861–868, Sept. 2006.

[70] M. Horner, L. Ries, M. Krapcho, N. Neyman, R. Aminou, N. Howlader, S. Altekruse, E. Feuer, L. Huang, A. Mariotto, B. Miller, D. Lewis, M. Eisner, D. Stinchcomb, and B. Edwards (eds). Seer Cancer Statistics Review. 1975-2006.

[71] Z. Hossain, U. R. Alim, and T. Moller. Toward high-quality gradient estimation on regular lattices. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):426–439, April 2011.

[72] Y. Hu, S. G. Kobourov, and S. Veeramoni. Embedding, clustering and coloring for dynamic maps. *Pacific Visualization*, pages 33–40, 2012.

[73] A. Huang, D. Roy, M. Franaszek, and R. M. Summers. Teniae coli guided navigation and registration for virtual colonoscopy. *IEEE Visualization*, pages 279–285, 2005.

[74] A. Huang, R. Summers, and A. Hara. Surface curvature estimation for automatic colonic polyp detection. *SPIE*, pages 393–402, 2005.

[75] H. W. Jensen. Global illumination using photon maps. *Proceedings of the Eurographics Workshop on Rendering Techniques*, pages 21–30, 1996.

[76] H. W. Jensen. Rendering caustics on non-lambertian surfaces. *Computer Graphics Forum*, 16(1):57–64, 1997.

[77] H. W. Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

[78] G. Jiang and L. Gu. An automatic and fast centerline extraction algorithm for virtual colonoscopy. *IEEE Engineering in Medicine and Biology Society*, 5:5149–5152, 2005.

[79] M. Jin, J. Kim, and X. D. Gu. Discrete surface Ricci flow: Theory and applications. *Proceedings of the 12th IMA International Conference on Mathematics of Surfaces XII*, pages 209–232, 2007.

[80] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete surface ricci flow. *IEEE Transaction on Visualization and Computer Graphics*, 14(5):1030–1043, 2008.

[81] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete surface Ricci flow. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1030–1043, 2008.

[82] C. D. Johnson and A. H. Dachman. CT colography: The next colon screening examination. *Radiology*, 216(2):331–341, 2000.

[83] T. Ju, Q.-Y. Zhou, and S.-M. Hu. Editing the topology of 3D models by sketching. *ACM Transactions on Graphics*, 26(3):42, July 2007.

[84] M. Kac. Random walk and the theory of Brownian motion. *The American Mathematical Society*, 54(7):369–391, August 1947.

[85] L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics*, 25(2):412–438, Apr. 2006.

[86] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. *Proceedings of the IEEE Symposium on Volume Visualization*, pages 79–86, 1998.

[87] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, July 2002.

[88] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1571–1578, November/December 2009.

[89] P. M. Knupp. Matrix norms and the condition number: A general framework to improve mesh quality via node-movement. *Eighth International Meshing Roundtable*, pages 13–22, 1999.

[90] S. G. Kobourov and K. Wampler. Non-euclidean spring embedders. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):757–767, 2005.

[91] Y. Koren, L. Carmel, and D. Harel. Drawing huge graphs by algebraic multigrid optimization. *SIAM Multiscale Modeling and Simulation*, 1(4):645–673, 2003.

[92] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):805–812, 2006.

[93] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *SIGGRAPH*, pages 451–458, July 1994.

[94] S. Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, May 2004.

[95] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. *Proceedings of ACM symposium on Solid and Physical Modeling*, pages 183–191, 2008.

[96] J. Lamy and R. M. Summers. Teniæ coli detection from colon surface: Extraction of anatomical markers for virtual colonoscopy. *International Symposium on Visual Computing*, pages 199–207, 2007.

[97] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, pages 80–89, 2001.

[98] B. Lee, M. Czerwinski, G. G. Robertson, and B. B. Bederson. Understanding eight years of infovis conferences using paperlens. *Proceedings of the IEEE Symposium on Information Visualization*, 2004.

[99] M. Leif. *Stability, Riemann Surfaces, Conformal Mappings*. Ventus Publishing, 2010.

[100] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, Aug. 2004.

[101] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[102] B. Levy. Laplace-Beltrami eigenfunctions towards an algorithm that "understands" geometry. *Proceedings of the IEEE International Conference on Shape Modeling and Applications*, pages 13:1–8, 2006.

[103] W. Li, K. Mueller, and A. Kaufman. Empty space skipping and occlusion clipping for texture-based volume rendering. *IEEE Visualization*, pages 317–324, 2003.

[104] S. Lim, H.-J. Lee, and B.-S. Shin. Surface reconstruction for efficient colon unfolding. *International Conference on Geometric Modeling and Processing*, pages 623–629, 2006.

[105] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, August 1987.

[106] H. M and S. K. Cortical cartography using the discrete conformal approach of circle packings. *NeuroImage*, 23:S119–S128, 2004.

[107] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[108] M. Meila and J. Shi. A random walks view of spectral segmentation. *AISTATS - AI and Statistics*, pages 8–11, 2001.

[109] M. Mirzargar and A. Entezari. Quasi interpolation with Voronoi splines. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1832–1841, Dec. 2011.

[110] D. Mohr and G. Zachmann. Continuous edge gradient-based template matching for articulated objects. *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 519–524, February 2009.

[111] L. Mroz, H. Hauser, and E. Gröller. Interactive high-quality maximum intensity projection. *Computer Graphics Forum*, 19(3):341–350, 2000.

[112] C. Muelder and K.-L. Ma. Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1301–1308, 2008.

[113] Z. Nehari. *Conformal Mapping*. New York:Dover, 1982.

[114] A. Noack. An energy model for visual graph clustering. *Proceedings of Symposium on Graph Drawing*, pages 425–436, 2003.

[115] S. C. North. Incremental layout in dynadag. *Graph Drawing*, 1027:409–418, 1995.

[116] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. J. Guibas. One point isometric matching with the heat kernel. *Proceedings of the Eurographics Symposium on Geometry Processing*, 29(5):1555–1564, 2010.

[117] D. S. Paik, C. F. Beaulieu, R. B. J. Jeffrey, C. A. Karadi, and S. Napel. Visualization modes for CT colonography using cylindrical and planar map projections. *Journal of Computer Assisted Tomography*, 24:179–188, 2000.

[118] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro real-time ray-casting system. *SIGGRAPH*, pages 251–260, 1999.

[119] H. Pfister and A. Kaufman. Cube-4 - A scalable architecture for real-time volume rendering. *Proc. of Symposium on Volume Visualization*, pages 47–54, 1996.

[120] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–22, May 2001.

[121] L. A. Piegl and W. Tiller. *The NURBS Book*. Springer, 1996.

[122] U. Pinkall, S. D. Juni, and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[123] S. M. Pizer, G. Gerig, S. C. Joshi, and S. R. Aylward. Multiscale medial shape-based analysis of image objects. *Proceedings of the IEEE*, 91(10):1670–1679, 2003.

[124] M. Pohl, F. Reitz, and P. Birke. As time goes by: Integrated visualization and analysis of dynamic networks. *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 372–375, 2008.

[125] J.-S. Praßni, T. Ropinski, J. Mensmann, and K. H. Hinrichs. Shape-based transfer functions for volume visualization. *Proceedings of the IEEE Pacific Visualization Symposium*, pages 9–16, Mar 2010.

[126] H. C. Purchase, R. F. Cohen, and M. James. Validating graph drawing aesthetics. *Proceedings of Graph Drawing*, pages 435–446, 1996.

[127] D. Raviv, M. M. Bronstein, A. M. Bronstein, and R. Kimmel. Volumetric heat kernel signatures. *Proceedings of the ACM Workshop on 3D Object Retrieval*, pages 39–44, 2010.

[128] D. Reniers, A. Jalba, and A. Telea. Robust classification and analysis of anatomical surfaces using 3D skeletons. *Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine*, pages 61–68, 2008.

[129] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo. Discrete Laplace-Beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33:381–390, Jun 2009.

[130] M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, Apr 2006.

[131] R. M. Rustamov. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. *Proceedings of the Eurographics Symposium on Geometry Processing*, pages 225–233, 2007.

[132] Y. Sato, C. F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3D local intensity structure for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180, 2000.

[133] R. Schoen and S.-T. Yau. *Lectures on Differential Geometry*. International Press, 1994.

[134] J. Shi, P. M. Thompson, and Y. Wang. Human brain mapping with conformal geometry and multivariate tensor-based morphometry. 7012:126–134, 2011.

[135] M. Smelyanskiy, D. Holmes, J. Chhugani, A. Larson, D. M. Carmean, D. Hanson, P. Dubey, K. Augustine, D. Kim, A. Kyker, V. W. Lee, A. D. Nguyen, L. Seiler, and R. Robb. Mapping high-fidelity volume rendering for medical imaging to CPU, GPU and many-core architectures. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1563–1570, November/December 2009.

[136] B. Springborn, P. Schröder, and U. Pinkall. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics*, 27(3):77:1–77:11, Aug. 2008.

[137] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. *Proceedings of Volume Graphics*, pages 187–195, 2005.

[138] M. Styner, I. Oguz, S. Xu, C. Brechbhler, D. Pantazis, J. Levitt, M. Shenton, and G. Gerig. Framework for the statistical shape analysis of brain structures using spharm-pdm. *Proceedings of Image Computing and Computer Assisted Intervention Open Science Workshop*, 2006.

[139] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man & Cybernetics*, 11(2):109–125, 1981.

[140] J. Sun, M. Ovsjanikov, and L. J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Proceedings of the Eurographics Symposium on Geometry Processing*, 28(5):1383–1392, 2009.

[141] M. E. Taylor. *Partial differential equations. I. , Basic theory*. Applied mathematical sciences. Springer, 2011.

[142] C. Thomassen. The graph genus problem is NP-complete. *J. Algorithms*, 10(4):568–576, 1989.

[143] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. *Eurographics Symposium on Geometry Processing*, pages 201–210, 2006.

[144] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. *Proceedings of European Conference on Computer Vision*, pages 596–609, 2008.

[145] G. Uhlenbeck and L. Ornstein. On the theory of Brownian motion. *Physical Review*, 36(5):823–841, September 1930.

[146] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. *IEEE Symposium on Information Visualization*, pages 199–206, 2004.

[147] A. Vaxman, M. Ben-Chen, and C. Gotsman. A multi-resolution approach to heat kernels on discrete surfaces. *ACM Transactions on Graphics*, 29:121:1–121:10, July 2010.

[148] G. Wang, S. B. Dave, B. P. Brown, Z. Zhang, E. G. Mcfarland, J. W. Haller, and M. W. Vannier. Colon unraveling based on electrical field: Recent progress and future work. *Proceedings SPIE*, 3660:125–132, 1999.

[149] G. Wang, E. G. Mcfarland, B. P. Brown, and M. W. Vannier. GI tract unraveling with curved cross section. *IEEE Transactions on Medical Imaging*, 17:318–322, 1998.

[150] G. Wang and M. W. Vannier. GI tract unraveling by spiral CT. *Proceedings SPIE*, 2434:307–315, 1995.

[151] M. C. Wang and G. Uhlenbeck. On the theory of Brownian motion II. *Reviews of Modern Physics*, 17(2-3):323–342, April 1945.

[152] Y. Wang, X. Gu, T. F. Chan, P. M. Thompson, and S.-T. Yau. Conformal slit mapping and its applications to brain surface parameterization. *MICCAI*, pages 585–593, 2008.

[153] Z. Wei, J. Yao, S. Wang, and R. M. Summers. Teniae coli extraction in human colon for computed tomographic colonography images. *Virtual Colonoscopy and Abdominal Imaging*, pages 98–104, 2010.

[154] L. Westover. Footprint evaluation for volume rendering. *SIGGRAPH*, 24:367–376, Sept. 1990.

[155] Y.-L. Yang, J. Kim, F. Luo, S.-M. Hu, and X. Gu. Optimal surface parameterization using inverse curvature map. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1054–1066, Sep 2008.

[156] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of dynamic graphs with radial layout. *Proceedings of the IEEE Symposium on Information Visualization*, pages 43–50, 2001.

[157] X. Yin, J. Dai, S.-T. Yau, and X. Gu. Slit map: Conformal parameterization for multiply connected surfaces. *Geometric Modeling and Processing*, pages 410–422, 2008.

[158] W. Zeng, J. Marino, K. C. Gurijala, X. Gu, and A. E. Kaufman. Supine and prone colon registration using quasi-conformal mapping. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1348–1357, Nov. 2010.

[159] X. Zhao, Z. Su, X. D. Gu, A. E. Kaufman, J. Sun, J. Gao, and F. L. 0002. Area-preservation mapping using optimal mass transport. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2838–2847, 2013.

[160] Q.-Y. Zhou, T. Ju, and S.-M. Hu. Topology repair of solid models using skeletons. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):675–685, July 2007.

[161] L. Zhu, S. Haker, and A. Tannenbaum. Flattening maps for the visualization of multibranched vessels. *International Conference Image Processing*, 24:945–948, 2005.

[162] V. Zobel, J. Reininghaus, and I. Hotz. Generalized heat kernel signatures. *Journal of WSCG*, 19(3):93–100, 2011.