공학박사 학위논문

Blood Flow Visualization using Flowline Extraction and Opacity Modulation based on Vascular Structure Analysis

혈관 구조 분석 기반 혈류선 추출과 불투명도 변조를 이용한 혈류 가시화 기법

2016년 2월

서울대학교 대학원

컴퓨터공학부

권 오 재

Blood Flow Visualization using Flowline Extraction and Opacity Modulation based on Vascular Structure Analysis

지도교수 신 영 길

이 논문을 공학박사 학위논문으로 제출함
2015년 10월

서울대학교 대학원
컴퓨터공학부
권 오 재

권오재의 박사 학위논문을 인준함
2015년 12월

위 원 장 김 명 수 (인)

부위원장 신 영 길 (인)

위 원 염 헌 영 (인)

위 원 김 보 형 (인)

위 원 이 정 진 (인)

Abstract

Blood Flow Visualization using Flowline Extraction and Opacity Modulation based on Vascular Structure Analysis

Ohjae Kwon
School of Computer Science and Engineering
The Graduate School
Seoul National University

With recent advances in acquisition and simulation of blood flow data, blood flow visualization has been widely used in medical imaging for the diagnosis and treatment of pathological vessels. The integral line based method has been most commonly employed to depict hemodynamic data because it exhibits a long term flow behavior useful for flow analysis. This method generates integral lines to be used as a basis for graphical representation by tracing the trajectory of a massless particle released on the vector field through a numerical integration. However, there are several unsolved problems when this previous method is applied to thin curved vascular structures. The first one is to locate a seeding plane, which is manually performed in the existing methods, thus yielding inconsistent visual results. The second one is the early termination of a line integration due to locally reversed flow and narrow tubular structure, which results in short flowlines comparing with the vessel length. And the last one is the line occlusion caused by the dense depiction of flowlines. Additionally, in blood flow visualization for clinical uses, it is essential to apparently exhibit abnormal flow relevant to vessel diseases. In this paper, we present an enhanced method that overcomes problems related to the integration based flow visualization and depicts hemodynamics in a more informative way

for assisting the diagnosis process. Using the fact that blood flow passes through the inlet or outlet but is blocked by vessel wall, we firstly identify the vessel inlet or outlet by the orthogonality metric between flow velocity vector and vessel surface normal vector. Then, we generate seed points on the detected inlet or outlet by Poisson disk sampling. Therefore, we can achieve the automatic seeding that leads to a consistent and faster flow depiction by skipping the manual location of a seeding plane to initiate the line integration. In addition, we resolve the early terminated line integration by applying the tracing direction adaptively based on flow direction at each seed point and by performing the additional seeding near the terminated location. This solution enables to yield length-extended flowlines, which contribute to faithful flow visualization. Based on the observation that blood flow usually follows the vessel track if there is no obstacle or leak in the middle of a passage, we define the representative flowline for each branch by the vessel centerline. Then, we render flowlines by assigning the opacity according to their shape similarity with the vessel centerline so that flowlines similar to the vessel centerline are shown transparently, while different ones opaquely. Accordingly, our opacity modulation method enables flowlines with unusual flow pattern to appear more noticeable, while minimizing visual clutter and line occlusion. Finally, we introduce HSV (hue, saturation, value) color coding to simultaneously exhibit flow attributes such as local speed and residence time. This color coding gives a more realistic fading effect on the older particles or line segments by attenuating the saturation according to the residence time. Hence, it supports users in comprehending intuitively multiple information at once. Experimental results show that our technique is well suitable to depict blood flow in vascular structures.

Keywords: Blood flow visualization; Integration based flow visualization; Automatic seeding; Flowline extraction; Opacity modulation; Vascular structure.

Student Number: 2012-30190

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Motivation

Cardiovascular disease (CVD) is a class of diseases that involve the heart or blood vessels and is the leading cause of death globally [1], with a mortality rate of 31% in 2013 [2]. Its initiation and evolution depends strongly on the blood flow characteristics [3]. Hence, the inspection of hemodynamics is necessary for the diagnosis and treatment of CVDs. In practice, blood flow is observed with catheter based examination, an invasive method that is still considered as the gold standard [4] or by a cost-effective and non-invasive ultrasound. However, these modalities are very local and require substantial user knowledge to interpret the acquired images. Recently, quantitative blood flow data have been obtained through in-vivo measurement or computational fluid dynamics (CFD) simulation. Real measurement of blood flow is achieved by 4D PC MRI (Phase Contrast Magnetic Resonance Imaging) that takes advantage of the principle of phase shift proportional to the flow speed. In addition, studies have shown that image-based patient-specific CFD models are capable of reproducing the flow structures observed in-vivo during angiographic examinations [5]. Furthermore, CFD simulation enables to study the

hemodynamics in different vessel geometries and to assess treatment options [6]. Hence, the visual exploration of blood flow data gains importance due to the increased availability of measured and simulated blood flow data [7].

In order to depict blood flow, the standard flow visualization techniques have been generally adopted. Flow visualization is the research area that visualizes fluid dynamics in a more intuitive and understandable form. The straightforward approach is the integration based visualization, which represents the flow velocity vector field by the integral lines. This method traces trajectories of massless particles released on the flow field and draws them by connecting the traced points. With recent advances in the acquisition and simulation of blood flow data, flow visualization has been widely used in the medical imaging to inspect hemodynamics in cerebral and cardiovascular vessels. The blood flow visualization techniques assist medical doctors to diagnose and treat the vessel diseases. Unlike the normal flow pattern of healthy people, blood flows showing unusual patterns imply a potential problem in vessels. The abnormal blood flow plays an important role to indicate the progressing vascular diseases such as stenosis, aneurysm and rupture since the distorted vessel induces unstable and turbulent flow around the problematic area. Therefore, the visualization technique to apparently exhibit abnormal flow is essential for a clinical use since it enables physicians to save time in diagnosing and instead to concentrate on investigating suspicious candidates of vascular deformation.

1.2 Problem Statement

There are three major issues in visualizing the blood flow in vascular structures such as vessels: seeding, tracing and rendering. Seeding is to decide the starting positions for initiating the line integration. Previous methods [8-10] usually determined seeds through user interaction. These methods require the manual location of a seeding plane as a prerequisite, thus yield different results depending on the position of the seeding plane. Hence, if this process is automated, we are able to obtain consistent result rapidly by skipping the user intervention. Tracing is to construct integral lines by starting at seed points and consecutively searching for trajectory points. However, even if a lot of seed points are assigned, many of them are early terminated especially in narrow vascular structures where they are prone to go out of the vector field. As a result, we fail to appropriately unveil the overall flow behavior. In addition, the use of dense lines for the visualization of 3D flow field causes visual clutter and line occlusion [13]. Hence, we need to devise a method to effectively convey flow features, simultaneously overcoming these problems.

Many seeding methods [8-11] for blood flow visualization have been proposed. The basic methods [8, 9] require a user to position the vessel cross section to be used as a seeding plane. Several times of user interventions may be needed until obtaining a satisfactory result through trial and errors. The more advanced method [10] presents candidate regions found by a preceding geometric analysis and allows a user

to select one or multiple regions of interests among them. Although this method reduces manual operations in a semi-automatic way, it still demands user interaction. Flow based seeding method [11] groups areas with similar flow features by a hierarchical clustering and places seed points at cluster centers. Since this method depends on flow data, the positions of seed points differ over time in unsteady flow. Therefore, the temporally changing seed points prevent users from capturing quickly the flow behavior due to a lack of frame coherence.

Using integral lines to represent a vector field is general in flow visualization. However, such a line based visualization suffers from visual clutter and line occlusion particularly in 3D field. In order to resolve these problems, line selection and opacity modulation have been proposed. Line selection methods [11-14] handle these problems by choosing more important lines from the generated lines and sparsely rendering a reduced number of lines. One simple method [12] eliminates the insignificant or overlapped lines by filtering based on line attributes such as length, winding angle, and positional similarity. The improved method [13] selects flowlines by the criterion considering view dependent line occupancy as well as line entropy. This method selectively renders flowlines which hold high entropy and occlude others less. This view dependent method is successful in avoiding line occlusion. However, it is not frame-coherent and has high computational complexity by frequent updates due to the view dependent criteria. Another class of methods [11,

14] are based on extracting the representative lines after clustering in order to cope with high visual complexity of flowlines. Pelt et al. [11] perform clustering on vector field by dissimilarity measures such as velocity, position, and the merging cost. They produce the representatives at cluster centers. Meanwhile, the method proposed by Oeltze et al. [14] classifies the generated flowlines based on the interline similarity and selects the representatives for each cluster. However, these approaches have the limitation in that a few cluster representatives do not grasp the entire flow structure, thus miss the significant flow pattern. Therefore, these methods mainly aim at visualizing an abstract depiction of blood flow.

The opacity modulation methods [15-19] generate flowlines densely and adjust the opacity of each line according to the certain properties of interests. Neugebauer et al. [16] employ the opacity modulation to correlate inversely with the velocity so that the faster laminar flow does not occlude the slow flow relevant to thrombosis of interest. Günther et al. [17] adjust the opacity of a flowline based on its screen contribution and their subsequent methods [18, 19] assign the opacity to a line segment via the optimization process pursuing a balance between information presentation and occlusion avoidance. However, these methods focus on providing as much information as possible without line occlusion rather than the important information. Lawonn et al. [15] presented the opacity assignment based on the line shape and vortex detection. This method hides the convex flowlines close to a viewer,

while showing the concave flowlines far from a viewer. Thereby, it reveals the internal vortices otherwise occluded by the exterior flowlines, concurrently preserving the surrounding lines. This method is suited to expose vortex cores in a view independent fashion, but fails to illustrate important flows with different shapes. This shape based method has a restriction in representing abnormal flows except vortices because it is impossible to detect every deformable shape of abnormal flows.

In summary, the selection based method renders lines either visible or invisible. On the other hand, the opacity modulation method depicts lines in various visibility with a varying opacity. Therefore, the former has the risk of missing the important flow features comparing to the latter. Moreover, the opacity modulation method is more desirable since it shows the normal and abnormal flow together, emphasizing the abnormal flow of major interest. For these reasons, we adopt favorably the opacity modulation. Additionally, by defining firstly the normal flow and then regarding flows dissimilar to it as abnormal, we attempt to extend the coverage of abnormal flows.

1.3 Main Contribution

The purpose of this study is to overcome problems related to the integral line based visualization i.e., the automation of seeding, the early terminated tracing, and line occlusion, and to depict hemodynamics in a more informative way for assisting the diagnosis process. By restricting our visualization to the blood flow in vascular structures, we can take into account more optimized approaches regarding aforementioned problems. In general, blood flows within the vessel, entering the inlet, exiting the outlet, and bifurcating into branches. Using this fact, we firstly identify the inlet or outlet by the relationship between blood flow and vessel geometry. Then, we generate randomly seed points on the detected inlet or outlet. Thereby, we can achieve the automatic seeding that leads to a consistent and faster flow depiction. In addition, we resolve the early terminated line integration by applying the tracing direction adaptively based on flow direction at each seed point and by performing the additional seeding near the terminated location. This solution enables to yield length-extended flowline, thus results in abundant flow visualization.

The blood flow usually follows the track guided by vessel geometry, if there is no obstacle or leak in the middle of the passage. Therefore, we can define the flow following the vessel track as normal, while the flow deviating from it as abnormal. Based on this observation, we define the representative flow for each branch by the vessel centerline and differentiate the normal and abnormal flow by the shape

matching with this vessel centerline. Afterwards, we render the normal flow transparently and the abnormal flow opaquely using a variable opacity. As a result, we can render abnormal flow more noticeable, while avoiding line occlusion at the same time. Finally, we introduce HSV (hue, saturation, value) based color coding to simultaneously exhibit flow attributes such as local speed and residence time. HSV color coding also can give a more realistic fading effect on the older particles or line segments by attenuating the saturation according to the residence time. Hence, it supports users in intuitively comprehending multiple information at once.

1.4 Organization of the Dissertation

The reminder of this thesis is organized as follows. In Chapter 2, we briefly introduce flow visualization and describe the related works for its application to the blood flow depiction. In Chapter 3, we explain the line integration method for extracting flowlines from a vector field. In Chapter 4, we present the proposed method of blood flow visualization based on flow feature and geometry analysis. Subsequently, we provide the experimental results with a comparison to the existing method in Chapter 5. Finally, we conclude the dissertation and discuss the future works in Chapter 6.

Chapter 2. Related Works

2.1 Flow and Velocity Vector

Generally, flow is described by velocity vectors. According to whether the vector field is constant or varying over time, the flow can be divided into steady and unsteady. The flow velocity vector v is typically expressed with a differential equation of the location r(t) with respect to the time t as follows:

$$\frac{dr(t)}{dt} = v(r(t), t) \tag{2.1}$$

In steady flow, since the velocity vector v is not dependent on the time t, the above equation can be reduced into a simpler form by using $v(r(t), t) = v\big(r(t)\big)$:

$$\frac{dr(t)}{dt} = v\big(r(t)\big) \tag{2.2}$$

The common way to portray flow is using an integral line. This line is obtained by numerically integrating the differential equation in (2.1) over time and by continuously connecting the traced points. Given a position r(t), a new position $r(t + \Delta t)$ at the time $t + \Delta t$ is determined by:

$$r(t + \Delta t) = r(t) + \int_t^{t+\Delta t} v(r(t), t)dt \tag{2.3}$$

These integral lines are mainly used to visualize a long term flow behavior. The representatives are streamline, pathline and streakline. Streamline represents a line that is everywhere tangent to velocity vectors at a specific time instance. Pathline is defined as the trajectory obtained by tracing a particle. Streakline is determined as a line generated by connecting particles injected successively at the same position. Streamline, pathline and streakline coincide in steady flow, but differ in unsteady flow. Flowlines will be used as the term to indicate these lines together throughout the dissertation.

2.2 Flow Visualization

Flow visualization is a well-established branch of scientific visualization [20]. Its goal is to present the behavior of flow data in a meaningful manner from which important flow features and characteristics can be easily identified and analyzed [21]. Flow visualization has been steadily researched and increasingly utilized in a rich variety of application fields i.e., automotive industry, aerodynamics, turbomachinery design, weather simulation, meteorology, climate modeling, ground water flow and medical imaging [22]. Its broad spectrum of applications prove its high level of usability. A large range of techniques have been developed accordingly with various

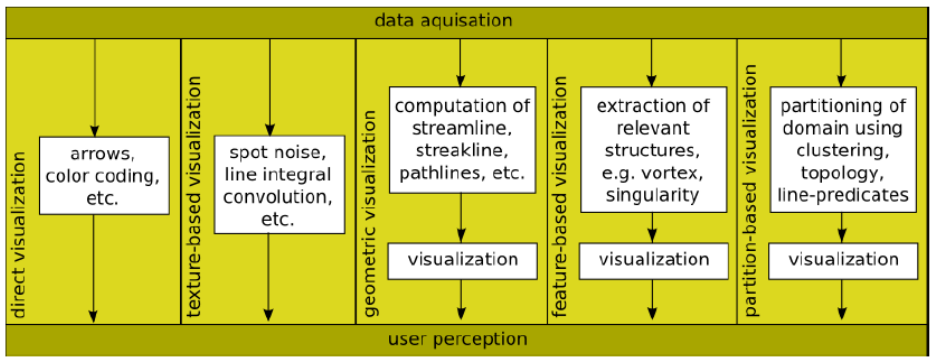


Figure 2.1. Classification of flow visualization techniques [24].

applications. Hence, rather than enumerating relevant techniques individually, we'll introduce in a categorized form.

The most widely accepted categorization suggested by Post et al. [23] classified flow visualization techniques into direct, texture-based, geometric, and feature-based techniques according to several factors such as data type, the dimensionality of the domain, and the information level such as elementary, local, global. Afterwards, Salzbrunn et al. [24] added the class of partition-based techniques. In this categorized order, the abstraction increases as shown in Fig. 2.1. Direct methods are the simplest and computationally inexpensive techniques in the sense that the flow data is directly mapped to graphical representation. They are implemented by placing an arrow to indicate the velocity vector at each grid point or by mapping color to velocity magnitude [see Fig. 2.2]. However, these methods are improper to

express the vector field in dense grids due to visual complexity of overlapped arrows. Texture-based techniques ensure a dense representation of the flow in higher spatial resolution by warping noise textures according to local flow. The widely used method is Line Integral Convolution (LIC), first introduced by Cabral et al. [25], that convolves noise texture along the vector direction [see Fig. 2.3]. Texture-based method is effective for two-dimensional, but has some problems such as clutter and occlusion in three-dimensional domain.

For a better depiction of the long term behavior induced by flow dynamics, geometric methods integrate flow data and use geometric primitives as a basis for flow visualization [22]. Although generating as many flowlines as possible is desirable to avoid missing important flow features, this is susceptible to visual clutter and line occlusion [see Fig. 2.4]. As an alternative to resolve these problems, feature-based methods represent the flow sparsely by visualizing features extracted from the vector fields [see Fig. 2.5]. The widely used flow features are vortices, vortex core lines, critical points, and shock waves. These features are calculated by vector operations such as divergence, curl, Jacobian matrix. Feature-based techniques are suitable for depicting large data due to its sparse representation, yet may demand high computational complexity to extract features before visualization.

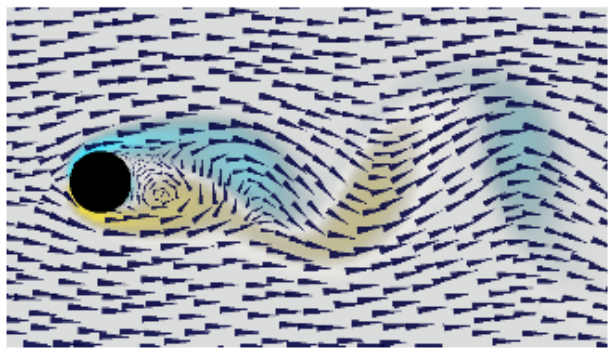
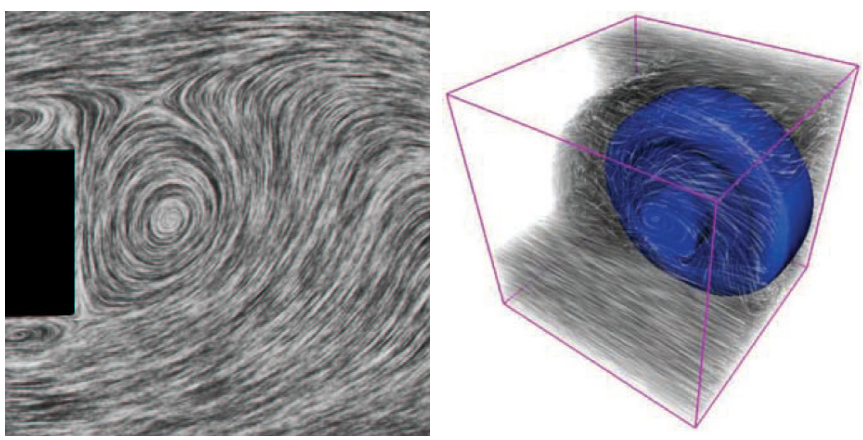


Figure 2.2. An example of direct flow visualization method [26]. The arrowhead at each grid point indicates the flow direction and magnitude. The color representes the rotation direction of the flow, the clockwise (blue) and the counter clockwise (yellow).



(a) (b)

Figure 2.3. Examples of texture-based methods. (a) Line intgeral convolution (LIC) is used to depict 2D flow around a block [22]. (b) LIC is used to show a simulation of 3D flow around the wheel [27].

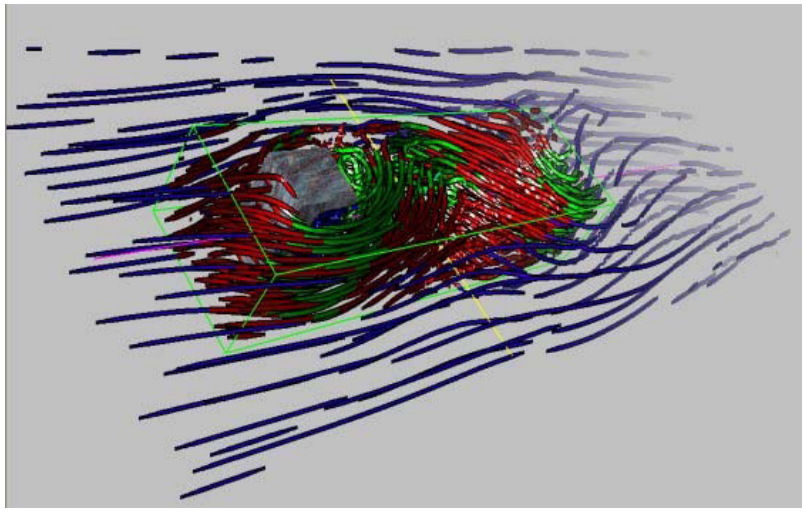


Figure 2.4. An example of geometric flow visualization [28]. Flow around block is illustrated by flowlines. Color is used to enode the pressure.

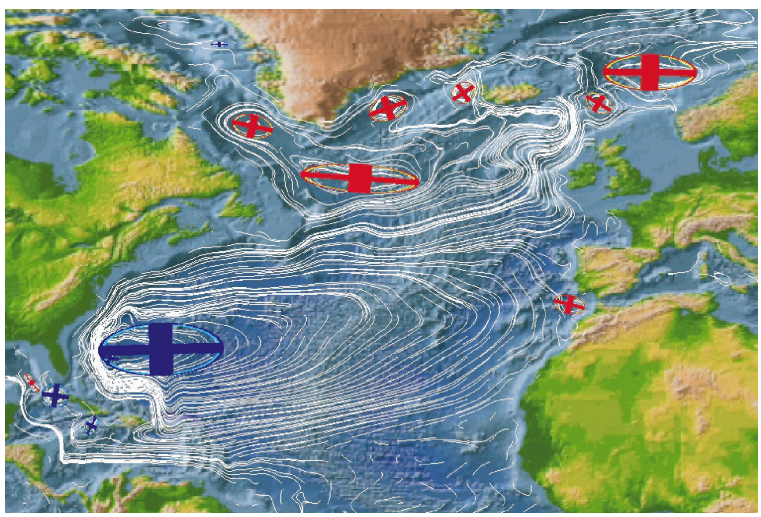
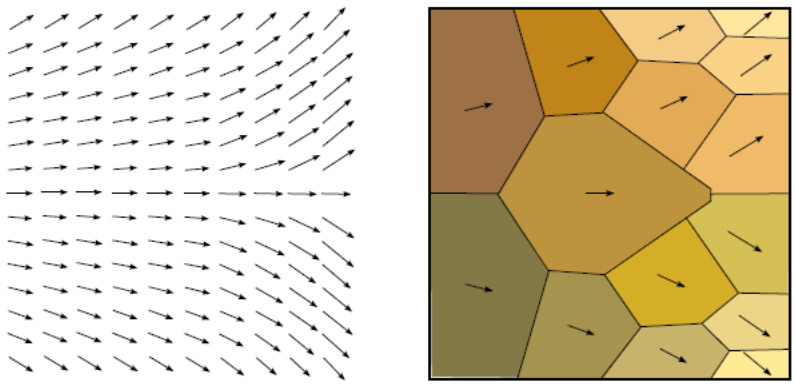


Figure 2.5. An example of feature-based techniques [29]. Flow in the Atlantic Ocean is visualized by streamlines and ellipsoids reprensenting vortices. Color is used to indicate the rotation direction of the vortices i.e., clockwise (red) and counter clockwise (blue), respectively.



(a) (b)

Figure 2.6. Clustering of a flow field [24]. (a) The flow field represented by a direct method. (b) An abstract flow depicition by partition-based method.

Partition-based approaches aim at effectively partitioning the spatial or spatiotemporal domain according to flow properties. The partition procedure comprises computation of a similarity measure and extraction of the representatives. As suggested by Salzbrunn et al. [24], these methods can be divided into two approaches according to whether the partition is based on the velocity vectors or the generated integral lines as shown in Fig. 2.6. Cluster based methods group flow regions with similar vectors by the similarity measure on vector velocity and position. Integral line based methods cluster the flow field according to similar behavior of flowlines. These approaches provide visually simplified results. However, partitioning is computationally expensive especially for larger data sets, thus may be a bottleneck in visualizing the overall flow.

Summarizing, texture based methods enable a dense coverage of the flow field over the direct method. Geometric methods makes it possible to show a long-term flow behavior. Feature-based approaches focus on depicting merely flow features of interest or importance. And partition-based techniques are successful in illustrating flow abstraction.

2.3 Blood Flow Visualization

The blood flow visualization methods have been developed by adopting general flow visualization techniques adaptively for specific purposes such as diagnosis and treatment planning. Therefore, by classifying the existing methods according to the aforementioned categorization of flow visualization, we attempt to obtain useful insights into future works as well as previous approaches.

2.3.1 Geometric Method

Pelt et al. [8] proposed blood flow visualization method using the interactive placement of a seed plane. The seeding process is conducted by two-step user interaction i.e., placing a seed plane and selecting a seeding template. The selected

vessel cross-section is used to seed streamlines for flow inspection. Therefore, in order to capture the overall flow behavior, several times of moving a seed plane may be required. Also, they presented various color coding methods to convey speed, to discriminate the flow from seeding planes, and to show line propagation. In their second paper [9], they presented the improved seeding approach to depict blood flow in the heart which is non-vascular structure. They introduced a virtual probe as the seeding basis, which was automatically aligned with the orientation of the flow field. They also presented visualization techniques including particle traces and halo enhanced pathlines.

Born et al. [30] proposed a line predicate approach that sorts integral lines into bundles with similar properties. Line predicate is a Boolean function that evaluates to either true or false depending on whether a streamline meets a certain characteristic. They introduced several line predicates that enabled to visualize the flow according to various features such as velocity distribution, vortices, and flow paths. Therefore, through a flexible combination of these predicates, users could create flow structures of interests.

Neugebauer et al. [10] proposed a seeding method for inspecting near-wall hemodynamics. This method extracts cap-shaped surfaces that resemble bleb formations. These surfaces are identified automatically by a curvature-based energy measure using the shape index and curvedness. Based on these regions, circular

seeding regions are defined with seed points uniformly distributed underneath for a subsequent streamline integration.

Gasteiger et al. [31] introduced a ghosted view approach that displays the vessel surface and reveals the blood flow by applying a view-dependent transparency. They assigned more opacity to regions facing away from a viewer and more visibility at regions facing to a viewer. This opacity adjustment for vessel surface enables to show internal flows better. Additionally, in order to increase depth perception between overlapped vessel parts, they employed shadow casting and atmospheric attenuation.

Lawonn et al. [32] improved Gasteiger's ghosted view approach by modifying the vessel opacity of the front faces by means of the suggestive contour measure. Furthermore, they utilized depth blurring instead of atmospheric attenuation for more natural depth perception. They extended adaptive surface visualization approach to pathline animation by emphasizing nearby surface regions.

2.3.2 Feature-based Method

Born et al. [33] aimed at an intuitive and simple visualization. So, this method tried to look for a minimal number of lines representing all features by extracting the representative lines from the topologically correct skeleton. They also identified

vortices as the important flow feature and displayed them. The process for finding the representative lines is performed through three following steps. Each line bundle is voxelized and subsequently smoothed by a Gaussian filter. From the resulting voxel mask, the skeleton is generated by topology-preserving thinning. Streamlines are identified as the representatives if they cross the skeleton voxels with a certain amount of crossings. The representative streamlines are drawn by fused stream-tapes and color coding of the flow speed. Moreover, the vortex core lines are visualized using tube-like structures with hatching textures.

Lawonn et al. [15] proposed a combination of view-dependent attenuation of all streamlines and concurrently view-independent emphasis of important features, vortices. This method distinguishes between convex and concave streamlines according to the current viewpoint and set convex streamlines transparent and concave lines opaque. Additionally, they extracted vortices as the important feature and rendered them opaquely. Thereby, they could emphasize vortices, otherwise occluded by the external streamlines.

2.3.3 Partition-based Method

Pelt et al. [11] presented a sparse representation of the flow field based on a spatiotemporal hierarchical clustering. The clustering employs a bottom-up

approach consistently generating the hierarchy. Iterative merging of individual clusters leads to the tree nodes based on the dissimilarity measures between neighboring clusters. They performed the clustering by two dissimilarity measures. The one is elliptical dissimilarity measure comparing the averaged velocity and position. And the other one is a local linear expansion defined by the cost to merge two clusters into a new cluster. They showed the clustering result by representative patharrows seeded at the cluster centers.

Oeltze et al. [14] presented a fully automatic approach to provide visual summary of flow patterns by clustering streamlines and computing cluster representatives. For this, they employed spectral clustering. Spectral clustering maps the original streamlines to a spectral embedding space where each line is represented by a point. This clustering can be formulated as a graph partitioning problem. Streamlines are represented by a weighted, fully-connected, undirected graph. The nodes are the streamlines and the edge weights are calculated according to the mean of closest point distance. The weights are then transformed such that similar streamlines have a high weight while dissimilar lines a low weight. Next, the graph is partitioned into sub-graphs by minimizing the sum of weights of edges to be removed and at the same time balancing the sum of edge weights of the partitions. After clustering, the representative is determined by the streamline corresponding to the point closest to the cluster centroid.

In summary, geometric based methods mainly deal with seeding, color-coding of flowlines, and vessel surface rendering. Seeding method has been gradually improved from the manual method of positioning a seed plane to the semi-automatic way of presenting seeding candidates. And adaptive surface visualization methods tried to show the internal flowlines better by applying view-dependent transparency to vessel surfaces. Feature-based methods defined the vortices as the important features. So, they displayed the vortices dominantly over other flowlines. Partition-based methods focused on abstract flow depiction. Vector-based clustering showed local flow information, whereas integral line based clustering represented long-term flow patterns covering the entire vector field.

Geometric methods enable to portray long-term flowlines in 3D vector field, but suffer from line occlusion and visual clutter. In order to solve these issues, the straightforward approach is to select more important lines from the generated lines and sparsely render a reduced number of lines. This is achieved by selecting flowlines with features of interests or by showing the representatives after clustering. However, feature-based method has a small coverage of important flowlines because of a definition limited to vortices. Partition-based method is mainly for flow summary and abstract depiction, so this method is prone to miss important features.

Chapter 3. Integration based Flowline Extraction

Most of blood flow visualization methods utilize primarily the integral lines as a basis for graphical representation to depict the flow vector field in feature-based and partition-based as well as geometric methods, although a few approaches [33, 34] use the integral surfaces. In this chapter, we introduce the process to generate the integral lines from the vector field in more detail. This procedure is composed of four processing steps; seeding, cell searching, vector calculation and the advection to the next point. For ease of explanation, we describe the line construction with regard to streamlines. The same principle can be applied to extract pathlines or streaklines except for the use of vectors adjacent temporally as well as spatially. Streamlines are generated from only one time step of flow data, while pathlines and streaklines are created by employing multiple time steps of flow data, also requiring the temporal interpolation of velocity vectors.

3.1 Overview

We extract flowlines from the velocity vector field as follows. Starting at a seed point, we find the cell containing this point. After identifying the host cell, the vector

at the point is interpolated with velocity vectors at irregular grids. Then, the position of its next point is calculated by the numerical integration in (2.3). This procedure is repeated by continuously adding a new point to the flowline until the flowline reaches a physical boundary or the number of time steps exceeds a predefined limit [35]. At last, when the line integration is completed, the flowline is generated by sequentially connecting the seed point and its subsequent ones. The above process is equally applied to all seed points, yielding one flowline at each seed point. Cell searching and vector interpolation are challenging especially in unstructured grids, since these tasks greatly govern the overall computational complexity and accuracy. Therefore, we employ the tracing algorithm [35-38] that is currently proven to be efficient in the tetrahedral grids. In the following sections, we present a detailed description on each step of tracing algorithm.

3.2 Seeding

Seeding is to determine the starting point of line integration. Hence, seeding should be firstly performed to initiate line tracing. Seeding involves two major steps: locating a seeding plane and generating seed points on that plane. In general, the seeding plane is chosen by a user because it is difficult to locate a suitable seed plane,

while seed points are generated automatically by random sampling. In particular, when visualizing blood flow in vascular structures, the vessel cross-section is commonly employed as a seeding plane.

3.3 Barycentric Coordinate Conversion

The core problem in all particle tracers is: given an arbitrary point, to seek the host cell including this point and to calculate the velocity vector at the point [38]. In general, barycentric coordinates is used to decide if a point lies inside a tetrahedron, and to interpolate a value within a tetrahedron because the use of barycentric coordinates greatly simplifies and accelerates these operations [39]. Hence, we take advantage of barycentric coordinates like that used in [35, 38] to efficiently perform cell searching and vector calculation for flowline construction.

Given a tetrahedron composed of four vertices r_1, r_2, r_3, and r_4, a point r located inside this tetrahedron can be written by:

$$r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3 + \lambda_4 r_4, \tag{3.1}$$

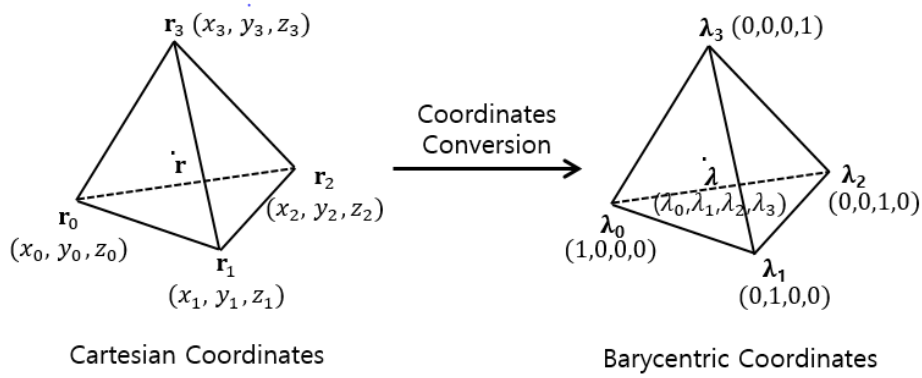


Figure 3.1 Cartesian to barycentric coordinate conversion.

where $\lambda_1, \lambda_2, \lambda_3,$ and λ_4 are the barycentric coordinates of the point r with respect to a tetrahedron, satisfying two conditions of $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$ and $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$ [see Fig. 3.1]. By substituting $\lambda_4 = 1 - \lambda_1 - \lambda_2 - \lambda_3$ and $\mathrm{r_i} = (x_i, y_i, z_i)^T$ into the above equation, the equation for converting Cartesian coordinates to barycentric coordinates can be formulated as [39]:

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = T^{-1}(\mathrm{r} - \mathrm{r}_4),$$

$$\lambda_4 = 1 - \lambda_1 - \lambda_2 - \lambda_3,$$

$$\text{where } T = \begin{bmatrix} x_1 - x_4, & x_2 - x_4, & x_3 - x_4 \\ y_1 - y_4, & y_2 - y_4, & y_3 - y_4 \\ z_1 - z_4, & z_2 - z_4, & z_3 - z_4 \end{bmatrix}. \tag{3.2}$$

r_1-r_4, r_2-r_4, and r_3-r_4 are linearly independent because four vertices of a tetrahedron, r_1, r_2, r_3, and r_4 are not coplanar. Hence, the matrix T is invertible and the barycentric coordinates can be calculated by Eq. (3.2).

3.4 Cell Searching

It is not a straightforward task to identify the host cell containing a given point. We use an algorithm presented in [36] to solve a cell searching problem. Firstly, we compute the barycentric coordinates of a given point with respect to the current tetrahedron by using Eq. (3.2). If its barycentric coordinates are within the range between 0 and 1, this point is located inside the current tetrahedron and the cell searching is completed. Otherwise, the searching continues in the next cell adjacent to the face opposite to the vertex with the minimum coordinate. This next candidate cell is decided by referring to the neighbor list that defines the neighbors of each cell based on its shared face. This process iterates over all cells until the host cell is found. In case of no found cell, it means that the point is outside all cells. Accordingly, the line integration finishes. Meanwhile, when the host cell is identified, the velocity vector at a given point is calculated.

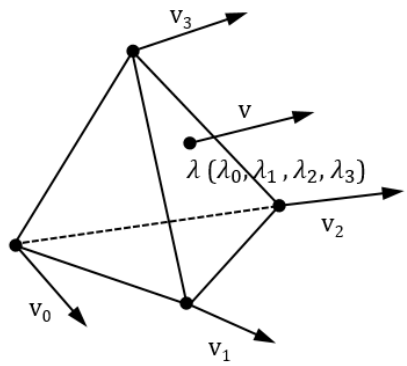


Figure 3.2. Velocity vector calculation.

3.5 Velocity Vector Calculation

Since the velocity vectors are defined only at unstructured tetrahedral grids, the vector at an arbitrary position should be spatially interpolated with vectors at its adjacent grids. As shown in Fig. 3.2, the barycentric coordinates of a given point in the tetrahedron reflect the proportion between the volumes of four sub-tetrahedrons defined by that point and three points of each triangular face. Hence, the velocity vector can be efficiently calculated by reusing as weights the barycentric coordinates obtained in the preceding cell searching as follows:

$$V = \lambda_1 V_1 + \lambda_2 V_2 + \lambda_3 V_3 + \lambda_4 V_4 \tag{3.3}$$

where $\lambda_1, \lambda_2, \lambda_3$, and λ_4 are the barycentric coordinates. V_1, V_2, V_3, and V_4 denote the velocity vectors at four vertices of the host tetrahedron.

To construct pathlines and streaklines in unsteady flows where the velocity vector field changes over time, velocity vectors should be temporally interpolated. The velocity vector $\mathrm{v}(\mathrm{r}(\mathrm{t}), \mathrm{t})$ at the time t and the spatial position r(t) is obtained by a linear interpolation between two closest time instances t_l and t_{l+1} containing the vector field as follows:

$$\mathrm{v}(\mathrm{r}(\mathrm{t}), \mathrm{t}) = (1 - \delta)\mathrm{v}(\mathrm{r}(\mathrm{t}), t_l) + \delta\mathrm{v}(\mathrm{r}(\mathrm{t}), t_{l+1}),$$

$$\text{where } \delta = (t - t_l)/(t_{l+1} - t_l). \tag{3.4}$$

Therefore, in case of the time-varying flow, the velocity vectors used in Eq. (3.3) should be replaced by vectors interpolated temporally.

3.6 Advection

Since the current point and its velocity vector are known, the next point of a flow path is approximately computed by numerical integration methods. The simplest approach is the first-order Euler method. However, this method is generally not used

because it is inaccurate and error-prone. Instead, the fourth-order Runge-Kutta method [40] has been widely exploited since it is more accurate and stable. This method employs not only the vector at the current position but also three additional vectors at positions consecutively predicted by the proceeding vectors. These vectors are incorporated to obtain the next position as follows:

$$v_1 = v(r(t), t),$$

$$v_2 = v(r(t) + v_1 \frac{\Delta t}{2}, t + \frac{\Delta t}{2}),$$

$$v_3 = v(r(t) + v_2 \frac{\Delta t}{2}, t + \frac{\Delta t}{2}),$$

$$v_4 = v(r(t) + v_3 \Delta t, t + \Delta t),$$

$$r(t + \Delta t) = r(t) + \frac{1}{6}(v_1 + 2v_2 + 2v_3 + v_4)\,\Delta t \qquad (3.5)$$

where r(t) is the position of the traced point at the time t and v(r(t), t) is the velocity vector at the specified position r(t) and time t. $v_1, v_2, v_3,$ and v_4 represent vectors at the start position, two predicted mid positions, and the predicted final position respectively. Here, in order to obtain three vectors, v_2, v_3, v_4 at the predicted positions, cell searching and vector calculation should be repeatedly carried out for each vector.

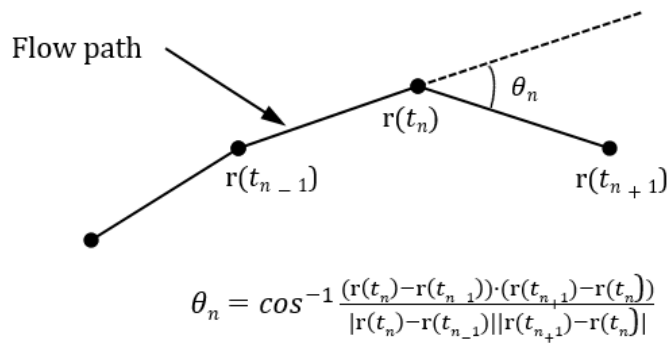


Figure 3.3 Step size adaptation is based on the angle of line segments at successive points along a path [38].

3.7 Step Size Adaptation

If the integration step size is fixed to a constant value, a flowline may not properly trace the flow around bifurcations or strongly curved vessels where the flow direction changes rapidly. This can be prevented by exploiting an adaptive step size control scheme where the integration step size is changed according to an error tolerance [38]. The error tolerance is evaluated by the difference between two destination points found after advecting respectively by a step size Δt and by two steps of its half size Δt/2. The half size is used when the distance between two end-points is larger than a predefined tolerance. However, this technique was too expensive because it unconditionally performs the tracing twice at every point location.

Darmofal et al. [41] and Kenwright et al. [38] have proposed more efficient methods that perform selectively the tracing of a half step size in the interval where a flow direction changes rapidly. They estimated the change of flow direction by the angle of velocity vectors [41] or by the angle of line segments [38] at successive points along a path as shown in Fig. 3.3. We employ the latter method for our implementation.

Chapter 4. Blood Flow Visualization using Flow and Geometric Analysis

In this chapter, we present an approach for the extraction and depiction of flowlines in the ascending aorta and coronary arteries as shown in Fig. 4.1. The aorta is the main artery in human body, originating from the left ventricle of the heart. The ascending aorta is the beginning part of an aorta located between aortic valve and aortic arch. Coronary arteries are composed of three main coronary arteries (RCA, LAD, and LCX) and their small branches. The figures presented throughout the dissertation were obtained from this dataset.

Our approach consists of four processing steps; preprocessing, seeding, tracing, and rendering, as shown in Fig. 4.2. In the preprocessing, we extract the neighbor list and surface meshes from the indexed tetrahedrons given as an input dataset. Afterwards, we detect the inlet or outlets of blood vessels and then place seed points on them. Next, a line integration is performed incorporating the complementary means to prevent the early termination. Finally, opacity-modulated flowlines are rendered overlaying vessels.

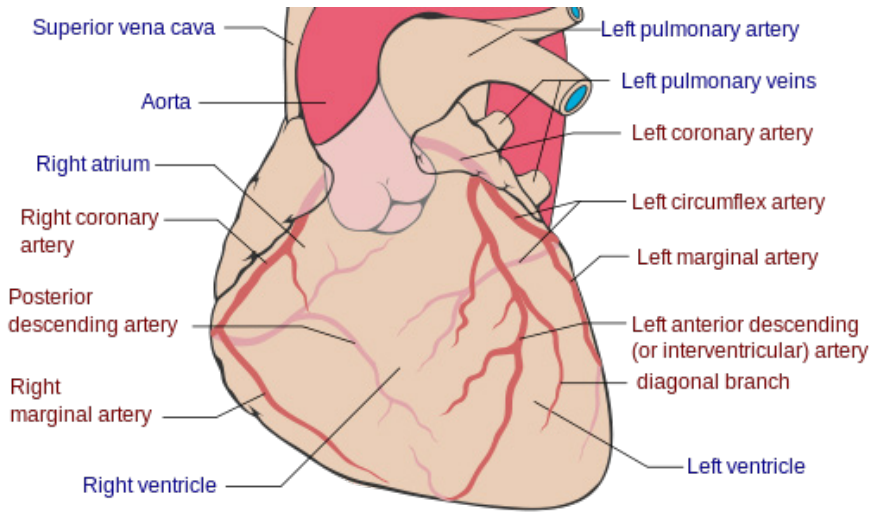


Figure 4.1 Aorta and coronary arteries [42].

4.1 Preprocessing

Blood flow within vessels is described by velocity vectors contained in the indexed tetrahedron set, which stores an array of vertices and encodes tetrahedrons as sets of indices into this array [43]. Here, each vertex contains its 3D coordinates and velocity vector. In the preprocessing step, when loading such a flow dataset, we prepare neighbor list and surface mesh for later use. Firstly, we make the neighbor list for each tetrahedron by finding other tetrahedrons sharing its faces. The tetrahedron located inside has neighbors at its all faces, whereas the boundary one has at least one non-shared face. Hence, surface meshes can be extracted by

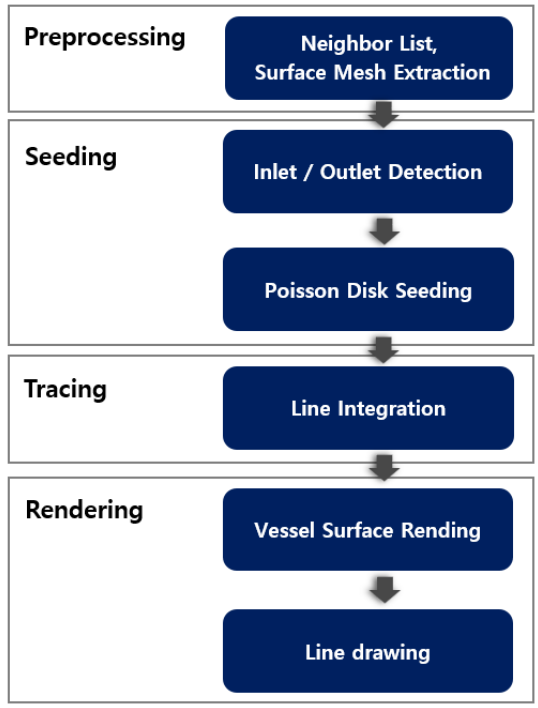
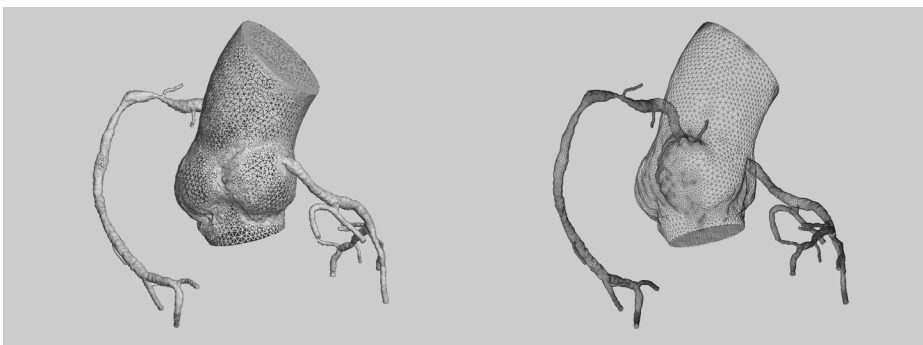


Figure 4.2 The proposed method consists of four processing steps; preprocessing, seeding, tracing, and rendering.

collecting non-shared faces only. The extracted data are utilized to efficiently gain the local geometric information needed for the subsequent processing steps such as seeding, tracing and rendering the vessel surface. Fig. 4.3 depicts the volume mesh filled with tetrahedrons and its extracted surface mesh which are rendered in a form of wire frame.



(a) (b)

Figure 4.3 Dataset of ascending aorta and coronaries. (a) Volume mesh consists of tetrahedrons. (b) Surface mesh is extracted from the volume mesh in the preprocessing step.

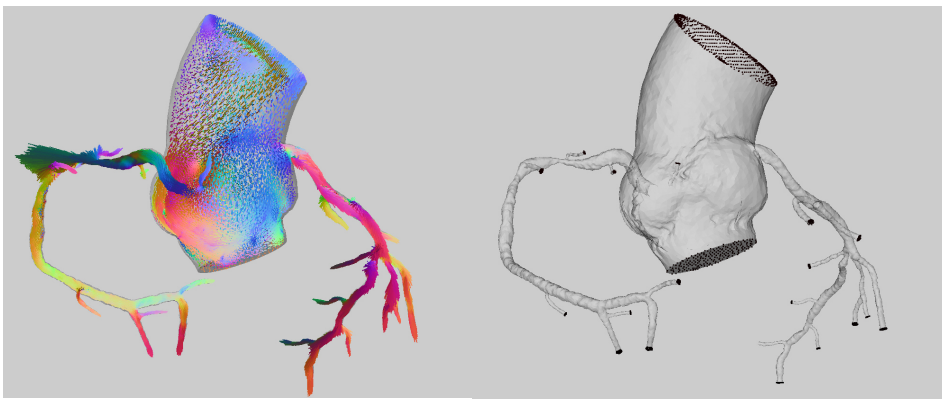
4.2 Inlet or Outlet based Seeding

In general, vessel surface mesh can be divided into three parts, i.e., the inlet, outlet and wall. The inlet or outlet is the input face or output face that blood enters or exits from a given vessel. From the observation that blood flow passes through the inlet or outlet, but is blocked by vessel wall, we get a hint to separate the inlet or outlet from vessel wall. The flow velocity vector and the face normal vector are perpendicular to each other in vessel walls, while not perpendicular in the inlet or outlet. Therefore, we are able to identify all surface vertices as the inlet or outlet and wall ones by the orthogonality metric between these vectors as follows:

$$C_i = \begin{cases} inlet\ or\ outlet, & if\ \left|\frac{V_i}{|V_i|} \cdot N_i\right| > \theta \\ wall, & otherwise \end{cases}, \tag{4.1}$$

where V_i and N_i denote the velocity vector and the outward normal vector at the i-th vertex of surface mesh respectively. θ is the threshold for orthogonality decision, in which we use 0.7 approximately corresponding to 45 degrees as the angle between two vectors. The inlet or outlet vertices are classified into clusters by performing k-means clustering algorithm. Afterwards, we find the best fitting plane for vertices of each cluster by minimizing their orthogonal distance from to the plane. As a plane is defined by one point on it and its normal vector, the fitted plane is specified by the cluster centroid and the eigenvector corresponding to the minimal eigenvalue of the covariance matrix of vertices [44]. Fig. 4.4 illustrates the detected inlet or outlet vertices.

We employ Poisson disk sampling [45] as presented by Sobel et al. [46] to place seed points on the plane estimated from the inlet or outlet vertices. This method ensures a uniform distribution of seed points by generating all points at least the predefined disk radius apart. The number of seed points is relevant to the ability of visualizing densely flow behavior. Hence, dense seeding with a smaller radius is required not to miss important flow characteristics. However, since the calculation time is directly proportional to the number of seed points, the disk radius needs to be



(a) (b)

Figure 4.4 Inlet or outlet detection for automatic seeding. (a) flow velocity vectors at all vertices. (b) The inlet or outlet vertices detected by the orthogonality metric between flow velocity vector and face normal vector.

determined considering a trade-off between quality and performance. As a breakthrough solution, we adjust locally the seeding density according to the vector consistency. To apply this strategy to Poisson disk seeding, instead of a fixed radius, we exploit a variable radius depending on the vector variance that is calculated from velocity vectors at four vertices of the host tetrahedron including the current point. A small radius is used in area with high vector variance, while a large one in area with low variance. As a result, vector variance based seeding has the effect that more seed points are placed in the region of a large variance. Consequently, Poisson disk seeding with a variable radius makes it possible to exhibit the overall flow behavior with a limited number of seed points by allocating more in dynamic and less in

homogeneous area. In order to apply the same seed points to every flow data during a cardiac cycle, the vector coverage for variance calculation is extended temporally as well as spatially, or restricted to the flow data in diastole with the peak vector variance.

Unlike the existing methods that demand user placement of a seeding plane, our seeding method is automatic in that it does not require user intervention to determine seed points. The conventional methods cannot visualize the blood flow until the user locates the seeding plane within the vessel. However, our method allows to depict blood flow immediately on loading flow data by employing the detected inlet or outlet as a seeding plane without such a manual operation. In addition, the inlet or outlet based seeding also enables to investigate the branching behavior of blood flow by classifying flowlines according to their seeded inlet or outlet. Although the proposed seeding method was initially developed for the ascending aorta and coronary arteries, it can be universally applied to other vessels as well since our seeding strategy is based on the inlet and outlet that every vessel contains.

4.3 Tracing

4.3.1 Flow based bidirectional tracing

Inflow is expected in the inlet of the ascending aorta connected to the heart that pumps blood. However, in practice, blood flow is not unidirectional in the ascending aorta near aortic valve because it is locally reversed due to the cardiac cycle changing from systole to diastole. If we apply forward tracing to seed points on the inlet, we'll fail to generate flowlines at seed points with outflow since the traced points go beyond the vessel boundary. In addition, the number of generated flowlines is varying during cardiac cycle due to a difference in the prevalence of reversed flows between systole and diastole. Thus, when initiating the tracing in the ascending aorta under consideration, we perform adaptively forward or backward tracing according to the flow direction. Since the inlet or outlet plane is oriented outward, we can discern inflow from outflow based on the directional correlation between flow vector and normal vector. Therefore, the flow direction F_i at the i-th seed point is determined as inflow or outflow by:

$$F_i = \begin{cases} inflow, & if\ V_i \cdot N_i < 0 \\ outflow, & otherwise \end{cases}, \tag{4.2}$$

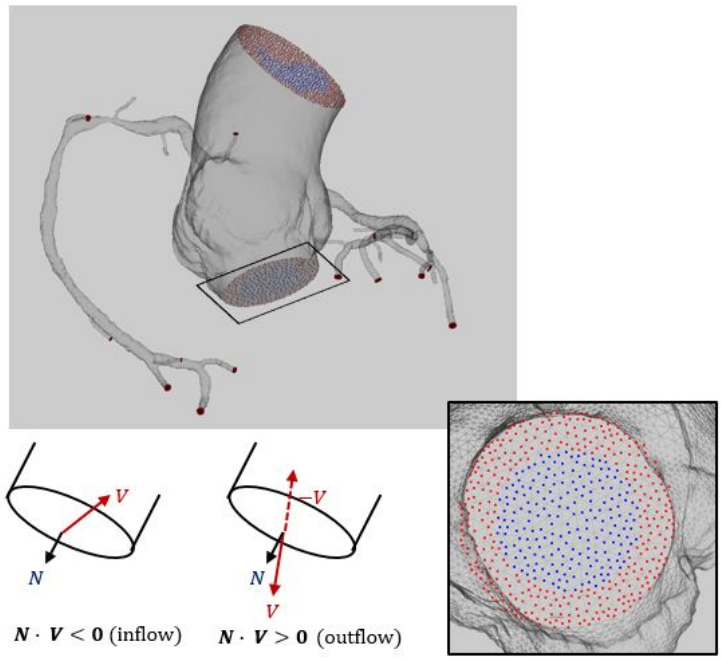


Figure 4.5 Flow based directional tracing adaptively applies the tracing direction according to the flow direction at each seed point, i.e., forward tracing to the inflow points (blue) and backward tracing to the outflow points (red).

where V_i and N_i are the velocity vector and the outward normal vector respectively. These vectors are interpolated from vectors stored at four vertices of the tetrahedron by using as weights the barycentric coordinates of a seed point, as described earlier in Section 3.5. The sign of inner product of these vectors indicates the flow direction, i.e., inflow or outflow, as shown in Fig. 4.5. Our method applies forward tracing to inflow points by integrating flowlines with a positive time step. On the other hand, it conducts backward tracing at outflow points using a negative time step. This flow

based directional tracing enables the line construction at all seed points and additionally allows to produce flowlines consistently during cardiac cycle.

4.3.2 Additional seeding for length extended line integration

We construct flowlines by starting at seed points and consecutively connecting trajectory points traced by a fourth order Runge-Kutta scheme [40], as explained in Section 3.6. This procedure terminates generally when the flowline goes out of the vector field or approaches the critical points with zero vector, or when the number of time steps exceeds a predefined limit [35]. Particularly in narrow vascular structures like branch vessels, early termination frequently occurs, which leads to short flowlines compared with the vessel length. In order to prevent early termination, we relax the termination conditions by sufficiently increasing the limit of time step and by stopping the line integration before going beyond the vessel boundary. Since the flow vectors are available only within the vessel, a further integration is impossible when exceeding vessel boundary. Thus, restricting integration to the vessel inside is of crucial importance in integration based methods, until reaching the other end of the vessel passage. This is accomplished by pausing the advection when the advancing direction points to the vessel outside after arriving at the vessel boundary cell during the cell searching. Nonetheless, this relaxation does not

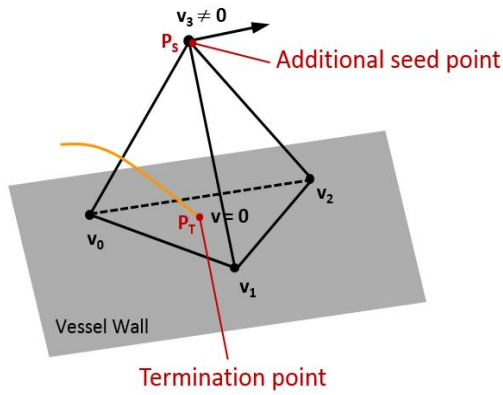


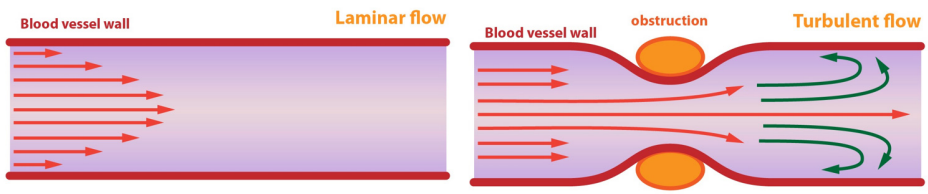
Figure 4.6 Additional seeding is performed at the vertex with nonzero velocity vector among four vertices of the tetrahedron including the termination point when the line integration terminates due to zero vector.

completely resolve the early termination problem because it is primarily caused by flowlines trapped in zero vector that makes them stay at the same position. This is typical in thin branch vessels where flowlines are prone to arrive at arterial walls with zero vector. In simulation, blood flow is assumed to stop and stay at vessel walls with zero vector. However, this is impossible in reality since the subsequently following flow pushes the preceding flow. Therefore, we attempt to keep a stationary flow moving in a similar way to this situation. To escape from such a standstill, we add a new seed point and restart the line integration at the vertex with nonzero vector among four vertices of the tetrahedron containing a termination point as depicted in Fig. 4.6. Since at least one vertex of the boundary tetrahedron is located inside the vessel, we are able to seek the vertex with nonzero vector. Hence, anywhere within

vessel, we can exploit the additional seeding as a solution to avoid the termination due to zero vectors. Instead, some care must be taken. We perform the additional seeding only when the line integration terminates abnormally, i.e., in the middle of the passage. The abnormal termination is determined by the terminated location. For example, if the termination occurs at vessel walls identified by the equation in (4.1), it is regarded as abnormal.

4.4 Opacity Modulation

In order to visualize blood flow information, we adopt the strategy generating flowlines densely and instead showing important ones opaquely and the others transparently. Therefore, the method to evaluate the importance of flowlines is firstly needed. In general, abnormal flow is significant because it is formed by the deformed vascular structure due to vessel diseases. Blood flow can be assessed as normal or abnormal in terms of laminar or turbulent flow as shown in Fig. 4.7. If the flow runs unidirectionally in parallel with the boundary wall without disruptions, it is assessed as laminar. On the contrary, if the flow is unpredictably irregular and random in direction as well as velocity, it is regarded as turbulent. In healthy vessels, laminar flow is observed, while turbulent flow in pathological vessels. Therefore, in blood



(a) (b)

Figure 4.7 Blood flow patterns in the vascular structures [48]. (a) Laminar flow is observed in the healthy vessel. (b) turbulent flow is found in the pathological vessel.

flow visualization for diagnosing the cardiovascular pathologies, the key is to distinguish the laminar and turbulent flow and to express the turbulent flow more noticeably so that the users pay more attention on such an unstable flow. The vessel centerline shows the shape similar to vessel geometry since it is directly derived when extracting the vessel for CFD simulation. Therefore, we employ the vessel centerline as a reference to separate into laminar and turbulent flow. As a result, we can determine flowlines as normal or abnormal by comparing the shape between flowlines and the vessel centerline.

By utilizing the tubular structure of vessels, the centerline is extracted by the tracking algorithm based on Bayesian sided test with branch point detection [47]. For computational simplicity in CFD simulation, the vessel morphology is assumed not to change during the cardiac cycle although blood vessels are non-rigid structures slightly transformed by a regular heartbeat. Therefore, the vessel centerlines are

initially extracted once from the vessel geometry in a flow independent fashion and need not be updated for every flow data acquired at each phase of the cardiac cycle.

4.4.1 Global opacity

In order to determine the opacity of each flowline to be rendered, we compute the interline similarity between flowline and the vessel centerline. For this purpose, we make use of the mean of closest point distances less sensitive to the line length, as presented in [49]. Here, we reformulate the distance metric by exploiting 3D coordinates of vertices displaced by the starting position of each line rather the original ones to minimize the influence of spatial distance as follows:

$$d(\mathrm{L}^c, \mathrm{L}^f) = mean_{\mathrm{p}_i^f \in \mathrm{L}^f} \min_{\mathrm{p}_j^c \in \mathrm{L}^c} |(\mathrm{p}_j^c - \mathrm{p}_0^c) - (\mathrm{p}_i^f - \mathrm{p}_0^f)|, \tag{4.3}$$

where L^f and L^c represent the flowline and the vessel centerline with points p_i^c and p_j^c respectively. p_0^f and p_0^c denote the starting point of each line. The distance is mapped to the opacity, thus focusing on flowlines more dissimilar to the vessel centerline. We render flowlines by attenuating the opacity according to the interline similarity in an exponential manner as follows:

$$\alpha_g = e^{-\frac{(D - d(\mathrm{L}^c, \mathrm{L}^f))^2}{\sigma^2}}, \tag{4.4}$$

where D is a half diameter of the inlet or outlet, which is calculated by the distance between the centroid and its farthest vertex in each cluster of inlet or outlet vertices detected during the seeding process. σ is a parameter to control how fast the opacity decreases. As a consequence, we are able to render the flowlines resembling the representative line transparently and the different ones opaquely.

4.4.2 Local opacity

However, since the global opacity modulation assigns equally a single opacity to all line segments, it may give a high opacity to line segments aligned in parallel with the vessel centerline. This leads to false interpretation unlike our intention. Hence, it is more desirable to assign the opacity on a line segment basis by the local interline similarity. Moreover, the local opacity modulation enables an intensive depiction of abnormal flows by narrowing the target of opaque rendering from the entire line to specific line segments showing unusual flow patterns.

Local opacity is determined by the shape similarity between two line segments at the flowline point and its corresponding centerline point. To calculate the shape similarity at each flowline point, we firstly look for its closest centerline point based on the Euclidean distance. The interval between consecutive points is consistent in vessel centerline, but irregular in flowlines due to the flow speed. Therefore, the

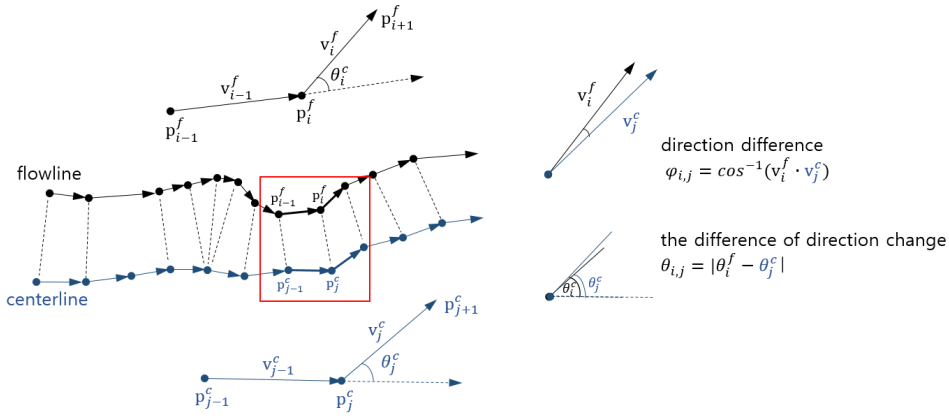


Figure 4.8 The shape similarity between flowline and ceterline segment is calculated on a point basis by the difference of direction vectors, $\varphi_{i,j}$ and that of direction changes, $\theta_{i,j}$.

same centerline point can be selected as the closest one from several flowline points with very slow speed, as shown in Fig. 4.8. We evaluate two metrics i.e., the advancing direction and the direction change at these points respectively. These metrics focus on shape similarity only, thus not sensitive to the distance between two points. Finally, we calculate the opacity according to the difference between these measured values in such a way that the larger the difference is, the higher an opacity is.

The equations for calculating the local opacity on a point basis are formulated as follows. The centerline point p_j^c closest to the *i*-th flowline point p_i^f is determined as:

$$\mathrm{p}_j^c = \underset{\mathrm{p}_k^c \in \mathrm{L}^c}{\arg\min}\, d\left(\mathrm{p}_i^f, \mathrm{p}_k^c\right), \tag{4.5}$$

where L^c is an ordered set of points in the vessel centerline propagating from the vessel trunk to the outlet and $d\left(\mathrm{p}_i^f, \mathrm{p}_k^c\right)$ is the Euclidean distance between p_i^f and p_k^c. The direction vector v_i^f at the *i*-th flowline point is defined by the normalized vector pointing from the current point p_i^f to its next point p_{i+1}^f as follows:

$$\mathrm{v}_i^f = \frac{\mathrm{p}_{i+1}^f - \mathrm{p}_i^f}{\left|\mathrm{p}_{i+1}^f - \mathrm{p}_i^f\right|}. \tag{4.6}$$

Likewise, the direction vector v_j^c at the corresponding *j*-th centerline point p_j^c is:

$$\mathrm{v}_j^c = \frac{\mathrm{p}_{j+1}^c - \mathrm{p}_j^c}{\left|\mathrm{p}_{j+1}^c - \mathrm{p}_j^c\right|}. \tag{4.7}$$

The direction difference $\varphi_{i,j}$ is calculated by the angle between direction vectors at two corresponding points:

$$\varphi_{i,j} = cos^{-1}(\mathrm{v}_i^f \cdot \mathrm{v}_j^c). \tag{4.8}$$

This direction difference catches the reversed flow compared with normal flow represented by the propagating direction of the centerline. To map $\varphi_{i,j}$ to the opacity value α_i^d in the range between 0 and 1, the following equation is used as:

$$\alpha_i^d = saturate(\frac{\varphi_{i,j} - \varphi_m}{\varphi_M - \varphi_m}), \tag{4.9}$$

where *saturate*() is the function to truncate the input value into the range between 0 and 1. φ_m and φ_M are the user specified minimum and maximum thresholds, which set to 5.5° and 150° respectively in our implementation.

As the second metric, the direction change which indicates sudden or slow change of flow direction catches a rapid fluctuation of the abnormal flow. The direction change also is determined by the angle between directional vectors at two successive points along a flowline. The direction change θ_i^f at the *i*-th flowline point is computed as:

$$\theta_i^f = cos^{-1}(\mathrm{v}_{i-1}^f \cdot \mathrm{v}_i^f). \tag{4.10}$$

And the angle θ_j^c at its corresponding *j*-th centerline point is:

$$\theta_j^c = cos^{-1}(\mathrm{v}_{j-1}^c \cdot \mathrm{v}_j^c). \tag{4.11}$$

The difference between two angles is defined as:

$$\theta_{i,j} = |\theta_i^f - \theta_j^c|. \tag{4.12}$$

Similarly, the equation to map $\theta_{i,j}$ to the opacity value α_i^c in the range between 0 and 1 is defined as:

$$\alpha_i^c = saturate(\frac{\theta_{i,j}-\theta_m}{\theta_M-\theta_m}), \tag{4.13}$$

where θ_m and θ_M are the minimum and maximum parameters of the range specified by users, which set to 0° and 45° respectively.

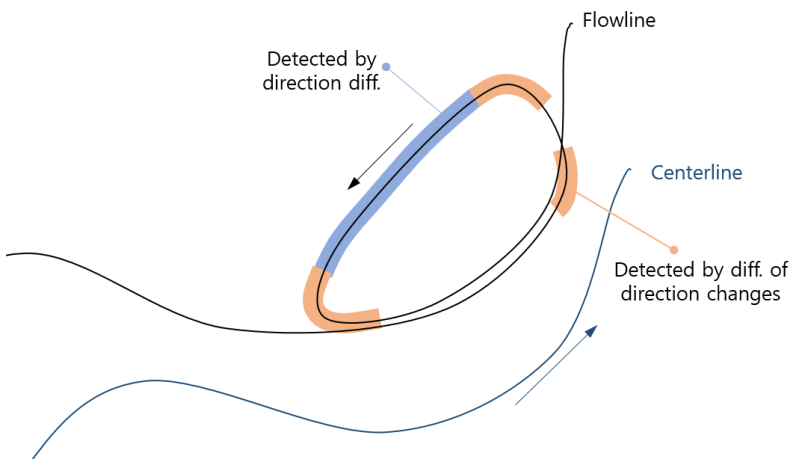


Figure 4.9 The shape similarity between flowline and centerline segments is calculated on a point basis by the difference of direction vectors, and that of direction changes.

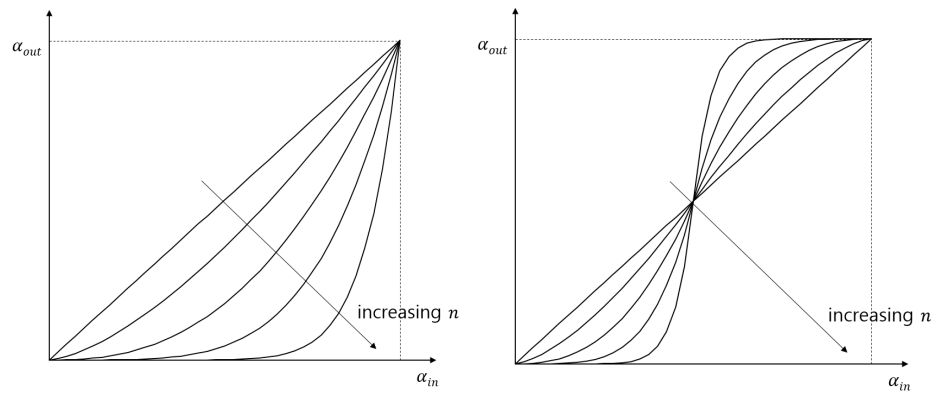
Since turbulent flow has hemodynamic features such as large winding angles, frequent fluctuations and reversed flow direction, the difference of direction and that of direction change enable to detect the abnormal pattern of turbulent flow. A high opacity is assigned to the current point when the difference of direction or that of direction change between two matched points is large. Otherwise, a low opacity is assigned. The final opacity is computed in a combined form of two metrics so that if at least one of two metrics is high, the final result becomes high:

$$\alpha_i = 1 - \left(1 - \alpha_i^d\right)\left(1 - \alpha_i^c\right). \tag{4.14}$$

Accordingly, two metrics are complementary to each other because the one detects what the other doesn't recognize as dissimilar in flow pattern as shown in Fig. 4.9.

The aforementioned global opacity is consistently calculated and not sensitive to the outliers by taking the mean distance over all flowline points. However, the resulting local opacity may be significantly different from its neighbors, since it is entirely calculated on a point basis. Thus, a high dissimilarity in flow pattern may occur at bifurcations or strongly curved vessels. However, these line segments are limited to a very small in the length, while line segments detected as abnormal flow in the narrowed vessel are relatively long. Therefore, in order to exclude the outliers and ensure a smooth transition in opacity along a flowline, we apply the Gaussian filter. Additionally, it is worthy to note that since the presented criteria for computing local opacity are view-independent, we need not recalculate the opacity whenever changing the view point.

The visibility of flowlines can be locally controlled by means of the opacity determined according to their importance. Thus, unimportant line segments are transparently drawn, whereas important ones are opaquely expressed. As a consequence, this causes the viewer to naturally pay more attention to important line segments, while suppressing distracting information of less important ones.



(a) (b)

Figure 4.10 Opacity remapping functions are designed to differentiate the specific range of opacities. (a) The first function differentiates the upper range of opacities, giving up the depiction of lower opacities. (b) The second function differentiates the middle range of opacities, while sacrificing the discrepancy of low and high values.

4.4.3 Opacity Adjustment

We provide a user-adjustable parameter to enhance the differentiation of the opacity so that the user can interactively distinguish abnormal flow from normal flow. This differentiation is implemented by introducing an adjustable exponent parameter n in a power-law expression:

$$\alpha_{out} = {\alpha_{in}}^{n}, \tag{4.15}$$

where the input opacity, α_{in} and output opacity, α_{out} are typically in the range between 0 and 1. The exponent *n* is between 1 and 10. The output opacity decreases

rapidly as *n* increases, as shown in Fig. 4.10. Another function for opacity remapping can be used to differentiate the middle range of opacities as follows:

$$\alpha_{out} = \begin{cases} 0.5 \cdot (\frac{\alpha_{in}}{0.5})^n, & if\ \alpha_i < 0.5, \\ 1.0 - 0.5 \cdot (\frac{1.0-\alpha_{in}}{0.5})^n, & otherwise. \end{cases} \tag{4.16}$$

The first function focuses on differentiating both middle and high opacities, sacrificing the depiction of low opacities. Whereas the other function widens the middle opacities sacrificing both low and high opacities. Through these functions, the user can selectively hide or unveil insignificant flows besides important flows.

4.4.4 Blending

Drawing semi-transparent flowlines is not trivial because a depth sorting for blending should be performed in advance at the cost of storing and sorting fragments. To avoid it, the order independent method that alters the compositing operator was proposed by McGuire et al. [50]. The blending equation is expressed by:

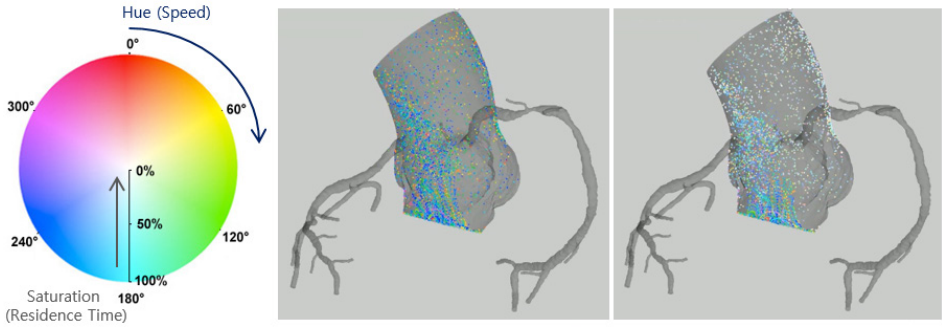
$$C_f = \frac{\sum_{i=1}^{n} C_i \cdot w(z_i, \alpha_i)}{\sum_{i=1}^{n} \alpha_i \cdot w(z_i, \alpha_i)} (1 - \prod_{i=1}^{n}(1 - \alpha_i)) + C_0 \prod_{i=1}^{n}(1 - \alpha_i), \tag{4.17}$$

where C_i and α_i denote the color and opacity of the *i*-th fragment. C_0 and C_f represent the background color and the pixel color respectively. z is a camera-space value that is zero at the center of projection and decreases away from the

camera towards -∞. Any monotonically decreasing, non-zero function of |z| on the expected depth range, such as $w(z,\alpha) = |z|^{-k}\ or\ w(z,\alpha) = z - z_{far} + \varepsilon$, grants nearer surfaces higher weights and thus creates an occlusion cue between transparent surfaces [50]. To render flowlines semi-transparently, we conduct this blending method by means of a two-pass off-screen rendering. In the first pass, we render the blended colors and its accumulated weights into two textures of the frame buffer object. In the second pass, we calculate the final pixel color by mixing the blended color and the background color with the accumulated weights.

4.5 HSV Color Coding

Color is one of the important visual cues that assist the viewer to figure out data characteristics at one glance. Therefore, color coding has been popularly used in flow visualization field to convey an additional information about flowlines. This is achieved by mapping predefined colors according to the flow attribute of interest e.g., local speed, seed position, and residence time. Although the color used therein can be defined in a variety of ways by users, rainbow colors are preferably employed as de facto standard in the medical literature since it is what medical domain experts are accustomed to viewing [51]. In addition, a perceptually appropriate color map

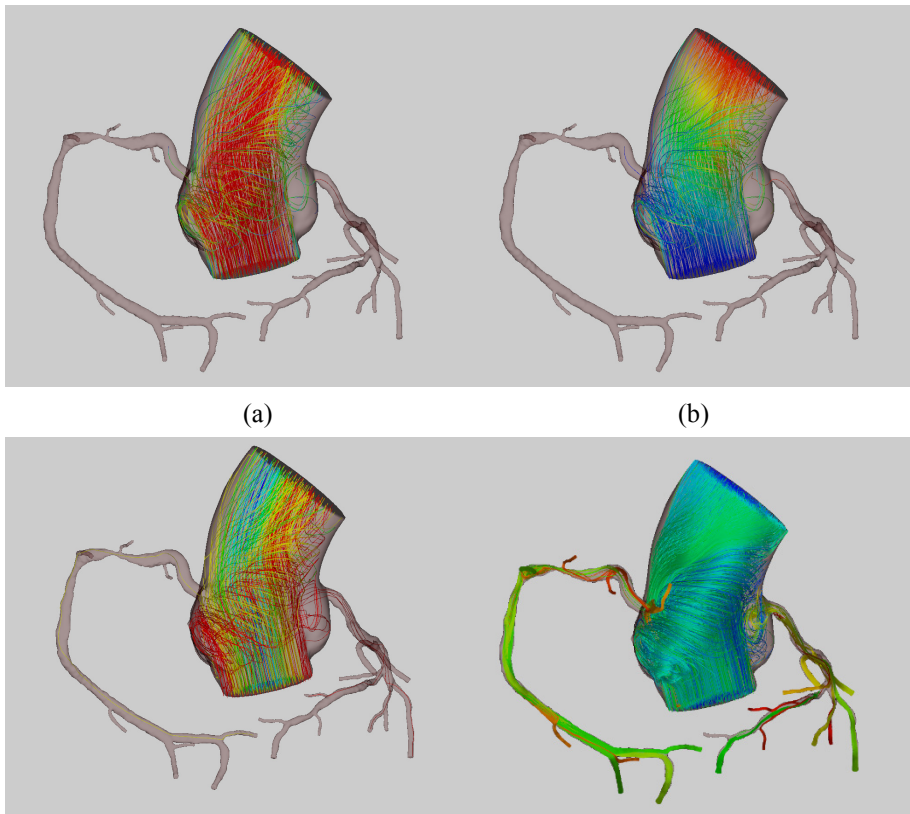


(a) (b) (c)

Figure 4.11 HSV color coding. (a) HSV color scheme uses hue variation and saturation to express local speed and residence time, respectively. (b) Rainbow color coding: Hue is mapped with local speed. (c) HSV color coding: Hue and saturation are mapped with local speed and residence time respectively.

leads to fewer diagnostic mistakes [51]. Hence, we employ a typical rainbow color coding in our implementation. Red is the most eye-catching as it has a pop-effect [52]. Therefore, in rainbow color scheme ranging from blue to red, blue is mapped on a low value whereas red is mapped on a high value relevant to a risk of a major interest.

However, rainbow color coding cannot represent multiple flow characteristics at the same time, because it may cause the color confusion. In order to simultaneously convey two kinds of information in a combined way, we introduce Hue-Saturation-Value (HSV) color scheme as depicted in Fig. 4.11. We employ hue values of the range between 0^{o} and 240^{o} corresponding to rainbow colors to indicate the flow speed in the same way as the conventional color coding. Instead, we decrease the



(a) (b)

(c) (d)

Figure 4.12 Color-coded flowlines based on various flow attributes. (a) Local speed. (b) Residence time. (c) The distance of a seed point from the center of the vessel cross-section. (d) Seeding plane.

saturation according to the residence time which indicates how long a particle takes to arrive at the current position from a seed point. The color is calculated by:

$$C_i' = w_i C_i + (1 - w_i) C_w,$$

$$\text{where } w_i = \alpha^i, \ (0 < \alpha < 1). \tag{4.18}$$

C_i is the color of the i-th point starting from the seed point and C_w is the white color. And α is the user specified value to control how fast the saturation decreases. This approach gives the fading effect like the dye losing its color after being dropped into fluid. In our implementation, older particles or line segments get faded into white [see Fig. 4.11(c)]. Thereby, both velocity and residence time can be intuitively recognized. We present a continuous color mapping as default. However, the discrete color mapping can be used since it contributes to an easy recognition of the value indicated by color.

Flowline color mapping can be switched to the flow attribute selected by a user. Fig. 4.12 shows the results of color mapping that represent various flow properties i.e., local speed, residence time, seeding plane, and the distance of a seed point from the center of the vessel cross-section. This color coding assists to inspect the characteristic behavior of flows with a similar attribute.

The vessel color should be carefully chosen to avoid being confused with colors of flowlines. Thus, to distinguishably render vessel and flowlines, we assign a desaturated color to the vessel. Therefore, the color of older particles gets more similar to the desaturated vessel color, giving an effect of disappearing gradually over time.

4.6 Vessel Rendering

In this section, we describe the rendering method used to draw the vessels in the visualization combined with blood flow. Blood vessels are drawn by the surface rendering that uses the surface mesh extracted from volume mesh in the preprocessing step. In order to understand the initiation, progression, and regression of vessel pathologies, the inspection of the interaction between blood flow and its enclosing vessel morphology is necessary. Therefore, both flow and vessel information should be visualized at the same time. However, the simultaneous depiction of both information generally causes the occlusion problem. As a correct interpretation of flow pattern is more important, we put the visual priority on the embedded flow rather than its surrounding vessels. Therefore, we adopt the strategy that overlays flowlines on vessel surface to guarantee a high visibility of flow information.

Furthermore, excessive details of vessels may interrupt users to comprehend flow patterns. Hence, we endeavor to simplify the geometry of vessel shape to minimize the distraction of vessel details and to achieve an easy interpretation. To attain this purpose, we apply a mesh smoothing to the vessel surface. Instead, we enhance vessel contours to emphasize the important shape information by performing a contour enhancement technique used in non-photorealistic rendering.

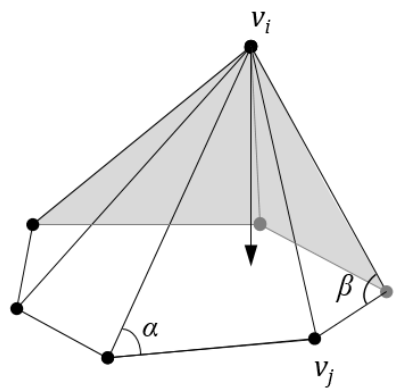


Figure 4.13 Cotangent Laplacian vector (arrow) for a vertex v_i and its 1-ring neighors as well as the angles for a vertex v_j [55].

4.6.1 Vessel Smoothing

To suppress details of vessel surface, we employ Laplacian mesh smoothing [53]. As derived by Meyer et al. [54], the Laplacian Δ at the i-th vertex v_i can be discretized with cotangent weights as follows:

$$\Delta \mathrm{v}_i = \frac{1}{2\mathrm{A}_i} \sum_{\mathrm{v}_j \in N_1(\mathrm{v}_i)} (cot\alpha_{ij} + cot\beta_{ij})(\mathrm{v}_j - \mathrm{v}_i), \tag{4.19}$$

where A_i is one third of the area of all incident triangles, $N_1(\mathrm{v}_i)$ is the set of one-ring neighbors of v_i. α_{ij} and β_{ij} are the angles opposing the edge (i, j) as shown in Fig. 4.13. The cotangent weights are preferably used because they gives a good

approximation of the surface normal. In order to smooth a surface mesh, we simply apply Laplacian smoothing as follows:

$$\mathrm{v}_i \leftarrow \mathrm{v}_i + \lambda \Delta \mathrm{v}_i, \tag{4.20}$$

where λ is a scalar coefficient to control the smoothing level.

4.6.2 Vessel Contour Enhancement

To reveal better the important features of vascular anatomy, we improve the visibility of vessel contours by darkening surfaces facing away from the viewer. This is achieved by mixing the originally assigned color and the black defined as a contour color according to the orientation of the surface normal to the viewer. As a result, the improved contrast around vessel boundaries leads to a better perception of vessel shape. The contour intensity $I_{contour}$ is computed based on the view-dependent surface orientation in the following equation [56]:

$$I_{contour} = (1 - |V \cdot N|)^n, \tag{4.21}$$

where V is the view vector and N is the surface normal vector. The exponent n determines what is classified as a contour depending on the angle between view vector and normal vector. In our implementation, n is empirically set to 4 showing good results. The final color $C_{vessel}{}'$ is calculated by blending the contour color

$C_{contour}$ with the vessel color C_{vessel} according to the contour intensity $I_{contour}$ as follows:

$$C_{vessel}{}' = (1 - I_{contour})C_{vessel} + I_{contour}C_{contour}. \quad (4.22)$$

This contour enhancement is implemented with Phong shading for lighting on vessel surface in fragment shader.

4.7 Flowline Drawing

In this section, we introduce two methods used to improve the visibility of flowlines. The first is line illumination for increasing depth perception of flowlines and the other is line halo for enhancing the spatial relationship between overlapped flowlines.

4.7.1 Line Illumination

Phong shading [57] is the most widely used method to give a lighting effect on target objects to be rendered. A normal vector is basically needed to calculate both diffuse and specular term in Phong equation:

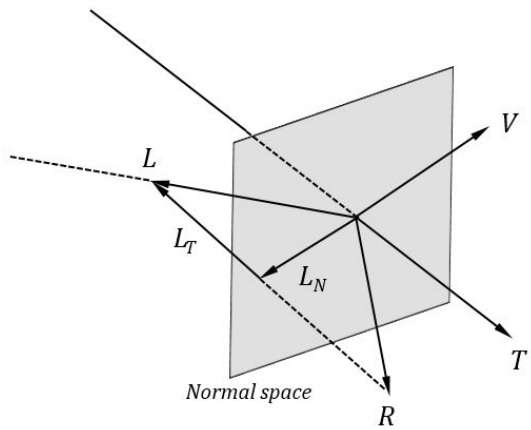


Figure 4.14 The light vector *L* can be decomposed into two orthogonal components, L_T and L_N, corresponding to the projection on the line's tangent and normal space, respectively [59].

$$I = k_a + k_d \cdot L \cdot N + k_s \cdot (V \cdot R)^n,$$

$$where\ R = L - (2N \cdot L)N. \tag{4.23}$$

However, no unique normal vector is defined in line segments. As a result, it is impossible to apply a shading effect for line illumination. As an alternative solution, cylinders can be employed instead of lines to express illuminated flowlines. However, this tuboid-typed line drawing has disadvantages in that it increases both geometric complexity and rendering time. In addition, it may cause occlusion and visual clutter especially in cases of densely placed flowlines. Therefore, the use of cylinders is desirable only when drawing a small number of lines for abstract depiction. Although line segments have no explicit normal vectors, Zöckler et al. [58]

and Stalling et al. [59] found the way to apply Phong shading to lines by defining the pseudo normal vector according to the maximum reflection principle. Choosing the normal vector N as the one that is coplanar to the tangent vector T at a flowline point p and the light vector L as shown in Fig. 4.14, the diffuse term $L \cdot N$ can be expressed without N [58]:

$$L \cdot N = |L_N| = \sqrt{1 - |L_T|^2} = \sqrt{1 - (L \cdot T)^2}. \tag{4.24}$$

Likewise, the specular term $V \cdot R$ can be rewritten without R [58]:

$$\begin{aligned} V \cdot R &= V \cdot (L_N - L_T) \\ &= V \cdot \big((L \cdot N)N - (L \cdot T)T\big) \\ &= \big((L \cdot N)(V \cdot N) - (L \cdot T)(V \cdot T)\big) \\ &= \sqrt{1 - (L \cdot T)^2}\sqrt{1 - (V \cdot T)^2} - (L \cdot T)(V \cdot T). \end{aligned} \tag{4.25}$$

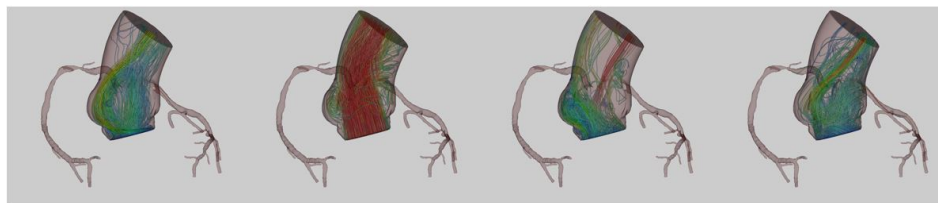
4.7.2 Line Halo

Furthermore, we incorporate a line drawing technique called line halo [28] to enhance depth perception of flowlines. Line halo is to add black border on both sides of a line. This is accomplished by drawing lines twice with OpenGL depth function set to GL_LESS, firstly drawing the original line and then redrawing the black line with a larger line width [28]. As a result, the black boundary only is additionally

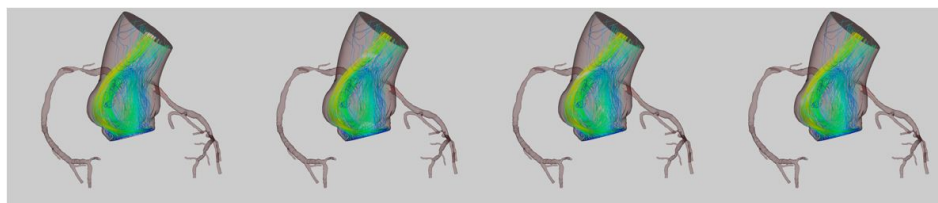
drawn, which contributes to clarify the spatial relationship between overlapped line segments.

4.8 Animation

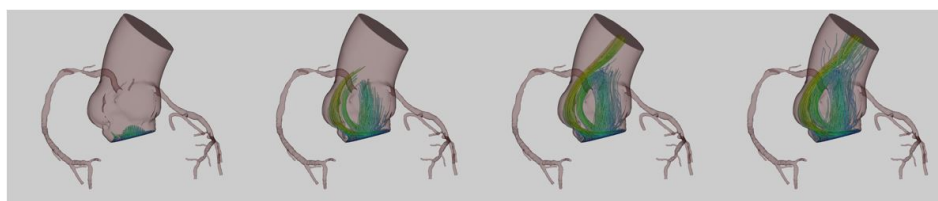
In static scene, stationary particles or flowlines alone are difficult to appropriately express their advancing direction and speed. So, the arrow and color coding are often used to represent both information. However, these are still not intuitive, thus need an effort to understand. As an alternative solution, the animation has been employed to exhibit blood flow dynamics in a more intuitive and comprehensible fashion. We implement the animation in three ways i.e., updating flowlines over time, local animated highlighting as presented in [8, 32], and line growing along a flow path, as shown in Fig. 4.15(a), (b), and (c). The first method depicts the blood flow by sequentially updating flowlines extracted at each phase of a cardiac cycle. Thereby, this method facilitates to observe flow patterns changing during a cardiac cycle. Meanwhile, the second and third methods mimic a more realistic flow behavior by moving forward the highlights or by extending the visible line length along flowlines with no update of flow data. These methods permit a quick perception of the relative flow direction and speed between flowlines that are ambiguous in static visualization.



(a)



(b)



(c)

Figure 4.15 Animation methods for blood flow depiction. (a) Frame update over time. (b) Local animated highlighting. (c) Line growing.

In order to efficiently perform the animation, we implement two-pass rendering that draws the vessel surface and flowlines separately. In the first pass, the vessel surface is once drawn into the frame buffer object (FBO). At this time, the vessel backface is drawn in a half-dissected form with OpenGL culling function. In the second pass, after filling the background with the rendered result of vessels stored in

FBO, flowlines are drawn so that they overlay the vessel surface. Afterwards, the second pass only is repeatedly performed skipping the first pass. Only when resized or rotated by user interaction such as zoom in/out and change of viewpoint, the first pass is carried out to redraw the vessel. This two-pass rendering enables to reuse the rendered result of vessel surface and to update flowlines alone. As a result, it significantly reduces the rendering time especially when visualizing through animation.

Chapter 5. Experimental Results

We implemented the proposed method using C++ with OpenGL library for visualization and Qt library for user interface. All tests were performed on 3.4 GHz Intel Core i7 PC with 16 GB memory and graphics hardware equipped with Nvidia's GeForce GTX 780 GPU and 3 GB memory.

We utilized the blood flow data that was acquired by the CFD simulation in the ascending aorta and coronary arteries. The first and second datasets consist of flow data with 120 frames spaced 8 msec. apart during a cardiac cycle. The third dataset contains 101 frames spaced 10 msec. apart. Each flow data is the volume mesh consisting of 0.46~1.82 million of tetrahedrons. Table 5.1 summarizes three datasets used to validate the performance of our approach.

The workflow of the proposed method for blood flow visualization is presented in Fig. 5.1. Our method receives multiple frames of flow data and the vessel centerline as the input data. In the preprocessing stage, we extract from volume mesh surface mesh which is used for the following seeding process and vessel surface rendering. Afterwards, in the seeding stage, we detect the vessel inlet or outlets and then place seed points on them. These seed points are applied equally to every flow

Table 5.1 Test Datasets

Dataset	No. tetrahedrons	No. frames	time step (msec.)
1	1,823,304	120	8
2	1,761,600	120	8
3	459,332	101	10

data during a cardiac cycle. Next, in the tracing stage, the line integration is performed to construct flowlines from the vector field with the complementary means to prevent the early termination. In the rendering stage, we firstly reduce excessive details on vessel surface by the Laplacian mesh smoothing and further improve the vessel shape by the contour enhancement technique. Afterwards, we assign the color to flowline segments with a color mapping of flow attributes. Additionally, we calculate the opacity according to the shape similarity of the vessel centerline. Then, we conduct additionally line-specific illumination and halo on flowlines to enhance depth perception. Since the flowlines are of main concern, overlaying flowlines on the vessel is desirable. Thus, in order to illustrate vessels in a half-dissected and semi-transparently overlapped form, we firstly draw vessel backface using the culling and blending functions supported by OpenGL. Finally, color-coded and opacity-modulated flowlines are rendered overlaying vessels.

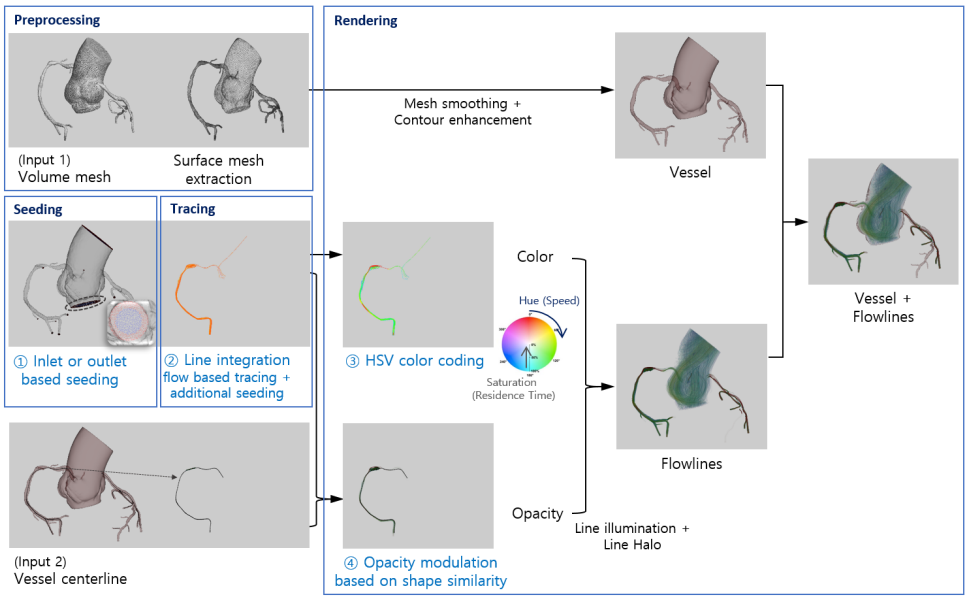


Figure 5.1 Workflow of the proposed method for blood flow visualization.

5.1 Evaluation on Seeding

We compared the seeded results in the inlet of the ascending aorta in order to evaluate the performance between the fixed radius and variable radius seeding method. The fixed radius based seeding (FRS) placed seed points with a uniform distribution as shown in Fig. 5.2(a). However, the variable radius based seeding (VRS) allocated more seed points on the inhomogeneous area where inflows and outflows coexist, while distributing fewer seeds on the central area with homogeneous flow [see Fig. 5.2(c)]. In spite of the difference in the seeded patterns,

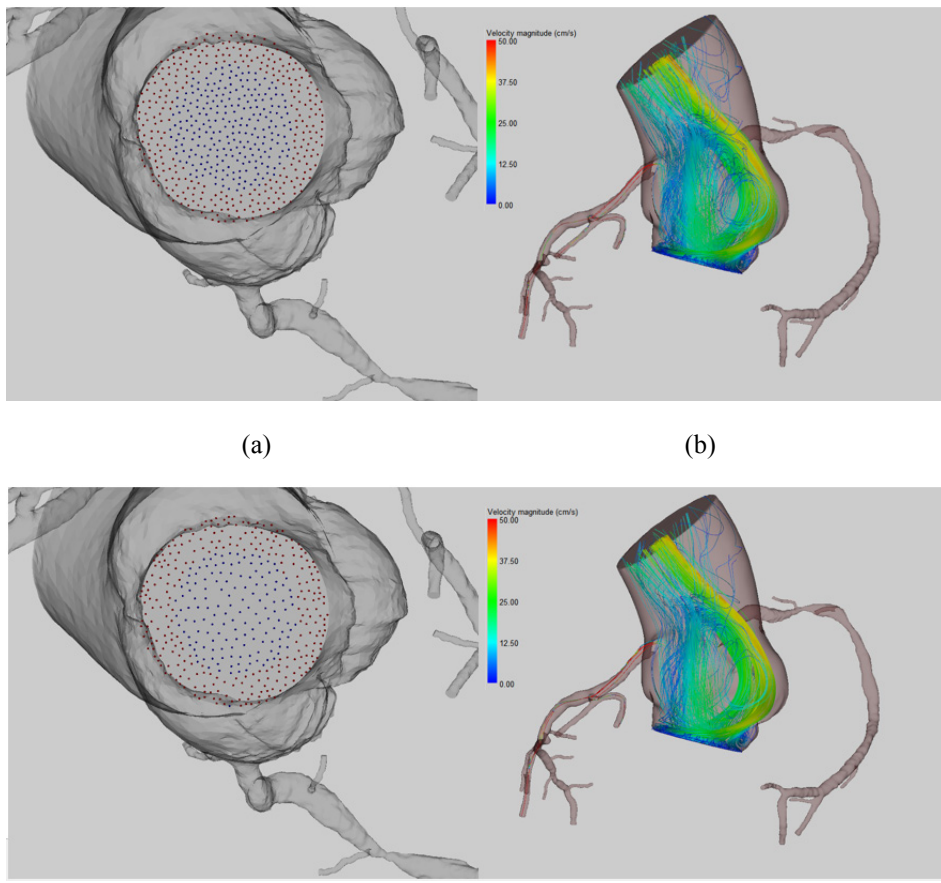


Figure 5.2 The comparison of seeded results between the fixed radius (FRS) and variable radius based seeding (VRS). (a) FRS yielded 560 seed points. The color of a seed point indicates the flow direction i.e., inflow (blue), outflow (red). (b) Streamlines generated from seed points of (a). (c) VRS yielded 417 seed points. (d) Streamlines generated from seed points of (c).

the overall flow patterns of two methods were nearly similar [see Fig. 5.2(b) and (d)]. This shows the efficiency of VRS, since this method enables to capture the overall

flow patterns with a reduced number of seed points (n=417), which are almost identical to those of FRS with more seed points (n=560).

In order to compare the flow patterns between FRS and VRS, we exploit the linear entropy [62] and the angular entropy [13] as the metrics to quantify the shape of a flowline. The linear and angular entropy indicate respectively the length and angle variation along a flowline. The linear entropy E_L is formulated by:

$$E_L = -\frac{1}{log_2(m+1)}\sum_{j=0}^{m}\frac{D_j}{L_S}log_2\frac{D_j}{L_S} \tag{5.1}$$

where m denotes the number of flowline segments, D_j is the j-th segment length and L_S is the total length of a flowline. Likewise, the angular entropy E_A is defined by:

$$E_A = -\frac{1}{log_2(m)}\sum_{j=0}^{m-1}\frac{A_j}{L_A}log_2\frac{A_j}{L_A} \tag{5.2}$$

where A_j is the absolute value of the angle at the j-th flowline joint and L_A is the total angular variation along a flowline. The flowlines with similar values in the linear and angular entropy are analogous in shape when generated in the same vector field. Therefore, we can evaluate the overall shape similarity of flowlines by the standard deviation of these entropies. A high standard deviation indicates that the flowlines are more various and less overlapped in shapes. Table 5.2 shows the mean and standard deviation of the linear and angular entropy of flowlines generated respectively by FRS and VRS. VRS had a higher standard deviation than FRS in both linear and angular entropy, indicating that VRS exhibits more various flow

patterns. Hence, VRS is more proper to capture the overall flow feature with the same number of seed points comparing to FRS. Fig. 5.3 depicts the generated flowlines and their corresponding scatter plots between FRS and VRS.

Table 5.2 The linear and angular entropy of flowlines (mean ± std.)

Seeding method	*FRS*	*VRS*
Linear entropy (E_L)	0.959±0.090	0.946±0.101
Angular entropy (E_A)	0.901±0.110	0.884±0.135

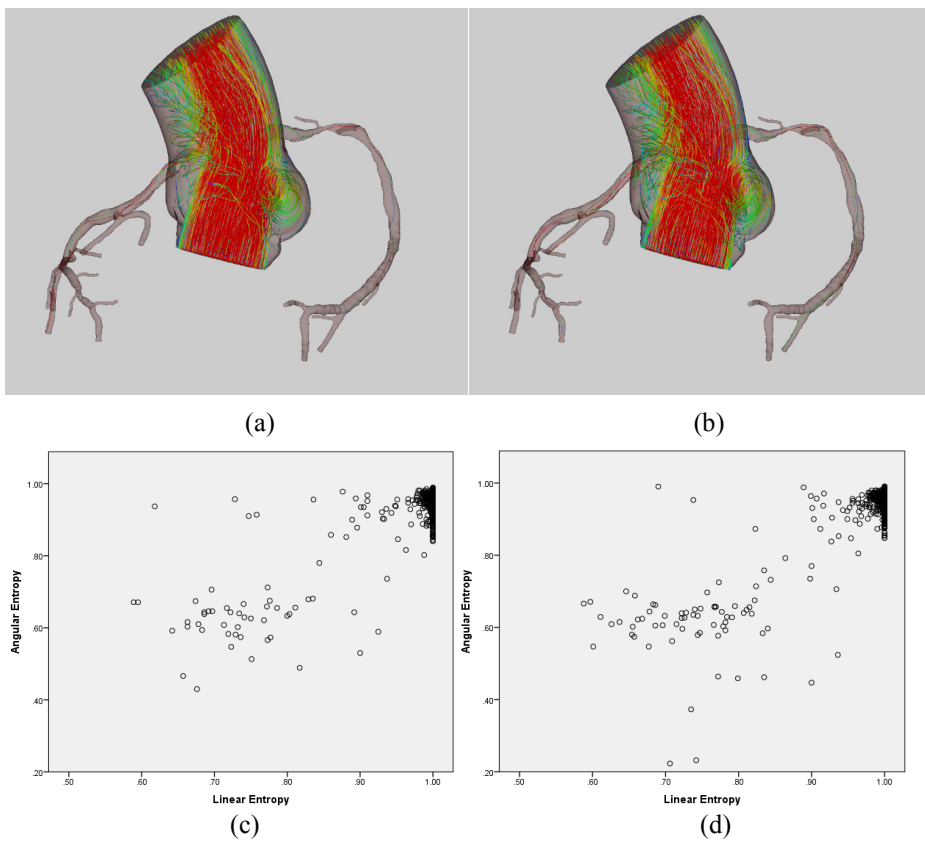


Figure 5.3 The comparison of the flow patterns between FRS and VRS. The scatter plots of flowlines using the linear and angular entropy. (a) Flowlines generated by FRS. (b) Flowlines generated by VRS. (c) The scatter plot of FRS. (d) The scatter plot of VRS.

5.2 Evaluation on Tracing

We introduce the line generation rate to evaluate the performance of the proposed flow based bidirectional tracing described in Section 4.3. This rate is defined by the ratio of the number of successfully generated flowlines divided by the number of seed points. A flowline is regarded as successfully generated if it traces at least ten successive points after starting at a given seed point. Table 5.3 shows the generation rate of flowlines that are seeded in the inlet of an ascending aorta at five time instances during the cardiac cycle. The unidirectional tracing showed that the line generation rate was varying according to the phase of a cardiac cycle, i.e., a high rate in systole and a low rate in diastole as depicted in Fig. 5.4. This resulted from the fact that the inflows ejected from the left ventricle are typical in systole, while the rotating and retrograde flows evolve in diastole. Therefore, the unidirectional tracing which performs unconditionally the forward tracing was successful in systole, while failed to construct flowlines at seed points with outflows in diastole. On the contrary, the flow based bidirectional tracing yielded successfully flowlines at all seed points regardless of a cardiac cycle, since it applied adaptively the tracing direction according to the flow direction.

Fig. 5.5 presents a comparison on line generation result of the flow based bidirectional tracing against that of the unidirectional tracing. An apparent difference

between two methods appears especially in diastole phase where both inflow and outflow coexist. The unidirectional tracing failed to construct flowlines at seed points with the reversed flow. As a result, this method could not adequately show flow characteristics near the vessel wall. However, the proposed bidirectional tracing was successful in grasping flow features in near-wall as well as central area consistently during a cardiac cycle.

In order to evaluate the effect of the proposed tracing method to prevent early terminated line integration, we utilize two metrics. The one is the length of flowlines measured in terms of the number of traced points. The other is the arrival rate $R_{arrival}$ that is determined by the ratio of the number of the flowlines arrived at the opposite end divided by the number of seed points. This arrival rate indicates that all generated flowlines arrive to the other end as the ideal case if $R_{arrival}$ =1.0 and some flowlines fail to arrive if $R_{arrival}$ <1.0. Table 5.4 presents the arithmetic mean and standard deviation of the line length and those of the arrival rate, which are measured in constructing streamlines during a cardiac cycle in four coronary arteries as shown in Fig. 5.6. The line integration method without additional seeding [38] produced a short length of flowline with 431 points and a low arrival rate of 5.27%, while the proposed integration method with additional seeding yielded a longer flowline with 847 points and a higher arrival rate of 30.84% on the average. Accordingly, the proposed method exhibited an enhancement with 1.96 times in line length and 5.86

times in arrival rate respectively against the method without additional seeding. An improved but still low arrival rate of our approach is attributed to the fact that it is impossible to further integrate a flowline when at least one of the predicted positions used in the fourth-order Runge-Kutta method is outside the defined vector field within the vessels.

Fig. 5.7 illustrates the resulting flowline integration in the narrow branch vessel between the existent method and our method. In the existing method, most of flowlines exceeded the vessel boundary on the way of line integration, which resulted in short flowlines. The proposed method enforced flowlines under integration to be constrained to the vessel inside by stopping the integration prior to crossing the vessel boundary and restarted the stopped integration at the adjacent vertex with nonzero vector closest to the termination position. Thereby, our method could output length-extended flowlines.

Table 5.3 Line generation rate in the inlet of an ascending aorta

Time (msec.)	*Unidirectional*	*Bidirectional*
0	0.978	1.000
0.2	1.000	1.000
0.4	0.763	1.000
0.6	0.785	1.000
0.8	0.854	1.000
Average	0.876	1.000

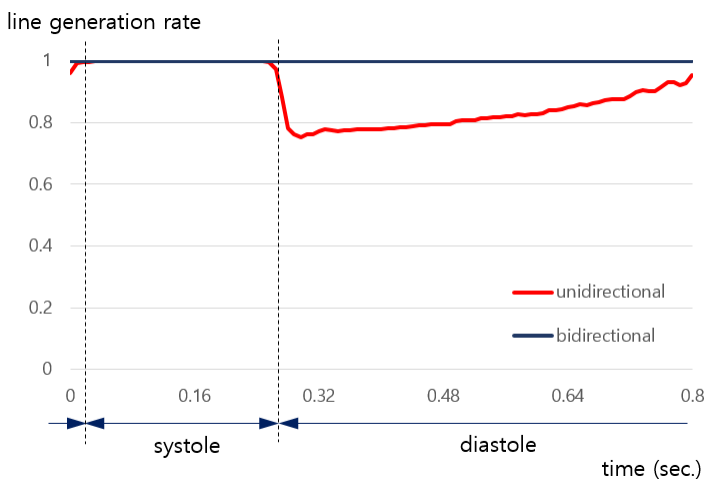
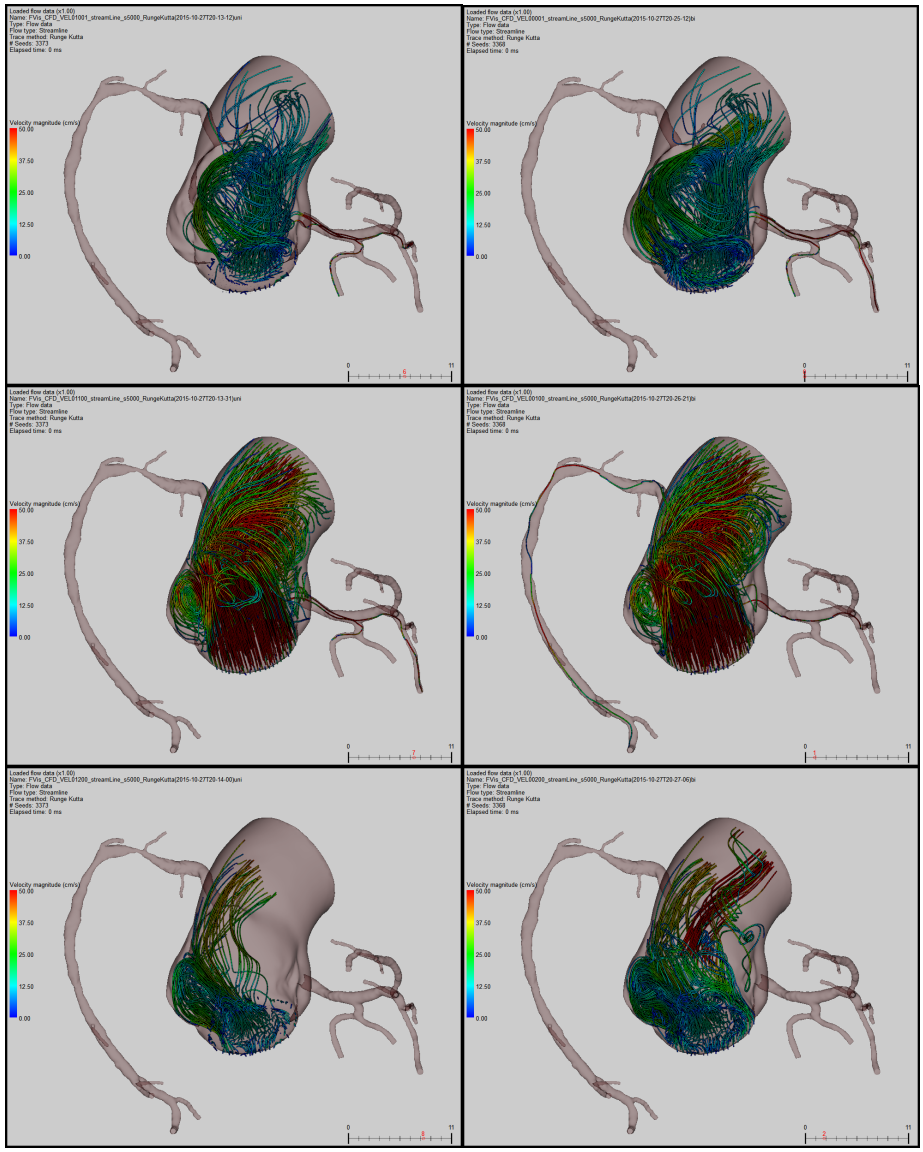


Figure 5.4 Line integration rate in the inlet of the ascending aorta during a cardiac cycle. The undirectional tracing (red) showed a lower rate due to reversed local flows in diastole, while the flow based bidirectional tracing (blue) generated flowlines at all seed points regardless of cardiac cycle.


Velocity magnitude (cm/s)
50.00
37.50
25.00
12.50
0.00

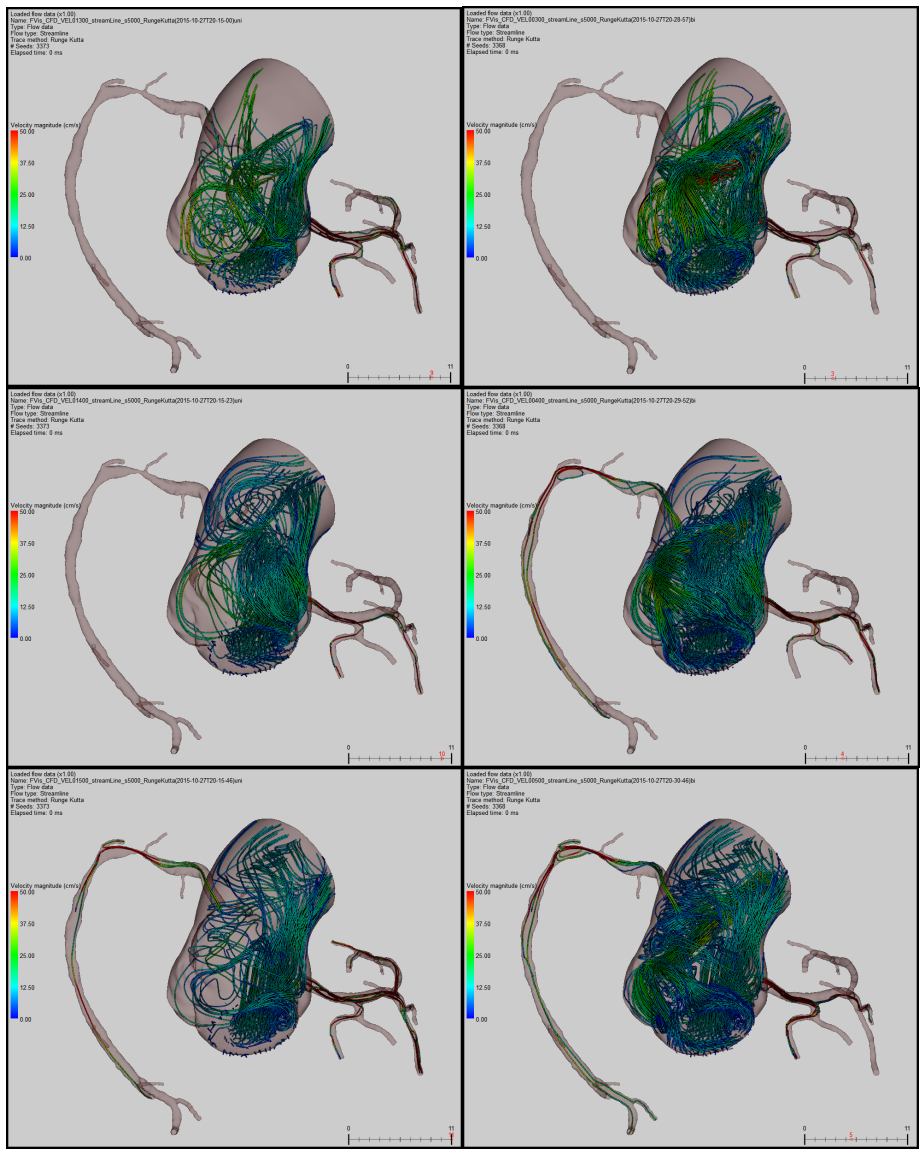


Figure 5.5 The comparison of streamline generation seeded in the inlet of an ascending aorta during a cardiac cycle (from top to bottom). The unidirectional tracing (left column) failed to conduct the line integration at near-wall outflow points and yielded varying flowlines depending on cardiac cycle. The flow based bidirectional tracing (right column) was successful at all seed points in depicting flowlines in near-wall as well as central area and yielded a consistent flow depiction regardless of cardiac cycle.

Table 5.4 Performance of streamline integration (mean ± std.)

Coronary arteries	*Method without additional seeding* [38]		*Method with additional seeding*	
	Length (pts.)	Arrival rate, $R_{arrival}$ (%)	Length (pts.)	Arrival rate, $R_{arrival}$ (%)
1	533.96±69.23	4.57±1.81	959.67±141.51	27.46±14.21
2	612.77±83.98	7.95±4.03	960.39±188.58	33.50±14.38
3	214.38±49.95	2.30±0.74	681.93±276.46	34.99±12.44
4	364.05±79.27	6.24±2.90	785.13±361.68	27.40±13.80
Average	431.29±70.61	5.27±2.37	846.78±242.06	30.84±13.71

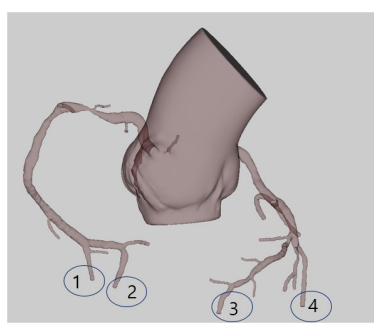
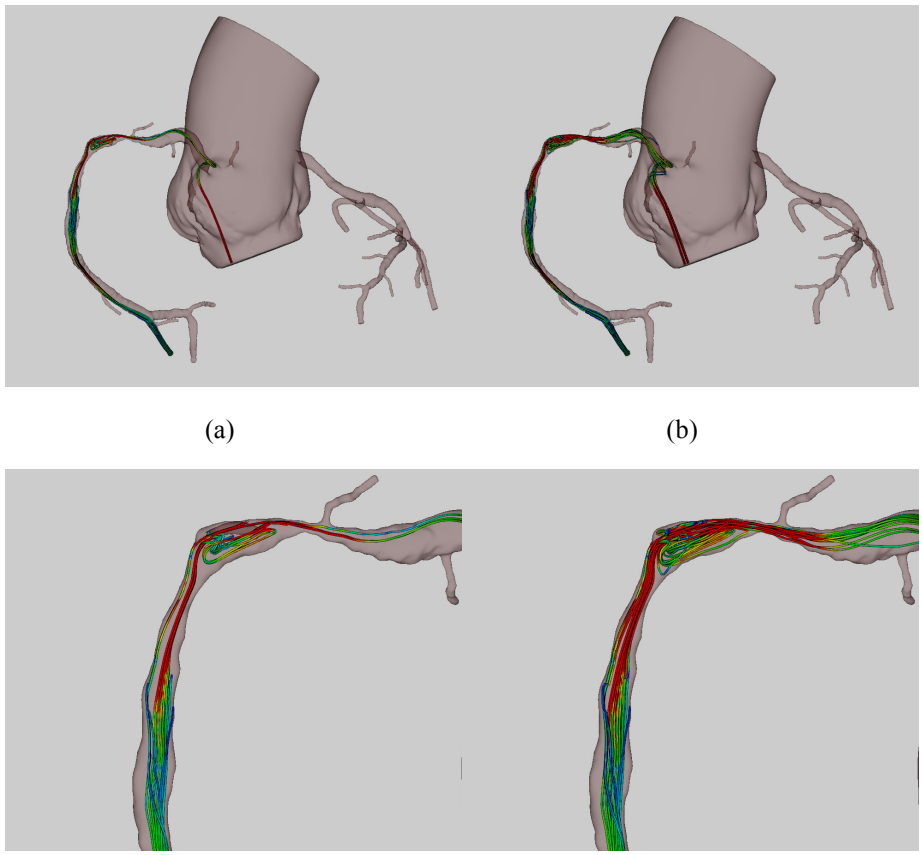


Figure 5.6 Four coronary arteries used to evaluate the performance of streamline integration as shown in Table 5.3.



(a) (b)

(c) (d)

Figure 5.7 Line integration in the narrow branch vessel. (a) short flowlines due to the early termination (3 arrivals of 45 seeds, $R_{arrival}$ = 6.67%). (b) flowlines length-entended by the additional seeding (12 arrivals of 45 seeds, $R_{arrival}$ = 26.67%).

5.3 Evaluation on Opacity Modulation

Fig. 5.8 shows the comparison between methods without and with the opacity modulation. The method with the fixed opacity (1.0) showed severe line occlusion, failing to depict the interior flowlines occluded by the exteriors, which makes it difficult to capture the overall flow behavior. On the contrary, the proposed method exhibited the internal and external flows together, while rendering opaquely flows with extraordinary patterns. Our method depicted the flowlines, which fluctuated along the centerline or had reversed flow direction, more noticeable over the others. The visibility of flowlines can be changed by the parameter that controls the opacity remapping as explained in Section 4.4. The second row of Fig. 5.8 shows the laminar and turbulent flows together for a purpose to observe the overall flow patterns, while still attenuating the opacity according to the shape similarity with the vessel centerline. The third row of Fig. 5.8 illustrates that line segments with turbulent flow dominantly appeared, while most of laminar flowline segments disappeared. Therefore, the user interaction for opacity adjustment enabled a fast recognition of the important flow patterns.

In the ascending aorta with a large diameter, researches [60, 61] have reported that a vortex occurs typically due to the opening and closing of the aortic valve repeated periodically during a cardiac cycle. Meanwhile, in coronary arteries,

regardless of a cardiac cycle, the laminar flows are consistently observed although the flow velocity is slightly varying according to the cardiac cycle. Based on this fact, if the turbulent flow is observed in coronary arteries, the distortion of vessel shape can be estimated as a sign of the progressing vessel pathologies. Therefore, such a flow pattern is directly associated with the stiffly narrowed vessel which results from the stenosis after a plaque formation. Such a typical example is presented in Fig. 5.8. Since the turbulent flow is expressed more opaquely, it is noticeable enough to grab more attentions.

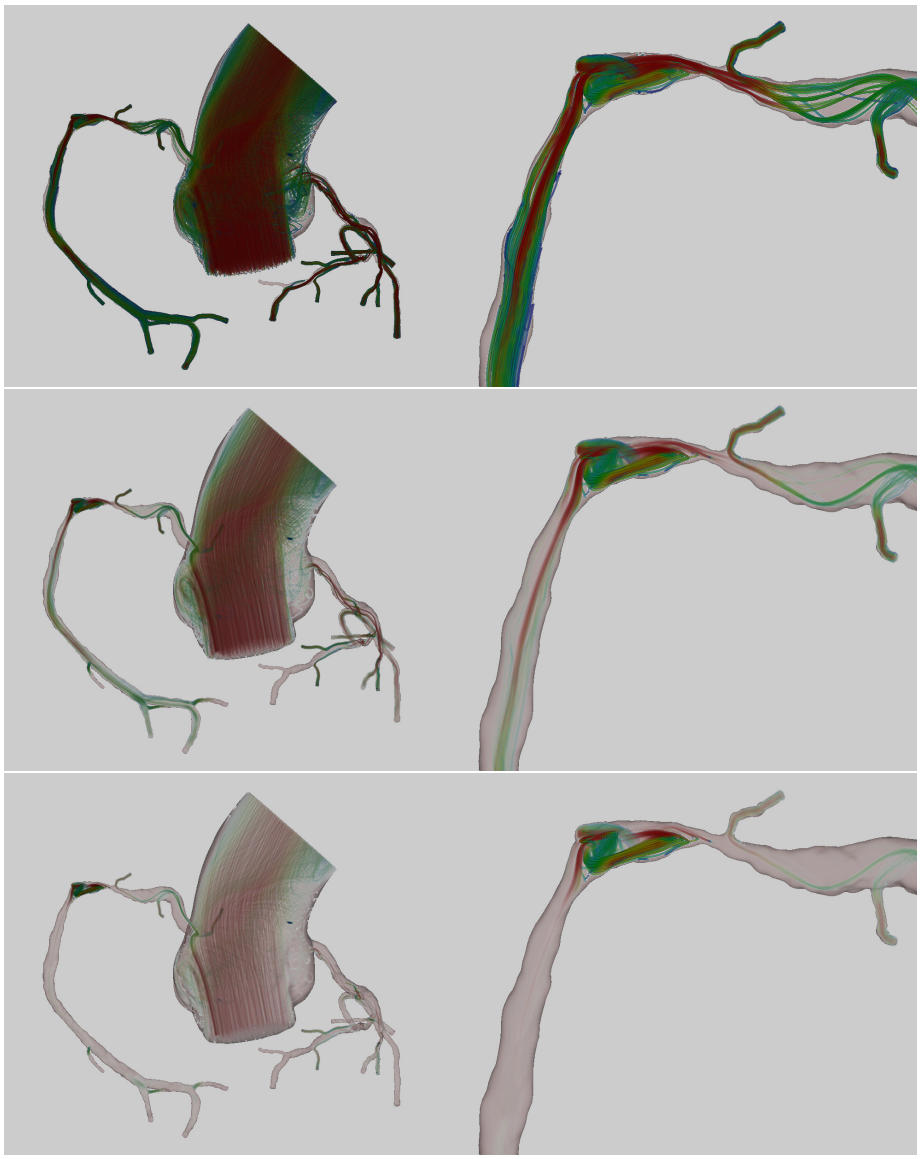


Figure 5.8 Flowlines rendered without and with opacity modulation. The first row shows the flowlines rendered using the fixed opacity value (1.0). The second and third rows illustrate the opacity-modulated flowlines with the control paramter of opacity adjustment in Eq. (4-15), n=1 (the second row), n=2.5 (the third row).

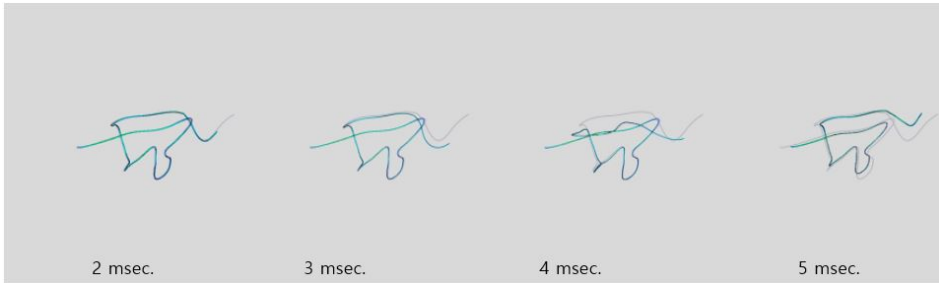


Figure 5.9 Four flowlines with time step of 2, 3, 4, and 5 msec. respectively are compared with the underlying gray line of time step of 1 msec. The flowline of time step of 2 msec is identical to the reference gray line, which means the convergence at the time step of 1 msec.

5.4 Parameter Study

In the following, we briefly discuss the practical issue that arise in the application of the methods described in this dissertation. The time step used to generate flowlines determines the spatial resolution between points constituting a flowline, which is related to both integration accuracy and computation time. Smaller time step improves the integration accuracy, although it causes longer calculation time. Since the accurate depiction of blood flow is more important, we endeavor to decide the optimal time step that ensures the integration accuracy to the best. In addition, the execution time can be reduced with GPU-acceleration of the flowline extraction which we have not implemented yet. The proper time step is chosen through the convergence test which evaluates by decreasing continuously the time step until the

generated flowlines are identical between the previous and current time steps as shown in Fig. 5.9. Through this convergence test, we determined 1 msec. as the optimal time step for extracting flowlines from the given flow dataset.

Chapter 6. Conclusion

In this thesis, we proposed a novel blood visualization method that improved seeding, tracing, and line-drawing by employing flow feature and vessel geometry. First, we presented the solutions to problems relevant to flowline extraction i.e., the automatic seed localization and the early terminated line integration. The inlet or outlet based seeding method automated the seeding process by detecting the inlet or outlet and by utilizing it as a seeding plane. As a consequence, the proposed approach led to a consistent and fast flow depiction by skipping a manual operation to locate the seeding plane. In addition, we presented flow based directional tracing and additional seeding method that contributed to length-extended flowline construction. Secondly, we provided a method that exhibited hemodynamics in a more informative way for assisting the diagnosis process. The proposed opacity modulation method attenuated the opacity of a flowline on a line segment basis according to its shape similarity with the vessel centerline. This approach made users pay more attention to abnormal flows as well as reduces line occlusion by depicting abnormal flow more noticeable, concurrently fading out normal flows predictable from the vessel geometry. In addition, we introduced HSV color coding to simultaneously exhibit a couple of flow attributes i.e., local speed and residence time. Experimental results demonstrated that the proposed method successfully constructed flowlines in narrow

branch vessels as well as vessels with locally reversed flows. As a result, our method permits clinicians to effectively grasp the entire flow behavior in vascular structures. Moreover, it enables a faster diagnosis through a noticeable depiction of abnormal flow.

Our approach is applicable to any kind of tubular structures with embedded flow information. Although the proposed seeding method employs the vessel inlet or outlet as a seeding plane, it can be extended to other anatomic landmarks of interests such as bifurcating branches, the conjunction between vessel trunk and its branch, and stiffly narrowed vessel region detectable by a more elaborate geometric analysis on vascular structures. This will enable a local and intensive inspection of hemodynamics in candidate geometries suspicious of vessel diseases. In the proposed method, the importance of flowlines was decided based on their geometric feature alone. In future work, we are planning to further enhance the accuracy of the importance metric by taking into account flow intrinsic factors (i.e., pressure, viscosity) as well. Our opacity modulation based on the shape similarity with the vessel centerline is suitable to coronary arteries with consistent flow characteristics, but not appropriate to the ascending aorta in which the flow pattern changes significantly during a cardiac cycle. Therefore, in order to evaluate the abnormality of flow in such vessels, we are planning to utilize the temporally changing statistical flow patterns instead of the fixed vessel centerline as a reference for shape matching.

Bibliography

[1] S S. Mendis, P. Puska, and B. Norrving, “Global Atlas on Cardiovascular Disease Prevention and Control,” World Health Organization in collaboration with the World Heart Federation and the World Stroke Organization. pp. 3–18. 2011.

[2] World Health Organization: Cardiovascular diseases (CVDs), Jan. 2015, http://www.who.int/mediacentre/factsheets/fs317/en/

[3] B. Kohler, R. Gasteiger, U. Preim, H. Theisel, M. Gutberlet and B. Preim, "Semi-automatic vortex extraction in 4D PC-MRI cardiac blood flow data using line predicates," IEEE Trans. on Visualization and Computer Graphics, vol. 19, no. 12, pp. 2773-2782.

[4] A. Hennemuth, O. Friman, C. Schumann, J. Bock, J. Drexl, M. Huellebrand, M. Markl, and H. O. Peitgen, “Fast Interactive Exploration of 4D MRI Flow Data,” In Proc. of SPIE Medical Imaging, pp. 489–492, Feb. 2011.

[5] J. R. Cebral, R. Pergolizzi, and C. M. Putman, “Computational Fluid Dynamics Modeling of Intracranial Aneurysms: Qualitative Comparison with Cerebral Angiography,” Academic Radiology, vol. 14, no. 7, pp. 804–813, 2007.

[6] R. Gasteiger, D. J. Lehmann, R. V. Pelt, G. Janiga, O. Beuing, A. Vilanova,

and B. Preim, “Automatic detection and visualization of qualitative hemodynamic characteristics in cerebral aneurysms,” IEEE Trans. on Visualization and Computer Graphics, vol. 18, no. 12, pp. 2178-2187, 2012.

[7] R. Gasteiger, M. Neugebauer, O. Beuing, and B. Preim, “The FLOWLENS: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms,” IEEE Trans. on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2183-2192, 2011.

[8] R. van Pelt, J. O. Bescos, M. Breeuwer, R. E. Clough, M. E. Groller, B.t.H. Romeny, and A. Vilanova, “Exploration of 4D MRI blood flow using stylistic visualization,” IEEE Trans. on Visualization and Computer Graphics, vol 16. no. 6, pp. 1339-1347 Oct. 2009.

[9] R. van Pelt, J. O. Besc´os, M. Breeuwer, R. E. Clough, M. E. Gröller, B. T. H. Romenij, and A. Vilanova. “Interactive virtual probing of 4D MRI blood-flow. IEEE TVCG, 17(12): pp. 2153–2162, Dec. 2011.

[10] M. Neugebauer, K. Lawonn, O. Beuing, P. Berg, G. Janiga, and B. Preim, “AmniVis–A System for Qualitative Exploration of Near-Wall Hemodynamics in Cerebral Aneurysms,” Computer Graphics Forum, vol. 32, no. 3, pp. 251-260, Jun. 2013.

[11] R. van Pelt, S. S. A. M. Jacobs, B. M. ter Haar Romeny, and A. Vilanova, “Visualization of 4D Blood-Flow Fields by Spatiotemporal Hierarchical

Clustering,” Comput Graph Forum, 31, pp.1065–1074, Jun. 2012.

[12] X. Ye, D. Kao, and A. Pang, “Strategy for seeding 3D streamlines,” IEEE Vis. pp. 471-478, Oct. 2005.

[13] S. Marchesin, C. Chen, C. Ho, and K. Ma, "View-dependent streamlines for 3D vector fields," IEEE Trans. on Visualization and Computer Graphics, vol. 16, no. 6, pp.1578-1586 Nov. 2010.

[14] S. Oeltze, D.J. Lehmann, A. Kuhn, G. Janiga, H. Theisel, and B. Preim, "Blood Flow Clustering and Applications in Virtual Stenting of Intracranial Aneurysms," IEEE Trans. on Visualization and Computer Graphics, vol. 20, no. 5, pp. 686-701. May 2014.

[15] K. Lawonn, T. Günther, and B. Preim, "Coherent View-Dependent Streamlines for Understanding Blood Flow," EuroVis, 2014.

[16] M. Neugebauer, G. Janiga, O. Beuing, M. Skalej, and B. Preim, “Anatomy-Guided Multi-Level Exploration of Blood Flow in Cerebral Aneurysms,” Computer Graphics Forum, vol. 30, no. 3, pp.1041-1050, Jun. 2011.

[17] T. Günther, K. Bürger, R. Westermann and H. Theisel, “A View-Dependent and Inter-Frame Coherent Visualization of Integral Lines using Screen Contribution.,” Vision, Modeling, and Visualization (VMV), pp. 215-222, 2011.

[18] T. Günther, C. Rössl, and H. Theisel, “Opacity optimization for 3D line

fields," ACM Trans. on Graphics (TOG), vol. 32 no. 4, 120:1-8, 2013.

[19] T. Günther, C. Rössl, and H. Theisel, "Hierarchical opacity optimization for sets of 3D line fields," Computer Graphics Forum, vol. 33, no. 2, pp. 507-516, May 2014.

[20] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser, "Illustrative flow visualization: State of the art, trends and challenges," Visibility-oriented Visualization Design for Flow Illustration, 2012.

[21] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen, "Over Two Decades of Integration-Based, Geometric Flow Visualization," In Computer Graphics Forum, vol. 29, no. 6, pp. 1807-1829, Blackwell Publishing Ltd., Sep. 2010.

[22] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, "The State of the Art in Flow Visualization: Dense and Texture-Based Techniques," In Computer Graphics Forum, vol. 23, no. 2, pp. 203-221, Blackwell Publishing Ltd., Jun. 2004.

[23] F. H. Post, B. Vrolijk, H. Hauser, R.S. Laramee, and H. Doleisch, "The state of the art in flow visualization: Feature extraction and tracking," Computer Graphics Forum, vol. 22, no. 4, pp. 775~792, Dec. 2003.

[24] T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Secheuermann, "The state of the art in flow visualization: partition-based techniques," In Simulation and

Visualization, pp. 75-92, 2008.

[25] B. Cabral and L. C. Leedom, “Imaging vector fields using line integral convolution,” In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 263-270. Sep. 1993.

[26] R. M. Kirby, H. Marmanis, and D. H. Laidlaw, "Visualizing multivalued data from 2D incompressible flows using concepts from painting," In IEEE Visualization, pages pp. 333–340, 1999.

[27] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In Proceedings IEEE Visualization ’99, pages 233–240. IEEE Computer Society, 1999.

[28] O. Mattausch, T. Theußl, H. Hauser, and E. Gröller, “Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines,” In Proceedings of the 19th spring conference on Computer graphics, ACM, pp. 213-222, Apr. 2003.

[29] I. A. Sadarjoen and F. H. Post, “Detection, quantification, and tracking of vortices using streamline geometry,” Computers & Graphics, vol. 24 no. 3, pp. 333-341, 2000.

[30] S. Born, M. Pfeifle, M. Markl, and G. Scheuermann, “Visual 4D MRI blood flow analysis with line predicates,” IEEE PacificVis Symposium, pp. 105-

112, Feb. 2012.

[31] R. Gasteiger, M. Neugebauer, C. Kubisch, and B. Preim, “Adapted Surface Visualization of Cerebral Aneurysms with Embedded Blood Flow Information,” VCBM, pp. 25-32, Jul. 2010.

[32] K. Lawonn, R. Gasteiger, and B. Preim, “Adaptive Surface Visualization of Vessels with Animated Blood Flow,” Computer Graphics Forum, vol. 33, no. 8, pp. 16-27, Dec. 2014.

[33] S. Born, M. Markl, M. Gutberlet, and G. Scheuermann, G. “Illustrative visualization of cardiac and aortic blood flow from 4D MRI data,” IEEE PacificVis Symposium, pp. 129-136, Feb. 2013.

[34] H. Krishnan, C. Garth, J. Gühring, M. A. Gülsün, A. Greiser, and K. Joy, “Analysis of time-dependent flow-sensitive PC-MRI data,” IEEE Trans. on Visualization and Computer Graphics, vol. 18, no. 6, pp. 966-977, Jun. 2012.

[35] S. K. Ueng, C. Sikorski, and K. L. Ma, “Efficient streamline, streamribbon, and streamtube constructions on unstructured grids,” IEEE Trans. on Visualization and Computer Graphics, vol. 2, no. 2, pp. 100-110, 1996.

[36] R. Löhner, and J. Ambrosiano, “A vectorized particle tracer for unstructured grids,” Journal of Computational Physics, 91(1), pp. 22-31, 1990.

[37] D. A. Lane, “UFAT: a particle tracer for time-dependent flow fields,” In Proceedings of the conference on Visualization'94. IEEE Computer Society

Press. pp. 257-264, Oct. 1994.

[38] D. N. Kenwright and D. A. Lane, "Interactive time-dependent particle tracing using tetrahedral decomposition," IEEE Trans. on Visualization and Computer Graphics, vol. 2, no. 2, pp. 120-129, Jun. 1996.

[39] Available at: https://en.wikipedia.org/wiki/Barycentric_coordinate_system

[40] P. J. Prince and J. R. Dormand, “High Order Embedded Runge-Kutta Formulae,” J. Computational and Applied Math., vol. 7, no. 1, pp. 67-75, 1981.

[41] D. Darmofal and R. Haimes, “Visualization of 3D vector Fields: Variations on a Stream,” AIAA 30th Aerospace Sciences Meeting and Exhibit, AIAA 92-0074, Reno, Nev., Jan. 1992.

[42] Partrick J. Lynch, Available at: https://commons.wikimedia.org/wiki/File: Coronary_arteries.svg

[43] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Levy, “Polygon Mesh Processing,” AK Peters, Natick, MA, USA, 2010.

[44] G. H. Golub and C. F. V. Loan, “Matrix computation,” Johns Hopkins University Press, 3rd edition, 1996.

[45] R. Bridson, “Fast Poisson disk sampling in arbitrary dimensions,” ACM SIGGRAPH, p. 5, 2007.

[46] J. S. Sobel, A. S. Forsberg, D. H. Laidlaw, R. C. Zeleznik, D.F . Keefe, I.

Pivkin, G. E. Karniadakis, P. Richardon and S. Swartz, “Particle flurries,” Computer Graphics and Applications, IEEE, 24(2), pp. 76-85, 2004.

[47] D. Han, et al., “MAP Vessel Tracking from CT Angiography”, PLOS. submitted. 2015.

[48] Available at: http://fblt.cz/en/skripta/x-srdce-a-obeh-krve/2-krevni-obeh/.

[49] I. Corouge, S. Gouttard, and G. Gerig, “Towards a Shape Model of White Matter Fiber Bundles Using Diffusion Tensor MRI,” Proc. IEEE Int’l Symp. Biomedical Imaging: Nano to Micro (ISBI), pp. 344-347, 2004.

[50] M. McGuire, and L. Bavoil, “Weighted Blended Order-Independent Transparency. Journal of Computer Graphics Techniques” (JCGT), vol. 2, no. 2, pp. 122-141. Dec. 2013.

[51] M. Borkin, K. Z. Gajos, A. Peters, D. Mitsouras, S. Melchionna, F. J. Rybicki, and H. Pfister, “Evaluation of artery visualizations for heart disease diagnosis," IEEE Trans. on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2479-2488, 2011.

[52] B. Bauer, P. Jolicoeur, and W. B. Cowan, “Distractor heterogeneity versus linear separability in colour visual search,” Perception, vol. 25, no. 11. pp. 1281-1294, 1996.

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

[54] M. Meyer, M. Desbrun, P. Schröder, and A. H.Barr, “Discrete differential-geometry operators for triangulated 2-manifolds,” In Visualization and mathematics III, pp. 35-57, Springer Berlin Heidelberg, 2003.

[55] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, “Laplacian mesh optimization,” In Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, pp. 381-389, Nov. 2006.

[56] K. Engel, M. Hadwiger, J.M. Kniss, C. Rez-Sajama, and D. Welskopr, “Real-Time Volume Graphics,” A K Peters Ltd., 2006.

[57] B.T. Phong, “Illumination for computer generated pictures,” Communications of the ACM, vol. 18, no. 6, pp.311-317, Jun. 1975.

[58] M. Zockler, D. Stalling, and H. C. Hege, “Interactive visualization of 3D-vector fields using illuminated stream lines,” IEEE Visualization'96. Proceedings, pp. 107-113, Oct. 1996.

[59] D. Stalling, M. Zöckler, and H. C. Hege, “Fast display of illuminated field lines. Visualization and Computer Graphics,” IEEE Transactions on, 3(2), pp. 118-128, 1997.

[60] H. G. Bogren, and M. H. Buonocore, “4 D magnetic resonance velocity mapping of blood flow patterns in the aorta in young vs. elderly normal subjects,” Journal of Magnetic Resonance Imaging, vol. 10, no. 5, pp. 861-

869, 1999.

[61] T. A. Hope, M. Markl, L. Wigström, M. T. Alley, D. C. Miller, and R. J. Herfkens, “Comparison of flow patterns in ascending aortic aneurysms and volunteers using four-dimensional magnetic resonance velocity mapping,” Journal of Magnetic Resonance Imaging, vol. 26. no. 6, pp. 1471-1479, 2007.

[62] S. Furuya and T. Itoh., “A streamline selection technique for integrated scalar and vector visualization”, IEEE Visualization, Poster Session, 2008.

초 록

혈류 데이터 획득 및 시뮬레이션 기술의 발달로 혈관 질환을 진단하고 치료하기 위한 용도의 혈류 가시화 기술이 의료 영상분야에서 보편적으로 사용되고 있다. 유적선 기반의 유동 가시화 방법은 장시간 유동 특성을 표현하는 장점으로 혈류 분석을 위해 가장 널리 사용된다. 이 방법은 속도 벡터장에 놓여진 유체 입자의 궤적을 추적함으로써 유동 특성의 시각적 표시를 위한 유적선을 추출한다. 그러나 이러한 방법을 혈관 구조에 적용할 때 다음과 같은 문제점이 있다. 첫 번째가 혈류선 추출을 위한 시작점을 결정하는 문제로, 기존 방법에서는 보통 수동으로 정해지며 그 결과 일관성이 결여된 가시화 출력을 보인다. 두 번째는 부분 역류와 협소한 벡터장으로 인해 혈류선 추출이 조기 종료되는 문제로, 혈관에 비해 짧은 혈류선을 생성한다. 마지막은 밀집된 형태의 혈류선 표시에 의한 혈류선 간 간섭 문제로, 혈관 안쪽에 위치한 혈류선을 가려서 혈류 분석을 어렵게 한다. 또한 의료 진단용 혈류 가시화에서는 혈관 질환과 관련된 비정상 혈류 위주로 표시하는 것이 중요하다. 따라서 본 논문에서는 유적선 기반의 혈류 가시화 과정에서 발생하는 문제점을 해결하고 추출된 혈류선을 의료 진단 목적에 유용하도록 가시화하는 방법을 제안한다. 혈류는 혈관 입구면과 출구면을 통과하고 혈관 벽과는 평행하게 흐른다는 점에 착안하여 혈류 속도 벡터와 혈관 표면 법선 벡터 간 직교성 판단에 의해 혈관 입구면 또는 출구면을 검출한다. 검출된 평면 위에 푸아송 원반 샘플링 (Poisson Disk Sampling)을 사용하여 혈류 시작점을 자동 생성함으로써 일관되고 빠른 혈류 가시화 결과를 제시한다. 또한 시작점의 혈류 방향에 따라 혈류선 추적 방향을 적응적으로 적용하고 비정상적 종료점에 시작점을 추가하는 방법을 사용함으로써, 혈류선 추출의 조기 종료를 방지하고 추적된 혈류선 길이를 확장하여 혈류 가시화 결과

를 충실하게 표현한다. 정상 상태 혈관에서는 혈관벽과 평행한 혈류 흐름을 보이며 질환 상태 혈관에서는 혈관 변형으로 인해 불규칙한 속도 및 방향을 갖는 난류가 관찰된다. 이러한 점에 착안하여, 혈관을 따라 흐르는 정상 혈류를 대변하는 혈관 중심선과 각 혈류선 간 형태 유사도에 따라 불투명도를 변조하여 혈류선을 표시한다. 이러한 불투명도 변조 방법은 혈류선 간 간섭과 시각적 혼란을 최소화하면서 혈관 질환을 암시하는 비정상 혈류에 더 주목하도록 가시화한다. 또한 HSV 색상 표시 방법을 도입하여 혈류 속도와 잔류 시간을 각각 색조(hue)와 포화도(saturation)에 대응 표시함으로써 복수의 혈류 정보를 동시에 인식하도록 한다. 제안 방법을 대동맥 및 관상 동맥 혈류 데이터에 적용한 결과, 제안 방법이 혈관 내 혈류 가시화에 적합함을 보였다.

주요어: 혈류 가시화, 적분 방식의 유동 가시화, 시작점 자동 선정, 유동선 추출, 불투명도 변조, 관상 구조
학번: 2012-30190