Physicalizing Cardiac Blood Flow Data via 3D Printing

Ang, Kathleen Danielle

http://hdl.handle.net/1880/111069

Downloaded from PRISM Repository, University of Calgary
UNIVERSITY OF CALGARY

Physicalizing Cardiac Blood Flow Data via 3D Printing

by

Kathleen Danielle Ang

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA
SEPTEMBER, 2019

© Kathleen Danielle Ang 2019
Abstract

Blood flow data from cardiac 4D Flow MRI holds much potential for research and diagnosis of flow-related diseases. However, understanding this data is quite challenging – after all, it is a volumetric vector field that changes over time. One helpful way to explore the data is by flow visualization, but most traditional flow visualizations are designed for 2D screens and thus suffer from limited depth perception and restricted screen space. We present a framework for creating physical flow models as a complementary means of visualizing the flow data. The model design respects the conventional method of viewing medical imagery (i.e. in cross sections) but has the advantages of engaging one’s sense of touch, and being fabricable by affordable 3D printers. We apply our technique to different representations of blood flow data and demonstrate that the framework is capable of transforming volumetric flow data into tangible, fabricable models.
Acknowledgements

Making it through grad school is probably a little bit like raising a child – “it takes a village”, as they say. So, this is about acknowledging that village.

First, a huge thanks goes out to my supervisor, Faramarz Samavati. I’m sure supervising students is tough, but it gets even tougher when you have someone who has little computer science background and skill. The countless times where he would take the time to teach me different concepts and mentor my progress through the program were invaluable. I was also privileged to be under the tutelage of some great professors in the courses I took – Sonny, Usman, Dr. P – thank you for all your help. Thank you especially to Dr. P and Lora for being my examiners and giving me valuable feedback; hopefully you had more fun reading this than I did writing it.

I also (literally) wouldn’t have survived without the generous funding from NSERC, Alberta Innovates, and our industry partner, Circle Cardiovascular Imaging Inc., so a warm thank you to these institutions.

Much gratitude is also owed to the villagers (the lab family). When I first started, I got a lot of help from Alex Brown, who went out of his way to help me fix the molasses-like code I had written. Along the way there was feedback and company from different labmates who have since moved on (Gaurav, Kamyar, Erika, Hasan, Mark, and Shima) – every bit of advice was interesting and useful to me. And to the rest of the math sci 6th floor, the ones who were around as I finished this off – I’m glad I took one year more than the supposed two, because the last one was the best one. From the random, to the mathematical, to the absolutely ridiculous, it has been a fun ride. The oft-times lunch crew (Andrew, Jeremy, Lee, Desmond, Philmo, Alex, Surbhi, Mik, Pascal, Dr. P, Sonny) was always around for mid-day entertainment. My lab family – Samin, my collaborator and friend to whom much is owed in this work, complete with many a coffee break; Troy, my office neighbour and eerily smart TA with an uncanny ability to nap in the lab; Ben, almost always ready for a good [conversation] complete with potentially controversial opinions; John, with his
positive attitude accompanied by very habitual eating habits (the soup was pretty good, I’ll admit); Mia, full of smiles, laughter, jokes, and quips (just not too early in the day); Lakin, who spent many an extra hour helping me work out all the math that I struggled with; Hooman, ready for a friendly conversation along with door-pull-up shenanigans; Fatemeh, always helping with the greater community and also getting us the inside track on free shirts; Hessam (and Mahsa), with whom I can struggle through the nuances of 3D printing, or nuances of the English language, to name a few; Tim, who always came up with interesting questions during lab meetings; and Xi, my supposed protege who has already surpassed me – it’s been a good time. Yes, I did just make that one giant run-on sentence. A shout-out also goes to our post-docs, old and new – Mohammed Elbaz, for much of his mentorship in the early stages of my work, and Katy for her perspective and help on many things, especially relating to user studies and ethics applications. Also, many thanks to Julio Garcia Flores, our collaborator at Libin – so much of this work could not have happened without his support and guidance.

Finally, much thanks to my family outside the lab – all the dinners, home meetings, Bible studies and fellowship times kept me grounded, even amidst the craziness that grad school can bring. These are the things that have eternal worth.

To my home – Jacky and Teddy Bear – it’s finally done. Thank you for your patience, encouragement and always being a willing audience for many a presentation.

“Let everything that has breath praise Jehovah. / Hallelujah!” (Psa. 150:6)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>List of Symbols</td>
<td>xi</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem statement and challenges</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Goals and scope</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Method</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Contribution</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Thesis organization</td>
<td>8</td>
</tr>
<tr>
<td>2 Background and related work</td>
<td>10</td>
</tr>
<tr>
<td>2.1 The heart</td>
<td>10</td>
</tr>
<tr>
<td>2.1.1 Cardiac anatomy and physiology: basics</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2 A broken heart</td>
<td>12</td>
</tr>
<tr>
<td>2.1.3 Cardiac imaging: CMR and blood flow imaging</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4 Cardiac imaging: 4D Flow MRI</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Flow</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 Flow vector fields</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 Flow visualization: glyphs</td>
<td>19</td>
</tr>
<tr>
<td>2.2.3 Flow visualization: stream objects</td>
<td>20</td>
</tr>
<tr>
<td>2.2.4 Vortices</td>
<td>26</td>
</tr>
<tr>
<td>2.3 Scientific visualization</td>
<td>30</td>
</tr>
<tr>
<td>2.3.1 Related work: blood flow visualization</td>
<td>31</td>
</tr>
<tr>
<td>2.4 Physicalization</td>
<td>32</td>
</tr>
<tr>
<td>2.4.1 3D printing</td>
<td>34</td>
</tr>
<tr>
<td>2.4.2 Related work: physicalizing motion/flow</td>
<td>35</td>
</tr>
<tr>
<td>2.4.3 Related work: 3D printing in medical applications</td>
<td>36</td>
</tr>
<tr>
<td>2.4.4 Related work: surface representation</td>
<td>37</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>39</td>
</tr>
<tr>
<td>3 Multi-slice models</td>
<td>41</td>
</tr>
<tr>
<td>3.1 Slice-based design</td>
<td>43</td>
</tr>
<tr>
<td>3.2 Glyph model</td>
<td>44</td>
</tr>
<tr>
<td>3.2.1 Physicalization of relative context</td>
<td>46</td>
</tr>
<tr>
<td>3.2.2 Fabricated glyph model</td>
<td>47</td>
</tr>
<tr>
<td>3.3 Streamline model</td>
<td>48</td>
</tr>
<tr>
<td>3.3.1 Generating streamlines</td>
<td>49</td>
</tr>
<tr>
<td>3.3.2 From lines to meshes</td>
<td>51</td>
</tr>
<tr>
<td>3.3.3 Fabricated streamline model</td>
<td>53</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>54</td>
</tr>
<tr>
<td>4 Anatomical context</td>
<td>55</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Summary of glyph model Likert questionnaire. . . . . . . . . . . . . . . . . . . . 73
5.2 Summary of streamline model Likert questionnaire. . . . . . . . . . . . . . . . . . 73
### List of Figures and Illustrations

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Cardiac magnetic resonance images – multiple 2D images represent a 3D volume. (Screen capture of software program Segment [Heiberg et al., 2010].)</td>
</tr>
<tr>
<td>1.2</td>
<td>Some initial tests: (A) Attempting to print a small sample of thin arrow glyphs failed due to the many fragile features; and (B) Attempting to print a small sample of streamlines shows that thin tubes are quite breakable.</td>
</tr>
<tr>
<td>2.1</td>
<td>Heart anatomy and simplified schematic of blood flow within the heart. (Both diagrams courtesy of BruceBlaus, CC BY 3.0, [Blausen.com staff, 2014].)</td>
</tr>
<tr>
<td>2.2</td>
<td>Different types of cardiomyopathy. (Diagram courtesy of Nicholas Patchett - CC BY-SA 3.0)</td>
</tr>
<tr>
<td>2.3</td>
<td>Four chamber long-axis view.</td>
</tr>
<tr>
<td>2.4</td>
<td>Example of raw 4D Flow MRI data for one timeframe. (Image adapted from Markl et al., 2011, CC BY 2.0)</td>
</tr>
<tr>
<td>2.5</td>
<td>Vortex core (red isosurface) of a sample subject, detected during early filling. (Image adapted from Elbaz et al., 2014, CC BY 4.0)</td>
</tr>
<tr>
<td>2.6</td>
<td>Basic idea of vector field function ( \mathbf{f} : D \rightarrow \mathbb{R}^3 ), illustrated with one point ( s_0 \in D ).</td>
</tr>
<tr>
<td>2.7</td>
<td>Two-dimensional vector field data, shown with arrow glyphs.</td>
</tr>
<tr>
<td>2.8</td>
<td>Physical example of streamlines: iron filings aligning in the magnetic field created by a bar magnet. (Courtesy of Berndt Meyer, CC BY-SA 3.0)</td>
</tr>
<tr>
<td>2.9</td>
<td>Physical example of pathlines: lines drawn by car lights in long exposure photography. (Photo free for use, CC0 1.0)</td>
</tr>
<tr>
<td>2.10</td>
<td>Difference between streamlines, pathlines and streaklines in a simple 2D vector field example. (Diagram inspired by “Vector Field Visualization” slides, T. Möller and H. Shen)</td>
</tr>
<tr>
<td>2.11</td>
<td>Visualizing wind data with varying numbers of streamlines.</td>
</tr>
<tr>
<td>2.12</td>
<td>Other stream objects: stream ribbons and stream surfaces.</td>
</tr>
<tr>
<td>2.13</td>
<td>Example of line integral convolution texture, which illustrates the vector field behaviour. (Image courtesy of Anders Sandberg - CC BY-SA 3.0)</td>
</tr>
<tr>
<td>2.14</td>
<td>Visualizing a scalar elevation field using a surface.</td>
</tr>
<tr>
<td>2.15</td>
<td>Examples of 3D vector field visualization using glyphs (left) and streamlines (right). (Image courtesy of van der Geest and Garg, 2016, CC BY 4.0)</td>
</tr>
<tr>
<td>2.16</td>
<td>Physicalization of geospatial data via affordable 3D printing. (Image courtesy of Allahverdi et al., 2018, ©2018)</td>
</tr>
<tr>
<td>2.17</td>
<td>Surface extracted from MRI data using the marching cubes algorithm. (Image courtesy of wikimedia user Dake - CC BY-SA 2.5)</td>
</tr>
<tr>
<td>3.1</td>
<td>Overview of multi-slice physical visualization framework.</td>
</tr>
<tr>
<td>3.2</td>
<td>The left ventricle (highlighted in red in the short axis images on the left) can be segmented from MR image data (a stack of 2D slices which make up a 3D volume). Such a segmentation represents the chamber’s volume. For physicalization, this volume can be “sliced” into thin slabs (shown on the right), which provide a natural support structure for physicalization while also maintaining a sense of the data.</td>
</tr>
</tbody>
</table>

viii
3.3 Arrow glyphs have small, fragile parts (circled in black), whereas cones can portray directional information but are not as breakable.

3.4 Cone glyphs are embedded within a slice for printability.

3.5 Each slice model includes a supplementary handle (shown in blue, top), which allows for easy assembly into a stack of slices (bottom).

3.6 Picture of fabricated left ventricle glyph model during late filling (late diastole).

3.7 Picture of fabricated left ventricle late filling (left) and early filling (right) vector field glyph models; both are sliced such that corresponding slices can be compared between the two.

3.8 The similarity distance is measured in a local neighbourhood at the point \( p \) where the streamline \( S_i \) is growing.

3.9 Naïvely growing all streamlines for a given set of seed points results in a very cluttered visualization (left), whereas applying the similarity distance metric when generating the streamlines from the same seed points results in a cleaner, physicalizable collection (right).

3.10 We build each streamline mesh by sweeping a circular cross section along the streamline (left). Rather than having an ambiguous streamline mesh (middle), we add a conical arrowhead (right) to indicate the direction of the streamline.

3.11 Picture of the printed left ventricle streamline model (early filling).

4.1 Comparison of surface reconstruction using marching cubes (left) vs. our method (right).

4.2 High level overview of LV surface reconstruction process: (A) Input data, a volumetric binary mask. (B) Initialization - an initial mesh \( M_0 \) is placed within the object; surface points are shown in black. The surface mesh evolves, somewhat similar to a balloon. (C) Triangles highlighted in yellow indicate fronts which are active. (D) Active fronts are adaptively subdivided. (E) The final coarse surface mesh. (F) The final subdivided and smoothed mesh, \( M_F \).

4.3 Illustration of distance field: grayscale values reflect distance field values, black (0) approximates the target surface.

4.4 Detecting approximate surface points based on binary mask data [left]: a given pixel (voxel in 3D) is flagged as a surface point [middle] if its 4-connected neighbours include at least one 1 and one 0. Then a signed distance field is calculated based on the surface points [right], here illustrated with Manhattan distances.

4.5 2D illustration of placing \( M_0 \): point \( p \) (shown with a star) marks the “innermost” point within the object, and \( D(p) = m \) is used to initialize the radius (for instance, \( r = m/2 \)). \( M_0 \) is shown with a red circle.

4.6 A mesh may be near the approximate surface points \( \mathcal{P} \) but the surface is not adequately reconstructed. (Note: black pixels in the distance field represent approximate surface points.)

4.7 Surface fitting steps.

4.8 Schematic illustrating how vertex normals are scaled based on the distance map \( D \) (shown as a greyscale grid in this image). This “pushes” the vertex towards the approximate surface points.
4.9 The vertex \( v_i \) (highlighted in red) is pushed toward the centroid of its one-ring neighbours (shown in grey). .................................................. 64

4.10 A representation of the LV anatomy can be included as part of the physical flow model to provide context. .................................................. 67

4.11 LV mesh (red) and thin box (blue) can be intersected [B] to generate a smooth long-axis slice [C]. .................................................. 68

5.1 (A) User study setup, with screenshots of corresponding digital visualizations for (B) the glyph models and (C) the streamline model. .......................... 71

5.2 For both the glyph model comparative tasks (top) and the streamline model comparative tasks (bottom), most users were faster using the physical model (blue) over the digital model (grey). Three users chose the incorrect streamline when working with the digital model (red). .................................................. 77

6.1 Two colour glyph model, printed using MakerGear M3-ID. ................................. 79

6.2 Screenshots from Paraview showing (A) unfiltered streamlines (generated within Paraview), and (B) the filtered streamline model for printing. Pictures of the fabricated single slice left ventricle streamline model in (C) its final form, and (D) printed in two halves. .................................................. 81

6.3 The number of pathlines is reduced to a simple set which summarizes the main flow path over the cardiac cycle. .................................................. 82

6.4 The mesh generated for a pathline involves scaling the cross sectional radius as it is swept along the pathline; this gives the path tube a tapered appearance. .......... 83

6.5 Picture of 3D printed pathline predicate model for a healthy subject. ..................... 83

6.6 Pictures of healthy vortex core summary model. (A) Long-axis view, with vortex cores marked by arrows. (B) Short-axis view, a vortex core ring structure (C) can be observed. .................................................. 85

6.7 Comparison between LV of healthy subject (left) to cardiomyopathy patient (right) in photos of (A) two printed streamline (half-)models, (B) pathline predicate models, and (C) vortex core summary models. .................................................. 87
### List of Symbols, Abbreviations and Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRI</td>
<td>Magnetic resonance imaging</td>
</tr>
<tr>
<td>CMR</td>
<td>Cardiovascular magnetic resonance</td>
</tr>
<tr>
<td>PC(-MRI or -CMR)</td>
<td>Phase contrast</td>
</tr>
<tr>
<td>3D</td>
<td>Three dimensional</td>
</tr>
<tr>
<td>4D</td>
<td>Four dimensional</td>
</tr>
<tr>
<td>LV</td>
<td>Left ventricle</td>
</tr>
<tr>
<td>FDM</td>
<td>Fused deposition modelling</td>
</tr>
<tr>
<td>FFF</td>
<td>Fused filament fabrication</td>
</tr>
<tr>
<td>VSD</td>
<td>Ventricular septal defect</td>
</tr>
<tr>
<td>venc</td>
<td>Velocity encoding range</td>
</tr>
<tr>
<td>RKF-45</td>
<td>Runge-Kutta-Fehlberg method</td>
</tr>
<tr>
<td>STL</td>
<td>Abbreviation for stereolithography; common mesh file format</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

A young child lays on the hospital bed, various cords connected to machines surrounding her. A rhythmic beeping is heard, the number 100 flashes on the screen. Another machine displays two numbers, 95/60. They are monitoring her heart rate and blood pressure, her vital signs; a drastic change will trigger an alarm, a cry for help. These vital signs represent the well-being of her cardiovascular system, the system that is responsible for circulating blood through her entire body – a system so critical that the continuous success of its operation can be the difference between life and death.

Given the importance of the cardiovascular system, it is natural that we desire to understand it. From both medical and scientific perspectives, the human body fascinates our intellect. And with present-day technology, we have a window into our fascination: medical imaging. Medical imaging illuminates the unseen in modern-day medicine; it empowers our ability to understand the human body in both its beauty and its struggles, with usage in areas such as research, diagnosis, and communication. However, the resulting medical imagery is not always easy to understand. A dynamic three-dimensional organ such as the heart might be captured as a stack of two-dimensional images (Fig. 1.1). It is left to the mind of the radiologist to reconstruct this complex geometry, which is not always a simple feat – and even if it becomes natural for a physician after years of practice, explaining such images to a patient in a way that they will truly understand remains a challenge. This becomes much more difficult when abnormalities appear, such as small holes between heart chambers that should not exist.

To alleviate this challenge, the medical research community is increasingly turning to three-dimensional (3D) visualization formats. Such formats include 3D digital models and 3D printing (additive manufacturing), which are tools that improve understanding of intricate spatial relation-
ships. While digital models are conventionally still viewed on 2D screens, 3D printing allows physicians, nurses, trainees, and patients to engage with the data in a tangible manner. In the last two decades, there has been a marked increase in the use of 3D printed models in medicine; prints of patient-specific anatomy can improve patient and trainee education, or can even be used for presurgical planning and surgery rehearsal. Research in this area has shown much promise and there are now ongoing multicenter studies looking to establish improvements in clinical outcomes.

While 3D printing of cardiovascular anatomy is continuing to develop, we must note that cardiology is not just about anatomical structure. In fact, Richter and Edelman [2006] would argue that “cardiology is flow”; the pumping mechanism of the heart drives, controls, and maintains transport of blood through an immense vascular network of arteries, veins, and capillaries around the body. It is about flow, and this flow has much to do with the health of a person: it can impact remodelling of anatomical structures, cause lesion formation, and can effectively serve as the metaphorical red light, warning you of problems to come. Since flow is so crucial, it makes sense that recent advances in imaging technology have reached a point where blood flow data in the heart and great vessels can be acquired. One such technology, known as 4D Flow MRI (magnetic resonance imaging), captures time-varying three-dimensional data that represents the magnitude
and direction of blood flow [Markl et al., 2012]. 4D Flow MRI holds much potential for research and diagnostic tests of flow-related diseases [Dyverfeldt et al., 2015], but to really understand this data we need a way to visualize it.

Due to the complexity of the data – it is volumetric, time varying, and encodes vector field information – creating good visualizations can be quite challenging. Common ways for visualizing vector field data include using glyphs as a visual representation of the vectors at each point in space, or using integral lines such as streamlines or pathlines, which are tangent to the vector field at every point and can depict complex flow patterns. Some medical imaging software programs are also equipped with tools for volume rendering, accompanied by interaction techniques such as rotating, zooming in/out, and panning. Although such programs are beneficial, there are some drawbacks: it can take some time to learn how to use the software (even interaction with mouse/keyboard can limit accessibility to a broader audience), and the visualizations are ultimately displayed on a 2D screen. Thus, challenges with screen space, depth perception, and lack of tangibility are inevitable.

Rather than navigating the difficulties imposed by 2D screen visualizations, we propose the use of physical visualization or physicalization (we use these terms interchangeably) as a technique to represent blood flow data. A fabricated model offers natural depth perception (since it inherently exists in 3D space) as well as the advantage of tangibility. There are a number of reported benefits of physical visualization, such as enabling active perception, appealing to non-visual senses, making data more accessible to a broader audience (e.g. those who are visually impaired or who are less amenable to digital technology), and engaging people [Jansen et al., 2015].

The idea of physicalizing medical flow data is very much in the same vein as 3D printing anatomical structures. In fact, synthesis of data from different sources (e.g. anatomical structures from cine SSFP (steady-state free precession) MRI and blood flow from 4D Flow MRI) is often desirable for increased understanding and analysis; hence, physicalization of blood flow holds promise even for enhancing the more typical anatomical 3D printed models in medicine. But despite the large body of work on this topic, to our best knowledge there have been no studies
exploring the use of 3D printing for visualization of blood flow data.

1.1 Problem statement and challenges

Blood flow data is critical in understanding the health and function of the cardiovascular system. Visualization of said data assists our interpretation of the data, yet traditional flow visualization is confined to a flat 2D screen even though the blood flow is indeed volumetric. For pathological anatomy, researchers have investigated physicalization of the data via 3D printing to help with comprehending the data; such physical models have been shown to enable better patient education and improved surgical planning. But what about blood flow? Herein lies the overarching question which we attempt to answer in this thesis: “Can we use affordable 3D printing to visualize blood flow information?”

Indeed, physicalizing volumetric vector field data is challenging. To encourage accessibility to a wider audience, we chose to use affordable material extrusion 3D printers (often referred to as “fused deposition modelling (FDM)” or “fused filament fabrication (FFF)” printers). From here, we must consider how vector field data can be visualized. Typical flow visualization styles such as vector glyphs or streamlines are common for digital formats, but it is not straightforward to fabricate a comparable physical visualization with potentially many glyphs, or long and thin streamlines, since such fine features are susceptible to breakage \cite{Allahverdi2018}. Moreover, there are other subtleties to consider when creating a physical visualization – unlike virtual visualizations, physical models are subject to gravity and need appropriate mechanisms for existing in the real world. The inability to print in “thin air” necessitates support structures that can be removed after printing to circumvent the issue; however, this assumes that the model can eventually be free-standing without the support structures. This is certainly not guaranteed for any volumetric vector field dataset, represented with either glyphs or a collection of integral lines. Hence, naïvely attempting to print some glyphs or streamlines is essentially guaranteed to fail (Fig. 1.2).

Aside from these inherently physical limitations, we encounter another challenge which many
visualizations face: clutter. In the digital realm, a multitude of visual objects can be generated but this usually creates clutter – ultimately obstructing data from the user’s view, making the visualization convoluted, and limiting its utility. Good visualizations, therefore, aim to reduce clutter without sacrificing too much information. In the physical realm, extremely cluttered visualizations are not only incomprehensible, they are not even fabricable – for instance, a one-to-one mapping between flow vector field data and 3D printed visual objects (e.g. a glyph) is basically impossible. Thus, for the sake of both printability and comprehension, some kind of filtering is required.

1.2 Goals and scope

The primary goal of this thesis can be summarized as follows: Design and fabricate a 3D printable physical visualization to represent blood flow data.

We can unpack this main goal into the following sub-goals:

- Our physical visualization should be designed such that it is fabricable by an affordable desktop 3D printer. We specifically use a material extrusion 3D printer, often known as an FDM or FFF printer.
The physicalization(s) should depict some facet of the blood flow data. To this end, different visualization styles can be used (much in the same way that traditional flow visualization leverages various techniques such as vector glyphs, streamlines, pathlines, etc.).

The final fabricated model should also convey contextual information in some way – it is not just flow anywhere, but flow within a particular chamber of the heart.

With these goals in mind, we also clarify the scope of this thesis. Since physicalization of blood flow has not been extensively studied, we chose to focus on the left ventricle (a particular and most critical chamber of the heart – after all, it is responsible for pumping oxygenated blood to the rest of the body). By narrowing the scope in this way, we can explore different visualization styles. Moreover, as flow physicalization is the focus, our methods do not focus on segmenting the anatomical structure and/or flow data. Our input data has already been segmented – usually semi-automatically, since this is typical in medical research. Finally, this work is intended to demonstrate proof of concept: the emphasis is placed on creating and fabricating a physicalizable design. As such, it is not yet designed for routine clinical use and validation in a clinical context is outside the scope of this thesis; to reach this stage, much more extensive research would be required.

Given our scope, we can re-state our primary goal with some added detail: Design and fabricate 3D printable physical visualizations which represent (some aspect of) blood flow data within the left ventricle.

1.3 Method

Our approach to the problem of physicalizing blood flow data involved iterations of design, testing, and evaluation. The overall methodology revolved around three main building blocks: (1) visualization style, (2) filtering, and (3) slicing. Since flow data can be represented in a variety of ways, we explored different visualization styles (e.g. glyphs or integral lines). For each style, we
would approach the challenges of its physicalizability through some combination of filtering and/or slicing. In this section, we first explain the notion of “slicing”, then briefly describe the overall research method undertaken over the course of this research.

To tackle the core challenges of 3D printing flow data, we were inspired by traditional methods that are natural to most medical professionals: slice-based (i.e. cross section-based) visualization. Typical practice in the medical community involves viewing “slices” or cross sections (2D images) of the data, as this better supports detailed analysis [Tietjen et al., 2006] and is the format of most medical images, even if a volumetric space was acquired. This concept of slicing became a key “hammer” that we used in building our flow physicalizations.

Our initial application of slicing was somewhat akin to slicing a loaf of bread. The basic idea is to slice the volumetric dataset or set of visual objects (the bread) into several subsets such that they give an overall picture of the data when combined together. We applied this multi-slice concept to two styles of flow visualization: glyphs and streamlines. For both styles, we aimed to create clear, uncluttered visualizations that are 3D printable and give an overview of the flow behaviour. We achieved this by using filtering techniques that are dependent on the chosen visualization style. Aside from filtering and slicing, we found that creating some kind of physical context made understanding the behaviour of the flow more convenient. To this end, we designed a simple stand which included a representation of the inner surface of the left ventricle; this helped maintain a sense of anatomical context.

With these two styles of flow physicalization (multi-slice glyph and streamline models) in hand, we sought to evaluate our physical prototypes by conducting a user study. The chief intent of our study was to observe and understand how users would interact with tangible flow visualizations, especially in comparison with similar digital flow representations. Since this work is quite exploratory in nature, the user study was also kept relatively open-ended; qualitative responses were a valuable part of the study’s results.

Such results and general observations from the user study fed into a second iteration of physical
visualization design. One of the main issues raised in the user study was the multi-slice design for the streamline model – while it seemed natural for the physical glyph model, several participants felt it was awkward for the streamline model. Consequently, we re-designed the streamline model such that only a single slice was used; this provided adequate support for the physical streamline objects and added a better sense of anatomical context. We then asked for feedback from a few cardiovascular experts, which led to the idea of creating simplified summary models. Thus, we explored two more visualization styles: pathlines and vortex cores. These two model designs are intended to provide a higher level perspective of the flow behaviour within the left ventricle.

1.4 Contribution

The main contributions of this work include the following:

- Developing a new framework for physical visualization of cardiac blood flow from 4D Flow MRI data using affordable/accessible 3D printers.

- Evaluating the feasibility of such a framework by fabricating several example physical models using affordable 3D printing. These models represent blood flow data from an actual human heart (specifically the left ventricle).

- Comparing the usability of the developed physical visualization against conventional digital 3D flow visualization formats by conducting a user study.

The methods described in this thesis (especially those covered in Chapters 3, 5, and 6) have also been accepted for publication in the journal *Computer & Graphics* under the title “Physicalizing cardiac blood flow data via 3D printing” (K.D. Ang, F.F. Samavati, S. Sabokrohiyeh, J. Flores, and M.S. Elbaz).

1.5 Thesis organization

The remaining chapters in this thesis are organized as follows:
Chapter 2 describes background information (particularly in the areas of heart anatomy and physiology and imaging, flow basics, scientific visualization, and physicalization) and related work, specifically in the areas of blood flow visualization and flow/motion physicalization.

Chapter 3 details the development of two models: (1) the multi-slice glyph model, and (2) the multi-slice streamline model. Design and fabrication are described for both.

Chapter 4 provides an explanation of creating anatomical context, namely, a smooth representation of the left ventricle endocardial surface.

Chapter 5 explains our user study design and results.

Chapter 6 outlines further visualization styles. It includes improvements on the models described in Chapter 3, as well as new designs based on expert feedback.

Chapter 7 reports key implementation details for all aspects of the work.

Finally, Chapter 8 concludes with a summary of the work described and ideas for future work.
Chapter 2

Background and related work

The work presented in this thesis overlaps several areas of study: the human heart and medical imaging, flow, scientific visualization (particularly, flow visualization), and physicalization (with an emphasis on 3D printing). We discuss the necessary background information in each topic, as well as related work in the realm of 4D Flow MRI blood flow visualization (Section 2.3.1, a subsection of scientific visualization), and in the area of physicalization: flow/motion physicalization (Section 2.4.2), medical 3D printing (Section 2.4.3), and creating smooth surface representations from volumetric data (Section 2.4.4).

2.1 The heart

The circulatory system, often also known as the cardiovascular system, enables rapid transport of molecules such as oxygen within the human body. Three components are key to this system: blood, blood vessels, and the heart. In this section (much of which is based on material from Drake et al. [2010], Widmaier et al. [2011]), we discuss some background information on the heart and blood flow within the heart, a few examples of diseases that are related to the left ventricle, and some general cardiac imaging basics. This will bring us to one particular imaging modality which is used for capturing blood flow information: 4D Flow MRI.

2.1.1 Cardiac anatomy and physiology: basics

In its essence, the heart is a muscular organ located in the chest. It is contained within a fibrous sac known as the pericardium. The heart wall itself, known as the myocardium, is principally made up of cardiac muscle cells. The inner walls of the heart are known as the endocardium. Overall, the heart can be divided into two halves – the left and the right – each of which has an atrium
and a ventricle (Fig. 2.1). Thus, the heart is composed of four main chambers. Between each atrium/ventricle pair (on the same side) there are one-way atrioventricular valves, and between the ventricles there is a muscular wall known as the interventricular septum. These structures help direct the flow of blood appropriately.

![Anatomy of the Heart and Normal Blood Flow](image)

Figure 2.1: Heart anatomy and simplified schematic of blood flow within the heart. (Both diagrams courtesy of BruceBlaus, CC BY 3.0, [Blausen.com staff][2014])

The heart functions as a double pump to move blood in the human body. One pump – the right atrium and right ventricle – receives deoxygenated blood from the body and pushes it to the lungs for oxygenation. The second – the left atrium and left ventricle – receives oxygenated blood from the lungs and pushes it to the rest of the body. Blood travels through the body through a network of vessels and delivers oxygen, eventually becoming deoxygenated and returning to the right atrium, where the cycle repeats. Pumping of blood through the heart can be divided into two phases: systole (when the ventricles contract, pushing blood to the lungs/body) and diastole (when the ventricles relax, allowing blood to flow into these chambers). A complete cycle of diastole and systole is known as a cardiac cycle.

Due to the complexity of the heart’s structure and function, we chose to focus on the left
ventricle (LV) in this thesis. What happens in the LV during one cardiac cycle? During diastole, blood flows from the left atrium through the mitral valve (sometimes also referred to as the bicuspid valve) toward the LV apex. During systole, the LV contracts and blood flows through the aortic valve to the aorta, which delivers the blood to the rest of the body via a network of blood vessels.

2.1.2 A broken heart

An organ as complex as the heart can fail in many ways; hence, we limit our discussion to a few examples particularly relevant to the LV: cardiomyopathy, valvular disease, and ventricular septal defect (VSD). Cardiomyopathy is a disease of the heart muscle and can be categorized as either dilated, hypertrophic, or restrictive. In dilated cardiomyopathy, the LV is dilated (enlarged) and contraction during systole is impaired. In hypertrophic cardiomyopathy, the ventricular walls are thickened and relaxation is impaired; sometimes the interventricular septum causes obstruction of the left ventricular outflow tract. In restrictive cardiomyopathy, the elasticity of the heart muscle is impaired, thus causing reduced diastolic filing. Mitral valve disease is most commonly some combination of incompetence (insufficiency) where the valve functions poorly, and/or stenosis (narrowing) where the valve does not open fully. Aortic valve disease (either stenosis or regurgitation) can result in marked heart failure. VSD is a congenital condition where a hole exists between the left and right ventricles. This causes inefficient short-circuiting of blood flow: some of the oxygenated blood is pumped into the right ventricle and back to the lungs, rather than the rest of the body.

2.1.3 Cardiac imaging: CMR and blood flow imaging

Various imaging modalities, such as CT (computed tomography), MRI, and ultrasound, can be used to capture data about the cardiovascular system, e.g. heart structure and function. In this section we broadly provide some background on two main topics: cardiovascular magnetic resonance (CMR) imaging, and different methods for capturing blood flow data. The following section will discuss one specific technique for capturing flow data which is known as 4D Flow MRI; this was
the primary means of data acquisition used in this thesis.

To capture MR images, the patient lays within the bore of a very powerful magnet, which causes hydrogen atoms (abundant in water and fat molecules) to align themselves within the magnetic field \cite{Novelline2004}. Radio waves are then passed through the patient in specific sequences of short pulses, causing the protons (hydrogen) to be knocked out of alignment. As they return to equilibrium (i.e. aligned with the magnetic field), they emit the absorbed radio-frequency (RF) waves; these waves are analyzed by the computer to produce MR images.

In MRI, images are acquired as tomographic slices through any selected plane. There are a number of standard planes for imaging, which we briefly explain here \cite{Nacif2010}. At the beginning of an MR exam, a few localizing sequences based on thoracic planes (i.e. coronal, sagittal and axial) are used to scout the location of the heart. The heart itself does not align with the body’s typical planes (one can think of an object’s coordinate system within the world coordinate system), so from these scouting images the technologist can plan two types of “heart-centric” long-axis views (two-chamber and four-chamber). The four-chamber view allows for identification

Figure 2.2: Different types of cardiomyopathy. (Diagram courtesy of Nicholas Patchett - CC BY-SA 3.0)
of the left and right atria and ventricles, as described in the previous section (see Fig. 2.3). Once the long-axis views have been obtained, short-axis views (orthogonal to the long axis of the LV) can be prescribed from the base of the LV to the apex.

![Figure 2.3: Four chamber long-axis view.](image)

The heart beats. Its dynamic nature introduces an element of complexity when trying to capture it, especially when compared to imaging other parts of the body. To address this challenge, imaging needs to be synchronized with electrocardiogram (ECG). This can be done in one of two ways: prospective gating or retrospective gating [Tseng et al., 2016]. In prospective gating, the ECG R-wave is used as a trigger to acquire the data. In retrospective gating, the data is acquired continuously and then reassigned based on the relative timing of the ECG.

To capture motion of the heart structure over the cardiac cycle, cine MRI can be used [Tseng et al., 2016]. At present, cine MRI typically uses balanced steady-state free precession (bSSFP) with retrospective gating. This technique acquires images with high tissue contrast between myocardium and blood, allowing for assessment of cardiac function. For instance, ventricular function is often assessed by calculating stroke volume and ejection fraction, which can be calculated based on multiple short-axis views from cine MRI.

Although the anatomical structure of the heart is important, capturing blood flow information is also crucial. After all, the main purpose of the cardiovascular system is to control and transport blood throughout the body. For observing intracardiac blood flow, three main techniques dominate [Rodriguez Muñoz et al., 2013]:

1. Echocardiographic particle image velocimetry (echo-PIV) – ultrasound is used to
track groups of contrast agent particles.

2. Vector flow mapping (VFM) – based on colour-Doppler data, a series of equations are solved to uncover laminar and vortical flow components.

3. Phase contrast (PC) CMR – traditionally captures through-plane velocities in a 2D imaging plane over time, based on the MR signal’s sensitivity to motion. Phase contrast imaging has also been expanded to 3D; this technique is often referred to as 4D Flow MRI (or 3D PC-MRI). We discuss 4D Flow MRI with more detail in Section 2.1.4.

2.1.4 Cardiac imaging: 4D Flow MRI

While there are several techniques for cardiac flow imaging, we focus our attention on one particular technique: 4D Flow MRI. This technique has gotten particular attention in recent years, as it enables access to the full 3D blood flow dynamic within the heart and great vessels over the cardiac cycle [Dyverfeldt et al., 2015]. 4D Flow MRI, also known as 4D Flow CMR, can be described as “three-dimensional (3D) cine (time-resolved) phase-contrast CMR with three-directional velocity-encoding” [Dyverfeldt et al., 2015]. It is three-dimensional since a volumetric space is acquired. Cine implies that it is time-resolved, specifically with respect to the cardiac cycle. In fact, a given dataset represents a time-averaged vector field since it is currently not possible to acquire the entire cardiac cycle in real time; thus, the data captures persistent flow features over the cardiac cycle. The term phase-contrast refers to how flow information is derived: in essence, a signal from MRI has both a magnitude and a phase, and researchers have found that the phase (or more specifically, phase difference) is proportional to the velocity of the flow. Three-directional velocity-encoding points to the fact that velocity information (in all three spatial directions, i.e. \(v_x, v_y, v_z\)) is encoded in greyscale values. Interpretation of these values depends on the venc (velocity encoding range): 0 represents -venc and the maximum greyscale value represents +venc. For intracardiac imaging, the venc is usually set between 100-150 cm/s; it is generally recommended that it be set 10% higher.
than the expected maximum velocity [Dyverfeldt et al., 2015]. In summary, 4D Flow MRI data encodes flow velocity within a volumetric space in all three spatial directions over time along the cardiac cycle (3D + time = 4D). The acquired data for one time frame (phase) comprises four volumes: magnitude, velocity in x ($v_x$), velocity in y ($v_y$), and velocity in z ($v_z$) (Fig. 2.4). The entire time-varying dataset includes these four volumes for each time frame over the cardiac cycle (typically around 30 phases).

![Figure 2.4: Example of raw 4D Flow MRI data for one timeframe. (Image adapted from Markl et al. [2011], CC BY 2.0)](image)

4D Flow MRI opens up much possibility for understanding flow within the chambers of the heart and great vessels. In light of this technology, a group of physicists, physicians and biomedical engineers came together to produce a consensus paper [Dyverfeldt et al., 2015] which provides a summary of many aspects in the 4D Flow MRI workflow, from acquisition to processing. The work we present in this thesis fits within the last stage of this workflow, specifically aiming to introduce a novel technique for blood flow visualization.

A recent systematic review of over 70 clinical intra-cardiac applications [Crandon et al., 2017] shows that there are many proposed methods that use 4D Flow MRI to quantify and analyze intra-cardiac flow. Within this literature, the left ventricle has been the most extensively studied chamber of the heart [Crandon et al., 2017]. One of the particular applications relevant to our work is the exploration of flow within the left ventricle during diastole [Elbaz et al., 2014, Kumar et al.,]
Kumar et al. [2011] showed using colour vector analysis that healthy subjects had rapid, organized early diastolic flow in the left ventricle that reached from the mitral valve to the apex of the left ventricle, whereas patients with left ventricle diastolic dysfunction had organized high-velocity flow that stopped in the mid-left ventricle (i.e. had a shorter distance of propagation). By analyzing vortex features in the left ventricle, Suwa et al. [2016] demonstrated that vortex formation during diastole may be critical for both filling of and ejection from the left ventricle. In fact, Elbaz et al. [2014] showed that vortex formation in the left ventricle during early and late filling could be quantified and may be useful for analyzing normal left ventricular function (Fig. 2.5). Together, these studies indicate that distinctive flow patterns and features exist within the left ventricle, dependent on the health condition of the subject.

Figure 2.5: Vortex core (red isosurface) of a sample subject, detected during early filling. (Image adapted from Elbaz et al. [2014], CC BY 4.0)

2.2 Flow

Flow is everywhere: in commercial processing plants, in the pouring of a glass of water, in wind patterns and ocean currents, and of course, in our heart, our arteries, our veins. Although we focus
on visualizing blood flow specifically in this thesis, we first provide some basics about flow data and visualization in a general sense. To do this, we introduce the notion of flow vector fields and some mathematical notation used to describe vector fields. From here, we discuss visualization using discrete symbols (glyphs), and stream objects. Finally, we discuss vortices, one particular flow feature that is of interest in the realm of blood flow. (Note: The following two sections are largely based on [Telea 2014]).

2.2.1 Flow vector fields

Flow data can be represented as a vector-valued function \( f \) defined on a subset \( D \) of \( \mathbb{R}^n \) over time. Usually, \( n \in \{2, 3\} \); though for our purposes, \( n = 3 \) is assumed unless otherwise stated. Hence, we have \( D \subset \mathbb{R}^3 \), and

\[
\mathbf{f} : D \times \mathbb{R}^+ \rightarrow \mathbb{R}^3.
\]  

(2.1)

The subset \( D \) defines the volume of interest, such as the volume within a pipe or within a particular vessel. When we look at this function at a single snapshot \( t \) in time, \( f_t : D \rightarrow \mathbb{R}^3 \), we often refer to this as a vector field (see Fig. 2.6). Generally speaking, vector field data can encode any vector quantity (e.g. pressures, directions, forces, etc.), but within the context of this thesis, we assume that the vector at each location in space represents the velocity of the flow at that point. We refer to this velocity \( \mathbf{v} = (u, v, w) \), where \( u = v_x \), \( v = v_y \), and \( w = v_z \), i.e. the components of the velocity in each spatial direction \( x, y, \) and \( z \). If this vector field is constant over time (i.e. time-independent), we call this steady flow. But often, flow has a naturally dynamic quality to it – over time (\( \mathbb{R}^+ \)), the vector field changes (or is unsteady).

Flow is naturally a continuous phenomenon, but when flow data is acquired, it is discrete. Vector field data usually consists of samples of the vector field at regular spatial intervals; for instance, a 3D vector field might have a spatial resolution of 1 mm x 1 mm x 1 mm. Depending on the sampling resolution, subsequent analysis and/or visualization might require re-sampling. Fig. 2.7a illustrates how acquired vector field data is discrete in nature.
2.2.2 Flow visualization: glyphs

Now that we have described the basics of vector field data, we can discuss different ways to visualize it. One simple strategy to visually represent a vector field is to use *glyphs* – a glyph, for our usage within the context of this thesis, is a visual symbol or representation of a vector at a given point in space. Perhaps the most common glyph used for vectors is an arrow, as it naturally encodes the concept of direction. The orientation of the glyph at a point \( p \in \mathcal{D} \) can be defined using \( \mathbf{f}(p) \). The size of the glyph can also encode some information; for instance, it can be scaled proportional to \( ||\mathbf{f}(p)|| \), which is the magnitude of the velocity (or speed) at \( p \). A simple two-dimensional example is shown in Fig. 2.7.

More complex glyphs can be designed to encode additional information, such as divergence, material density, pressure, or temperature. Unfortunately, having so many degrees of freedom often
results in confusing visualizations and ultimately has limited effectiveness.

The dimensionality of the vector field also impacts the visualization approach: for 2D vector fields, glyph representations often adequately portray the data in a clear way; however, in 3D using glyphs can easily clutter the display and create occlusion, thus becoming more challenging to interpret. Moreover, due to the discrete nature of glyphs, visual interpolation between glyphs can be challenging.

2.2.3 Flow visualization: stream objects

Stream objects offer a more continuous way to represent flow, opting to create some kind of line or surface which communicates the trajectory of the flow. The most common type of stream object is known as a streamline. Closely related are pathlines and streaklines, which we introduce here briefly.

A streamline is an integral curve $s(\tau) = (x(\tau), y(\tau), z(\tau))$ which is tangent to the vector field everywhere at a single snapshot in time ($\tau$ parameterizes the curve). Conceptually, it portrays the flow pattern at an instant in time, much in the same way that iron filings align within a magnetic field and create a picture of the magnetic field (Fig. 2.8).

Figure 2.8: Physical example of streamlines: iron filings aligning in the magnetic field created by a bar magnet. (Courtesy of Berndt Meyer, CC BY-SA 3.0)
Mathematically, a streamline satisfies the following:

\[ \dot{s}(\tau) = f(s(\tau)), \quad (2.2) \]

with the initial condition \( s(0) = s_0 \), where \( s_0 \in D \). This initial condition is often called a seed, which is some point in space within the vector field domain \( D \) that the streamline passes through.

Streamlines are commonly calculated for flows which are time-independent, but can also be calculated for time-dependent (unsteady) flow – in these cases, streamlines can be calculated for each snapshot in time, giving a picture of the flow character at each snapshot.

Another option for time-dependent vector fields is to draw pathlines. Pathlines are quite similar to streamlines, but are traced over physical time instead. Consequently, pathlines trace the trajectory that a massless particle would follow when placed into the flow field at a given seed point \( p(t_0) = p_0 \), where \( p_0 \in D \) (at time \( t = t_0 \)). Intuitively, one can think of the lines traced in long exposure photography (e.g. those created by a moving car on the highway, see Fig. 2.9) as pathlines.

Figure 2.9: Physical example of pathlines: lines drawn by car lights in long exposure photography. (Photo free for use, CC0 1.0)

Mathematically, pathlines are defined very similarly to streamlines but are integrated over
physical time \((t)\):

\[
\dot{p}(t) = f_r(p(t), t).
\]  

\(2.3\)

**Streaklines** are another variation of integral curve which can depict vector field data. They are best explained by illustration: imagine taking a fixed initial point \(p_0\) within \(D\) and releasing ink continuously from this point over time. The ink will get advected in the flow field over time, and the curve drawn by this ink is known as a streakline. We can represent streaklines \((x(\tau, t))\) with the following:

\[
\dot{x}(\tau, t) = f(x(\tau, t), t),
\]  

\(2.4\)

where \(x(\tau = t, t) = p(t)\), i.e. the streakline’s end point (when \(\tau = t\)) is the same as the end point of a pathline \(p\) which started at the same seed point \(x(0, 0) = x_0\) and had been traced for the same time \(t\).

To compare and understand these three types of stream objects, we refer to a simple example shown in Fig. 2.10. The top row shows a simple time-varying vector field at 4 snapshots in time. In the bottom row, there is an example streamline traced at \(t = t_0\) on the left. In the middle is a pathline, seeded at the solid blue dot (see vector field at \(t_0\)) and traced until \(t = t_3\). Finally, on the right is a streakline, which connects particles that have been released from the same seed location over the four time frames.

Of these three types of integral curves, we focus on streamlines for the remainder of our discussion. However, much of the discussion can be extended to the other integral curves since many practical aspects, such as how to numerically compute the integral curves, are generally quite similar to one another. Also, streamlines, pathlines and streaklines are all equivalent in steady (time-independent) flow fields.
Calculating streamlines

To calculate streamlines from a discrete vector field in practice, we begin with the seed point and trace it through the vector field using some kind of numerical integration technique. Ultimately, we will find a number of points along the streamline curve which we use to represent that streamline; this set of points we denote with $S$.

The simplest numerical integration technique is Euler integration [Telega, 2014]: starting at a seed point $s_0$, we iteratively step through the vector field and store points which constitute our (piecewise linear) integral curve. Each step is determined by the velocity vector at our current position and a user-specified parameter increment, $\Delta \tau$:

$$s_{i+1} = s_i + f_t(s_i) \Delta \tau,$$

where $f_t(x)$ represents the vector field at a single snapshot (e.g. $t = t_0$). The integral lines traced in Fig. 2.10 are examples of applying Euler integration.

The collection of points found from integration is our streamline, $S = \{s_0, s_1, ..., s_k\}$. Terminat-
ing the streamline tracing process can be based on “integration time” (i.e. a certain maximal value of $\tau$), or a maximum arclength of $S$. In addition, a streamline naturally terminates at a zero-velocity point, when $f(s_k) = (0, 0, 0)$, or when it leaves the vector field domain $D$.

Despite the simplicity of Euler integration, it is often not a favourable choice since it has poor accuracy (the error is in $O(\Delta \tau^2)$). Using a higher order method can improve this accuracy and Runge-Kutta methods are frequently chosen for this purpose. There are several variants of Runge-Kutta integration; we chose to use an adaptive variation which is often called the Runge-Kutta-Fehlberg method, or RKF-45 [Mathews and Fink, 2004]. In this method, the step size $\Delta \tau$ is adaptively adjusted by comparing a 4th order approximation to the problem with a 5th order calculation. If these two are within some error tolerance, the step size can be increased; if not, the step size is decreased. By using RKF-45 over Runge-Kutta 4, for example, the approximation error is reduced and fewer steps are required [Mathews and Fink, 2004].

When tracing streamlines, it is also common to trace both forwards and backwards as well. This can be accomplished simply by using both $-\Delta \tau$ and $\Delta \tau$, e.g. in Eq. 2.5.

**Creating streamline visualizations**

Visualizing flow using streamlines can create a more continuous perception of the flow, which probably aligns with our intuition about flow as a continuous phenomenon. A single streamline can be traced through a vector field, which would give some sense of a very local part of the data. More commonly, however, a group of streamlines would be used to portray the general flow behaviour (Fig. 2.11).

Since streamline visualizations generally involve tracing a collection of streamlines, this leads us to another important topic: seeding. How many seeds should be used, and where should they be placed? This can depend on the goal of the visualization; for a dataset with a question such as “where does flow from the inlet go?”, it makes sense to seed the streamlines at the inlet and trace them throughout the vector field domain. However, the question is often not so simple to answer. We frequently want to understand the dataset as a whole, which brings in challenges related to
Figure 2.11: Visualizing wind data with varying numbers of streamlines.

visual occlusion and clutter. For overview visualizations, we usually seek the following properties:

Coverage: We would like the streamlines to capture important flow features. Such features are often application- or task-specific, but could include generally known features such as critical points (points with zero velocity, e.g. sources, sinks, saddles).

Uniformity: Density of streamlines should not vary too wildly; areas which are too densely seeded will end up becoming cluttered and areas which are sparsely covered may be difficult to understand, particularly the regions where no streamlines pass through (Fig. 2.11).

Continuity: Typically, we prefer long streamlines to short ones since they tend to be easier to interpret.

Other stream objects

Extensions of streamlines include other stream objects such as stream tubes, stream ribbons, and stream surfaces. Stream tubes are three-dimensional extensions of streamlines, constructed by sweeping a circular cross section along the streamline [Telea, 2014]. We adopt the use of streamtubes in our method and will therefore discuss this further in Chapter 3. Stream ribbons are also three-dimensional extensions of streamlines; these are created by tracing two streamlines from two nearby seed points and creating a surface from these two lines (Fig. 2.12a). Finally, stream surfaces are a generalization of stream ribbons, where a number of streamlines seeded along a curve are converted into a surface (Fig. 2.12b).
Aside from stream objects, there are two other common ways to visualize vector field data: texture-based methods, and simplified flow representations. Texture-based methods (such as line integral convolution) are most commonly used for 2D domains or surfaces. The basic idea involves an input texture (made up of small-scale black and white noise) and tracing streamlines for each pixel in the domain. Using the streamlines, the original noisy texture is converted into one where the “grain” of the resulting texture reflects the flow character (Fig. 2.13). Although texture-based methods can also be applied in 3D, this is generally less common and as such, is outside the scope of our discussion here. Simplified flow representations are based on flow features such as vortices, speed extrema, or separation lines between regions of laminar flow. In keeping with the focus of this thesis, our primary feature of interest is vortices, which we discuss in the next section.

2.2.4 Vortices

To most of us, the idea of a vortex is somewhat intuitive – it usually comprises swirling or rotating flow about some central region. Unfortunately, this type of definition does not translate to an easy way to detect vortices – in fact, robust detection and quantification of vortices is an ongoing area of research. For now, we summarize the main detection techniques outlined by Jiang et al. [2005], and present some of the mathematical details of one particular method which is recommended for flow in the cardiovascular system [Kheradvar and Pedrizzetti, 2012].
Jiang et al. [2005] present nine vortex detection methods, which we briefly summarize here:

1. **Helicity** method [Levy et al., 1990]: vortex core lines are extracted based on maximal points of normalized helicity.

2. **Swirl Parameter** method [Berdahl and Thompson, 1993]: an intrinsic swirl parameter, which is defined by the ratio of convection time to orbit time, is introduced to distinguish vortex core regions.

3. $\lambda_2$ method [Jeong and Hussain, 1995]: vortex core regions are defined based on eigenvalues of the decomposed velocity gradient tensor.

4. **Predictor-Corrector** method [Banks and Singer, 1995]: the algorithm predicts vortex cores based on vorticity lines and corrects the prediction using local minimum pressure.

5. **Eigenvector** method [Sujudi and Haimes, 1995]: after calculating the velocity gradient tensor eigenvalues and eigenvectors, velocity vectors are projected onto the plane normal to the eigenvector of the real eigenvalue to decide if a given point is part of a vortex core.
6. **Parallel Vectors** method [Roth and Peikert, 1998]: vortex core lines are built from locating where velocity is parallel to its second derivative.

7. **Maximum Vorticity** method [Strawn et al., 1999]: after calculating vorticity, local maxima of vorticity magnitude define vortex cores (in the plane normal to vorticity).

8. **Streamline** methods [Sadarjoen et al., 1998]: vortices are detected using a winding angle method.

9. **Combinatorial** method [Jiang et al., 2002]: combinatorial labeling based on *Sperner’s Lemma* is used on the velocity vector field to identify swirling flow centers.

Of all the methods presented, the $\lambda_2$ method [Jeong and Hussain, 1995] is considered to be the most accepted technique for detecting vortex structures in cardiovascular applications [Kheradvar and Pedrizzetti, 2012]. Because of this, we will only describe the $\lambda_2$ method in more detail.

The $\lambda_2$ method is a region-based method, which means that it identifies volume(s) of vortex-like behaviour [Gänther and Theisel, 2018]. To calculate vortex core regions using the $\lambda_2$ method, consider a given point $p \in \mathcal{D}$ at a time $t = t_0$. With the associated vector, i.e. $f_t(p) = (u, v, w)$, we first calculate the velocity gradient tensor, $J$:

$$J = \begin{bmatrix}
\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\
\frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\
\frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z}
\end{bmatrix} \quad \text{(2.6)}$$

The velocity gradient tensor describes how the velocity within the vector field is changing spatially. This matrix $J$ is then decomposed into the strain deformation tensor, which describes the rate of stretching and shearing [Gonzalez and Stuart, 2008],

$$S = \frac{J + J^T}{2}, \quad \text{(2.7)}$$
and spin tensor, which describes the rate of rotation \cite{Gonzalez and Stuart, 2008},

$$\Omega = \frac{J - J^T}{2}. \quad (2.8)$$

Eigenvalue analysis is then applied to $S^2 + \Omega^2$. A given voxel is classified as part of a vortex core if $\lambda_2 < 0$, assuming the eigenvalues are ordered such that $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Note that all eigenvalues are real numbers (i.e. there are no complex eigenvalues) since $S^2 + \Omega^2$ is real and symmetric \cite{Jiang et al., 2005}.

If there are multiple vortices, it can be difficult to use the method to distinguish between them. Moreover, all methods are susceptible to generating false positives and do not necessarily provide a good way to distinguish between false positives and real vortices \cite{Jiang et al., 2005}. The most straightforward way to verify if a detected line or region is indeed a vortex is by visual inspection, although an automated verification process has been proposed \cite{Jiang et al., 2005}.

\cite{Günther and Theisel, 2018} do note that region-based methods (including the $\lambda_2$ method) are specifically defined for divergence-free, incompressible flows. However, within the context of this thesis, this limitation is acceptable since blood is assumed to be incompressible \cite{Kheradvar and Pedrizzetti, 2012}.

**Vortices in blood flow**

Within the context of blood flow, vortical flow has received much attention in recent years \cite{Pedrizzetti et al., 2014, Kheradvar and Pedrizzetti, 2012, Sengupta et al., 2014} since vortices seem to transport fluid more efficiently and help multidirectional streams of blood merge without energy loss. \cite{Pedrizzetti et al., 2014} suggest that maladaptive intracardiac vortices may be involved in LV remodelling and could provide early indications of long-term outcomes. Indeed, the development and characteristics of vortices in the heart show correlation with the function of the heart and its pumping efficiency \cite{Kheradvar and Pedrizzetti, 2012}. As such, identification and quantification of vortices is an area of active research – for instance, \cite{Elbaz et al., 2017} evaluate 3D vortex ring formation in the LV during diastole and its association with viscous energy loss and
conclude that altered formation is associated with increased energy loss.

2.3 Scientific visualization

Understanding complex scientific phenomena (such as flow) from mere numbers can be difficult at best – it is more likely near impossible for the common researcher. Enter visualization, a standard tool whose pivotal value can be easily overlooked. Fundamentally, visualization aims to create images that convey some insight into a given process [Telea, 2014]. To reach this point, we typically follow a road of data acquisition and import, filtering/enriching, mapping to visual representations, and rendering; this road is often known as the visualization pipeline. In this thesis, we primarily focus on step 3 of the visualization pipeline: mapping.

Mapping is the process of associating visual elements to the dataset we are trying to visualize [Telea, 2014]. The question of mapping depends on the type of data attributes we look to represent. The three most common categories are scalars, vectors, and tensors. Scalar datasets (often also called scalar fields) represent functions $f : D \rightarrow \mathbb{R}$, where $D$ is usually a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. Elevation, for instance, could represent a scalar attribute of a 2D domain; to visualize it, a surface with elevation mapped to the $z$-coordinate could be used (Fig. 2.14). Regarding vector datasets (or vector fields), we have discussed these at length in Section 2.2. Finally, tensors are high-dimensional generalizations of vectors and matrices. Some examples of tensor attributes include curvature, diffusion, or stress and strain. While visualizing vector fields is difficult, the challenge of tensor field visualization is even greater, since each data point is typically associated with a $3 \times 3$ matrix. Although tensor field visualization can appear in medical imaging (for instance, diffusion tensor MRI), it is outside the scope of this thesis.

Between these three common areas of scientific visualization, our focus lies in vector field visualization; in particular, flow visualization. The next subsection discusses related work in the area of flow visualization specifically as it relates to our phenomenon of interest, that is, blood flow.
2.3.1 Related work: blood flow visualization

Processing and visualizing blood flow data has become an area of active interest within the field of computer graphics: a few recent survey papers relating to blood flow visualization [Vilanova et al., 2014], data processing of 4D Flow MRI (with a focus on the aorta) [Kühler et al., 2016] and medical flow visualization [Oeltze-Jafra et al., 2018] provide a comprehensive overview of flow visualization in the medical area. Within these surveys, we highlight one main topic related to our work, namely qualitative flow analysis.

Qualitative flow analysis can be broken down into three main categories [Kühler et al., 2016]: (1) direct methods, (2) geometry-based methods and (3) feature-based methods. Direct methods include, e.g. volume rendering of velocity magnitudes, or line/arrow glyphs to represent the vector field data and colour-coding based on velocities. Displaying a glyph at each voxel usually creates visual clutter, so it is common to display vectors with a given distance between each of them, or to only depict vectors within a particular region [van der Geest and Garg, 2016]. These vector displays can be used to qualitatively assess left ventricle inflow/outflow direction, stenotic jet direction, and regions of recirculating flow [van der Geest and Garg, 2016]. Geometry-based flow visualization includes using lines or particles to depict flow. Two of the most common geometry-based techniques
used for 4D Flow MRI are streamlines and pathlines: streamlines show the instantaneous nature of the flow, whereas pathlines depict a particle’s trajectory in an unsteady flow field over time. Both techniques require a careful seeding strategy and some numerical integration scheme (usually from the Runge-Kutta family). A number of methods have been proposed for appropriately seeding and growing integral lines [Chen et al., 2007, McLoughlin et al., 2010, Yu et al., 2012]. These lines can also be interactively seeded (e.g. by selecting vessel cross-sections) [Pelt et al., 2010] and rendered in a stylistic way to improve depiction of the blood flow dynamics [Lawonn et al., 2014, Pelt et al., 2010]. Finally, feature-based methods are used to highlight specific flow characteristics in the data, such as high-velocity jets, vortex cores, or vortex regions. Salzbrunn et al. [2008] introduce the idea of line predicates: Boolean functions which indicate if an integral line matches a certain criterion or not. They have been used for bundling flow lines which, e.g. pass through a certain region of interest, are part of a vortex, reach maximal velocity, are a minimum length, or reside in a particular area longer than a specified length of time [Born et al., 2013, Jankowai et al., 2016]. Bridging geometry-based and feature-based methods, there has also been work on clustering of integral lines to reduce visual clutter, classify flow structures, and aid in physicians'/experts' exploration and understanding of flow data [Meuschke et al., 2016, Oeltze et al., 2014, Pelt et al., 2012, Yu et al., 2012]. Additional software tools which allow the user to clip or slice the data (sometimes creating additional focus windows or inspection lenses [Gasteiger et al., 2011]) have been designed in an attempt to decrease the clutter and get a clearer understanding of the data, though the restriction of screen space still hinders comparison between multiple windows.

2.4 Physicalization

As an alternative to digital visualization, there has been recent interest in the area of physical visualization. For instance, the work of Jansen et al. [2013] demonstrates that physical visualization can improve user efficiency for information retrieval tasks. Jansen et al. [2015] subsequently highlight the opportunities and challenges for data physicalization. The authors define a data
physicalization as “a physical artifact whose geometry or material properties encode data”, and discuss numerous benefits of physicalization, such as enabling active perception (e.g. being able to turn a model around or move closer), engaging non-visual senses (e.g. touch, with nuances in perceiving texture, weight, etc.), and bringing data into the real world (the visualization is always “on”, which supports casual visualization).

Physicalization has a broad scope of application. For instance, complex geospatial data benefits from physicalization (Fig. 2.16) by improving interaction and understanding [Djavaherpour et al., 2017, Allahverdi et al., 2018]. In architecture and urban designs, physical models are well-established tools in the hands of designers – Hull and Willett [2017] discuss how such models allow for exploring spatial relationships and characteristics, support comparisons, provide context, and evoke emotion. Even data which we traditionally see in plots and charts can take on a physical form [Taher et al., 2015, Taher et al., 2017, Houben et al., 2016]. Herman and Keefe [2018] experimented with 3D printing scalar fields on different kinds of surfaces and found that box-shaped glyphs (“boxcars”) on spheroids (“potatoes”) were most compelling for tangible interaction (users were more likely to pick them up and inspect closer). Bader et al. [2018] exploit multimaterial voxel-printing to create physicalizations of 3D data such as point clouds, volumes from medical
imaging, and results from a computational fluid simulation. The example prints are visually striking but require a commercial 3D printer rather than an affordable one. This rather humble sampling of previous work demonstrates the power and profit of physicalizations in a broad sense.

Figure 2.16: Physicalization of geospatial data via affordable 3D printing. (Image courtesy of Allahverdi et al. [2018], ©2018)

2.4.1 3D printing

When it comes to creating a data physicalization, one of the fundamental questions to address is “how?”. Although there are many techniques (e.g. subtractive manufacturing, sculpting, building with modular pieces such as LEGO, etc.), we focus on affordable 3D printing using material extrusion printers, which are commonly known as FDM (fused deposition modeling) or FFF (fused filament fabrication) printers. These types of printers create models by layering melted thermoplastics – most popularly, PLA (poly-lactic acid) and ABS (acrylonitrile butadiene styrene). Aside from FDM, there are also more expensive types of 3D printers such as stereolithography (SLA) which use photopolymers that are cured by UV lasers and selective laser sintering (SLS) which fuse small particles of powders by laser [Rengier et al. 2010]. In the remainder of this section, we provide some basic background on 3D printing for physicalization in two parts: firstly, we describe the general workflow from virtual model to physical, and secondly, we discuss some of the particulars related specifically to affordable 3D printing.

The 3D printing workflow [Meckes 2016] starts with a 3D model, traditionally created in some kind of modelling software. Sources for 3D models have become more plentiful, with various platforms such as Thingiverse [MakerBot Industries, LLC 2019] providing models to the masses.
The 3D model is converted into a mesh file format for 3D printing, the most common being STL or OBJ. Once in a mesh file format, it is often checked for errors or “printability”, particularly looking for things like holes or non-manifold vertices. (This can be done with freely available tools such as Meshmixer (©Autodesk, CC-BY-NC-SA 3.0)). The mesh file is then loaded into a slicer software; in our work we use Simplify3D. Within the slicer software, the mesh is converted into gcode, a set of instructions for the 3D printer. The 3D printer then creates the model based on the gcode file.

Affordable desktop printers have grown in popularity due to their low cost and relatively easy maintenance; however, there are a number of challenges associated with them. The print bed size can limit creation of larger objects, though previous work has focused on partitioning models into 3D printable parts. When printing overhanging features, support structures are required for the printing process but are usually discarded after, resulting in material waste and damage to the printed model surface; hence, specifically aim to partition models so that they can be printed without requiring support. Of note, both of these studies show example prints which are all single models – such models undoubtedly have challenging thinner features (e.g. the arms in the fertility model) but are ultimately one single, solid model. On the other hand, printing many tiny objects (such as a forest of detailed trees) is quite challenging since they are susceptible to breakage.

2.4.2 Related work: physicalizing motion/flow

Since we are interested in physicalization of blood flow specifically, it is important to consider existing physical visualizations of flow or motion in general. Informally, there have been a number of attempts; for example, artistically 3D printed people’s movements in a lobby space over a 10-hour time frame, reports an example of 3D printing trajectory lines in an engineering application, and explore printing of abstract fluid flow structures. Of note, none of these examples have been studied or presented in a detailed and rigorous way. More formally, there has been work on building software tools for authoring
motion geometry [Kazi et al., 2016] and for generating motion sculptures from a series of 2D images [Zhang et al., 2018]. However, both of these studies generally emphasize the portrayal of human motion from a more “macroscopic” perspective (e.g. the movement of a person’s arms and legs while running, or the path swept by a tennis player’s swing), rather than a finer, “microscopic” application such as the intricacy of blood flow within the heart. More closely related to medical data, [Acevedo et al., 2004] explore the use of expensive colour 3D printing (specifically powder bed fusion) to create diffusion tensor MRI visualization models, employing thick image-based slabs as support. Their initial experiments suggest that physical models enhance usage and analysis of their digital equivalents.

2.4.3 Related work: 3D printing in medical applications

Within the medical field, physicalization (most commonly in the form of 3D printing) has increased in popularity over the last couple of decades. Medical 3D printing has a wide spectrum of applications, ranging from patient-specific anatomical models for pre-surgical planning to manufacturing of prosthetics [Mitsouras et al., 2015]. The scope of interest in previous visualization applications mainly focuses on printing (patient-specific) anatomical structures and related data, which can often be rather expensive to create.

One might then ask – what is the benefit of 3D printing medical data versus visualizing it in a traditional way (e.g. with 2D images or using computer software)? Aside from the general benefits mentioned earlier (e.g. enabling active perception, leveraging natural spatial perception, making data accessible, and bringing data into the real world [Jansen et al., 2015]), one study showed that personalized 3D printed models of patient anatomy (specifically of kidneys and tumours) improved patient understanding of their tumour and surgery [Bernhard et al., 2016]. Another study analyzed the differences between using physical models, textbooks, and 3D computer models to understand complex equine foot anatomy and demonstrated that physical models were significantly more effective as teaching aids [Preece et al., 2013]. With respect to cardiovascular applications specifically, [Giannopoulos et al., 2016] report that fabricated 3D models provide the advantage
of haptic feedback, direct manipulation, and enhanced understanding of cardiovascular anatomy and underlying pathologies. Sun and Lee [2017] provide a systematic review on cardiovascular 3D printing applications, highlighting findings in three main areas: (1) representing patient data with high diagnostic accuracy, (2) serving as an educational tool for parents, clinicians, healthcare professionals and medical trainees, and (3) using 3D printing as a tool for pre-surgical planning, medical device design, and simulation of diseases. Farooqi et al. [2018] discuss the role of 3D printing in preprocedural planning for various pathologies which can lead to heart failure, such as transcatheter aortic valve replacement, right-sided heart failure, hypertrophic cardiomyopathy (HCM), heart transplantation, and ventricular assist device placement. They conclude that a multidisciplinary approach is key to increase the use of this technology in caring for patients with heart failure.

2.4.4 Related work: surface representation

While physicalizing flow is the main focus of this thesis, we found that creating some kind of representation of anatomical context was also useful for understanding the flow data. In medical imaging, anatomical information is captured in volumetric form – but as mentioned in Section 2.4.1, one of the typical requirements in the 3D printing workflow is generation of a surface representation (commonly saved in a file format such as STL). As such, this section provides a brief discussion of work related to creating surfaces from volumetric data. The typical process within the context of medical 3D printing includes segmenting the organ of interest, creating a surface from that segmentation, and editing the created surface using CAD (computer-aided design) software to prepare it for printing [Giannopoulos et al., 2016].

To create a surface from medical imaging data, the first key step is segmentation of the organ or region of interest [Byrne et al., 2016]. Traditionally, segmentation is carried out by an expert clinician who manually draws contours over stacks of images in order to isolate a certain region of interest – in these cases, the contours can often be connected and formed into a surface. However, this process is quite labourious, which has led to the rise in popularity of semi-automatic and
automatic techniques for segmentation. Some techniques include (combinations of) brightness thresholding, region growing, active contouring, manual editing, etc. [Byrne et al., 2016]. The output of these methods is often a volumetric binary image that represents the segmented object; for 3D printing, this needs to be converted to a surface mesh representation.

One of the most popular techniques for converting a volumetric object to a surface is known as marching cubes [Lorensen and Cline, 1987]. By scanning (“marching”) through the whole voxel grid, surface patches are constructed based on how each cell matches one of 15 base configurations. These pieces are collected together to yield a (triangular) surface mesh approximation of the volume; however, the result frequently has issues such as high triangle complexity, topological problems, and staircase artifacts (Fig. 2.17) [Botsch et al., 2010, Lobos and Rojas-Moraleda, 2013]. To address these issues, various remeshing and simplification techniques can be applied [Botsch et al., 2010]. Many practitioners involved in medical 3D printing end up resorting to manual editing in order to ensure their model is fit for fabrication [Giannopoulos et al., 2016].

![Figure 2.17: Surface extracted from MRI data using the marching cubes algorithm. (Image courtesy of wikimedia user Dake - CC BY-SA 2.5)](image)

To reduce the issue of high triangle complexity (especially for large datasets), Westermann et al. [1999] modify the standard marching cubes algorithm to work for adaptively refined octrees. Of note, the emphasis of this work is on real-time interaction rather than smoothness; hence, the characteristic staircase effect is still an issue [Lobos and Rojas-Moraleda, 2013]. An alternative to marching cubes and its variants is a family of methods based on Delaunay refinement [Boissonnat et al., 2018].
et al., 2009]. Such methods generally require constructing a particular set of sample points with sufficient samples, especially in areas of finer features; however, they do not perform well with sparse sampling [Sharf et al., 2006]. To account for this type of input data, [Sharf et al., 2006] present a technique for multi-scale surface reconstruction using a coarse-to-fine approach.

The idea of creating a coarse surface representation for heart anatomy is especially appealing when considering the time-varying nature of its shape. Indeed, Orderud and Rabben [2008] explore the use of deformable Doo-Sabin subdivision surfaces for segmentation of the LV from echocardiographic images (i.e. ultrasound). While their tracking of the LV’s deformations over time seems to be fairly effective, it is worth noting that the initial subdivision surface is created manually; thus, in a sense, one must know a priori what mesh resolution will be required. To guide the deformable model in tracking the LV, normal displacement is used. Generally speaking, the idea of using normals to guide surface deformations is commonly used; this is the case, for instance, in the work of Sharf et al. [2006] and Pusch and Samavati [2010]. In both of these methods, the issue of smoothness is also addressed by involving either the cotangent Laplacian or discrete mesh Laplacian.

2.5 Summary

The cardiovascular system controls blood flow through the body. Its complex and critical function is the study of many curious minds. With modern-day imaging technology, we can now acquire four-dimensional (3D + time) blood flow information for the heart and great vessels; this provides us with information that can be used for research and eventually diagnosis. However, the flow data is not trivial to understand. One fundamental aspect to interpreting the blood flow data is visualization, which has become a topic of interest in recent years. On the one hand, several advanced rendering and processing techniques have emerged, but these are generally limited to 2D screen displays. On the other hand, the idea of physicalization has been gaining traction as a research area as well, as it allows for tangible interaction, leverages natural spatial perception, and generally
increases accessibility to a wider audience. In heart-related medical applications, physicalization is most frequently seen with tangible anatomical models. Thanks to the explosion in 3D printing accessibility over the past few decades, many studies have considered and demonstrated the utility of creating patient-specific models in medicine. However, specifically considering a physical representation of blood flow data is a void which we explore in this thesis.
Chapter 3

Multi-slice models

The overarching goal of this thesis is to design and fabricate a 3D printable physicalization of blood flow data within the left ventricle. Generally speaking, flow can be complex and challenging to visualize; physically visualizing the data via 3D printing presents additional subtleties which need to be addressed. In this chapter, we describe two physical model designs: one which uses glyphs, and the other which uses streamlines. Both of these models use the underlying idea of multiple slices to achieve 3D printability of flow data. We can summarize the general process by which these multi-slice physical visualizations are constructed with the following five main steps (see also Fig. 3.1):

1. Select a visualization style to represent the data. We chose two styles for multi-slice visualization, glyphs and streamlines.

2. Filter the visualization objects (i.e. glyphs or lines) so that they are fabricable, either by subsampling (for the glyphs), or by using a similarity-guided placement strategy [Chen et al., 2007] (for the streamlines).

3. “Slice” the model into 3D printable parts which are subsets of the original dataset.

4. Augment the model with auxiliary structures (e.g. base/connecting parts) to provide relative context.

5. Fabricate the model using an affordable 3D printer.

Section 3.1 presents our input data and the overall concept of using slices. The subsequent two sections (Sections 3.2 and 3.3) will describe the glyph model and the streamline model respectively, including their construction and fabrication.
Physicalization of blood flow (steps)

INPUT
segmented
left ventricle
4D Flow data

1. Choose style
e.g. glyphs or integral lines

2. Filter
e.g. by slice or
distance metric

3. Slice

4. Context

5. 3D print
e.g. glyphs or integral lines
segmented
left ventricle
4D Flow data
e.g. by slice or
distance metric

Figure 3.1: Overview of multi-slice physical visualization framework.
3.1 Slice-based design

The blood flow data, obtained from 4D Flow MRI, can be represented as a vector-valued function \( \mathbf{f} \) defined on a subset \( D \) of \( \mathbb{R}^3 \) over time, according to Equation 2.1 (see Section 2.2.1). The segmented volume, \( D \), is the chamber or vessel of interest; in our work, \( D \) represents the left ventricle. Segmenting \( D \) based on 4D Flow MRI data alone is challenging due to its known low anatomical contrast; therefore, \( D \) was obtained using an established semi-automated algorithm: the LV is segmented from cine SSFP MRI by manually tracing endocardial contours, then this segmentation is automatically registered to the 4D Flow MRI velocity data by a mutual information algorithm [Elbaz et al., 2017]. Masking the 4D Flow data with the registered LV segmentation provides us with the input data to our system: a given subject’s blood flow data within the LV at snapshots in time over the cardiac cycle (for each snapshot, \( \mathbf{f}_t : D \rightarrow \mathbb{R}^3 \)).

Given this input, we are faced with a challenging question: how does one convert volumetric flow data into a fabricable model? To begin, we decided to target two styles of visualization that are relatively common in traditional flow visualization: (1) glyphs, which represent the “raw” vector field data, and (2) streamlines (a type of integral line), which portray the flow character at a snapshot in time.

Both vector glyphs and streamlines are usually thin and disparate, making them challenging to 3D print. To overcome this complication, we drew from physicians’ typical use of cross sections or “slices” to acquire and explore medical image data [Tietjen et al., 2006]. Thus we arrive at the concept of our proposed design: slice-based physical visualization. The core idea is to use slices of the segmented volume \( D \) (Fig. 3.2) as a natural support structure, which alleviates some of the core difficulties in 3D printing (e.g. supporting floating structures/objects, fragility of thinner features, etc.), while also providing some contextual cues (since the shape of the slice represents a cross section of the segmented volume). In addition, the slices can be spaced out such that the slicing frequency is suitable for the desired visualization – the number of slices can be as many or as few as preferred for a given dataset. Note that “slices” here refer to thin slabs (subsets of \( D \), as
an extremely thin 2D plane would be nearly impossible to 3D print.

Figure 3.2: The left ventricle (highlighted in red in the short axis images on the left) can be segmented from MR image data (a stack of 2D slices which make up a 3D volume). Such a segmentation represents the chamber’s volume. For physicalization, this volume can be “sliced” into thin slabs (shown on the right), which provide a natural support structure for physicalization while also maintaining a sense of the data.

The slice-based design brings with it numerous advantages: it creates an inherent support structure which can be easily fabricated by low-cost 3D printers, it affords a way to compare between different parts of the data if desired (e.g. comparing slices from two different time frames), and it naturally adds contextual information. The palpability of the overall model can also be seen as a complementary means to grasp the data. We applied this conceptual design to two types of flow representations – vector field glyphs and streamlines – using segmented left ventricle 4D Flow MRI data.

3.2 Glyph model

The basic idea of our glyph model is to represent the data points in the LV and their associated vectors with some directional glyphs. However, representing every data point (in a Cartesian grid with 1.57 x 1.32 x 4.43 mm\(^3\) spacing, for instance) with a glyph would result in an extremely cluttered model, not to mention the high likelihood of failure to print. Hence, only a subset of the
data should be chosen. We select that subset using the slice-based design idea: each slice is a thin slab which represents some subset of the vector field data. Data points contained within each of these thin volumes are candidates for the final model; however, converting every data point even within these thin slabs will still result in many overlapping glyphs. Consequently, within each slab, we downsample by keeping every $n$th data point ($n = 10$ for our example models) and converting those into directional glyphs. Our goal was to strike a balance between comprehensiveness and comprehensibility – we want enough glyphs to represent the flow in its entirety (comprehensively), while limiting cognitive overload and visual clutter (comprehensibly).

This led to the following question: how many slices should be used? We initially experimented with including the entire volume by constructing slices which had no spacing in between them. However, the drawback to this approach was in the final model assembly – having no space between slices meant that the glyph representations would collide with each other in physical space. Therefore, we chose to evenly space the slices (for improved aesthetic quality) and uniformly scale the glyphs to prevent inter-slice collisions.

![Figure 3.3: Arrow glyphs have small, fragile parts (circled in black), whereas cones can portray directional information but are not as breakable.](image)

In addition to slicing, selecting an appropriate glyph required particular consideration. As observed in Fig. 3.3, arrow glyphs – while common for 2D vector field visualizations – are challenging to print. Thin arrow tails break easily once fabricated, and do not provide adequate support for the arrowhead. This is unfortunate, since arrows have the benefit of encoding direction, as well as the potential to represent another variable (e.g. vector magnitude could be shown with the size of the arrow). Based on these desirable attributes, the most natural alternative glyph was a cone. Cones are similar in shape to arrows and can likewise portray direction and speed based
on orientation and scaling, but are more printable since they do not have fragile features. Once embedded into a slice, a collection of glyphs becomes much easier to print. Moreover, scaling the glyphs based on vector magnitude gives a clear impression of the predominant flow direction (Fig. 3.4).

Figure 3.4: Cone glyphs are embedded within a slice for printability.

3.2.1 Physicalization of relative context

The slice-based model described so far accounts for many of the challenges associated with affordable 3D printing, but a collection of mere slices won’t hold up in the real world – however nice a virtual model may seem, once physicalized it must stand the test of gravity. Therefore, some additional design considerations were necessary.

We needed some additional physical parts which would support the model’s existence in the real world without losing spatial context (i.e. the position of one slice relative to another). We achieved this by designing slice handles and a stand with a wheel and axle-type mechanism (Fig. 3.5). The base, a rectangular prism with a small cylinder, holds a vertical post. The vertical post, in turn, holds the slices of the model: a “handle”, made up of a thin rectangular piece and a hollow cylinder, is affixed to each slice, and each cylinder can be slid onto the vertical post. The cylinder heights are designed such that the spatial relationships between slices are preserved. Although this design is relatively simple, it includes some important features: the relative spatial positions of the
slices are maintained, and having the post inside each handle cylinder (similar to a wheel and axle) allows for rotation and inspection of individual slices while preserving contextual awareness.

![Diagram of slice model with handle](image)

**Figure 3.5:** Each slice model includes a supplementary handle (shown in blue, top), which allows for easy assembly into a stack of slices (bottom).

### 3.2.2 Fabricated glyph model

Two glyph models were created from two datasets (i.e., two time frames), both derived from a 4D Flow MRI scan of one healthy subject. The models were printed using the MakerGear M2. Because the cone glyphs were embedded in each slice and could appear on either side of the slice (Fig. 3.4), we printed each slice model in halves and then glued the slices together. This prevented the additional hassle of removing support. The final physical models were completely 3D printed except for the vertical post used to support the slices; we used a wooden dowel for this post. Spacing between slices was approximately 12 mm and the thickness of each slice was 2 mm.

Although an individual fabricated model is static and represents a single snapshot in time (Fig. 3.6), we wanted to incorporate the idea of time-varying data as well. For this, we acknowledge the importance of the phases in the cardiac cycle, i.e., systole and diastole. Within these phases,
there are certain key events [Elbaz et al., 2014]; for example, during ventricular diastole, there is early filling (when the ventricles relax and blood flows in from the atria due to pressure difference) and late filling (when the atria contract and push blood into the ventricles). Thus, we represent the flow data by creating models at key time frames, which aligns with common practice in medical textbooks and research publications. Since our focus was the left ventricle, we chose to use early filling and late filling. Both models were sliced with the same spacing so that the slices would be comparable (Fig. 3.7). To get a finer sense of the flow evolution over time, one could print each time frame in between, similar to how LAIKA studio creates models for stop-motion animation [Stratasys Ltd., 2017].

Figure 3.6: Picture of fabricated left ventricle glyph model during late filling (late diastole).

3.3 Streamline model

Although representing the raw vector field may be the most faithful to the original data, it might not be the easiest to interpret at a glance. For this reason, we also developed a method for creating a physical streamline model. However, producing a physicalizable streamline representation is
not simple: careful thought must first be given to how the streamlines are seeded, and once the streamlines are formed, each of them must be converted into a mesh which can be 3D printed.

3.3.1 Generating streamlines

When generating streamlines, one is always faced with competing goals. On the one hand, producing many streamlines can reveal interesting flow characteristics, such as areas of vortical flow, sources, sinks, saddles, etc. On the other hand, a large number of streamlines quickly clutters the visual field, ultimately obscuring whatever feature(s) we originally intended to discover. To manage these competing goals, we employed the idea of generating streamlines using a similarity-guided placement strategy [Chen et al., 2007], as this technique reportedly achieved a balance between uncovering interesting flow behaviour and limiting visual clutter. Specifically, streamlines should have some natural spacing (e.g. a minimum Euclidean distance between lines) to reduce clutter and occlusion, but should also represent “interesting” or “unique” features in the flow. For instance, a large number of streamlines with similar trajectory could be portrayed with perhaps two or three
lines. But, if the directions and shapes of streamlines are distinct from one another, more of these lines should be preserved in the final visualization to give a more complete picture of the flow behaviour.

Figure 3.8: The similarity distance is measured in a local neighbourhood at the point \( p \) where the streamline \( S_i \) is growing.

To obtain a set of streamlines for our physical model, we used the formulation of similarity distance presented by [Chen et al. 2007]. Similarity distance is defined between a point \( p \) on a growing streamline \( S_i \) to another (existing) streamline \( S_j \) (Fig. 3.8). First, the closest point \( q \) on \( S_j \) is found (using Euclidean distance as the metric for closeness). Two sample windows of the same size are formed, one about point \( p \) and the other about point \( q \), and are uniformly sampled with \( m \) samples along the streamline. In the event that a symmetric window cannot be formed about one or both points \( p \) and/or \( q \) (e.g. the point is at the end of a streamline), one-sided windows are used instead. Once these two sets of sample points have been formed, the similarity distance can be calculated [Chen et al. 2007]:

\[
d_{sim} = \| p - q \| + \alpha \left( \frac{1}{m} \sum_{k=0}^{m-1} \| p_k - q_k \| - \| p - q \| \right).
\] (3.1)

This metric has two main components or terms. The first term is simply the Euclidean distance between \( p \) and \( q \), and the second term accounts for how similar the two streamlines are within a local neighbourhood: if both streamlines have the same shape and orientation, this term will be
zero. (The shape coefficient, $\alpha$, is used to specify the relative importance of the two terms.) Hence, the more distinct two streamlines are, the greater the similarity distance between them. A new streamline $S_i$ which has a large similarity distance between all existing streamlines indicates that this streamline $S_i$ should be included in the final visualization.

To obtain a set of streamlines for physicalization, seed point locations are initialized in a Cartesian grid covering the volume and each streamline is traced (both forwards and backwards) using the Runge-Kutta-Fehlberg (RKF45) method [Mathews and Fink, 2004]. Growing a given streamline is stopped if it is too similar to an existing streamline ($d_{\text{sim}} < d_{\text{min}}$), too similar to itself ($d_{\text{sim}} < d_{\text{self}}$), or its length is greater than a user-specified maximum length.

Generating streamlines using this technique creates a set of lines that are good for fabrication: they have a natural element of spacing (thanks to the Euclidean distance term in Equation 3.1), yet they capture the interesting character of the flow. When compared to a set of streamlines created without any filtering, the number of streamlines generated using the similarity-guided strategy was at least 90% less (e.g. 646 lines vs. 59 lines), but the overall flow behaviour is still captured (Fig. 3.9). Furthermore, the user-established thresholds (e.g. $d_{\text{min}}$ and $d_{\text{self}}$) offer flexibility when creating a printable set of streamlines; for example, they can be customized to suit a specific 3D printer’s resolution.

### 3.3.2 From lines to meshes

Once a set of streamlines has been generated, it is necessary to convert it into a mesh for 3D printing. We used sweep surfaces to accomplish this task. A cross section shape is selected and “swept” along each streamline (trajectory curve), adjusting its orientation using the parallel transport approach [Hanson and Ma, 1995]. For simplicity, we used a circular cross section since it generates a tube (Fig. 3.10), which is a natural three-dimensional extension of a line; however, any arbitrary 2D curve could be used. We used a cross section diameter of 3 mm to ensure printability, at the cost of some overlap with voxels surrounding the streamline. Specifically, a given cross section might partially overlap with at most nine voxels based on the spatial resolution of the data, though we
Figure 3.9: Naïvely growing all streamlines for a given set of seed points results in a very cluttered visualization (left), whereas applying the similarity distance metric when generating the streamlines from the same seed points results in a cleaner, physicalizable collection (right).

would expect a typical overlap of 2-4 voxels.

Many previous streamline visualizations do not seem to include directional information [McLoughlin et al., 2010], making the overall character of the flow somewhat ambiguous. Therefore, we enhance our model by adding cone-like arrowheads at the end of each tube. These were constructed by interpolating the vector field at the endpoint of each streamline ($v_i$) and attaching a cone whose axis aligns with $v_i$ (Fig. 3.10).

After obtaining a complete set of streamline meshes we might think that we are ready to print, but this is not the case. While the set of streamlines is technically printable, the fabricated model would require a lot of additional support material since the streamlines are generally disconnected from one another and almost all would need to be floating in mid-air. This, of course, is not physically possible; therefore, we apply our proposed slicing method to provide intermediate support for the streamlines (see Section 3.2.1). Slicing reduces the total amount of support material required and allows for assembling into a model that maintains the overall shape of the streamlines. Unlike the vector field model, the streamlines do not need to be embedded in the slice. Consequently, we chose
to use fewer slices than the vector field model: too many slices in the streamline model becomes obtrusive, as they interrupt the shape of any streamline spanning multiple slices.

3.3.3 Fabricated streamline model

The final model was printed using the MakerGear M3-ID. One advantage of the M3-ID over the M2 is that it is equipped with two extruders, making it possible to print with two colours in the same model. We leveraged this by using a translucent filament for the slices and support material, and using a red filament for the streamlines themselves. This highlights the streamlines compared to the slices, support, and base, thus emphasizing the flow features in the model (Fig. 3.11). To assist with inspection of the data, the model was scaled up 1.5x in all dimensions within Simplify3D. The complete streamline model was fabricated with a total print time of approximately 1.4 days and required about 330 grams of PLA filament. In the printed model, spacing between slices was approximately 22 mm and streamline radii were 2.25 mm each.
3.4 Summary

Using the idea of slice-based viewing, which is common practice in the medical field, we design two models for visualizing blood flow data. One model depicts the flow data using cone-shaped glyphs which are embedded in a slice for printing. The cone shape and slice-based support allow for relatively easy 3D printability. Moreover, the glyphs are scaled in size based on the magnitude of the vector field at each point; this allows for a clearer understanding of the predominant flow direction. The second model uses streamlines to portray the blood flow data. A 3D printable set of streamlines is generated using the similarity-based distance metric [Chen et al., 2007], and is sliced into multiple slabs for printing. Both models capture the character of the flow at a snapshot in time (i.e. during diastole).
Although the main goal of this work is to create a physical representation of blood flow, having anatomical information helps place the flow in physical context. The supporting slice structures can also be derived from reconstructing the anatomy. Hence, we aim to augment the physical flow model(s) with a smooth representation of the heart chamber (in our case, the endocardial surface of the LV).

The original input data to our physicalization framework consists of segmented 4D Flow MRI data. It inherently encodes the anatomical shape of the LV, but in a volumetric form (for example, a binary volumetric mask could be provided). For 3D printing, our goal is to convert the volumetric binary mask,

\[
I(p) = \begin{cases} 
1, & \text{if } p \in \Omega \\
0, & \text{otherwise,}
\end{cases}
\]

into a smooth surface mesh, \(M\). The mesh \(M\) comprises a set of vertices \(V\) and faces \(F\).

The de facto standard algorithm for creating such a surface from volumetric data is known as \textit{marching cubes} [Lorensen and Cline, 1987]. However, there are some well-known issues with this type of technique: the resulting surface often has staircase artifacts and high triangle complexity [Botsch et al., 2010]. We would like to avoid creating a surface with jagged edges, as this does not reflect the known anatomical geometry of the LV, and we would also like to generate a surface mesh with fewer vertices as such models are simpler and often easier to 3D print. Hence, we developed a method which converts our volumetric binary mask into a smooth surface mesh which is suitable for providing anatomical context.

Our approach is primarily inspired by the work of Sharf et al. [2006] and Pusch and Samavati.
Sharf et al. [2006] describe a multiscale deformable model for surface reconstruction of point clouds. Pusch and Samavati [2010] present a least-squares based method for deforming surfaces based on constraints. We combine aspects of both methods to generate a smooth surface approximation from the volumetric segmentation mask. Compared to basic marching cubes, our surface is smoother, requires fewer vertices, and generally does not suffer from topological issues such as holes (Fig. 4.1).

![Comparison of surface reconstruction using marching cubes (left) vs. our method (right).](image)

Figure 4.1: Comparison of surface reconstruction using marching cubes (left) vs. our method (right).

4.1 Method overview

The method for surface reconstruction proceeds in two main stages, the initialization stage (Fig. 4.2A-B) and the fitting stage (Fig. 4.2C-F). In the initialization stage, we pre-calculate approximate surface points, a signed distance field to those surface points (Fig. 4.3), and initialize the surface mesh within the volume. During the fitting stage, we iteratively grow the mesh based on the distance field while keeping the surface smooth using an energy term. The next two sections will delve into the details of each of these stages further.
Figure 4.2: High level overview of LV surface reconstruction process: (A) Input data, a volumetric binary mask. (B) Initialization - an initial mesh $\mathcal{M}_0$ is placed within the object; surface points are shown in black. The surface mesh evolves, somewhat similar to a balloon. (C) Triangles highlighted in yellow indicate fronts which are active. (D) Active fronts are adaptively subdivided. (E) The final coarse surface mesh. (F) The final subdivided and smoothed mesh, $\mathcal{M}_F$.

4.2 Initialization

Our first step in the initialization stage is to find a collection of points, $\mathcal{P}$, which represents the surface that we aim to reconstruct. In 2D, one can think of using some kind of edge detector on the binary image to obtain this set of points. At this stage, a rough approximation is sufficient; thus it can be obtained in different ways. One simple way is to scan through the volume ($I$) and check each voxel to see if its immediate 6-connected neighbours are different from each other (i.e. there is at least one 1 and one 0). This indicates that the voxel is close to the boundary of the object and can be flagged as an approximate surface point. Fig. 4.4 illustrates the idea in 2D.
Figure 4.3: Illustration of distance field: grayscale values reflect distance field values, black (0) approximates the target surface.

Figure 4.4: Detecting approximate surface points based on binary mask data [left]: a given pixel (voxel in 3D) is flagged as a surface point [middle] if its 4-connected neighbours include at least one 1 and one 0. Then a signed distance field is calculated based on the surface points [right], here illustrated with Manhattan distances.

Once we have found $\mathcal{P}$, we build a signed Euclidean distance field, $D$. The distance field encodes values as 0 (on the surface), negative (outside the object), or positive (inside the object); Fig. 4.4 illustrates this concept in 2D. To calculate this distance transform, we use a fast scanning algorithm that accounts for anisotropic voxels ([Mishchenko, 2015]), which are typical in 3D medical imaging.

After calculating the distance field, we place a regular icosahedron within the object as our starting mesh, $\mathcal{M}_0$. At this stage, we do not know how many vertices will be required to reconstruct the LV surface, but we do know that we generally prefer fewer vertices; therefore, we begin with a shape that is relatively simple. The idea of using a sphere-like shape is based on the work of [Sharf et al., 2006]. Other genus 0 shapes (such as a tetrahedron or octahedron) could also be candidate options for starting meshes, though these have far fewer vertices than we would expect of our final
surface and were thus considered less suitable.

To place $M_0$, we can either use a manual strategy or an automatic one using some kind of heuristic. For our work, we first find the maximum value $m$ in $D$. The location $p$ ($D(p) = m$) is the starting location of $M_0$. Some proportion of $m$ (say $r = m/2$) initializes the radius of $M_0$. By choosing $r < m$, our initial mesh is completely contained within the object (Fig. 4.5). With both $D$ and $M_0$ in hand, we are ready to begin the surface fitting stage.

Figure 4.5: 2D illustration of placing $M_0$: point $p$ (shown with a star) marks the “innermost” point within the object, and $D(p) = m$ is used to initialize the radius (for instance, $r = m/2$). $M_0$ is shown with a red circle.

### 4.3 Surface fitting

Our end goal is to generate a surface which closely matches our input mask, which is now represented as a set of approximate surface points $\mathcal{P}$. This brings us to the following question: how do we measure the reconstruction quality of our mesh? What is our error function that we seek to minimize? One initial, natural thought is to use our distance field $D$ and vertex positions of our candidate mesh. If, for every $v \in \mathcal{V}$, we have $D(v) < \epsilon$, then we might consider that the mesh is a good fit since it is very close to the surface points. In this case, our error function would be

$$E(\mathcal{V}, D) = \max(D(v)), \forall v \in \mathcal{V}. \quad (4.2)$$
But if we consider the case illustrated in Fig. 4.6, we can see that it is very plausible to have a mesh whose vertices are close to \( \mathcal{P} \), yet have surface points \( p \in \mathcal{P} \) that are not “happy” – they have not been adequately reconstructed.

![Figure 4.6](image.png)

Figure 4.6: A mesh may be near the approximate surface points \( \mathcal{P} \) but the surface is not adequately reconstructed. (Note: black pixels in the distance field represent approximate surface points.)

Therefore, we also look at the question of reconstruction quality from the perspective of the approximate surface points – are all points \( p \in \mathcal{P} \) within some minimum distance \( \epsilon \) of the mesh \( \mathcal{M} \)? We can address this question by first defining a function \( \Phi(p, \mathcal{M}) \) that returns the shortest Euclidean distance between a point \( p \) and the surface of a mesh \( \mathcal{M} \). This would give us the following error function:

\[
E(\mathcal{M}, \mathcal{P}) = \max(\Phi(p, \mathcal{M})), \forall p \in \mathcal{P}. \tag{4.3}
\]

With this error function, we have a well-defined metric to describe the quality of surface reconstruction. However, to minimize this error is a non-linear optimization problem. We decided to approach this complex non-linear optimization problem by iteratively solving a simpler linear problem. At a high level, we deform the mesh (step by step) by allowing it to expand toward the surface points, somewhat analogous to inflating a balloon with a specified bounding volume (Fig. 4.2). The surface deforms at each step based on the vertex normals and the distance field, \( D \). At each step, we also use the discrete mesh Laplacian to keep the surface relatively smooth. As the shape grows and deforms, it may get stuck at different features such as turns and tunnels. This
occurs since we initially underestimate the number of vertices that will be required to reconstruct the whole surface. To account for this, we adaptively subdivide the mesh as required. This whole process can be summarized by Fig. 4.7. The next subsections will describe each of these steps in more detail.

4.3.1 Iterative linear least-squares

A typical linear system can be expressed as $Ax = b$, where $x$ is our vector of unknowns. We first consider our unknowns – what are we trying to solve for in each iteration? We are interested in
finding all mesh vertices’ new positions:

\[
v_{k+1} = v_k + \Delta_k,
\]

where \( v_k \) is a vector that represents all vertex positions at iteration \( k \), and \( \Delta_k \) is a vector that represents the changes in vertex positions from \( k \) to \( k+1 \). For any given iteration, we know \( v_k \) and want to find \( v_{k+1} \), which we can easily calculate using Eq. 4.4 if we can determine \( \Delta_k \). Hence, our unknowns \( x_k = \Delta_k \).

How do we determine \( \Delta_k \) at each iteration? The displacement of each vertex is based primarily on the mesh expanding toward the surface, which we discuss in the next section. We then discuss how to maintain smoothness with an energy minimization term.

**Expansion**

To reconstruct the surface, the initial mesh iteratively moves (“inflates” or “expands”) towards the surface points, \( \mathcal{P} \). As a starting point, we are interested in finding how much we should push each vertex toward the surface. We formulate this “push” by moving the vertex in the direction of its normal (calculated using an area-weighted per-vertex normal [Jacobson et al., 2018]), and then scaling the normals based on the distance map \( D \) (see Fig. 4.8). We store this in a vector of scaled normals, \( n \):

\[
\mathbf{n} = \begin{bmatrix}
D(v_1)\hat{n}_1 \\
D(v_2)\hat{n}_2 \\
\vdots \\
D(v_n)\hat{n}_n
\end{bmatrix}
\]

For the expansion component, we could simply set \( \Delta_k = \mathbf{n}_k \). We can also re-write this in the following form:
Figure 4.8: Schematic illustrating how vertex normals are scaled based on the distance map $D$ (shown as a greyscale grid in this image). This “pushes” the vertex towards the approximate surface points.

\[ \mathbf{B} \Delta = \mathbf{n}, \quad (4.5) \]
\[ \mathbf{B} = \mathbf{I}_{n \times n} \quad (4.6) \]

where $n$ is the number of vertices in $\mathcal{V}$. However, only using this term means that each vertex is essentially moving independent of the rest, which can result in surfaces that are not smooth or are even self-intersecting. Thus, we include an energy minimization term to maintain smoothness.

**Energy minimization**

Our energy minimization term is based on the discrete mesh Laplacian operator [Pusch and Samavati, 2010]. In simple terms, we push each vertex toward the centroid of its immediate one-ring neighbours (see Fig. 4.9). Thus, the surface favours smoother, sphere-like configurations.

We can calculate this using the following formulation:

Recall from Equation 4.4 that we are looking for $\Delta_k$. If we re-write Equation 4.4 and multiply it by a matrix $\mathbf{T}$, it takes on the following form:

\[ \mathbf{T} \Delta_k = \mathbf{T} v_{k+1} - \mathbf{T} v_k. \quad (4.7) \]

Now we consider, what is $\mathbf{T}$? We want to use $\mathbf{T}$ such that each entry of $\mathbf{T} v$ represents the discrete
The vertex $v_i$ (highlighted in red) is pushed toward the centroid of its one-ring neighbours (shown in grey).

Mesh Laplacian for the corresponding vertex. Thus,

$$T = I_{n \times n} - A,$$

where $A$ represents the adjacency matrix of $\mathcal{V}$, and each row has been normalized by the vertex’s valence (i.e. the $i$th row has been divided by $\text{valence}(v_i)$, element-wise).

Referring back to Equation 4.7, we are interested in minimizing the discrete mesh Laplacian for our new set of vertices, hence we can set $Tv_{k+1} = 0$ and we have the following:

$$T\Delta_k = -Tv_k.$$  \hfill (4.9)

We can also write $-Tv_k$ as a vector of energy minimization vectors, $y$, where $y_i$ (for a given vertex, $v_i$) is defined by:

$$y_i = \frac{\sum_{j \in N(v_i)} v_j}{\text{valence}(v_i)} - v_i.$$  \hfill (4.10)
Complete formulation

By combining Equations 4.5 and 4.9, we can assign the variables found in the typical form of least-squares problems, $Ax = b$, with the following:

$$A = \begin{bmatrix} B \\ T \end{bmatrix},$$

(4.11)

$$x = \Delta,$$

(4.12)

$$b = \begin{bmatrix} n \\ y \end{bmatrix},$$

(4.13)

The mesh deforms iteratively by solving for $\Delta$ at each iteration and updating the vertices accordingly. After updating the positions of all vertices, all vertices are also updated with their current status—either active or inactive. If a given vertex $v_i$ is inactive, this means that it is already within some threshold $t_d$ of the surface points, i.e. $|D(v_i)| < t_d$. All of the vertex activity flags are stored in a vector $a$, where $a_i \in [1, 0]$ represents the active status of vertex $v_i$. Inactive vertices become fixed in position: if $a_i = 0$, then $\Delta_i = 0$ and the position of $v_i$ remains unchanged.

4.3.2 Adaptive subdivision

As the mesh expands, certain areas can be reconstructed with relatively few vertices, whereas finer structures (such as the apex of the left ventricle) require higher mesh resolution. We use linear adaptive subdivision to account for such areas. By using adaptive subdivision, we prevent unnecessary refinement of the mesh in areas where the resolution is already sufficient for representing the surface (see Fig. 4.2C).

To determine which faces will be adaptively subdivided, we calculate the barycenter of each face (for a given face $f_i \in \mathcal{F}$, we calculate its barycenter $b_i$). Using the distance map $D$, we find $D(b_i)$ for all faces; any face for which $D(b_i) > t_d$ is flagged as an “active” triangle and is linearly subdivided
(see Fig. 4.2C-D). Mesh connectivity is maintained by adding T-junctions where necessary. All new vertices become a part of the overall mesh and evolve as described in the previous section.

4.3.3 Convergence

The surface fitting stage ends when the following criteria are met:

- All vertices are inactive (each vertex is considered sufficiently close to the approximate surface); i.e. Eq. 4.2 is minimized.

- All surface points $P$ are “happy” (they are sufficiently close to the mesh surface $M$); i.e. Eq. 4.3 is minimized.

Convergence is not always guaranteed, depending on the dataset and the thresholds chosen. Typically $t_d$ is set based on voxel size – either the largest of the three dimensions, or the L2 norm (representing the longest diagonal of the voxel). This was chosen experimentally and worked relatively well for the datasets processed. However, in cases where the convergence criteria were not met, a maximum number of iterations was used to prevent the algorithm for continuing indefinitely; this maximum was usually set to 50. Visual inspection of the result, which is necessary prior to 3D printing, confirms that an adequate representation of the LV anatomy was obtained.

4.4 Anatomical context for flow

The final surface representation is used in two main ways: (1) as a thin, half-shell surface which provides context for the vessel, and (2) as a smooth surface which can be “sliced” to provide support for the flow representation objects (i.e. glyphs or streamlines).

In its final form, our mesh $M_F$ is a surface representing the LV endocardium (inner surface of the LV). One way to add anatomical context is by embedding the structure into the stand (described in Section 3.2.1). Since our aim is to provide contextual information, we do not want the anatomical structure to obscure the flow information. Thus, we cut $M_F$ with a plane according
to an appropriate axis; usually a long-axis plane is chosen so that the shape of the LV from base to apex can be appreciated. For 3D printing purposes, we convert the surface into a shell by extruding along the normals. The complete stand with embedded shell can be seen in Fig. 4.10 as well as in photos of the fabricated multi-slice models (Figs. 3.6 and 3.11).

Figure 4.10: A representation of the LV anatomy can be included as part of the physical flow model to provide context.

Another way to provide anatomical context is by using the mesh $\mathcal{M}_F$ to create long-axis slices. To generate these slices, a mesh Boolean operation was used [Zhou et al., 2016, Jacobson et al., 2018]. By intersecting $\mathcal{M}_F$ with a thin box mesh $\mathcal{M}_{\text{box}}$, a cross-section of the LV shape can be obtained (Fig. 4.11). The box thickness determines the eventual slice thickness for 3D printing; this usually ranged between 0.5-2.0 mm to achieve printability without being too intrusive.

4.5 Summary

Anatomical context can be valuable for better understanding flow within the left ventricle. Hence, while the main focus of this thesis is creating a physical representation of blood flow data, we also created a simple surface to represent the LV endocardium (inner surface of the chamber). This was
constructed using a “balloon-filling” method: a basic coarse shape was placed within the volume to be reconstructed, and allowed to expand based on some simple rules. As the surface evolved, adaptive subdivision was used for reconstructing finer features without creating an overly fine mesh. The final result can be used to show the LV surface (e.g. using a half-shell shape), or sliced to show a long-axis cross section of the LV. Both methods help convey the idea of anatomical context for the flow data.
Chapter 5

User study

If this thesis were a story, this chapter might be the climax of the typical story-telling arc, the point of main conflict: how does a physicalized flow model compare to more typical screen-based visualizations? Thus far, we have presented two main designs in Chapter 3 both of which were fabricated with affordable 3D printers (see Figs. 3.6 and 3.11; both include the anatomical context design described in Chapter 4). With these physical visualization prototypes in hand, we wanted to see how they would compare with digital visualizations in the hands of potential users. Hence, we conducted a user study to evaluate our physical visualization prototypes.

5.1 Goals

The goal of the study was to discover how people would interact with the physical visualization models, and to see how they differ with respect to comparable digital visualizations. Within the concept of interaction, we sought to answer the question: “how quickly does a given medium (physical vs. digital) enable some sample task completion?”. In addition, we aimed to learn about user perspectives regarding the overall concept of physical visualization.

5.2 Participants

We recruited 16 subjects (8 male, 8 female) through acquaintance, all of whom had some prior medical training/knowledge. Fourteen participants were second-year medical students (all with varying prior backgrounds), one was a family medicine resident and one was a general internal medicine doctor. There were two main reasons for recruiting users with some medical training: (1) having medical background eliminated the need for the experimenter to explain basic blood flow
patterns in the body, and (2) people within the medical field are promising candidates for using these tools in practice. Formal consent for all participants was obtained at the beginning of each session.

5.3 Procedure

In each session, users had the opportunity to view and interact with the physical models (two vector field glyph models and one streamline model), as well as comparable digital models of each (see Fig. 5.1) which were displayed using a freely available, conventional scientific visualization software: Paraview [Ayachit, 2015]. There were three phases of the study: (1) tasks, (2) post-task questionnaire/survey, and (3) qualitative interview. A brief description of the basic interaction controls (i.e. rotating, panning, and zooming in/out using a mouse) for Paraview was provided before the study commenced as none of the participants had prior experience with Paraview. Video recording (with a phone camera set up on a tripod) was used to collect data for the task and interview phases of the study. Occasional notes were also taken by hand.

For the task phase, users were asked to complete three different tasks. Tasks marked with * indicate comparative tasks, which were executed once using the physical model and once using the digital model. The ordering of the tasks alternated between participants (i.e. one participant would do the task using the digital model first, and the next participant would do the task using the physical model first).

1. Between two slices of the vector field glyph model, determine which has the higher flow magnitude. *

2. After viewing and interacting with both physical and digital streamline models, select one and use it to briefly describe what you see happening in the flow.

3. Compare two streamlines (as identified by the experimenter) and report which one you believe is closer to you, using your initial viewpoint as a reference. (Users are

70
Figure 5.1: (A) User study setup, with screenshots of corresponding digital visualizations for (B) the glyph models and (C) the streamline model.

allowed to interact with the model.)*

The post-task questionnaire had four statements which were ranked on a 5-point Likert scale (1-strongly disagree, 3-neutral, 5-strongly agree) for each of the four visualization types (physical glyph model, digital glyph model, physical streamline model, and digital streamline model). The first three statements were the same for all visualizations, the last statement differed only between glyph/streamline models:

- The visualization was clear and easy to understand.
- The visualization was easy to interact with.
- It was easy to see/navigate different parts of the data.
• The interaction technique allowed me to easily compare different parts of the data (glyph models) and understand the shape and direction of the data (streamline models).

Finally, the user study concluded with three questions in the interview phase:

• In general, what do you think of working with physical models vs. digital models?
• What is your overall opinion about physical visualization?
• What do you think of the physical glyph model vs. streamline model?

5.4 Results and discussion

When comparing slices using the glyph models (Fig. 5.2, top), users were generally faster when using the physical model over the digital version of the same – an average time of 14 seconds with the physical model vs. 21 seconds with the digital (statistical $p$-value = 0.031). In all cases (i.e. using either physical or digital glyph models), all participants picked the correct slice.

Based on the scores of the post-task questionnaire, it appeared that users generally preferred the digital glyph models over the physical models (see Table 5.1). However, this difference was not statistically significant for any given aspect (Table 5.1). Given these results, we consider the physical glyph model to be at least comparable to a similar digital representation, with the advantage of enabling faster comparisons between slices.

As for the streamline models, slightly more participants (10 of 16) chose to use the digital version to describe the flow behaviour, rather than the physical. Regardless of which model was chosen, all participants described some aspect of the flow accurately. Those who chose to use the physical model tended to gesture with their hands while describing the flow. For the comparative task, it appeared that majority of participants were faster using the physical model over the digital – an average time of 10 seconds with the physical model vs. 15 seconds with the digital (Fig. 5.2, bottom) – but this was not considered statistically significant at the 5% significance level ($p$-value =
Table 5.1: Summary of glyph model Likert questionnaire.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Median answer for physical model</th>
<th>Median answer for digital model</th>
<th>p-value from Wilcoxon signed-rank test</th>
</tr>
</thead>
<tbody>
<tr>
<td>The visualization was clear and easy to understand.</td>
<td>4</td>
<td>4.5</td>
<td>0.125</td>
</tr>
<tr>
<td>The visualization was easy to interact with.</td>
<td>4</td>
<td>4</td>
<td>0.883</td>
</tr>
<tr>
<td>It was easy to see/navigate different parts of the data.</td>
<td>4</td>
<td>5</td>
<td>0.766</td>
</tr>
<tr>
<td>The interaction technique allowed me to easily compare different parts of the data.</td>
<td>4</td>
<td>4.5</td>
<td>0.285</td>
</tr>
</tbody>
</table>

0.062). (Note: only 15 participants’ data were used since one participant’s records of working with the streamline models were partially lost due to equipment malfunction.) However, perhaps more importantly, depth perception accuracy was 100% when using the physical model, as opposed to 80% when using the digital model. This supports the idea of physical visualization enabling better, more natural depth perception. In terms of the Likert questionnaire results, neither the physical nor digital model emerged as better than the other in any aspect (see Table 5.2).

Table 5.2: Summary of streamline model Likert questionnaire.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Median answer for physical model</th>
<th>Median answer for digital model</th>
<th>p-value from Wilcoxon signed-rank test</th>
</tr>
</thead>
<tbody>
<tr>
<td>The visualization was clear and easy to understand.</td>
<td>4</td>
<td>4</td>
<td>0.148</td>
</tr>
<tr>
<td>The visualization was easy to interact with.</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>It was easy to see/navigate different parts of the data.</td>
<td>4</td>
<td>4</td>
<td>0.380</td>
</tr>
<tr>
<td>The interaction technique allowed me to easily understand the shape and direction of the data.</td>
<td>4</td>
<td>4</td>
<td>0.271</td>
</tr>
</tbody>
</table>

Overall, the results related to the streamline models seemed inconclusive; there was no distinct advantage of physical over digital or vice versa. However, this user study did give us some insight
into our physical streamline model design. We had originally hypothesized that the intermittent slices (which necessarily split most streamlines over at least two sections) would not interfere with perceiving each line as a whole. This hypothesis was based on the ideas of Gestalt theory [Desolneux et al., 2007], which suggest that people tend to understand things as a whole rather than as individual parts. But, at least three participants mentioned that they found the slices obtrusive and/or that the lines were difficult to follow through the model. Since there are quite a number of lines, it makes sense that the cognitive demand of interpreting the lines within each slice and simultaneously trying to combine the lines between slices would be quite high.

During the interview phase, majority of participants expressed appreciation for both physical and digital models. Some noted that they found manipulation easier with the digital model (e.g. being able to rotate freely) while handling the physical model was less fluid due to the stand design: one of the medical students noted that although the physical model “added tactile sensation”, it was “a bit clunkier because you can only rotate [the slices, so] it’s hard to pick it up and manipulate.”

The idea of physical size was also highlighted during the interview phase; at least seven of the participants noted the importance of understanding real-world scale using physical models. For instance, one of the medical students (who has background experience in radiology research) noted that “if [he was] thinking about actual size, [the physical model] is better if it’s to scale, whereas digital ... doesn’t work as well because you can zoom in/zoom out.”

Some participants mentioned that they appreciate the tangibility of physical models and indicated its particular usefulness as a tool for teaching and learning, especially for people who prefer hands-on learning. One participant believed that physical models would have less of a learning curve, especially for those who are not technologically inclined. Another participant noted that “physical models are super helpful, especially if you’re trying to explain something to a patient. ...it adds another level of interaction with that object, so instead of just looking at it...in a two-dimensional way, now [you can hold] it.”

This appreciation of both physical and digital visualization is evident in the results – although
majority of participants completed the tasks faster with the physical models, majority also ranked the digital models higher than their physical counterparts. We believe that this may be due to the demographic studied: almost all of the participants were between 18-35 years of age and all seemed relatively comfortable with manipulating the digital model on screen. Advanced image rendering and interaction technology is increasingly ubiquitous, whereas physical tools for (flow) visualization are perhaps unheard of. Thus, there is typically a natural hesitation to adopt an unfamiliar method of communicating complex flow data; as such, we suppose that increased exposure might change users’ level of comfort with physical visualization. (Indeed, one participant described their own high level of familiarity with digital interfaces and considered that physical visualization is “a new and different way of looking at data that I need more experience with to understand better.”)

Two participants (one medical student and the other, a general internal medicine doctor) mentioned lack of portability as a drawback to using physical models in the context of healthcare: “there’s a limitation to carry physical things”, and “although [physical models] are easy for me to visualize, [they’re] space-consuming.” However, a different participant (who had previous experience working in pediatrics) liked the idea of interacting with patients using a physical model over a digital one. This participant described their prior experience of using a laptop with children, which would create distractions: “when I worked in [pediatrics], kids would want to play Fortnite or whatever.” Furthermore, the participant anecdotally explained their frustration with technology that would sometimes fail to work (“every time I tried to pull something up on [a] computer, technology failed on me”); this experience resonates with the idea of physical visualizations being always “on” [Jansen et al., 2015].

5.5 Summary

By conducting a user study, we gained valuable insight into our physical model design. Many participants noted the merit of both physical and digital media for understanding complex data. While the physical models generally enabled accomplishing certain tasks faster, there was no
definitive evidence that the physical models were superior to comparable digital ones. However, the feedback and observations related to the physical models provided us with critical insight into our model design (particularly with regard to the multi-slice streamline model), which we explore further in Chapter 6.
Figure 5.2: For both the glyph model comparative tasks (top) and the streamline model comparative tasks (bottom), most users were faster using the physical model (blue) over the digital model (grey). Three users chose the incorrect streamline when working with the digital model (red).
Chapter 6

Alternate designs

In this thesis, we aim to design and fabricate 3D printable physical visualizations of (some aspect of) blood flow data. The particular “aspect” of blood flow data is not strictly defined, but since the idea of physicalizing blood flow has not been extensively studied, we started with aiming to represent the original vector field data at a single snapshot in time, first using glyphs and then using streamlines. After conducting the user study (Chapter 5), we realized that there were characteristics of our physicalization design that could be improved. For instance, in light of our results and observations, we improved our glyph model by creating a dual-colour print. This highlights the value of a dual extrusion 3D printer. We also improved interaction with the streamline model by reducing the number of slices used for support, and re-sizing the model such that it can be easily held in hand.

After these design improvements, we received some feedback from four medical experts, primarily radiologists (three experts in cardiac MRI and one surgeon). Although some of the more senior radiologists did not feel that a physical representation was necessary (after many years of experience, they felt that they could adequately reconstruct 3D geometry from 2D images in their minds), some of the radiologists specifically involved with pediatrics expressed interest in having physical models to complement digital visualizations. This aligns with the user study results and supports the idea of using physicalization, particularly for pediatric applications. Additionally, one expert noted that more simplified flow models would also be useful.

Based on the expert feedback that we received, we considered alternative styles for visualizing flow. Our original glyph and streamline models aimed to be as comprehensive as possible, without becoming overly cluttered. This came from the mindset of representing as much data as possible. However, since one of the experts suggested that creating more simplified flow models would also
be useful, we decided to explore this idea with two example cases: a pathline predicate model, and a vortex core model. Both of these can be considered as “summary models”, designed to give a simplified overview of (some aspect of) blood flow character over the cardiac cycle.

The following sections describe and showcase both our design improvements (Sections 6.1 and 6.2), as well as the new summary models we developed (Sections 6.3 and 6.4). We conclude the chapter by demonstrating how our proposed models can illustrate the differences between a healthy subject and a patient with cardiomyopathy (Section 6.5).

6.1 Two colour glyph model

During the user study, a few participants mentioned that it would be nice if the glyphs and slices had distinct colours. This was not possible using the MakerGear M2 since it has a single extruder, but it is possible with the M3-ID. A sample slice was printed to illustrate the potential (Fig. 6.1). The difference in colour allows for definite distinction between the slice (context) and vector glyphs.

Figure 6.1: Two colour glyph model, printed using MakerGear M3-ID.
6.2 Single slice streamline model

Using the results and observations from our user study, we decided to revisit the slice-based design of the streamline model. In particular, we wanted to know if there was an alternative set of slices that could adequately support the streamlines but with less obstruction of the streamline data, as several participants had commented on the difficulty of understanding the streamlines when broken into multiple slabs. During the user study, we also noticed that the wheel-and-axle construction seemed more natural when handling the glyph model (e.g. for comparing slices or inspecting a specific subset of data); the idea of comparison between slices using the streamline model is not that valuable. The slice(s) in the streamline model primarily provide support and anatomical/data context.

In light of this, we decided to use a single slice design. We chose a slice that approximates a long-axis cross sectional view, since it is one of the basic cardiac imaging views. Additionally, it has a relatively large cross sectional area, which provides enough support for the streamline structures. Once printed and assembled (Fig. 6.2), the model can be handled with relative ease. Furthermore, the long-axis slice combined with the overall extent of the streamlines seem to provide sufficient anatomical context, so no additional stand or structures were printed. By creating a somewhat irregular shape (akin to a “potato”) we hope to encourage user interaction with the model – picking it up, inspecting it, etc. [Herman and Keefe, 2018]. The user study also elucidated the benefit of physical scale, so we ensured that the single slice streamline model was printed to scale.

6.3 Summary model: pathlines

Aside from our design improvements, we also developed two additional summary models. The first of these uses *pathlines* to convey the flow data. Pathlines are integral lines, very similar to streamlines, which are derived for time-varying, unsteady vector fields. A pathline is often described as the trajectory that a massless particle would follow in the flow field over time. As such, a pathline \( p(t) \) is defined similarly to a streamline (Eq. 2.2) but is parameterized by \( t \) (physical
time) rather than $\tau$. We create an initial set of pathlines $\mathcal{P}$ as follows: we use the same spatial seeding strategy described in Section 3.3 at $t = 0$, then we trace pathlines from the seeds over the cardiac cycle ($t \in [0, t_f]$) using RKF-45 [Mathews and Fink, 2004]. Note that $t = 0$ represents early diastole and $t = t_f$ corresponds to phase just before early diastole, which is typically end systole.

The set of pathlines, $\mathcal{P}$, is likely to be dense, confusing, and nearly impossible to 3D print. As with glyphs and streamlines, minimizing clutter is always desirable and pathlines are no exception. Moreover, the goal of this model is to create a summarized depiction of the flow and should arguably be even less cluttered than the glyph and streamline models. Hence, to filter $\mathcal{P}$ into a printable and simplified set of pathlines, we chose to use pathline predicates. Pathline predicates [Salzbrunn et al., 2008, Jankowai et al., 2016] are Boolean functions which can classify and filter pathlines based on user-defined attributes (e.g. residence time, maximum velocity, passing through a region of interest, etc.). Combining different predicates using Boolean logic allows users to answer questions, such as “which pathlines pass through a particular region and have the highest speed?”.

For our example pathline predicate model, we decided to ask the question “which pathlines have the highest speed and are the longest?”, thus combining a predicate for maximal speed (at some point along the pathline) and length. We define our predicates based on the length and maximum velocity predicates described by [Jankowai et al., 2016]. By applying these predicates, we reduce
the number of pathlines from 106 to 10 (Fig. 6.3). (Note that the exact number of pathlines in the final visualization is chosen by the user.)

Figure 6.3: The number of pathlines is reduced to a simple set which summarizes the main flow path over the cardiac cycle.

Thus far, our physical models have represented flow behaviour at a snapshot in time. However, since pathlines are derived from time-varying data, the physical model can be designed to reflect this. We modified our sweep surface algorithm (Section 3.3.2) such that the cross-sectional radius is linearly scaled based on the time that the “particle” has travelled along the pathline (Fig. 6.4). This gives the pathtube a tapered appearance, suggesting the idea of movement over time; these are similar to motion lines which are common in traditional comic book art [Cohn, 2013].

By leveraging the single slice technique described in Section 6.2 we create a physicalizable pathline predicate summary model (Fig. 6.5). The summary print has only ten pathlines but provide an overview of the predominant flow behaviour – for instance, three of the pathlines are seen merging into one general direction, corresponding to ejection during systole.
6.4 Feature-based summary model: vortex cores

Up to this point, we have focused on some of the fundamental flow/vector field visualization techniques, such as glyphs and integral lines. Creating an overview visualization of the flow conveys the overall flow behaviour but does not necessarily highlight interesting features. Often, we would like to explore distinct characteristics in the flow – for instance, in the study of cerebral aneurysms, *residence time* is a feature of interest [Oeltze et al. 2014]. There are a number of flow-related features and hemodynamic parameters, such as pressure, wall shear stress, etc. One feature that has garnered particular interest within the context of intracardiac blood flow in recent years is *vortical flow* [Pedrizzetti et al. 2014, Kheradvar and Pedrizzetti 2012, ElBaz et al. 2014].
et al., 2014, Sengupta et al., 2014]. Pedrizzetti et al. (2014) suggest that maladaptive intracardiac vortices may be involved in LV remodelling and could provide early indications of long-term outcomes. As such, vortex cores seemed to be a suitable flow feature worth exploring in the context of cardiac blood flow. Thus, we use this as a case study for physicalizing flow features.

Robust vortex extraction is a challenging problem, with a formal definition of a vortex still lacking [Jiang et al., 2005]. Numerous methods for vortex core detection have been proposed [Jiang et al., 2005]; the method presented by Jeong and Hussain [1995] known as the $\lambda_2$ method, is generally regarded as the most suitable for vortex core extraction for blood flow in the cardiovascular system [Kheradvar and Pedrizzetti, 2012]. Therefore, we chose this method for vortex core extraction.

To create the vortical flow model, we take the following steps:

1. Use the $\lambda_2$ method for vortex core region extraction.
2. Create isosurfaces of the vortex core regions.
3. Slice the model.
4. Generate a pathline $P$ to represent the predominant flow direction.
5. Project $P$ to the slice plane.

In step 1, the $\lambda_2$ method is applied to a re-sampled version of the vector field. Rather than using the original data, which usually consists of anisotropic voxels, we re-sample (using trilinear interpolation) such that our new vector field has isotropic 1 mm$^3$ voxels. In addition, the velocity data can be noisy; therefore, we use a user-tuned threshold $\tau_{\lambda_2} < 0$ to distinguish the final vortex core regions [ElBaz et al., 2014]. Generally speaking, if $\mu$ represents the mean of all $\lambda_2$ where $\lambda_2 < 0$, then $\tau_{\lambda_2} = 4\mu$ gives a reasonable result.

In step 2, isosurfaces are created for the vortex core regions. Due to the sensitivity of the method, some very small voxels are detected as vortex cores. therefore, we set a minimum number of voxels as a threshold for a given connected component to be considered a valid vortex core.
Steps 3-4 involve applying previously described methods. For slicing, the long-axis single slice technique described in Section 6.2 is again suitable for supporting the 3D printed vortex core structures (Fig. 6.6). Moreover, a summary trajectory line (e.g. a pathline derived from a combination of various predicates, such as seeding plane, maximum velocity and/or length) to provide time-varying information can be included in the physical vortex core model; this can be generated as discussed in Section 6.3. Since the trajectory line is not the focus of the visualization, it can be projected onto the slice plane, acting as part of the model’s context.

![Image](image-url)

Figure 6.6: Pictures of healthy vortex core summary model. (A) Long-axis view, with vortex cores marked by arrows. (B) Short-axis view, a vortex core ring structure (C) can be observed.

6.5 Patient example

To further test our improved designs, we applied them to a dataset of a patient with cardiomyopathy (a disease of the heart muscle). Fig. 6.7 shows comparisons between the healthy subject and patient using three of the different flow models (streamlines, pathline predicates and vortex cores).
In Fig. 6.7A, the difference in streamline flow pattern structure show the disorganized flow in the patient’s LV during early filling when compared to the healthy subject. Fig. 6.7B shows that a greater proportion of blood flow enters/exits the LV within one cardiac cycle in the healthy subject as compared to the patient, suggesting that the healthy subject can transport oxygenated blood more efficiently. Finally, in Fig. 6.7C, an early vortex ring-like structure can be seen in both cases but the vortex core structure of the patient is not as well-formed. Hence, these examples show the potential of using physicalization to portray the hemodynamics of different cases.

6.6 Summary

Using the user study results and expert feedback, we explored alternate blood flow physicalization designs. Some of these designs were simple improvements on our existing glyph and streamline models (namely, dual-extrusion printing for the glyph model and a single long axis slice for the streamline model). The other designs were developed based on the idea of fabricating simplified summary models, which give a much higher level view of the flow behaviour. To do this, we developed a pathline model which utilized pathline predicates, and we also designed a vortex core model since vortices within the LV have been a popular area of recent research. The design framework for these newer single slice models was used to generate physicalizations for both a healthy subject and a patient who has cardiomyopathy; these examples illustrate the potential of using flow physicalization to understand healthy vs. pathological cases.
Figure 6.7: Comparison between LV of healthy subject (left) to cardiomyopathy patient (right) in photos of (A) two printed streamline (half-)models, (B) pathline predicate models, and (C) vortex core summary models.
Chapter 7

Implementation

In this chapter, we present some of the algorithms used for creating and filtering the various flow representations. Here we focus on streamlines, pathlines and vortex cores. We also summarize the various tools used for creating the different flow physicalizations. Note that, within the context of this thesis, the emphasis was on creating a design framework for physicalizing flow data – as such, optimizing for efficiency was outside the scope of our work.

7.1 Generating and filtering flow objects

To create physicalizable flow representations requires consideration of how to generate and select flow objects such that the resulting visualization is not overly cluttered. For generating a filtered set of streamlines, we based our implementation on the work of Chen et al. [2007]. A sketch of the algorithm is provided in Algorithm 1. For generating pathline summary models, we used a sequential form of pathline predicates (i.e. applying a speed predicate, followed by applying a length predicate); this is sketched in Algorithm 2. An alternative method for predicates could adopt the use of Boolean operations, such as intersection. Finally, for detecting and visualizing vortex cores, we generally follow the method presented in ElBaz et al. [2014] (Algorithm 3).

7.2 Tools used

Aside from developing our own algorithms, we also use some existing tools to create our physical flow models. Here we briefly mention the key ones used.
Algorithm 1 Similarity-based streamlines

1: function INITIALIZE()
2: initialize bounding volume \( D' \leftarrow \text{BoundingBox}(D) \)
3: initialize seeds, \( s_i \in D' \)
4: randomize seeds
5: end function

6: function GENERATESTREAMLINES()
7: for all seeds \( s_i \) do
8: \( S_i[0] \leftarrow s_i \)
9: \( \text{currLen} \leftarrow 0 \)
10: while \( \text{currLen} < \text{maxLength} \) do
11: \( \text{GrowStreamline}(S, S_i, \text{forward}) \)
12: \( \text{GrowStreamline}(S, S_i, \text{backward}) \)
13: \( \text{currLen} \leftarrow \text{ArcLength}(S_i) \)
14: end while
15: end for
16: end function

17: function GrowStreamline(\( S, S_i \), direction)
18: while AdaptiveRKF45Step(\( S_i \), direction) succeeds do
19: if \( \text{SimilarityDistance}(S_i, S_i) < d_{self} \) then
20: return
21: end if
22: for all \( S_j \in S \) do
23: if \( \text{SimilarityDistance}(S_i, S_j) < d_{sim} \) then
24: return
25: end if
26: end for
27: end while
28: end function

7.2.1 Multi-slice models

For creating the glyph models, a macro was written in Paraview to automatically generate the slice-based model with user-specified orientation, slice spacing, and glyph size scaling (to prevent collisions between slices). Paraview’s Clip tool, which can extract a subset of the data, was used for slicing the models. The glyphs themselves were also generated within Paraview.

The program for generating the set of streamlines using the similarity distance metric and converting the streamlines into meshes was implemented in MATLAB (R2016a).

Meshmixer (©Autodesk, CC-BY-NC-SA 3.0)) was used for building the handles and base, as
Algorithm 2 Pathline predicates

1: function \textsc{GeneratePathlines}()
2: \textbf{for all} seeds \( s_i \) \textbf{do} \hspace{1cm} \triangleright \text{Trace each pathline}
3: \hspace{1 cm} \( P_i[0] \leftarrow s_i \)
4: \hspace{1 cm} \textsc{RKF45Pathline}(P_i)
5: \hspace{1 cm} \( \mathcal{P} \leftarrow P_i \)
6: \textbf{end for}
7: end function

8: function \textsc{SequentialPredicates}(\mathcal{P}) \hspace{1cm} \triangleright \text{Target number of pathlines in final set}
9: \hspace{1 cm} \( m \leftarrow 10 \)
10: \hspace{1 cm} \textsc{SortByMaxSpeed}(\mathcal{P})
11: \hspace{1 cm} \mathcal{P}' \leftarrow \mathcal{P}[1 : 2m]
12: \hspace{1 cm} \textsc{SortByLength}(\mathcal{P}')
13: \hspace{1 cm} \textsc{ConvertToMesh}(\mathcal{P}'[1 : m]) \hspace{1cm} \triangleright \text{Top } m \text{ longest of top } 2m \text{ fastest pathlines}
14: end function

Algorithm 3 Vortex cores

1: function \textsc{VortexCoreDetection}()
2: \( f' \leftarrow \text{LinearlyResampleVectorField}(f, [1, 1, 1]) \)
3: \hspace{1 cm} \( V_{\lambda_2} \leftarrow \textsc{Lambda2}(f') \)
4: \hspace{1 cm} \( I_v \leftarrow V_{\lambda_2} < \tau_{\lambda_2} \)
5: \hspace{1 cm} \( I'_v \leftarrow \text{ConnectedComponentAnalysis}(I_v) \) \hspace{1cm} \triangleright \text{Keep sufficiently large connected components in final vortex core visualization}
6: \hspace{1 cm} \textsc{Isosurface}(I'_v, 0.5)
7: end function

well as for ensuring that the models would be 3D printable (i.e. having no open boundaries).

7.2.2 Anatomical context

Creating a smooth representation of the LV endocardium was implemented in MATLAB (R2016a), using some helper tools (e.g. calculating point-mesh closest distances and barycenters, writing .stl files) from the Geometry Processing toolbox [Jacobson et al., 2018]. For generating slices, we also used the mesh boolean operation implemented in Jacobson et al. [2018]. Comparison to marching cubes was done using Hammer [2013].
7.2.3 Single slice models

The single slice streamline model used the same set of streamlines generated in the multi-slice flow model section. Generating the summary pathline and vortex core models was also implemented in MATLAB (R2016a).

7.2.4 3D printing

Prior to 3D printing, the model (e.g. exported as an .stl file) needs to be converted into gcode (instructions for the printer). For this, we used Simplify3D to load the .stl file and automatically generate support structures, as well as generate the final gcode files. Ultimately, all physical models were fabricated using MakerGear’s M2 and M3-ID printers with AMZ3D PLA filament.
Chapter 8

Conclusions and future work

4D Flow MRI is an exciting technology which captures volumetric blood flow data over time and has much potential for both research and clinical use. One of the important pieces of the 4D Flow MRI processing pipeline is flow visualization, which can help various users (e.g. doctors, patients, researchers, students, etc.) better understand the data. There are many ways to visualize flow data, including vector field maps, integral lines, and feature-specific displays. However, to truly grasp the three-dimensional (or four-dimensional, when considering time) nature of the data, it may be beneficial to explore it using a three-dimensional, physical visualization. To this end, we designed a novel slice-based physicalization method for visualizing 4D Flow MRI data, specifically focusing on blood flow within the left ventricle.

Overall, this study provides a proof-of-concept on the feasibility of a novel physical visualization of blood flow within an actual human heart. We demonstrate that our proposed slice-based design is easily fabricable, and has the potential to be useful for physicalizing blood flow data. Since this area has not yet been extensively explored, we initially focused on two styles of visualization, glyphs and streamlines: one can consider glyphs as the lowest level of visualization since they most closely correspond with the raw vector field data, and streamlines can be thought of as one level higher, being derived from the vector values in space. We evaluated our physical model design in comparison to a more conventional digital format, and found that the two were in some senses comparable, with different users finding pros and cons for each. Generally speaking, the study provided valuable feedback and insight into the physicalization design, and confirmed that there were uses for physical models in addition to digital ones.

Based on the user study results and further expert feedback, we decided to go beyond the original two styles and explored two simplified summary model designs. One model was based on the idea
of pathline predicates, and the other created a physicalized visualization of detected vortex cores. Both summary models represent the time-varying aspect of the data as well.

8.1 Limitations and future work

While our presented workflow (as-is) is not intended for routine clinical use, it provides a tool for research applications, particularly in studies which utilize 3D printing. As an example, specific flow features (for instance, regurgitant flow) could be modelled and physicalized to complement other anatomical 3D prints, thus augmenting structural information with hemodynamics. This could be particularly useful for conditions where flow plays an important role; for example, in congenital heart disease, it could be valuable to understand both the structural defect and how much the flow pattern changes as a result. Such future designs will be made in collaboration with clinical experts, tailored to potential use cases/audiences (e.g. pediatrics).

Our current framework respects the original data and ensures 3D printability; however, we have only explored a couple of possibilities for higher level visualizations (e.g. feature-based). Various clustering techniques have been applied to 4D Flow streamline data [Meuschke et al., 2016, Oeltze et al., 2014]; these could be used to create more simplified flow overviews for physicalization as well. A number of different hemodynamic parameters can also be derived from 4D Flow data, which could enrich the content of the physicalization. We hope to investigate such future designs in collaboration with clinical experts as well. These types of feature-based models could also have utility in patient/trainee education, which we hope to study in more depth (e.g. with longitudinal user studies).

Additionally, while 4D Flow MRI provides data over the cardiac cycle, we focused on visualizing flow with 3D models that were ultimately static in nature (sometimes focusing on discrete snapshots in time, such as early and late diastolic filling, other times summarizing the time-varying information in a single model). In the future, we hope to also explore methods for dynamic physical visualization of flow over the cardiac cycle.
Aside from the more medically-focused future work, we also note that our current slicing strategy was mainly rooted in traditional approaches to slicing (e.g. creating long-axis or short-axis slices). Although this was sufficient for our purposes, it might be worthwhile to explore an optimization framework for determining the best slice(s) for a given model. For instance, the optimization could attempt to minimize the amount of support material required while still preferentially searching for known slice orientations.

In terms of evaluating our physicalized models, we also note that testing other visualization methods in the future (such as more sophisticated rendering techniques, stereoscopic displays, and virtual reality) would be valuable to understand the benefits and drawbacks of physical flow models in a broader visualization context.

8.2 Concluding remarks

This thesis presents a framework for physicalizing cardiac blood flow data coming from 4D Flow MRI. For the purpose of proving the concept, we focused on blood flow within the left ventricle and demonstrated four different styles (glyphs, streamlines, pathlines, and vortex cores) that could be used to physically visualize real blood flow data. The ideas presented in this thesis can be used in further research of topics such as creating patient-specific physical models (with both structural and hemodynamic information) for surgical planning.
Bibliography


Mohammed S. M. ElBaz, Boudewijn P. F. Lelieveldt, Jos J. M. Westenberg, and Rob J. van der Geest. Automatic extraction of the 3d left ventricular diastolic transmitral vortex ring from 3d whole-heart phase contrast mri using laplace-beltrami signatures. In Oscar Camara, Tommaso


99

Jochen Jankowai, Rickard Englund, Timo Ropinski, and Ingrid Hotz. Interactive 4d mri blood
flow exploration and analysis using line predicates. In Proceedings of SIGRAD 2016, May 23rd
and 24th, Visby, Sweden, number 127, pages 35–42. LinkÃämping University Electronic Press,
LinkÃämpings universitet, 2016.

Yvonne Jansen, Pierre Dragicevic, and Jean-Daniel Fekete. Evaluating the efficiency of physical
2481359.

Yvonne Jansen, Pierre Dragicevic, Petra Isenberg, Jason Alexander, Abhijit Karnik, Johan Kildal,
Sriram Subramanian, and Kasper Hornbæk. Opportunities and challenges for data physicaliza-

Jinhee Jeong and Fazle Hussain. On the identification of a vortex. Journal of Fluid Mechanics,

Ming Jiang, Raghu Machiraju, and David Thompson. A novel approach to vortex core region
536-X. URL http://dl.acm.org/citation.cfm?id=509740.509777

Ming Jiang, Raghu Machiraju, and David Thompson. Detection and visualization of vortices. In


Alex Meckes. Today’s 3D printing workflow: at least 6 steps, Jun 2016. URL [https://blog.grabcad.com/blog/2016/06/06/todays-3d-printing-workflow/](https://blog.grabcad.com/blog/2016/06/06/todays-3d-printing-workflow/).


ISSN 1861-6429. doi: 10.1007/s11548-010-0476-x. URL https://doi.org/10.1007/s11548-010-0476-x.


Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing

Simplify3D. Simplify3D 4.0. [https://www.simplify3d.com/](https://www.simplify3d.com/), a.


Kunihiko Taira, Yiyang Sun, and Daniel Canuto. 3d printing of fluid flow structures, 2017.


Appendix A

User study supplementary material

The following pages contain the supporting documentation from the user study described in Chapter 5 including:

- Blank pre-questionnaire
- Blank post-questionnaire.
DEPARTMENT OF COMPUTER SCIENCE 
PRE-SESSION QUESTIONNAIRE

CREATIVE VISUALIZATION AND INTERACTION OF/WITH MEDICAL DATA.

Demographic Information
This information is collected for demographic purposes only. All questions are optional.

Age:  ☐ 18-25  ☐ 26-35  ☐ 36-45  ☐ 46-55  ☐ 56-65  ☐ 66 or older

Gender:  ☐ Male  ☐ Female

Are you currently a student?  ☐ Yes  ☐ No

If you are a student, please indicate your current level of education. If you are not a student, please indicate the highest level of education you have completed:

☐ High school or equivalent  ☐ Master’s degree
☐ Vocational/technical school (2 year)  ☐ Doctoral degree (PhD)
☐ Some university  ☐ Professional degree (MD, JD, etc)
☐ University graduate (4 year)  ☐ Other ____________________

What is your field of study or expertise, or area of work and experiment?

Are you colour-blind?  ☐ Yes  ☐ No

[FOR RESEARCHER USE ONLY] SESSION ID: ___ PARTICIPANT ID: ___
Please indicate the **approximate length of time you have trained, worked, or had an active interest in the following areas:**

<table>
<thead>
<tr>
<th>Area</th>
<th>none</th>
<th>&lt; 1 year</th>
<th>1 – 2 years</th>
<th>3 – 4 years</th>
<th>5 + years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive data visualization</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Medical data visualization</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3D printing and/or rapid prototyping</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Please indicate approximately **how often** you use following technologies. Place a checkmark on the most suitable choice:

<table>
<thead>
<tr>
<th>Technology</th>
<th>never</th>
<th>a few times a year, or sporadically</th>
<th>a few times a month</th>
<th>a few times a week</th>
<th>almost every day or more</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scientific visualization interfaces</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Modelling software (e.g. 3D Max, Maya, Rhino, etc.)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
**CREATIVE VISUALIZATION AND INTERACTION OF/WITH MEDICAL DATA.**

*Please indicate which statements best summarize your experience when interacting with the physical vector field model:*  

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The visualization was clear and easy to understand</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The visualization was easy to interact with</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>It was easy to see/navigate different parts of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The interaction technique allowed me to easily compare different parts of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

*Please indicate which statements best summarize your experience when interacting with the digital vector field model:*  

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The visualization was clear and easy to understand</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The visualization was easy to interact with</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>It was easy to see/navigate different parts of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The interaction technique allowed me to easily compare different parts of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
Please indicate which statements best summarize your experience when interacting with the **physical streamline model**:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The visualization was clear and easy to understand</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The visualization was easy to interact with</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>It was easy to see/navigate different parts of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The interaction technique allowed me to easily understand the shape and direction of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Please indicate which statements best summarize your experience when interacting with the **digital streamline model**:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The visualization was clear and easy to understand</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The visualization was easy to interact with</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>It was easy to see/navigate different parts of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>The interaction technique allowed me to easily understand the shape and direction of the data</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>