Build a 4D MRI Blood-flow Velocity Atlas

Submitted by
Hongping Feng

Advised by
Dr. Anna Vilanova
Department of Computer Graphics & Visualization Group
Delft University of Technology
Tutored by
Lluís A. Belanche
Computer Science Department
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

18th June, 2019
Abstract

Cardiovascular diseases (CVDs) are the global number one cause of death. Current diagnosis of CVDs is mainly based on functions and morphology of cardiovascular structures. Magnetic resonance imaging (MRI), especially the 4D flow MRI has become a new and powerful tool for CVD diagnosis, which allows both anatomical and functional analysis in a single measurement. However, it is still difficult to efficiently utilize this data source to diagnose CVDs. One of the main reasons is the lack of standard analysis and understanding of this rather new imaging modality. The common method to understand the healthy behaviour and its variation is to build an atlas. Current blood flow atlas does not support 3D velocity vector field full information. We have built a statistical vector field model (SVFM) based on 4D flow MRI, which is similar to statistical shape and appearance models by applying Principal Component Analysis (PCA) to get the mean and main variations of the blood flow fields. The visualization of this atlas would help physicians understand the 4D MRI data and assist them to identify whether the data is abnormal or not. In order to get more efficient representations of variations for vector fields, we also tested Complex PCA and Quaternion PCA, besides traditional Real PCA. Finally, we visualize the results such that they can be interpretable.
Acknowledgement

Firstly, I would like to sincerely thank my thesis supervisor, Dr. Anna Vilanova, for helping me with kindness and patience in the registration process, and providing me invaluable guidance throughout the whole research process. I really enjoy the time discussing with you every week, and feel grateful for your effort helping me tackling every obstacle I encountered during the project.

I also want to thank my thesis tutor, PhD. Lluís A. Belanche, who is so kind to agree to accept me as his student and give me guidance on my thesis.

I would like to thank PhD. Klaus Hildebrandt, who has provided me a lot of help solving the problems with PCA.

I would like to thank Ruud de Jong, who is so nice and has helped me solve the installation problems of Elastix.

I want to thank David Gobbi, who is so patient and knowledgeable, helping me solve the problem of building vtk from source when I wanted to use Tkinter to embed vtk rendering.

I also want to thank Mrs. Inge Grootjans from TUDelft international office, and Carol and Ginesta from FIB, UPC, in guiding me to successfully finish the registration process in TuDelft.

Finally, I want to thank my parents, who are always supportive for everything I am dedicated to do and every audacious decision I have made.
Contents

1 Introduction and Motivation 8
  1.1 Background ................................................. 8
  1.2 Motivation ................................................. 9
  1.3 Main Objective ............................................. 10
  1.4 Challenges ............................................... 10
  1.5 Principal Contributions ................................. 11
  1.6 Thesis Structure .......................................... 11
  1.7 Specification .............................................. 12

2 Related Work 13
  2.1 4D MRI Blood Flow Data ................................. 13
  2.2 Traditional PCA and Applications ..................... 14
    2.2.1 PCA in Statistical Shape Models (SSM) ............ 15
  2.3 Complex PCA .............................................. 17
  2.4 Quaternion PCA ........................................... 18
    2.4.1 Quaternions Algebra ................................. 19
    2.4.2 Quaternions Matrix ................................ 20
    2.4.3 SVD of a Quaternion Matrix (QSVD) ................. 21

3 Dataset 23
  3.1 Real Vessels .............................................. 23
  3.2 Synthetic Curved Tubes ................................. 23
    3.2.1 Overview ........................................... 23
    3.2.2 The Method of Generating Curved Tubes ............ 24

4 4D Flow Atlas Generation 27
  4.1 Overview ................................................ 27
  4.2 Registration of Vector Field Data Volumes ........... 28
    4.2.1 Temporal Maximum Intensity Projection (tMIP) .... 29
    4.2.2 Image Registration ................................ 30
    4.2.3 Velocity Reorientation ............................... 32
  4.3 Masking .................................................. 34
  4.4 Building a Statistical Vector Field Model (SVFM) .... 34
    4.4.1 Representation of a Blood Flow Velocity Vector ... 34
    4.4.2 Real PCA ........................................... 35
    4.4.3 Complex PCA ........................................ 36
    4.4.4 Quaternion PCA .................................... 37
    4.4.5 The Effect of Rotation ............................... 39
  4.5 4D Flow Visualization ................................... 43

5 Implementation 46
  5.0.1 Mathematica .......................................... 46
6 Result
6.1 QPCA theory test ........................................ 49
6.2 Experiments of Synthetic Dataset ......................... 50
   6.2.1 Preprocessing ....................................... 50
   6.2.2 RPCA and QPCA .................................... 50
   6.2.3 Interpretation ....................................... 53
6.3 Experiments of Real-world Dataset ....................... 57
   6.3.1 Sub-Volume Selection ............................... 57
   6.3.2 Registration and Reorientation ........................ 58
   6.3.3 Masking ............................................ 58
   6.3.4 RPCA and QPCA .................................... 58
   6.3.5 Interpretation ....................................... 59

7 Conclusion ................................................. 65
7.1 Main Findings and Implications ............................. 66
7.2 Future Work ............................................... 67

A Appendix ..................................................... 72
A.1 GUI ......................................................... 72
A.2 Visualization of 3rd Time Step of Real Vessels using Glyphs ........ 73
A.3 Visualization of 6th Time Step of Real Vessels using Glyphs .......... 74
A.4 Parameters of the Real Vessels ............................ 75
List of Figures

2.1 4D MRI blood flow data ................................. 14
2.2 Variation of mean shape. Adopted from the paper [20] .... 16
2.3 The mean shape and shape instances generated from the first four eigenvectors for the real, complex shape representations of human face. Adopted from the paper [26] ................. 18

3.1 3D blood flow visualization in the thoracic aorta ................. 24
3.2 Six examples of synthetic tubes ............................. 24
3.3 BÉZIER curve ................................................. 25
3.4 Created stream flow, vortex and helix .......................... 26

4.1 The four steps of the proposed solution ........................ 28
4.2 Population-based atlas construction ............................. 32
4.3 Velocity reorientation flow .................................... 33
4.4 The signs of eigenvectors in RPCA could be different in different methods, but they could represent the same eigenvectors .... 36
4.5 The angles of eigenvectors in CPCA could be different in different methods, but they could represent the same eigenvectors .... 37
4.6 The angles of eigenvectors in QPCA could be different in different methods, but they could represent the same eigenvectors .... 38
4.7 The representations before and after Rotation using the first eigenvector calculated in RPCA and CPCA ........................ 41
4.8 Glyphs of the thoracic aortas ................................ 44
4.9 Streamlines of the thoracic aortas .............................. 45

6.1 Explained variance in RPCA and QPCA for simulated data ... 49
6.2 Explained variance in RPCA and QPCA for synthetic tubes ... 51
6.3 Reconstruction experiments of RPCA and QPCA .................. 52
6.4 Reconstruction experiments using 10, 12, 18 samples ............ 53
6.5 Variation of mean volume of 24 synthetic tubes from RPCA ... 54
6.6 Variation of mean volume of 24 synthetic tubes from QPCA ... 55
6.7 Variation of mean volume of 6 synthetic tubes from RPCA ... 56
6.8 Variation of mean volume of 6 synthetic tubes from QPCA ... 57
6.9 Explained variance in RPCA and QPCA for thoracic vessels ... 59
6.10 Reconstruction experiments of RPCA and QPCA ............... 60
6.11 Variation of mean volume of five thoracic vessels from RPCA using streamlines ........................................... 61
6.12 Variation of mean volume of five thoracic vessels from QPCA using streamlines ........................................... 62
6.13 Variation of mean volume of 5 thoracic vessels from RPCA ... 63
6.14 Variation of mean volume of 5 thoracic vessels from QPCA ... 64

A.1 GUI panels ....................................................... 73
A.2 Three additional parameter panels of the GUI ................. 74
A.3 Variations of five thoracic vessels in the 3\textsuperscript{rd} time step from RPCA 74
A.4 Variations of five thoracic vessels in the 3\textsuperscript{rd} time step from QPCA 75
A.5 Variations of five thoracic vessels in the 6\textsuperscript{th} time step from RPCA 76
A.6 Variations of five thoracic vessels in the 6\textsuperscript{th} time step from QPCA 77
List of Tables

6.1 Relationship of real and quaternion eigenvectors . . . . . . . . . 54
A.1 Key parameters of thoracic aortas (1) . . . . . . . . . . . . . . 75
A.2 Key parameters of thoracic aortas (2) . . . . . . . . . . . . . . 76
Chapter 1

Introduction and Motivation

1.1 Background

According to World Health Organization (WHO) report in 2017, cardiovascular disease (CVD) is the global number 1 cause of death annually, which take away 17.9 million people’s lives every year, accounting for 31% of all global deaths. CVD is expected to be the major cause of mortality in most developing nations by 2020 [34]. What is a cardiovascular disease? We human bodies are composed of various organs, which ensure our homeostasis and viability. The cardiovascular system (also called the circulatory system) is an organ system that comprises the blood, heart and blood vessels. Pumped by the heart, the vessels permit blood to circulate and transport nutrients and oxygen to and waste materials away from all body tissues. Arteries move blood away from the heart and veins return it. Lymph vessels are part of a cleaning system that removes damaged cells from human bodies and protects the body from infections and cancer.

Any condition affects this system is regarded as a vascular disease. The diseases range from problems with people’s arteries, veins, and vessels that carry lymph to disorders that affect how blood flows. A disease can limit the flow of oxygen-rich blood to the human body organs and other parts of the human body, called atherosclerosis, which can lead to serious, even life-threatening problems, including heart attack, stroke, or even death. There are also many other types of vascular diseases, such as cerebrovascular disease, peripheral arterial disease.

Vascular diseases are caused due to a variety of reasons, and many of them can be prevented with healthy living habits, such as reducing tobacco use, having a healthy diet and engaging in physical activity [34]. Also, early detection and health management can help prevent and reduce acute problems, mainly heart attack and stroke. Therefore, in order to save more lives and improve the quality of life, early detection and health management are essential for people at high risk of vascular diseases. However, since a person may have various symptoms, it may be difficult to diagnose the vascular disease. Usually, a detailed analysis of blood flow in the heart and vessels is needed in order to acquire a thorough understanding of the underlying pathomechanisms, using data from tests such as Cardiovascular Magnetic Resonance Imaging (MRI).

MRI is a powerful technique which uses magnets and radio waves to form pictures of organs inside human bodies. MRI has undergone substantial develop-
ments over the last decades and offers capabilities of measuring the anatomy and functions of organs or tissues, such as various cardiac vessels. Phase contrast (PC) MRI can be used to measure and quantify pulsatile blood flow in the human vascular system. The recent development of medical imaging technologies brings a method called "4D flow MRI", which refers to a time-resolved three-dimensional (3D) velocity encoding in all three orthogonal spatial directions. 4D flow MRI, still a rather new data source for physicians, provides a large amount of data but limited valuable information due to lack of insight. Some parameters, such as flow velocity, wall shear stress (WSS), turbulent kinetic energy (TKE) and pressure gradient, calculated from 4D flow MRI data have been proven to correlate with some specific vascular diseases [35]. However, the number of indicative parameters is still quite limited. Furthermore, there’s not any golden standard that helps determine whether the information presented in the 4D flow MRI images is abnormal. The average blood flow and its variations in the healthy people and patient groups are still relatively unexplored.

1.2 Motivation

A medical atlas is commonly used to identify or model anatomical structures. The mean and variations of blood flow within healthy people can be shown and interpreted using the hemodynamic atlas. A hemodynamic atlas is the atlas with information of blood flow parameters that helps understand the pathophysiology of CVDs. There are already some works in creating hemodynamic atlas using 4D flow MRI, which could help the physicians identify the potential abnormal patterns of blood flow. The authors [21][22] provide a method by combining rigid and non-rigid registration to create automatic hemodynamic atlases using 4D MRI data with minimal user interaction. They demonstrated their methods for parameters including kinetic energy (KE), helicity density (Hd) and turbulent kinetic energy in the aortic flow. These methods are limited in that they only provide the mean and standard deviation of selected parameters (such as KE). They are good to show what is generally normal but still difficult in CVD diagnosis because little information is provided regarding the variations. Furthermore, these atlas are all built for scalar values, such as KE. We haven’t seen any atlas for the 3D blood flow vector fields.

Statistical analysis is also commonly used in creating the atlas, such as principal component analysis (PCA). PCA, which is commonly used in creating an average shape and variations in Statistical Shape Models, such as Active Shape Model (ASP), can also be applied in creating an atlas for hemodynamic parameters, such as flow velocity. PCA can potentially identify the inherent features of the dataset, and remove redundant information and noise by projecting the original high-dimensional variable space onto low-dimensional variable space. Eigenface is one typical application using PCA, allowing a few numbers (hundreds) of components (eigenvectors) to represent the original dataset with a large quantity of samples. These new components represent the face features, such as poses, illumination, expressions, etc.

A large number of studies on PCA focus on the real domain (\(\mathbb{R}\)), since many datasets are composed of scalar data, such as the KE parameter calculated from in the 4D flow MRI data, and gray-scale intensity of image. In case of vector field data (such as 2D vector field flow or 3D vector field flow), classical approaches propose to concatenate the components into a long vector before
processing, or handle components separately. The use of long-vectors may be restrictive if all possible ways to build this vector are not examined [4][21][22]. A consequence of component-wise processing is the loss of the relations between the components[22]. An alternative of traditional PCA has been proposed, complex PCA (CPCA) for 2D vector field data, or quaternion for 3D vector field data, which have achieved better results than traditional PCA in applications such as computer graphics and seismology. But these methods have not been tried in building a hemodynamic atlas.

1.3 Main Objective

The goal of this research is to build a 4D MRI blood-flow statistical model (atlas) which can show the mean and variations of healthy people. Since the dataset we have is composed of vector field data, we would like to explore different alternatives for vector field data. Specifically, we want to first build a basic atlas using some methods similar as statistical shape model (SSM) by concatenating the three components of a velocity vector, and then compare it with the one built using the quaternion method. We also want to develop visualizations that help physicians understand the cardiac diseases using the atlas built on the 4D MRI data.

1.4 Challenges

During the research, we have come across some challenges originated from the dataset, the methods and the tools, respectively. After exploring related literature, we have summarized the potential problems as below. And we will attempt to solve these issues one by one to get the final results.

First, our dataset is composed of 4D flow MRI images of five individuals. Since the anatomy of heart and vessels vary for different individuals, as well as the data acquisition procedure, we do not have a mapping relationship of the voxels among different volumes. Therefore the registration operation is required. Registration is the process of transforming images or volumes into a common reference so that the pixels or voxels in different images or volumes are aligned with each other. The registration process is time consuming and prone to misalignment, and may bring error as a result of inappropriate reference selection. It is not trivial to design the registration process and tune the parameters to get accurate aligned images or volumes.

Furthermore, our 4D flow MRI dataset includes vector field data which contains the blood flow velocity information. Different from the alignment of volumes composed of scalar data which does not change its value during the alignment process, vector data changes its value (namely direction) accordingly. Vector reorientation in rigid registration is simple, but the difficulty is how to reorient the data in case of non-rigid transformation.

Another challenge is the huge dimension of the dataset (usually in millions, and around 2.5 million for our dataset). In this case, PCA is computationally costly and uses much memory access. However, in contrary, the data size (the number of samples) is limited, which may be too small to capture the full amount of
variations for a specific population. This is because the number of variations (or called eigenvectors or components) cannot exceed the number of the samples minus one.

With regard to the PCA methods, it is intuitive and common to separate the 3D velocity vector fields into three components and apply PCA separately, or concatenate them into a long vector and then apply PCA. Both methods may lose the potential non-linear relationship among the components. An alternative method is to use complex PCA (CPCA) for 2D vector fields or quaternion PCA (QPCA) for 3D vector field. However, many softwares do not support CPCA nor QPCA, such as Python. Currently Python does not support CPCA nor QPCA. Alternative solutions are required.

Quaternion algebra is not as developed as complex and real algebra. To the best of my knowledge, currently there is not a direct method to compute QPCA, and the only method I find is to apply CPCA on the complex adjoint form of the original quaternion matrix.

Finally, a quaternion number is composed of one real part and three imaginary parts. Only the three imaginary parts will be used to represent a 3D vector field. It’s still not clear how to interpret the real part of the eigenvectors.

1.5 Principal Contributions

Previous research on building medical hemodynamic atals has focused on the anatomy and scalar data, but there is not any available hemodynamic atlas for vector data. To the best of our knowledge, this is the first research in creating a hemodynamic atlas for 3D blood-flow velocity vector data. Unlike other atlas-building approaches in medical domain, which are based on merely mean and standard deviation of scalar fields, we have applied methods similar as SSM for vector field data to create the mean and variations of the population. These variations could potentially represent the main patterns of the blood flow of the studied population.

Additionally, we have also explored quaternion method, a more compact representation of 3D vector field data compared to that in real domain. This is also the first research attempting to use QPCA in building a medical atlas. Besides, we have employed an efficient method to calculate the PCA of covariance matrix when the number of samples is far less than the dimensions in both the complex and quaternion domains.

Finally, we develop interactive visualization to display the changes of the blood-flow velocity by tuning the parameters and show them in animation.

1.6 Thesis Structure

This master thesis is divided into seven chapters starting from a brief introduction of the background, motivation, objective and challenges in building a 4D flow MRI atlas. The remainder of the thesis is organized as follows.

Chapter 2 states the related work in building the 4D flow MRI atlas, mainly
including 4D MRI, PCA and its typical application SSM, complex PCA, quaternion algebra and quaternion SVD (QSVD).

Chapter 3 introduces our dataset obtained from the real world, and the synthetic data we have built and the method to generate the synthetic tubes.

Chapter 4 presents an overview of our solution and describe each step (registration, masking, building a statistical vector field model and visualization) in details.

Chapter 5 introduces the implementation of our solution, including the softwares and packages used and how we use them in our research.

Chapter 6 first illustrates the performance of QPCA based on small simulated dataset and then presents the experiments and results using different methods based on the real-world dataset and synthetic dataset respectively.

Chapter 7 discusses several interesting aspects of our study and then concludes the thesis.

1.7 Specification

In the following sections, if there is no special specification:

- a normal lowercase or uppercase letter represents a scalar value, for example, $v_x$ denotes the velocity value in the x direction at a voxel;
- a bold lowercase letter represents a vector, for example, $\mathbf{v}$ denotes the velocity vector at a voxel, $\mathbf{x}$ may represent the 3D coordinates of a voxel;
- a bold uppercase letter represents a matrix, such as $\mathbf{V}_t$ denotes a velocity vector volume.
Chapter 2

Related Work

Our research is conducted using methods like SSM on a group of 4D flow MRI volumes of five individuals. The main method used is PCA, which is an unsupervised learning method in machine learning. PCA has been extended to run on more complicated dataset in the past few years, such as complex data and the quaternions. And both of them have made a difference in the fields of computer graphics. Since our dataset is composed of vector field data, the intuition is that perhaps a more compact representation that uses the quaternions to express the 3D vector field data may be better. Therefore, we have also studied the state of art of PCA algorithms in the complex and quaternion domain.

2.1 4D MRI Blood Flow Data

Magnetic resonance imaging (MRI) is a powerful medical technique used to generate images of organs and tissues inside the body. It is a non-invasive diagnostic system without harm to human bodies, because it does not make use of X-ray and radioactive materials which are used in CT. MRI is not only able to provide anatomy information but also capable of presenting the functions of different organs and tissues. Therefore, it has become a significant tool in diagnosing cardiovascular diseases in recent decades. In the past decades, 2D phase contrast (PC) methods is very popular in diagnosing diseases such as brain injury and cancer. And now a more advanced imaging technique, 4D flow MRI, has already been developed and applied in practice to evaluate cardiovascular diseases in many regions of body [33]. 4D flow MRI is a time-resolved 3D phase contrast MRI (time + 3D), i.e. the velocity of blood flow is encoded along all three spatial dimensions throughout the cardiac cycle. Any time step of 4D flow MRI volumes can be divided into three flow datasets representing three components of velocities $V_x$, $V_y$, and $V_z$. The three components of 4D flow MRI can be represented in the form of 3D-vector in 3D space. The dataset of velocity magnitude, defined as $||V|| = \sqrt{V_x^2 + V_y^2 + V_z^2}$, helps to depict the anatomy of vessels. A more frequently-used method to visualize the vessels is time maximum intensity projection (tMIP), which captures the maximum magnitude value of each voxel inside the volume from all time phases. Figure 2.1 is an example of 4D flow MRI data, including three velocity components, magnitude image and tMIP image.

4D flow MRI helps to understand the underlying pathomechanisms of cardio-
vascular diseases, to detect pathologies prior to clinical manifestation, and to provide a prognostic evaluation. 4D-flow measurement seems to be a promising tool that provides functional information in addition to the morphological visualization of vessels. It could cover larger areas and record entire flow information, therefore the desired flow parameters can be evaluated in every region of interest. 4D-flow data provides extensive information that can be used to determine a number of parameters for analyzing flow velocity, flow volume, flow characteristics, and even relative pressure conditions. Among these parameters, the simplest yet most significant one is flow velocity (or flow rate), which is visualized in Figure 2.1. Low flow rate may indicate distal ischemia of the tissue. And local acceleration of the flow velocity is highly potentially a sign of stenosis [35]. Other useful parameters, including wall shear stress (WSS), turbulent kinetic energy (TKE), vortex flows, pressure gradient and pulse wave velocity, have also been used to detect CVDs, such as atherosclerosis and vessel dilation [35].

2.2 Traditional PCA and Applications

Principal component analysis (PCA) is the primary method used in SSM, and will also be the base of our research in building a hemodynamic atlas. PCA is probably one of the oldest and best known techniques of statistical analysis. It is a simple but popular and useful linear transformation technique that is used in numerous applications, such as SSM, eigenface, and many more. Nowadays in the big data era, the size and the dimensions of the dataset in numerous fields are so big that it is not only a challenge for storage but also a bottleneck for calculation. This is especially obvious in image processing, with more advanced imaging technique developed to generate ultra-high definition images.
The primary goal of PCA is to explore the hidden information and identify the inherent patterns of the dataset, such as different poses or expressions presented in the eigenface. In PCA, information is variance. The basic idea of PCA is to use orthogonal projection of data onto a lower dimensional space, such that the variance of projected data is maximized [15]. The new set of variables, called the principal components (PC) in the new space, are uncorrelated. If there are \( n \) observations with \( p \) variables, then the number of valid principal components is \( \min(n - 1, p) \). A valid PC is corresponding to a positive eigenvalue. The first principal component captures the largest amount of variance presented in the dataset. Each subsequent component has the highest possible variance under the constraints orthogonal to the previous components.

Another important linear algebra concept is the singular value decomposition (SVD). PCA and SVD are closely related. SVD, regarded as a more general form of PCA, can also be used to find the principal components that we want in PCA. PCA requires the matrix to be symmetrical, square, positive and semi-definite, which are the properties of covariance matrix [25]. While, SVD can factorize any real matrix \( A \) as: \( A = U \Sigma V^T \), where \( U \) and \( V \) are orthogonal matrices with eigenvectors from \( AA^T \) and \( A^TA \), respectively [25]. Both algorithms have been already packaged in many programming languages, such as numpy for python.

### 2.2.1 PCA in Statistical Shape Models (SSM)

Statistical shape models (SSM) provides a reference to obtain the mean and variations of a dataset. Specifically, SSM applies PCA to compute the mean shape and variations from a bunch of hand-segmented shapes. It is similar to the methodology in our research to calculate the variations of the 4D flow MRI dataset.

The term shape is generally defined as: “Shape is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.” [14]. The geometrical information of an object refers to the outline of an object in 2D or 3D space [20]. A shape can be represented using an ordered set of points. For example, a 2D shape can be defined [20] as: \( C = (x_1, y_1, \ldots, x_r, y_r) \), where \( (x_i, y_i) \) denotes a point. The point \( (x_i, y_i) \) is connected by its adjacent points, i.e., \( (x_{i-1}, y_{i-1}) \) and \( (x_{i+1}, y_{i+1}) \). In SSM, it is required that all training shapes must be represented by the same number of points and that all points have to be located at corresponding positions along all training shapes [20]. Therefore, before applying PCA, rigid shape alignment is done, which is a combination of translation, scale and rotation. The original shape can be approximately reconstructed with the mean shape and its variations, the author [20] gives a vivid example of variations of the hand shapes, shown in Figure 2.2. The shapes in the middle column are the same shape, all of which represent the mean shape. And the shapes on the left side are generated using the top first to fifth principal components (PC). They are calculated by subtracting \( 3\sigma_i \times \) corresponding PC from the mean shape, where \( \sigma_i \) is the \( i \)th eigenvalue. Similarly, the shapes on the right side are also synthesized from the top five components, by adding \( 3\sigma_i \times \) corresponding PC to the mean shape. How to match the model to a new target is for an extended discussion of shape matching we refer to the paper [5] and reference therein.

One potential problem of statistical shape models is a limited number of train-
ing data, i.e., the subset of shapes used to build the mean and variations. This is possibly due to the tedious and time-consuming process of manually extracting the points from given images. The small training sample size is a problem because it is assumed that all plausible shapes can be described by linear combinations of the training shapes, which is too restrictive for a small sample size, because the dimension of the shape space cannot be greater than the training size minus one, and the actual dimension could be even smaller because the maximum dimension is only achieved for linearly independent training shapes. Therefore, for a small number of training samples, it will be not enough to capture all variations of a specific dataset. There are some approaches to enhance this situation by obtaining more flexible SSMs, which are divided in three categories, as introduced and listed in the paper [20], including artificial enlargement of shape variations, relaxation of model-constraints and model partitioning. More details about these methods can be further found in these papers [7] [6] [8] [40] [9], etc. In our implementation, however, we do not extend

<table>
<thead>
<tr>
<th>$w_i = -3\sigma_i$</th>
<th>$w_i = 0$</th>
<th>$w_i = +3\sigma_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Hand" /></td>
<td><img src="image2.png" alt="Hand" /></td>
<td><img src="image3.png" alt="Hand" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Hand" /></td>
<td><img src="image5.png" alt="Hand" /></td>
<td><img src="image6.png" alt="Hand" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Hand" /></td>
<td><img src="image8.png" alt="Hand" /></td>
<td><img src="image9.png" alt="Hand" /></td>
</tr>
<tr>
<td><img src="image10.png" alt="Hand" /></td>
<td><img src="image11.png" alt="Hand" /></td>
<td><img src="image12.png" alt="Hand" /></td>
</tr>
</tbody>
</table>

Figure 2.2: Variation of mean shape. Adopted from the paper [20].
the size of the dataset using the above approaches considering the following reasons:

- 1) Our primary target is to prove the effectiveness of our solution but not to find accurate patterns representing the given dataset.
- 2) The added samples may bring in additional error.
- 3) Due to the complexity and unknown patterns in our dataset, it is difficult to evaluate the performance by enlarging the dataset.

Instead, we create a set of synthetic tubes simulating the blood flows in the vessels such that we can control the patterns of the dataset, for example, whether a synthetic tube contains vortex or helix, and where the special patterns are inserted. And we can add as many samples as we need. The main benefit of using synthetic data is that it makes it easier to interpret the results with the original dataset in mind.

### 2.3 Complex PCA

Most of the time, data values are real numbers or split into many parts of real numbers, such as the RGB color image, where the value of each pixel is composed of three components, red, green and blue. However, the representation of the nature of numerical values of data has been limited. In such cases, as mentioned, the traditional PCA method by concatenating the three components may lose the notion that they form a spatial vector. Therefore it can not capture the potential non-linear relationship among the components. There are plenty of cases where complex numbers can represent the data more compactly and retain the relationships between the components. Complex-valued numbers can be divided into two groups: those that are complex by design, and those that are made complex by convenience of representation [26]. One example of the former case is the image feature produced by applying complex-valued kernels, and for the other case, data like points with x and y coordinates in the statistical shape models can be represented as complex numbers [26]. Complex-valued data can be potentially used in many applications, and how we can extend the existing algorithms in real domain to complex domain is worth exploring.

Although the complex number is not the right choice to represent 3D vector field data in our research, it can be regarded as a trial on 2D vector field data before we use the quaternions (which will be introduced in the next section) to represent 3D vector field data. As the complex-valued number is a more general form of real-valued number, and the quaternion-valued number is a more general form of complex-valued and real-valued numbers, we intuitively assume that they share some common properties. Unless the complex representation of 2D vector field data is better than that of real representation, no matter in terms of effectiveness or accuracy or others, we will not consider to generalize the representation of 3D vector field data using the quaternions.

The author [26] illustrates that many algorithms can be directly used on complex-valued data the same way or similarly as those on real-valued data. Usually we assume the complex-valued data are proper and circular. "A complex variable is defined to be proper if it is uncorrelated with its complex conjugate, and that a complex random variable is circular if it has a probability distribution that is invariant under rotation in the complex plane, i.e. $x$ and $x' = e^{j\theta}x$ have
the same pdf for any given real \( a \) \cite{26}. And complex data that are circular is inherently rotation invariant.

Complex PCA (CPCA) has been proven more efficient than real PCA (RPCA) in that less components can represent more variance \cite{26}. This can be seen through the eigenvalues that the accumulative percentage of the top \( r \) \((r < \) the dimension of the dataset) eigenvalues of CPCA is higher than that of RPCA. It can be seen more intuitively from the visualization of top main eigenvectors. For example, Figure 2.3 (adopted from the paper \cite{26}) illustrates this conclusion. In this figure, the mean shape is located in the middle. The shapes on the top and bottom are generated from the first component, and the images on the left and right are synthesized using the second component. Finally, the shapes on the two diagonals are constructed using the third and fourth eigenvectors. The shapes on the top and left side are produced with a positive weight and the ones on the bottom and the right side are generated with a negative weight.

In the real representation, the first component and the second one represent the left/right pose and up/down pose respectively, while these can be explained using only the first component in the complex representation.

![Figure 2.3: The mean shape and shape instances generated from the first four eigenvectors for the real, complex shape representations of human face. Adopted from the paper \cite{26}.](image)

2.4 Quaternion PCA

In the previous section, we have already obtained some evidence that CPCA outperforms RPCA, and we want to further explore the quaternion representation of 3D vector field data. Quaternion algebra has been studied a lot in the past decades, the author \cite{38} introduces a brief history of the origin of the quaternions, and explains the basic algebra step by step with a lot of examples and practices, and finally combining it with computer graphics. The authors \cite{43} \cite{13} \cite{31} further explore the algebra of matrices of quaternions, which provides important contributions to our research. To the best of our knowledge, quaternion PCA (QPCA) or quaternion SVD (QSVD) is still not as widely used as RPCA or CPCA in a variety of applications. The existing related areas using QPCA or QSVD include color images processing \cite{23}, color texture segmentation \cite{30}, color image compression \cite{24}, 3D point cloud matching \cite{42}, polarized wave and vector-sensor signal processing \cite{22}, to name a few.
The authors [22] [23] show that QSVD is also invariant to spatial rotation and color space rotation, and has achieved better results than component-wise RPCA [22]. Besides the property of spatial rotation invariance, we also want to explore whether QPCA to quaternion-valued data is like CPCA to complex-valued data and outperforms RPCA for 3D vector field data with less components to represent more variance. These motivate us to explore the quaternion method in building an hemodynamic atlas based on 4D flow MRI data.

### 2.4.1 Quaternions Algebra

Quaternions are an extension of complex numbers from 2D plane to 3D space (as well as 4D space). Quaternions were invented in 1843 by W.R.Hamilton in 1843 [12] and are a specific class of hypercomplex numbers [16]. Quaternion algebra is the first non-commutative algebra to be discovered. A quaternion is made of four parts, one real part and three imaginary parts:

\[
q = s + ai + bj + ck, s, a, b, c \in \mathbb{R}
\]  

(2.1)

and

\[
\begin{align*}
i^2 &= j^2 = k^2 = ijk = -1 \\
i j &= k, \quad j k = i, \quad k i = j \\
j i &= -k, \quad k j = -i, \quad i k = -j
\end{align*}
\]  

(2.2)

A pure quaternion is a quaternion with a null real part \(s=0\), and quaternions with non-null real parts are sometimes called full quaternions. The conjugate \(\bar{q}\) of a quaternion \(q\) is

\[
\bar{q} = s - ai - bj - ck.
\]  

(2.3)

The norm of a quaternion is

\[
|q| = \sqrt{q \bar{q}} = \sqrt{\bar{q}q} = \sqrt{s^2 + a^2 + b^2 + c^2}.
\]  

(2.4)

The inverse of a quaternion is

\[
q^{-1} = \frac{\bar{q}}{|q|^2}.
\]  

(2.5)

The multiplication of quaternions is not commutative due to the relationships between the three imaginary parts. Given two quaternions \(q_1\) and \(q_2\), we have

\[
q_1q_2 \neq q_2q_1.
\]  

(2.6)

A vector-signal made of three orthogonal components \(\vec{q} = (q_x, q_y, q_z)\) can be represented using quaternions [22] by encoding the three components of a vector as a pure quaternion such that vector function is given by, \(q = q_xi + q_yj + q_zk\). In order to simplify the notation and explain the rotations in 3D space it is easier to consider the polar form of a quaternion. Euler’s formula for the complex
exponential generalizes to the hypercomplex form: \( q = |q| e^{\xi \theta} \), where \( \xi \) is a pure unit quaternion, called eigenaxis and \( \theta \) is the angle, called eigenangle.

\[
q = |q| e^{\xi \theta} \\
= |q| (\cos \theta + \xi \sin \theta) \\
= |q| (\cos \theta + \sin \theta \cos \phi \sin \psi i + \sin \theta \sin \phi \sin \psi j + \sin \theta k)
\]

where \( \phi \) and \( \psi \) are respectively the azimuth and elevation angles that define the position of \( \xi \) in 3D space. The polar form will be used to explain the quaternion representation of the velocity vector, and to show the availability of using gram matrix (which will be introduced in section 4.4.2) to efficiently calculate QPCA in case that the number of samples is far less than that of dimensions.

### 2.4.2 Quaternions Matrix

The study of quaternion matrices has been an active field of research for several years, which was started from 1936 [41], and is still in development. A quaternion matrix \( X \in \mathbb{H}^{N \times M} \) is a matrix, where \( N \) is the number of samples and \( M \) is the dimension of the dataset. The elements of \( X \) are quaternions, with each row \( x \in \mathbb{H}^{M} \) as a row vector space of dimension \( M \). (Note: We use row vector through the whole paper, which is different from the usual norm using a column representing a vector in a matrix.)

The following is lists some of the main properties of two quaternion matrices \( A \in \mathbb{H}^{N \times M} \) and \( B \in \mathbb{H}^{M \times P} \). Quaternion multiplication is non-communative, and this is also true for quaternion matrices. However, many equality relations for real and complex matrices are also valid for quaternion matrices.

1. \( AB^H = B^H A^H \),
2. \( AB \neq A B \) in general,
3. \( (A^H)^{-1} = (A^{-1})^H \),
4. \( (A^{-1}) \neq A^{-1} \) in general.

A quaternion vector \( x \in \mathbb{H}^{M} \) can be expressed with the Cayley-Dickson notation:

\[
x = x_1 + x_2 j
\]

where \( x_1 \) and \( x_2 \in \mathbb{C}^{M} \). A bijection can be defined \( f : \mathbb{H}^{M} \to \mathbb{C}^{2M} \)

\[
f(x) = \begin{bmatrix} x_1 \\ -x_2 \end{bmatrix}
\]

(2.9)

This definition is used during the process of quaternion eigenvector construction from its complex adjoint eigenvectors, which will be introduced in the following section.

Similarly, a quaternion matrix \( A \in \mathbb{H}^{N \times M} \) can be expressed using Cayley-Dickson notation:

\[
A = A_1 + A_2 j
\]

(2.10)
where \(A_1\) and \(A_2\) are complex matrices \((\in \mathbb{C}^{N\times M})\). The complex adjoint matrix, noted as \(\chi_A \in \mathbb{C}^{2N\times 2M}\), corresponding to the quaternion matrix \(A\) is like:

\[
\chi_A = \begin{pmatrix}
A_1 & A_2 \\
-A_2^T & A_1^T
\end{pmatrix}.
\] (2.11)

This complex notation for quaternion matrices can be used to compute quaternion matrix eigenvalue decomposition using complex decomposition algorithm. The conjugate transpose or Hermitian transpose of a \(N\)-by-\(M\) complex matrix \(A_1 \in \mathbb{C}^{N\times M}\) (or a quaternion matrix \(A \in \mathbb{H}^{N\times M}\)) is a \(M\)-by-\(N\) matrix \(A_1^H (A^H)\), therefore, the covariance matrix of the complex adjoint matrix \(\chi_A\) is

\[
\Sigma_{\chi_A} = \chi_A \chi_A^H = \begin{pmatrix}
A_1 & A_2 \\
-A_2^T & A_1^T
\end{pmatrix} \begin{pmatrix}
A_1^H & -A_2^T \\
A_2^H & A_1^T
\end{pmatrix}.
\] (2.12)

The following is a summary of the main properties for the given quaternion matrices \(A\) and \(B \in \mathbb{H}^{N\times M}\) and their complex adjoint matrices \(\chi_A\) and \(\chi_B \in \mathbb{C}^{2N\times 2M}\) [22]:

1. \(\chi_A\) is normal, hermitian or unitary if and only if \(A\) is normal, hermitian or unitary,
2. \(\chi_{AB} = \chi_B \chi_A\),
3. \(\chi_{A+B} = \chi_A + \chi_B\),
4. \(\chi_{A^{-1}} = (\chi_A)^{-1}\) if \(A^{-1}\) exists,
5. \(\chi_{A^H} = (\chi_A)^H\).

More properties of \(\chi_A\) are given in the papers [41][43].

### 2.4.3 SVD of a Quaternion Matrix (QSVD)

In linear algebra, the singular-value decomposition (SVD) is a factorization of a real or complex matrix. It can also be generalized to a quaternion matrix \(A \in \mathbb{H}^{N\times M}\) [43]. Any quaternion matrix can be decomposed as [43]:

\[
A = U \begin{pmatrix}
\Sigma_r \\
0
\end{pmatrix} V^H
\] (2.13)

where \(U \in \mathbb{H}^{N\times N}\) and \(V \in \mathbb{H}^{M\times M}\) are unitary quaternion matrices, which contain the left and right singular vectors of \(A\). \(\Sigma_r\) is a real diagonal matrix and has \(r\) non-null entries on its diagonal (singular values of \(A\)). The author [43] has demonstrated the existence of QSVD.

In SVD, the term rank of a matrix means "the dimension of the vector space generated (or spanned) by its columns" [2], which is equal to the number of linearly independent columns or the number of linearly independent rows (they are always the same). Simply put in SVD, it is equal to the number of non-null eigenvalues. Here, if the rank of a quaternion matrix \(A\) is \(r\), then the rank of its complex adjoint \(\chi_A\) is \(2r\) [41].

21
The QSVD can also be written as below [22]:

\[
A = \sum_{n=1}^{r} u_n v_n^H \sigma_n,
\]

(2.14)

where \(u_n\) and \(v_n\) are respectively the left singular vectors (columns of \(U\)) and the right singular vectors (columns of \(V\)), and \(\sigma_r\) is the real (\(\mathbb{R}\)) singular value.

A method to compute SVDQ of a quaternion matrix \(A \in \mathbb{H}^{N \times M}\) from its complex adjoint matrix \(\chi_A \in \mathbb{C}^{2N \times 2M}\) is given in the paper [22]. Here is a brief summary of those steps introduced in the paper [22].

1 First, obtain the SVD of its complex adjoint matrix \(\chi_A\):

\[
\chi_A = \hat{U} \begin{pmatrix} \Sigma_{2r} & 0 \\ 0 & 0 \end{pmatrix} \hat{V}^H = \sum_{\hat{n}=1}^{2r} \hat{u}_{\hat{n}} \hat{v}_{\hat{n}}^H \sigma_{\hat{n}}
\]

(2.15)

where the columns \(\hat{u}_{\hat{n}}\) and \(\hat{v}_{\hat{n}}\) are the columns of \(\hat{U}\) and \(\hat{V}\), and they are respectively elements of \(\mathbb{C}^{2N}\) and \(\mathbb{C}^{2M}\), and can be expressed as below:

\[
\hat{u}_{\hat{n}} = \begin{bmatrix} \dot{\hat{u}}_{\hat{n}} - \ddot{\hat{u}}_{\hat{n}} \\ \hat{u}_{\hat{n}} \end{bmatrix} \quad \text{and} \quad \hat{v}_{\hat{n}} = \begin{bmatrix} \dot{\hat{v}}_{\hat{n}} - \ddot{\hat{v}}_{\hat{n}} \\ -\hat{v}_{\hat{n}} \end{bmatrix}
\]

(2.16)

where \(\hat{u}_{\hat{n}} \in \mathbb{C}^N, \hat{v}_{\hat{n}} \in \mathbb{C}^N, \dot{\hat{v}}_{\hat{n}} \in \mathbb{C}^M\) and \(\ddot{\hat{v}}_{\hat{n}} \in \mathbb{C}^M\).

The diagonal matrix \(\Sigma_{2r}\) has the following special structure:

\[
\Sigma_{2r} = \text{diag}(\sigma_1; \sigma_1; \sigma_2; \sigma_2; \ldots; \sigma_r; \sigma_r)
\]

(2.17)

where \(\sigma_i \in \mathbb{R}\). This repeated structure in the diagonal entries is due to the special form of \(\chi_A\).

2 Associate the \(n_{th}\) right (resp. left) singular vector of \(A\) with \(\hat{n}_{th} = (2n - 1)_{th}\) right (resp. left) singular vector of \(\chi_A\), using the following equations:

\[
u_n = \hat{u}_{\hat{n}} + \hat{u}_{\hat{n}}j, \\
u_n = \hat{v}_{\hat{n}} + \hat{v}_{\hat{n}}j,
\]

(2.18)

3 Obtain the \(n_{th}\) singular value of the diagonal matrix \(\Sigma_r\), with \(\hat{n}_{th} = (2n - 1)_{th}\) diagonal element of the diagonal matrix \(\Sigma_{2r}\).

In this way, we obtain the mean and eigenvectors of a quaternion defined space.
Chapter 3

Dataset

The atlases in our research are built on both the real thoracic vessels from 4D flow MRI and a group of synthetic tubes, respectively. These datasets are introduced in the subsequent sections.

3.1 Real Vessels

Our real-world dataset consists of a cohort of five healthy people, with 19~24 time phases of 3D blood flow velocity vector per individual, and a tMIP per individual. Each time phase is a volume composed of 3D velocity vector field data \((v_x, v_y, v_z)\) in 3D space. Possibly due to different acquisition processes or different poses of these individuals, neither the number of time phases nor the number of voxels is the same among these five individuals. Figure 3.1 visualizes the 4\(^{th}\) time step of the volumes of these 4D MRI volumes, which corresponds to the systole phase of a cardiac cycle. These figures are generated using direct volume rendering technique and streamlines by putting a sphere source (center (-24.1,-51.3,18.5), radius 13.7) in the ascending aorta. The key parameters of this dataset are listed in appendix 4.

3.2 Synthetic Curved Tubes

3.2.1 Overview

The main reason to create the synthetic dataset is to be able to evaluate the behaviour of our algorithm with the knowledge of what to expect. The main variations of the real-world dataset are unkown and it is difficult to interpret without enough medical domain knowledge due to the complexity of the real-world dataset. But we can easily identify the variations of the synthetic dataset due to its controllable characteristics and relatively simplicity. Furthermore, the size of real-world dataset we have is small (only 5 samples), but the dimension per volume is large, which is difficult to manipulate. Due to its large dimension, the computational cost is so high that it usually takes more than hours to construct the original 3D vector into quaternion numbers.

The synthetic dataset is composed of a group of 24 curved tubes simulating the blood flow in the vessels. To create more variations, we add some perturbations,
such as vortex and/or helix, randomly located inside the tubes. Each synthetic volume has $32 \times 32 \times 32$ voxels, with origin at (0,0,0) and unit space between voxels. Some examples are shown in Figure 3.2.

### 3.2.2 The Method of Generating Curved Tubes

In order to synthesize the curved tubes, we first generate a polynomial curve and then generate a tube by finding the distances to the curve within a predefined threshold. Finally we insert vortices and helices randomly into the tube and compose a group of 24 different tubes. More details can be found as below:
Step 1: A parametric polynomial curve is generated. Here, we construct a BÉZIER curve from a set of control points \( P_i \in \mathbb{R}, 0 \leq i \leq n \), by

\[
X(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1 - t)^{n-1} P_i
\]  

(3.1)

where \( t \in [0, 1] \). Here we use four control points in 3D space. The generated curve is shown in Figure 3.3.

![BÉZIER curve](image)

Figure 3.3: BÉZIER curve

Step 2: We compute the distance from any point defined in the 32 × 32 × 32 space. We use the method of point to polynomial curve given in Chapter 10.6 in the book [29]. Namely, given a parametric curve \( Q(t) \) and a point \( P \), we can find the point on \( Q \) closest to \( P \). In another word, we would like to find the corresponding parameter \( t \). This leads us to solve an equation:

\[
(Q(t) - P)Q'(t) = 0
\]

(3.2)

where \( Q'(t) \) is the first derivative of \( Q(t) \).

The procedure to find the parameter \( t \) is briefly introduced as below. More details about this method can be found in the book [29].

1 Get a reasonable initial guess \( t_0 \) by evaluating 100 equally spaced parameter values with the range \([0,1]\), and the parameter value of the closest point is used as \( t_0 \).

2 Check whether the point coincidence is small enough:

\[
\|Q(t) - P\| \leq \epsilon_1
\]

(3.3)

where \( \|Q(t) - P\| \) is the distance between \( Q(t) \) and \( P \), and \( \epsilon_1 \) is a small enough number (such as \( 10^{-4} \)).

3 Check whether the angle between \( \|Q(t) - P\| \) and \( Q'(t) \) is close to 90°:

\[
\frac{\|Q(t) - PQ'(t)\|}{\|Q(t) - P\|\|Q'(t)\|} \leq \epsilon_2
\]

(3.4)

where \( \epsilon_2 \) is another small enough number (such as \( 10^{-4} \))
4 If either of the above two criteria are not met, then a Newton step is taken as below:

\[
t_{i+1} = t_i - \frac{(Q(t_i) - P)Q'(t_i)}{((Q(t_i) - P)Q'(t_i))^2} + \frac{(Q(t_i) - P)Q''(t_i)}{(Q(t_i) - P)Q''(t_i) + \|Q'(t_i)\|^2}
\]  

(3.5)

where \(Q''(t_i)\) is the second derivative of \(Q(t)\).

5 Then, check whether the parameter stays within the range \([0,1]\) by clamping it.

6 Check whether the parameter doesn’t change significantly:

\[
\|(t_{i+1} - t_i)Q'(t_i)\| \leq \epsilon_1
\]  

(3.6)

7 If the last criterion is not satisfied, repeat steps 2~6. If it is satisfied, the Newton iteration is stopped, and the closest point to \(P\) is \(Q(t_{i+1})\), and the distance \(d\) from \(P\) to the curve \(Q\) is \(\|Q(t_{i+1}) - P\|\).

Step 3: Create 3 basic volumes: stream flow (S), vortex (V), and helix (H). If the distance is smaller than a predefined value (here we define this value as the radius \(r\) of tube, which is equal to 3.0), the point will be given some value related to the corresponding point \(P\) on the curve, shown as below, according to the type of volume. Then we get the volume of stream flow, vortex, and helix by giving different values (shown as below) to the points.

- In stream flow (S): \((r - d)Q'(t_{i+1})\)
- In vortex (V): \(10 \times d \times (Q(t_{i+1}) - P) \times Q'(t_{i+1})\)
- In helix (H): calculated by \(S/2 + 2V\)

The created volumes of stream flow, vortex, and helix are shown in Figure 3.4.

Figure 3.4: Created stream flow, vortex and helix

Step 4: Randomly add vortex and helix patterns to the pure stream flow volumes, and generate 24 synthetic volumes.
Chapter 4

4D Flow Atlas Generation

4.1 Overview

Our primary goal is to generate a 4D MRI blood-flow atlas which can show variations of the studied population. The dataset we have is a group of 4D MRI images composed of thoracic vessels of five healthy individuals. Each volume of the dataset is composed of 3D vector field data, and the dimensions of these volumes are not identical between any two individuals. We have designed a method composed of a series of steps to get the main variations of this dataset and visualize them in a similar style shown in Figure 2.2.

Due to the difference in the number of voxels among different volumes, and the fact that we do not have a mapping relationship among them, the registration process is not avoidable. The process of registration is to align all the volumes. Different from scalar values in the image, vectors in the volumes should also change its direction accordingly. Just imagine the vertical blood flows through an upright vessel in your body when you stand up, they will become horizontal when you lie flat. Since we are only interested in the vessel region where the blood flows through, we will extract this region from the original volume, and then apply PCA on this truncated dataset. PCA will be first implemented on real values, by concatenating the 3D vectors into a long 1D vector. Complex PCA (CPCA) is then applied to some random generated datasets. We will then explore Quaternion PCA (QPCA), to compare the differences of the results among different approaches and try to find the inherent patterns, respectively. Finally, proper visualization is utilized that helps us understand the data and interpret the variations.

In sum, our solution including four steps shown in Figure 4.1, is briefly summarized as below:

1. Registration & Reorientation: This is to get all volumes aligned with each other.

   Register the tMIP images of all individuals and get the average tMIP image, the transformation matrix $T$ and its Jacobian matrix using population-based method. Then reorient the velocity vector volumes in the dataset accordingly.

2. Masking: This is to find the vessel region and then reduce the number of voxel per volume.
Create the mask by setting a threshold on the average tMIP image. Then APPLY the mask to the reoriented velocity vector volumes to get clearer vessel regions, and therefore reduce the cost of subsequent calculation.

3. Statistical Vector Field Model (SVFM): This is the core of our research, to build a statistical vector field model.

First select three typical time phases per individual in the 4D MRI dataset and form three datasets separately. Then explore different models including RPCA, CPCA and QPCA.

4. Visualization: This helps us interpret the information contained inside each variation.

Visualize the mean and variations using streamlines or glyphs, and dynamically show them by adjusting the parameters, similar to that shown in Figure 2.2.

### 4.2 Registration of Vector Field Data Volumes

Since we have a cohort of data volumes with different sizes and we do not know their mapping relation, first we need to get them aligned with each other. The main operation used is called image registration. Image registration is the process of transforming images into a common reference so corresponding pixels represent homologous biological points [17]. When the values of voxels of the volumes are scalar, there are multiple methods to get the aligned result after registration. However, our dataset is composed of velocity vector data, a further step is needed, which needs some special attention. In order to align all the data volumes composed of vector-valued blood flow, the following steps are done:

1. **Temporal Maximum Intensity Projection (tMIP)**

Temporal Maximum Intensity Projection (tMIP) captures the maximum magnitude value of each voxel inside the volume from all time phases. In order to segment the vessel lumens from the rest of body, we use the tMIP images of the 4D flow MRI dataset to roughly represent the vessels and use them to calculate the transform matrix.
2 Image Registration
In this step, we aim to align the tMIPs and get the transform matrix and Jacobian matrix, and then align velocity vector field data volumes. There are many methods available. It can be categorized by the number of images used to create the template, i.e., subject-based and population-based registration. It can also be divided into rigid and non-rigid registration, depending on whether the transformation is global or local. After the template is obtained, register all images to the template to get the corresponding transformation matrix $T_i, i \in 1...N$, and the Jacobian field of $T_i$, where $N$ is the number of blood flow volumes.

3 Velocity Reorientation
This is to correct the velocity vector field data volumes by reorienting the velocity vectors according to the Jacobian matrix calculated in the previous step.

In the next section, we will explain each of the modules in more details.

4.2.1 Temporal Maximum Intensity Projection (tMIP)

In medical visualization, a maximum intensity projection (MIP) is a compositing method for 3D data, which projects the voxels with the highest intensity value along the rays on the visualization plane [28]. An variant of MIP, temporal Maximum Intensity Projection (tMIP), searches the voxels with the highest intensity (or magnitude) values from all time phases. It is a typical method to segment and visualize the cardiovascular structure.

Quantitative analysis of vascular blood flow, acquired by phase-contrast MRI, requires accurate segmentation of the vessel lumen, which will enable more accurate quantification of velocity vector, as well as improved quality of visual representations. The author [3] employs the local coherence of the velocities, using the local phase coherence (LPC) and thresholding on the histogram of coherence values, to segment the vasculature, based on phase-contrast MRI blood-flow data. This method clearly indicates the boundaries of the vessel lumen, but the thresholding approach is noise prone and relies on various assumptions concerning the histogram distributions. Another paper [37] uses an active surface model and also introduces an approach to generate an approximate initial surface based on tMIP, to minimize the impact of numerical instability and potential to converge to local minima on the active surface model. This approach is validated and proven to be computationally efficient with the good approximate initial surface.

In our research, we will simplify the process by approximating the vessel lumen using tMIP. As the magnitude of velocity $||v|| = \sqrt{v_x^2 + v_y^2 + v_z^2}$ decreases towards the boundary of the vessel lumen and keeps decreasing (although discontinuously [32] due to the flow inside moving parallel to the boundary and the wall moving roughly normal to the boundary) to zero towards the static tissues, the tMIP volume can be directly applied as a rough representation of vessel regions. This is the starting point of subsequent registration.
4.2.2 Image Registration

Image registration is the process of transforming images into a common reference so corresponding pixels represent homologous biological points [17]. It is an important tool and commonly used in clinical applications, including segmentation of anatomical structures, computer-aided diagnosis, monitoring of disease progression, surgical intervention and treatment planning. Usually, it is difficult to detect the difference by comparison of separate images, in which case registration can be used to find the spatial relationship among them. Registration can also be used to create an anatomically normalized template to compare the images from different patients or obtained from different measurements, such as X-Ray, Computed Tomography(CT) and Magnetic Resonance Imaging(MRI).

Registration Framework
In the registration framework, two images are involved, one fixed image \( I_F(x) \) as the reference to which one moving image \( I_M(x) \) is deformed. \( x \) is the spatial coordinate of images, and \( I_F(x) \) and \( I_M(x) \) usually represent the intensity of pixels (or voxels) at location \( x \) (in our work, we will use the magnitude of velocity instead). After registration, the deformed moving image is expected to be spatially aligned to the reference, namely \( I_M(x + \mu(x)) = I_F(x) \), where \( \mu(x) \) is the spatial displacement of the moving image. The target of registration is to find the transformation \( T_\mu(x) = x + \mu(x) \). The quality of alignment is defined by a distance or similarity measure, such as the sum of squared differences (SSD), the correlation ratio, or the mutual information (MI) measure. For non-rigid transformations \( T_\mu(x) \), a regularization or penalty term is often introduced that constrains \( T_\mu(x) \).

Transformation model
The transformation model used for \( T_\mu \) determines the type of deformations between the fixed and moving image. In order of increasing flexibility, these are translation, rigid, similarity, affine, nonrigid B-spline and nonrigid thin-plate spline like transformations. A rigid transformation is defined as: \( T_\mu(x) = R_x + t \). A rigid transformation, also called Euler Transformation, including rotation \( R \) and translation \( t \) or their combinations, is a geometric transformation of a Euclidean space that preserves the Euclidean distance between every pair of points. Similarity transformation includes a combination of translation \( t \), rotation \( R \) and isotropic scales. In 3D space, \( t \) is a 3-element vector, \( R \) is a 3 × 3 matrix in 3D images, with three unknowns. Affine transformation is a more general form of transformation, defined as: \( T_\mu(x) = Ax + t \), where the matrix \( A \) is a 3 × 3 matrix in 3D images with no restrictions. This means that the image can be translated, rotated, scaled and sheared. These transformations are globally applied to all voxels of the whole 3D image.

Compared to the above transformations, the category of non-rigid transformations can be applied locally to some regions of the image to further achieve the correspondence between images. In real world, non-rigid transformations are necessary and commonly used due to the non-uniform variations among individuals. This will bring difficulty in the subsequent reorientation process of the blood flow velocity.

Template Construction
We need to generate a template for all the other tMIP volumes in order to get all the tMIP volumes aligned with each other. Generally, a template is an
average image, as the reference for other images to align to, can be used in cases such as segmentation where the atlas is deformed to the target image, as well as the segmentation map. The simplest way to construct a template is to select an individual in the image dataset as the template, but it is strongly biased to the selected individual. Thereafter, two categories of template construction are used. One is subject-based, which is based on single individual with minimal deformation, and the other is population-averaged with minimal deformation target (MDT) to counteract the bias brought by the single MDT template. A summary of these schemes is given in the paper [39], which experiments and validates the performance of these schemes. The population-based atlas is better regarding the the overlap of eigenvalue-eigenvector pairs between tensors (OVL) and FA accuracy [36], and global similarity index (GSI) obtained by averaging region similarity index (RSI) across all regions [39]. Therefore, a population-based atlas is a better representation of the given dataset, which is also the method we have used in our research. In practice, a population-based template is obtained by first constructing many subject-based templates and then averaging over all subject-based templates.

Assume we have N images, \( I_1, ..., I_N \), in the database. Let \( T_{ij} \) denote the deformation field that results from registration of source image \( I_i \) to target image \( I_j \). The deformed source image \( I_i \) after warping to the space of \( I_j \) is represented as \( \tilde{I}_{ij} = T_{ij}(I_i) \).

- **subject-based**
  First, we will introduce the basic subject-based method, that we use in our implementation. In subject-based template, an image \( I_i \) is selected as the initial reference. The reference image is warped to all the other images in the dataset to obtain the transformation matrix \( T_{ij} \). The minimal deformation target (MDT) template is derived from image \( I_i \) and defined as \( MDT_i = \overline{T}_i(I_i) \), with \( \overline{T}_i = \frac{1}{N} \sum_{j=1}^{N} T_{ij} \). The MDT template gets its name because it requires the least amount of deformation to all images in the dataset [18]. Since the registration algorithm preserves the topology, each \( MDT_i \) will be inevitably biased to the topology of the corresponding \( I_i \).

  To reduce the influence of the initial reference image, you can select the image that leads to the optimal MDT template [18]. A more accurate method is to iteratively refine the MDT template by warping all images again to the MDT template created in the previous step [11]. Another method, called intensity-based MDT, states that besides deforming an individual image \( I_i \) by the mean transformation \( \overline{T}_i \) to obtain \( MDT_i \), the new template \( \overline{T}_i(I_i) \) is obtained by transforming the intensity-average template \( \overline{I}_i = \frac{1}{N} \sum_{j=1}^{N} \tilde{I}_{ji} \) to its MDT template [39]. Although the bias is reduced, some bias towards \( I_i \) from which it is constructed can not be excluded.

- **population-based**
  In our work, we use the population-based method to build the average tMIP. The basic idea of the the population-based method is that all the \( MDT_i \) attained using the subject-based method described above are averaged, \( \frac{1}{N} \sum_{j=1}^{N} MDT_i \), in order to reduce the bias. This process is visually illustrated in Figure 4.2, modified based on the paper [36].
the group.

- In step B, a $MDT_i$ is created for each image.

- In step C, all images $I_i$ are averaged resulting in the final template. Theoretically, we cannot directly average all $MDT_i$ templates, since they are in different shapes. We assume that all the individuals are representative of the same population under study, and the image quality of all images is similar; therefore, all $MDT_i$ templates are very close to each other, except for some unresolved residual topological variations [39]. To this end, a population-based template has been constructed.

In step B, we have obtained the transformation matrix $T_i$ for each tMIP volumme, with $T_i = \frac{1}{N} \sum_{j=1}^{N} T_{ij}$. In addition, transformix in Elastix is also capable to compute the spatial Jacobian of the transformation. The determinant of the spatial Jacobian of the transformation matrix $T_i$ identifies the amount of local compression or expansion. Values smaller than 1 indicate local compression, while values larger than 1 represent local expansion [17]. We will use both the transformation matrix and the Jacobian matrix in the subsequent reorientation process.

![Figure 4.2: Population-based atlas construction](image)

### 4.2.3 Velocity Reorientation

For scalar-valued images, the value of the voxels will keep unchanged after registration process. Except when the voxel in the new image maps to a location be-
between voxels in the original image, interpolation is needed and the values will be calculated accordingly. Unlike scalar-valued data registration, handling vector-valued data, such as velocity vector field of blood flow, should also consider its orientation. This effect must be accounted for in order to ensure the anatomical correctness and preserve the integrity of the transformed image. This is quite intuitive to understand, for example, the velocity in a vertical narrow vessel lumen is directed up and down when the person stands up, which is expected to be horizontal and the magnitude remains the same when the person lies on a flat plane. The overall process of velocity reorientation is described in Figure 4.3. First, the velocity vector volume should also be transformed the same way as its corresponding tMIP, by moving the vector values from the points in the original velocity volume (denoted as $V_i$) to the corresponding points in the tMIP template. The mapping from original points to the corresponding points in the tMIP template can be obtained from the transformation matrix $T_i$, which has been already calculated from the population-based method in the previous section. And then the vectors in the transformed $V_i$ are reoriented by doing the dot product with the Jacobian field of $T_i$.

![Figure 4.3: Velocity reorientation flow](image)

The complexity of reorientation is in accordance with the order of transformation complexity. The problem is straightforward during rigid transformations, including translation and rotation. The blood flow velocity volume is composed of 3D vectors, noted as $v = (v_x, v_y, v_z)$, which should be rotated similarly as the tMIP. This can be achieved by applying the same rigid rotation matrix to each velocity vector in the image via a rigid transform. Thus, if $R$ is the rotation matrix representing the image transformation, after rotation, $v$ is replaced by $v', v' = Rv$. This reorientation does not affect the magnitude of the velocity, and only orientation is changed.

When the transformation applied to the image is rigid, it is straightforward to determine the required reorientation of the velocity. In general, we need to extend the method to cope with higher order transformations.

We assume that the population under study have similar vessels and blood flows, therefore the non-rigid transformation can be regarded as not significant. Therefore, we can reorient the blood flow velocity vector volumes by element-wise dot product of the aligned vector volumes and the full spatial Jacobian field of the transformation matrix $T_i$ calculated from the registration process of tMIPs.
4.3 Masking

After the registration and reorientation step, we get an aligned dataset with the same dimensions per volume (128 × 128 × 52). The resolution is so high that it brings two challenges. First, the larger the resolution, the higher the computation cost will be. There are two steps among the whole process that may consume massive computation resources. The first one is during the process of converting real-valued vectors to quaternion representation. The other is when calculating PCA using the covariance matrix, since the row size and column size of the covariance matrix will be the same and large (both are 128 × 128 × 52). It will reach the upper limit of the system memory, no matter for RPCA or QPCA. Although we have found a solution to calculate PCA efficiently by using the gram matrix (explained in section 4.4.2), the vector-to-quaternion process still remains a challenge. Second, since we are only interested in the vessel regions, other parts are are not significant. It is better to get a cleaner vessel region, which will also enable more accurate PCA calculation and interpretation.

Given the fact that the magnitude of velocity keeps decreasing from the center of the vessels towards the boundary of vessel lumen, and gets even smaller outside vessel region, we will apply a threshold to the average tMIP template. This is to remove the voxels which are lower than the threshold. Since all the voxels have been aligned in the first step, we can use the truncated dataset to do PCA calculation, and later recover the results (mean volume, and eigenvectors) to the original volume size for visualization.

4.4 Building a Statistical Vector Field Model (SVFM)

4.4.1 Representation of a Blood Flow Velocity Vector

4D flow MRI is a time-resolved PC-MRI technique with velocity encoding along all three flow directions. You can imagine any point (x, y, z) in your vessels within your body, where the blood keeps constantly flowing through. The velocity vxyz of the blood flow at this point can be expressed as a vector (vx, vy, vz). If vxyz does not change with time at this point, the blood flow is steady, or called stationary flow. On the contrary, it is called unsteady flow.

A 4D MRI blood flow velocity volume (defined as \( \vec{v}(\vec{x}, t) : (\mathbb{R}^3, \mathbb{R}) \rightarrow \mathbb{R}^3 \)) is composed of T (time steps) 3D volumes, with each element a velocity vector vxyz = (vx, vy, vz). Each 4D volume can be divided into T 3D sub-volumes (defined as \( \vec{v}(\vec{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \)), each of which is an instant volume. Depending on subsequent analysis methods, we represent each 3D sub-volume in the real (\( \mathbb{R} \)) domain and quaternion (\( \mathbb{H} \)) domain, respectively.

Assume we have a 4D flow MRI velocity dataset of N individuals, and the total number of voxels in the 3D space is M (i.e. \( M = XYZ \), with X, Y and Z being the number of voxels in the dimension of x, y and z). In real (\( \mathbb{R} \)) domain, each instant 3D sub-volume can be represented separately using three 1D vectors, i.e, \( v_x, v_y, v_z \in \mathbb{R}^M \), but the relationship among the three components are lost. It can also be represented using a long vector \( v \in \mathbb{R}^{3M} \), by concatenating the three components (\( v_x, v_y, v_z \)). Therefore, the dataset at each time step is
represented by the matrix $V \in \mathbb{R}^{N \times 3M}$, whose rows are the sub-volumes per individual. Although the inherent real relationship of the three components will be also lost, at least the linear relationship is reserved. Therefore, we will use the concatenated vector as the representation in RPCA.

The velocity vector has a more compact form of representation in the quaternion ($\mathbb{H}$) domain, shown as below:

$$v_H = v_x i + v_y j + v_z k \quad (4.1)$$

And a 3D sub-volume of each individual and the dataset of all individuals at each time step can be represented by $v_H \in \mathbb{H}^M$ and $V_H \in \mathbb{H}^{N \times M}$, respectively.

The velocity vector can also be presented in polarized notation, as below:

$$v_H = |v_H| e^{\xi \theta} \quad (4.2)$$

with $|v_H| = \sqrt{v_x^2 + v_y^2 + v_z^2}$, $\xi = \frac{v_x i + v_y j + v_z k}{|v_H|}$, $\theta = \pi/2$. The angle between real and imaginary part of the quaternionic velocity vector equals $\pi/2$, hence the real part is null.

### 4.4.2 Real PCA

Here, we use the concatenated long vector form. Now we have the 4D blood flow velocity volume $V \in \mathbb{R}^{N \times 3M}$, with each element denoted as $v_{ij}$, where $i \in [1, N], j \in [1, 3M]$. When $N > 3M$, the general steps to compute PCA are listed below. Here we use PCA calculation based on the covariance matrix directly, we call it the *direct method* or the *covariance method*, explained as below:

1. Compute the center of the points per dimension: $\overline{v}_j = \frac{1}{N} \sum_{i=1}^{N} v_{ij}, i \in [1, N], j \in [1, 3M]$
2. Compute the centered points: $v_i = v_i - \overline{v}, i \in [1, N]$
3. Compute the covariance matrix $\Sigma = V^T V$
4. Compute eigenvalues and eigenvectors of the covariance matrix: $\Sigma u_i = \lambda_i u_i$, where $u_i \in \mathbb{R}^{3M}$ are the orthogonal normalized eigenvectors and the eigenvalues are ordered $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 0$, $r \leq Min(N-1, 3M)$.
5. Convert the flattened $u_i$ to the vector from: $u_i = (v_{ix}, v_{iy}, v_{iz})$

In medical image processing, it is common that the space dimension is far bigger than the size of the dataset ($N < 3M$). We need to find an efficient method to compute the RPCA in this sparse space. First, we need to introduce the concept of gram matrix. Given a matrix $V$ composed of vectors $< v_1, \ldots, v_N >$ as its rows, its gram matrix $G$ is defined as: $G = V V^T$. We define the new efficient method the *indirect method* or the *gram method*, since this method is based on the gram matrix and then the eigenvectors of the covariance matrix is calculated by dot product of the original centered velocity data and the eigenvectors of gram matrix. The detailed procedure is listed as below.

1. Compute the center of the points per dimension: $\overline{v}_j = \frac{1}{N} \sum_{i=1}^{N} v_{ij}, i \in [1, N], j \in [1, 3M]$
2 Compute the centered points: \( v_i = v_i - \overline{v}, i \in [1, N] \)

3 Compute the gram matrix \( \mathbf{G} = \mathbf{V} \mathbf{V}^T \)

4 Compute eigenvalues and eigenvectors of the gram matrix: \( \mathbf{G} \phi_i = \lambda_i \phi_i \), where \( \phi_i \in \mathbb{R}^N \) are the orthogonal normalized eigenvectors and the eigenvalues are ordered \( \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_r \geq 0, r \leq \text{Min}(N, 3M-1) \).

5 Compute the eigenvectors by projecting them to the original space: \( \mathbf{u}_i = \frac{1}{\sqrt{\lambda_i}} \mathbf{V}^T \phi_i \)

The two method should yield the same principal components and variances. However, this does not imply all eigenvectors are identical. As is known, the eigenvectors are orthogonal in the space of real numbers, and the sign can be arbitrary (+, or -), but it does not matter. For example, in Figure 4.4, the direction of \( i^{th} \) eigenvector can be right (positive) or left (negative), or that of component 2 can be up (positive) or down (negative). No matter which direction it is, component 1 and component 2 are still orthogonal to each other. The signs of the eigenvectors calculated from the above two methods could be the same or different. Namely, both \( +\mathbf{u}_i \) and \( -\mathbf{u}_i \) represent the same eigenvector.

![Figure 4.4: The signs of eigenvectors in RPCA could be different in different methods, but they could represent the same eigenvectors.](image)

### 4.4.3 Complex PCA

CPCA is suitable for vector field data in 2D plane, and we do not directly use CPCA for 3D velocity vector data, but it is included in this section for two purposes. First, the implementation of QPCA will depend on the CPCA of the complex adjoint of the quaternion matrix. Therefore, QPCA share some similar properties as CPCA. Second, it will make it easier to understand some properties gradually from RPCA to CPCA, then to QPCA. CPCA is similar to RPCA. Here we will illustrate the CPCA process assuming the vector data is in 2D plane. The covariance and gram methods that calculate CPCA are similar to those explained in real domain. The main differences between CPCA and RPCA are listed below:

- The direction in 2D space (2D flow vector) is represented in a complex number. Since a complex number represents two real values, the dimension of the dataset will be reduced by half accordingly.
All the transpose operations (symbol: T) in RPCA will be replaced by conjugate transpose (symbol: H).

The space of complex eigenvectors can be represented in a polar coordinate, and the complex domain is spatially invariant to rotation in 2D space.

Similarly, when the size of samples is less than the dimensions of the dataset, we still efficiently construct the eigenvalues and eigenvectors of covariance matrix based on the gram matrix, which is introduced in RPCA. However, it is noteworthy that in the complex domain, an eigenvector can be expressed as \( u_i e^{\theta_i} \), and \( \theta \) could be any value within \([0, 2\pi]\). Figure 4.5 illustrates this using the \( i^{th} \) eigenvector. That means the items in the complex eigenvector could rotate any angle (\( \theta \)) along the unit circle, and the rotation angles should be the same for each item within each eigenvector. But the \( \theta \) values do not need to be the same among different eigenvectors. We can visually perceive this in the figure below: the two directions in the unit circle can represent the same eigenvector in the complex domain. And this will be further extended to a unit sphere in QPCA.

![Figure 4.5: The angles of eigenvectors in CPCA could be different in different methods, but they could represent the same eigenvectors.](image)

### 4.4.4 Quaternion PCA

The process of QPCA is similar to that of RPCA and CPCA. In the quaternion domain, the original dataset is represented as \( V_H \in \mathbb{H}^{N \times M} \), and its complex adjoint form is \( \chi_V \in \mathbb{C}^{2N \times 2M} \).

\[
\chi_V = \begin{pmatrix} V_1 & V_2 \\ -\overline{V_2} & \overline{V_1} \end{pmatrix}.
\] (4.3)

where \( V_H = V_1 + V_2j, V_1 \) and \( V_2 \in \mathbb{C}^{N \times M} \).

As described in section 2.4.3, there is no direct QPCA theory, therefore QPCA is obtained from CPCA on its complex adjoint form \( \chi_V \).
The direct method is based on the complex adjoint matrix $\chi_V$, described as below:

1. Compute the center of the points per dimension: $\bar{v}_j = \frac{1}{2N} \sum_{i=1}^{2N} v_{ij}, i \in [1, 2N], j \in [1, 2M]$
2. Compute the centered points: $v_i = v_i - \bar{v}, i \in [1, 2N]$
3. Compute the covariance matrix $\Sigma = \chi^H V \chi_V$
4. Compute eigenvalues and eigenvectors of the covariance matrix: $\Sigma \hat{u}^{V} = \lambda_i \hat{u}^{V}, \hat{u}^{V} \in \mathbb{C}^{2M}$ are the orthogonal normalized eigenvectors and the eigenvalues are ordered $\lambda_1 = \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_r \geq 0, r \leq \text{Min}(2N - 2, 2M)$.

We are interested in the situation that the number of sample $N$ is far less than that of dimension $M$ ($N < M$). We can still efficiently construct the eigenvalues and eigenvectors of covariance matrix using the gram matrix (indirect method), which is introduced in the RPCA and CPCA. Now, we need to extend the space of complex number (a unit circle) to the quaternion space (a unit sphere), which can be expressed in the polar form: $u_i e^{\xi \theta}$, with $\xi$ being a unit pure quaternion number instead of $i$ in a complex number, and $\theta$ can be any value within $[0, 2\pi]$. Figure 4.6 illustrates this using the first eigenvector. Therefore, the quaternion values of the quaternion eigenvector could lie in any position on the unit sphere. Similar to the complex eigenvector, the rotation angles of all the values within each quaternion eigenvector should be the same and there is no such constraint among different eigenvectors. We can visually perceive this in Figure 4.6. The two directions in the unit sphere can represent the same eigenvector in the quaternion domain.

![Figure 4.6: The angles of eigenvectors in QPCA could be different in different methods, but they could represent the same eigenvectors.](image)

The indirect method of QPCA is described as below:

1. Compute the center of the points per dimension: $\bar{v}_j = \frac{1}{2N} \sum_{i=1}^{2N} v_{ij}, i \in [1, 2N], j \in [1, 2M]$
2. Compute the centered points: $v_i = v_i - \bar{v}, i \in [1, 2N]$
3 Compute the gram matrix $G$: $G = \chi V \chi^H$

4 Compute eigenvalues and eigenvectors of the covariance matrix: $G \phi_i = \lambda_i \phi_i$, where $\phi_i \in \mathbb{C}^N$ are the orthogonal normalized eigenvectors and the eigenvalues are ordered $\lambda_1 = \lambda_2 \geq \lambda_3 = \lambda_4 \geq \ldots \geq \lambda_r \geq 0$, $r \leq \min(2N, 2M - 2)$.

5 Compute the basis vectors of the affine spaces: $\hat{u}_i^\chi \chi = \frac{1}{\sqrt{\lambda_i}} \chi^T \phi_i$.

After calculating CPCA on $\chi V$, we get its eigenvalues $(\sigma_1, \sigma_1, \sigma_2, \sigma_2, \ldots, \sigma_r, \sigma_r)$ and eigenvectors $\hat{u}_i^\chi \chi \in \mathbb{C}^{2M}$:

$$\hat{u}_i^\chi \chi = \begin{bmatrix} \frac{\hat{u}_i}{\sqrt{\lambda_i}} \\ \frac{-\hat{u}_i}{\sqrt{\lambda_i}} \end{bmatrix} \quad (4.4)$$

where $\sigma_i \in \mathbb{R}$, $\hat{u}_i$ and $\hat{u}_i$ $\in \mathbb{C}^M$.

The $i^{th}$ eigenvalue $\lambda_i$ of QPCA is given by:

$$\lambda_i = \sigma_i \quad (4.5)$$

The $i^{th}$ eigenvector $u_i$ of QPCA is given by the following equation, with $\hat{n} = 2i - 1$

$$u_i = \hat{u}_i + \hat{u}_i j = r_i + v_{ix} i + v_{iy} j + v_{iz} k \quad (4.6)$$

After the quaternion eigenvectors are obtained, a further step is needed to convert the quaternion numbers to vectors $(v_{ix}, v_{iy}, v_{iz})$. In the conversion process, we still use the three imaginary parts of the quaternion number to compose the velocity vector, and therefore the real part is discarded [22]. To be consistent with the representation of the real domain, we still define the eigenvector $u_i = (v_{ix}, v_{iy}, v_{iz})$.

### 4.4.5 The Effect of Rotation

Usually the volumes in 4D MRI dataset are not aligned due to the acquisition process, therefore alignment is usually not avoidable before computing PCA for the dataset. It could possibly bring errors accumulated during the non-rigid registration process. So you may ask: is that possible to skip over the alignment step to do PCA in the complex domain for 2D vector field data? The answer is yes, but conditionally. We still need to align the voxels, but the alignment of the co-domain (the value of the voxel) is not necessary. We would also like to find whether this answer also extends to the quaternion domain in 3D space since quaternion is a more general form of complex number and can represent 3D vector field data.

#### Rotation in 2D space

In 2D space, the standard rotation of $\theta$ degree, can be represented in the following matrix:

$$\begin{bmatrix} 
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta) 
\end{bmatrix}$$
\[ R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \quad (4.7) \]

In RPCA, we perform a same element-wise rotation on the simulated image dataset. Here, the rotation performs a geometric transform which transforms the vector \((v_x, v_y)\) in the position \((x, y)\) of a pixel element in an input volume into the vector \((z_x, z_y)\) in the position \((x, y)\) in an output volume by rotating it through a user-specified angle \(\theta\). For example, by rotating the image by \(\theta\) degree, we get:

\[
\begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}. \quad (4.8)
\]

The whole volume at a time step rotated by \(\theta\) degree, is represented as \(Z \in \mathbb{R}^{N \times 2M}, Z = V\Omega\), where \(\Omega \in \mathbb{R}^{2M \times 2M}\) is a sparse matrix composed of blocks of rotation \(R \in \mathbb{R}^{2 \times 2}\) along the diagonal in the following form:

\[
\Omega = \begin{bmatrix} R & 0 & 0 & \ldots & 0 \\ 0 & R & 0 & \ldots & 0 \\ & & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & R & 0 \\ 0 & \ldots & 0 & 0 & R \end{bmatrix}. \quad (4.9)
\]

After rotation, the covariance matrix \(\Sigma^\Omega\) is not equal to \(\Sigma\) before rotation. This can be easily proved as below:

\[
\Sigma^\Omega = Z^T Z
= \Omega^T V^T V\Omega
\neq V^T V = \Sigma \quad (4.10)
\]

However, the gram matrix \(G^\Omega\) is same as \(G\) before rotation due to \(\Omega\Omega^T = I\), where \(I \in \mathbb{R}^{2M \times 2M}\). Thus, the eigenvalues after rotation remain unchanged as before rotation. And the eigenvectors of the rotated volume has the following relationship with those before rotation:

\[
u_i^\Omega = u_i\Omega \quad (4.11)
\]

In CPCA, the rotation matrix \(R(\theta) \in \mathbb{R}^{2 \times 2}\) can be represented using a single complex number \(R(\theta) = \cos \theta + j \sin \theta\). The whole dataset \(V\) is also composed of complex numbers, and \(V_C \in \mathbb{C}^{N \times M}\). After rotation by \(\theta\) degree, the dataset can be represented as \(Z_C \in \mathbb{C}^{N \times M}, Z_C = V_C R\). After rotation, the covariance matrix \(\Sigma_C^\Omega\) is equal to \(\Sigma_C\) before rotation. This can be proven as below:

\[
\Sigma_C^\Omega = Z_C^H Z_C
= R^H V_C^H V_C R
= R^H R V_C^H V_C
= V_C^H V_C = \Sigma_C \quad (4.12)
\]

Similarly, the gram matrix \(G_C^\Omega\) is also equal to that before rotation.
Figure 4.7: The representations before and after Rotation using the first eigenvector calculated in RPCA and CPCA.

Figure 4.7 shows the representations before and after rotation using the first eigenvector calculated based on a simulated dataset in RPCA and CPCA.

Therefore, the answer to the question raised at the beginning of this section is yes if the voxels in the volumes have been already aligned with each other. For 2D vectors, the alignment process of the co-domain can be avoided before calculating PCA in the complex domain.

Rotation in 3D space

In 3D, a basic rotation about one of the axes of a coordinate system \((x, y, z)\) has the following form:

\[
R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

A general form of rotation can be obtained from the above basic rotations using matrix multiplication, \(R_n = R_z(\alpha)R_y(\beta)R_x(\gamma)\), where \(\alpha, \beta, \gamma\) are the angles rotated around the x-axis, y-axis, z-axis, respectively.

A more general form of rotation by an angle of \(\theta\) about a given axis in the direction of \(\mathbf{r} = (r_x, r_y, r_z)\), where \(r_x^2 + r_y^2 + r_z^2 = 1\), is:

\[
R_p = \begin{bmatrix} \cos \theta + r_x^2(1 - \cos \theta) & r_x r_y (1 - \cos \theta) - r_z \sin \theta & r_x r_z (1 - \cos \theta) + r_y \sin \theta \\ r_y r_x (1 - \cos \theta) + r_z \sin \theta & \cos \theta + r_y^2(1 - \cos \theta) & r_y r_z (1 - \cos \theta) - r_x \sin \theta \\ r_z r_x (1 - \cos \theta) - r_y \sin \theta & r_z r_y (1 - \cos \theta) + r_x \sin \theta & \cos \theta + r_z^2(1 - \cos \theta) \end{bmatrix}
\]

(4.13)

In RPCA, we perform element-wise rotation on the dataset. Here, the rotation performs a geometric transform which transforms the vector \((v_x, v_y, v_z)\) in the position \((x, y, z)\) of a voxel element in an input volume into the vector \((z_x, z_y, z_z)\)
in the position \((x, y, z)\) in an output volume by rotating it through a user-specified angle \(\theta\) around a given axis \(\mathbf{r}\). For example, rotate \(\theta\) degree around \(z\)-axis:

\[
\begin{bmatrix}
    z_x \\
    z_y \\
    z_z
\end{bmatrix}
= \begin{bmatrix}
    \cos\theta & -\sin\theta & 0 \\
    \sin\theta & \cos\theta & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    v_x \\
    v_y \\
    v_z
\end{bmatrix}
\]

(4.14)

The whole volume at a time step rotated by \(\theta\) degrees around a given axis, is represented as \(\mathbf{Z} \in \mathbb{R}^{N \times 3M}\), \(\mathbf{Z} = \mathbf{V}\Omega\), where \(\Omega \in \mathbb{R}^{3M \times 3M}\) is a sparse matrix composed of blocks of rotation matrix \(\mathbf{R} \in \mathbb{R}^{3 \times 3}\) along the diagonal in the following form:

\[
\Omega = \begin{bmatrix}
    \mathbf{R} & 0 & 0 & \ldots & 0 \\
    0 & \mathbf{R} & 0 & \ldots & 0 \\
    \vdots & \ddots & \ddots & \ddots & \vdots \\
    0 & \ldots & 0 & \mathbf{R} & 0 \\
    0 & \ldots & 0 & 0 & \mathbf{R}
\end{bmatrix}.
\]

(4.15)

After rotation, the covariance matrix \(\Sigma^{\Omega}\) is not equal to \(\sigma\) before rotation. This can be easily proved as below:

\[
\Sigma^{\Omega} = \mathbf{Z}^H\mathbf{Z} = \Omega^T\mathbf{V}^T\mathbf{V}\Omega \\
\neq \mathbf{V}^T\mathbf{V} = \Sigma
\]

(4.16)

However, the gram matrix \(\mathbf{G}^{\Omega}\) is same as \(\mathbf{G}\) before rotation due to \(\Omega\Omega^T = \mathbf{I}\), where \(\mathbf{I} \in \mathbb{R}^{3M \times 3M}\). Thus, the eigenvalues after rotation remain unchanged as before rotation and the eigenvectors of the rotated volume has the following relationship with those before rotation:

\[
u^{\Omega}_i = u_i\Omega
\]

(4.17)

In 3D space, rotations represented by quaternions take the form:

\[
\mathbf{R} = \left[\cos\frac{1}{2}\theta, \sin\frac{1}{2}\theta\mathbf{r}\right] \\
= \cos\frac{1}{2}\theta + \sin\frac{1}{2}\theta r_xi + \sin\frac{1}{2}\theta r_yj + \sin\frac{1}{2}\theta r_zk.
\]

(4.18)

where \(\theta\) is the angle of rotation and \(\mathbf{r}\) is the axis of rotation. For example, if we set \(\theta = 180^\circ\), we get a pure quaternion \(\mathbf{R} = r_xi + r_yj + r_zk\). Consider a rotation around axis \(\mathbf{r}\) with a rotation angle of \(\theta\), the vector \(v_H \in \mathbb{H}\) after rotation becomes \(v_H^{\Omega} = \mathbf{R}v_H\mathbf{R}^{-1}\), and \(\mathbf{R}^{-1} = \overline{\mathbf{R}}\).

The whole volume at a time step rotated by \(\theta^\circ\) degrees along a given axis, is represented as \(\mathbf{Z} \in \mathbb{H}^{N \times M}\), \(\mathbf{Z} = \mathbf{R}\mathbf{V}\mathbf{R}^{-1}\).

In quaternion domain, after rotation, neither the covariance matrix \(\Sigma^{\Omega}_H\) nor the gram matrix \(\mathbf{G}^{\Omega}_H\) is equal to that before rotation. The influence of rotation on
covariance matrix $\Sigma_H$ can not be ignored due to the non-communative property of quaternion values.

$$
\Sigma_H^\Omega = Z_H^H Z_H \\
= R^{-1} V_H^H RR^{-1} V_H R \\
= R^{-1} V_H^H V_H R \\
\neq R^{-1} RV_H^H V_H \\
\neq V_H^H V_H = \Sigma_H
$$

Similarly, for gram matrix, we have:

$$
G_H^\Omega = Z_H^H Z_H^H \\
= RV_H^H R^{-1} RV_H^H R^{-1} \\
= RV_H^H V_H R^{-1} \\
\neq RR^{-1} V_H V_H^H \\
\neq V_H V_H^H = G_H
$$

Therefore, the rotation influence can not be canceled in quaternion representation. The answer to the question raised at the beginning of this section is no. That is, for 3D vectors, the alignment of both the domain (voxels) and co-domain (values of voxels) can not be skipped before calculating PCA in quaternion domain. However, the authors all mention that Quaternion PCA (or SVD) is invariant to spatial rotation [22] [21] [24]. Therefore, further research is still needed to prove the correctness of this conclusion and then make use of it.

### 4.5 4D Flow Visualization

Visualization provides visual representations of datasets in order to help people carry out tasks more effectively. In order to perceive what the mean and variations of the vector-valued dataset represent, only numbers are not enough to give an intuitive understanding. The main purpose of visualization in our case is to understand the patterns of blood flow velocity vectors in 3D space. In order to present the flow patterns more clearly, the anatomical context will be also visualized using tMIP images. The method we use in our implementation is a popular compositing mode, called maximum intensity projection (MIP), which searches for the sample point with the highest magnitude value of the velocity. With MIP compositing, the vascular structure can be visualized efficiently because the highest velocity must be inside the vessels.

It is currently possible to visualize blood flow in the heart, aorta, liver vessels, kidney arteries, large intraabdominal vessels, carotids, and large intracranial vessels [35]. There are various options available for 3D blood flow visualization, such as glyphs, texture-based methods, topology-based methods and particle tracing.

Glyphs, such as directional arrows, are very commonly used to visualize medical data, such as blood flow. They can be also found in many other applications, because they are intuitive to understand, show full local information and convey
less error compared to other visualization techniques [1]. But you may get a lot of clutter if you extend the 2D visual plane to 3D space. Vector glyphs are a standard tool to convey the properties of flow data, which include both the direction and magnitude of the flow [27]. Two examples of blood flow visualization using glyphs are shown in Figure 4.8, where you can clearly identify the changes in a slice (4.8b) and see a lot of clutter of the whole thoracic aorta region (4.8a).

![Glyphs of Whole Thoracic Aorta Region](image1)

![Glyphs of a slice in thoracic aorta region](image2)

Figure 4.8: Glyphs of the thoracic aortas

Particle tracing methods are also generally used in many applications because it is easy to understand and recognize the flow patterns. The basic idea of particle tracing is to visualize the flow directions by releasing particles and calculating the trajectories based on the vector field, which can be further divided into instantaneous 3D streamlines, time-resolved 3D pathlines and streaklines. (The following definitions are paraphrased based on the course slides of Medical Visualization in TUDelft.)

- **Streamline (Steady)** 3D streamlines represent traces along the instantaneous 3D blood flow velocity vector field for an individual cardiac time-step.

- **Pathline (Unsteady)** Time-resolved pathlines can be used to visualize the temporal evolution of 3D blood flow over one or more heartbeats.

- **Streakline (Unsteady)** Streaklines are generated by continuously injecting a new particle at each time step, then advecting all the existing particles and connecting them together. It's better to display and view the pathlines and streaklines dynamically to fully catch the changes in blood flow over the cardiac cycle.

Considering the difficulty in identifying the differences among the motion graphs, we do not adopt the pathline and streakline methods which show the motion of flow patterns along time. Instead we focus on the volumes at a specific time-step. Therefore the streamline is used as another visualization method of blood flow in our implementation. Two examples of streamlines are shown in Figure 4.9, visualizing the blood flow in thoracic aortas using sphere source in the ascending aorta and the line source across the whole region respectively in Paraview.

We also develop a GUI that demonstrates the vector volumes using streamline and glyph and enables to interactively choose the main component and tune
the parameters to scrutinize the changes. The GUI is composed of a rendering panel that visualize the volumes and a parameter panel. More details can be found in appendix 1. To illustrate the main variations, only the top four eigenvectors, which correspond to the top four eigenvalues, are covered. The new volumes synthesized using the mean volume $\mathbf{\nabla}$ and the top four eigenvectors are shown the same way as Figure 2.2. In order to get a clear view of the changes among different eigenvectors, the parameters should be chosen carefully. If it is too small, you can not see the changes obviously. In the implementation, we set the parameters to have the same order of magnitude as the eigenvalues. For the synthetic dataset, the patterns contained in each eigenvector are clear, however this is not the case for the real-world dataset we have no matter how the parameters are set. More will be explained in the result section.
Chapter 5

Implementation

We follow the proposed workflow and apply our model to the large real dataset using python 3.6 and develop a GUI using python combined with Qt designer and Paraview. The main packages and softwares and their applications in our implementation are as below:

- Mathematica: for theory test of RPCA, CPCA and QPCA based on simulated data.
- Elastix/SimpleITK: for registration and reorientation of vector data.
- Numpy: for quaternion operations and PCA computation.
- vtk 8.2 and Paraview: for volume rendering and streamline/glyph visualization.
- qt 5.9 package in python and Qt designer: for GUI development.

5.0.1 Mathematica

Mathematica is the state of the art in technical computing, which is a powerful system covering plentiful mathematical operations and algorithms. Mathematica supports the basic quaternion algebra, RPCA and CPCA and is easy to use and understand, that is why we use it to test whether RPCA, CPCA and QPCA holds as we expected. But when the data size or the dimension is too large, it will run out of memory, and that is where we will turn to Python.

5.0.2 Elastix

Elastix is an open source software, based on the well-known Insight Segmentation and Registration Toolkit (ITK). The software supports many commonly used algorithms for medical image registration. A command-line interface is enabled to process large datasets by scripting. Elastix should be built from the source code using CMAKE. We implement the shell commands in the Python scripts to finish the whole registration and reorientation process.

For registration and reorientation, we use the code of population-based registration written in Python by Roy van Pelt. Since the version of python used in the code is v2.7.4, parts of the scripts have to be modified to run successfully in v3.6. Rigid registration and non-rigid registration are applied in order.
The first step is to get the transformation matrix which is mapping from each original tMIP to the average tMIP. Once the average tMIP and transformation matrices are obtained, the velocity vector volumes should also be aligned by applying the transformation matrix obtained in the previous step. Since the registration provided by Elastix is only applicable to scalar data, each vector volume should be first split into three separate volumes: \( V_x, V_y \) and \( V_z \), and then recombined after transformation.

After the vector volumes are aligned the same way as tMIP, they are reoriented accordingly. During reorientation process, the inverse of Jacobian matrix calculated from the registration process and the inverse of original transform matrix of the volume are multiplied by the vector volume in succession. (Note: the original transformation matrix is brought by the transform between the patient coordinates and the data coordinates. The original vector data is defined in patient coordinates by default, we need to define them in data coordinates).

### 5.0.3 SimpleITK

ITK is an open-source, cross-platform system that provides an extensive suite of software tools for image processing. SimpleITK is a simplified layer built on top of ITK, intended to facilitate its use in rapid prototyping, interpreted languages, etc. SimpleITK can be simply installed in Python through pip or conda command, and it is mainly used as reading and writing an image in .mhd format in our implementation.

### 5.0.4 Numpy

Numpy is a fundamental package for scientific computing in Python. The two main methods used in our project are numpy.linalg.svd and numpy-quaternion.

Since SVD is a more general form of PCA, we use numpy.linalg.svd instead of sklearn.decomposition.PCA. Another reason for this choice is that sklearn.decomposition.PCA does not support complex number calculation, while numpy.linalg.svd does. However, after many experiments, we find the eigenvalues calculated using numpy.linalg.svd and Mathematica are the same, but the eigenvectors are different. To further examine which result is correct, we reconstruct the original dataset using the top few (this number is dependent on the number of valid eigenvalues \( \lambda_i \), defined as \( \lambda_i > 10^{-8} \) ) eigenvectors. The reconstruction volumes using numpy.linalg.svd are far away from the original volumes no matter how many components are used, while the reconstruction error can be close to zero using Mathematica when we increase the components to a certain number. Therefore, after we calculate the covariance matrix/gram matrix in Python, we do CPCA calculation in Mathematica, and then return the eigenvectors to Python for the following calculation.

Another main package used is numpy-quaternion, which enables numpy to create a quaternion number and manipulate arrays of quaternions in Python. The usual algebraic operations, such as addition and multiplication are available. But multiplication of quaternion matrices is not supported. Therefore, we create a function to do such operation.
5.0.5  vtk 8.2

The Visualization Toolkit (VTK) is an open source software for 3D computer graphics, image processing and visualization. It has a collection of tools for 3D rendering and a suite of widgets for 3D interaction. The latest version of vtk can be simply installed in Python through pip or conda command. It is the basis of many popular visualization applications, such as Paraview. We use this package in Python for 3D rendering of the volumes with MIP, and visualizing the blood-flow velocity using streamlines and glyphs.

5.0.6  ParaView

ParaView is an open-source multiple-platform application for interactive, scientific visualization. It provides a flexible and intuitive user interface, which is easy to add various filters and visualize in multiple layers. Due to its friendly GUI and short time to display a large dataset, we use Paraview to visualize the original dataset and the representations generated from eigenvectors of the real vessels.

5.0.7  qt 5.9 in Python and Qt Designer

There are two commonly used GUI packages in Python, Tkinter and Qt, which are able to connect with vtk to insert the vtk renderings into a GUI. Tkinter is a light weight tool which is suitable for developing simple GUIs. However, it needs to be built from source using CMAKE to support linking with vtk, which is prone to error due to the parameter setting process. Therefore, we finally choose Qt, which is easily installed in Python using pip or conda command. Qt also provides a flexible user interface to design the GUI, and it can be automatically translated into scripts in Python or C++.
Chapter 6

Result

In this section, the experiments and results are divided into three section: QPCA theory test, experiments of synthetic dataset and experiments of real dataset.

6.1 QPCA theory test

In order to test whether QPCA works, we first apply the QPCA model to simulate a small dataset in Mathematica. Thanks to this step, we could identify the flaws of \texttt{numpy.linalg.svd} in calculating CPCA. From the paper [26], we know CPCA has different performances given different datasets. CPCA could outperform RPCA a lot for one dataset in terms of efficiency (the same number of components could represent more variance), and it could also perform similarly as RPCA for another dataset. The properties of the dataset that is more efficient using CPCA are not yet clearly figured out. In our experiments, we simulate three different datasets, by gradually increasing complexity and randomness. The definition of the three datasets and experiments are included in the Mathematica script (see annex 1. synthetic.3D.PCA_test.nb). All the three datasets compose of eight samples and each sample simulates a $4 \times 4 \times 4$ 3D space with 3-elements vector values. They can be represented as $X \in \mathbb{R}^{8 \times 192}$ in the real domain and $X \in \mathbb{H}^{8 \times 64}$ in the quaternion domain. The plot of accumulative variance explained in three scenarios is shown in Figure 6.1. The proportion of variance explained by each feature is calculated by taking the eigenvalue divided by the sum of all eigenvalues. The red line indicates accumulative proportion of variance explained by the top $n$ components in QPCA, and the blue line represents that in RPCA.

![Figure 6.1: Explained variance in RPCA and QPCA for simulated data](image)

The result of QPCA on simulated datasets from Mathematica shows that QPCA
outperforms RPCA in that the top few components explain more variance than RPCA. That is, \( \sum_{k=1}^{1} \lambda_H^k \geq \sum_{i=1}^{1} \lambda_i \) where \( 1 < k < r \), \( r \) is the rank of covariance matrix, \( \lambda_H^k \) is the \( i \)th eigenvalue in quaternion domain and \( \lambda_i \) is the \( i \)th eigenvalue in real domain. This behavior is especially obvious for the first component. This is just as expected and consistent with our intuition as quaternion representation is more compact than real representation. And we assume QPCA share some properties of CPCA, based on the fact that QPCA can be indirectly calculated using CPCA on its complex adjoint matrix. Interestingly, if the three components per velocity vector are linear, the eigenvalues of RPCA is the same as QPCA. The more complicated the relationship among the three components, the better the performance of QPCA.

### 6.2 Experiments of Synthetic Dataset

This section describes the experiments based on the synthetic tubes. As explained in chapter 3, the synthetic dataset is generated to evaluate the behavior of our algorithm with the knowledge of what to expect. We can control the number of samples in the dataset as well as the patterns (vortex and helix) and the locations of patterns. Therefore, we could easily compare the patterns contained in the eigenvectors and evaluate whether each eigenvector calculated in RPCA and QPCA is good to represent the dataset.

#### 6.2.1 Preprocessing

Since synthetic tubes are already aligned with each other when they are created, the first and second steps of our solution, namely registration & reorientation and masking, are not needed for the synthetic dataset. While, in order to reduce the cost of PCA calculation, we still have to truncate the volumes and only keep the voxels with values not equal to 0, which means only the voxels in the tube region are used to generate the high dimension (HD) space. After this step, the number of valid voxels per volume is reduced to 531 from 32768 (32 × 32 × 32).

In real domain, the vector field is concatenated into a long vector, therefore, the dataset \( \mathbf{V} \in \mathbb{R}^{24 \times 1593} \). In quaternion domain, the vector can be directly represented using a quaternion number, that is, \( \mathbf{V}_H \in \mathbb{H}^{24 \times 531} \).

#### 6.2.2 RPCA and QPCA

In this section we present a comprehensive comparison of the performance of RPCA and QPCA. The experiments are separated into two parts. First, we evaluate the eigenvalues and reconstruction error of RPCA and QPCA using all the 24 synthetic tubes. For QPCA, we also change the number of samples and explore how the quaternion eigenvalues and eigenvectors change by comparing them with the real eigenvectors.
Explained Variance

The top main eigenvalues calculated by RPCA and QPCA are nearly the same for the synthetic dataset. The proportion of variance explained by the first principal component is $\lambda_1/\sum_{i=1}^{24} \lambda_i$, which is about 23%. The proportion of variance explained by the second principal component is $\lambda_2/\sum_{i=1}^{24} \lambda_i$, about 19%. The top seven components explain 90% of variance of the dataset, and the top ten components explain 100% variance of the dataset. This shows that the volumes within the dataset have a high degree of correlation. This is easy to understand by visualizing the synthetic flows using streamlines, that the locations of vortices or helices in some synthetic tubes are overlapped with those in some other tubes.

![Figure 6.2: Explained variance in RPCA and QPCA for synthetic tubes](image)

Reconstruction Error

We evaluate the performance of reconstructed volumes based on the reconstruction error, which is also used in many researches [26][10][19]. The reconstruction of a velocity volume $v_{\text{reconstructed}}$ given the mean velocity volume $\bar{v}$, eigenvectors(components) $U$ and the original volume $v_i$ is defined as:

$$v_{\text{reconstructed}} = \bar{v} + UU^H(v_i)$$

(6.1)

Therefore, the reconstruction error is the distance between the reconstructed volume and the corresponding original volume:

$$e_{\text{reconstructed}} = \| V - V_{\text{reconstructed}} \|$$

(6.2)

where $V$ is the matrix of original synthetic vectors and $V_{\text{reconstructed}}$ is the matrix of reconstructed vectors using the given number of components. It should be noticed that for the above formula of reconstruction error, the conjugate transpose operation (symbol: $H$) in quaternion domain corresponds to the transpose operation (symbol: $T$) in real domain.

Figure 6.3 shows the reconstruction error of the 24 synthetic tubes for RPCA and QPCA. For the synthetic dataset, QPCA slightly outperforms RPCA, but the advantage is so small that we can not see the difference in Figure 6.3a. From Figure 6.3a, the reconstruction error is decreased to nearly 0 at the 14th components, so we know the valid number of components is 14, and others.
(a) Reconstruction Error of the top 14 components

(b) Reconstruction Error of all 24 components

Figure 6.3: Reconstruction experiments of RPCA and QPCA

can be regarded as not significant. And it is expected that the reconstruction error will continue to decrease infinitely close to 0 as the number of components increases. This is true for RPCA, but the reconstruction error in quaternion domain increases rapidly. In order to prove this is not coincidence, we changed the number of samples to 10, 12, 18, and did the experiment again. All of them present the same behavior, shown in Figure 6.4. It is not clear why this happens and further research is still needed.

Are the quaternion eigenvectors correct?

As introduced, the velocity volume is represented using only the three imaginary parts of a quaternion number, namely the real part is set to be zero. After QPCA, the real part of each quaternion number in the eigenvectors is not necessarily zero. The author [30] mentions that the real part of a quaternion eigenvector is somewhat related to the magnitude. From the polar form of quaternion numbers, \( v = |v| e^{i\theta} \), the real part is equal to the product of \( |v| \) and \( \cos \theta \). If we only take the three imaginary parts to compose a quaternion value within an eigenvector, the norm of the quaternion value is equal to \( |v| \sin \theta \). From this perspective, the real part is indeed related to the magnitude. Through experimentation, we find an interesting coincidence that the three component \((v_x, v_y, v_z)\) are not present in the expected elements \( v_xi + v_yj + v_zk \). Some components from both RPCA and QPCA are "opposite" to each other. Here, "opposite" refers to the fact that \( |v_x| \) in RPCA is close to the absolute value of \( j \) imaginary part in the eigenvectors of QPCA, similarly \( |v_y| \) and \( |v_z| \) in PRCA match with \( i \) imaginary part and real part in QPCA. In short, the quaternion number can be represented as \( v_z + v_yi + v_xj + ck \), where \( c \) can be zero or any real value. This result is not interpretable, since the quaternion eigenvector is constructed using the three imaginary parts, and the real part is not used. Furthermore, we suspect the values could have been exactly the same, if we do not introduce error due to the current low-precision calculation method.

In order to know whether this is a coincidence due to the special property of the dataset, we change the number of samples from 6 to 23 (randomly chosen from the original dataset) and repeat the experiment. Table 6.1 shows the result. In the result column, "same" is the inverse of "opposite", which means the absolute values of \( v_x, v_y \) and \( v_z \) of some eigenvectors in RPCA match with the \( i, j \) and \( k \) imaginary parts of corresponding eigenvectors in QPCA. And for those that do not have an obvious relationship between RPCA and QPCA are
Figure 6.4: Reconstruction experiments using 10, 12, 18 samples marked as "not sure".

As can be seen from Figure 6.1, this problem exists in many cases. There seems some inherent law behind this phenomenon. Therefore, this is still an open problem that needs further research.

6.2.3 Interpretation

For the 24 synthetic tubes, let us visualize the representations generated using the top four components in the GUI. Figure 6.5 shows the synthesized volumes using the first four components in RPCA. The volumes in the middle column is the mean volume $\mathbf{v}$ of the 24 synthetic tubes. Figure 6.5a and 6.5c are volumes generated using the first component $\mathbf{u}_1$, calculated by adding $-10\mathbf{u}_1$ and $10\mathbf{u}_1$ to the mean volume $\mathbf{v}$. The patterns shown in the first component $\mathbf{u}_1$ is obvious. There are helices near the top end of the tube and a vortex at the end of the straight part. Similarly, some clear patterns can be visualized from other components. These patterns can be found in the original dataset of the
Table 6.1: Relationship of real and quaternion eigenvectors

24 synthesized tubes.

The results of QPCA shown in 6.6 are disappointing. If the representations us-
ing the real eigenvectors are assumed to be correct and used as a standard, the representations using the quaternion eigenvectors are expected to be comparable or outperform the real ones in terms of the information/patterns explained by the main components. However, this is not the case for the 24 synthetic tubes. The first and third quaternion eigenvectors do not seem to provide any useful information, and the second and fourth are quite similar to the corresponding real representations. This visualization result is not surprising since the three components in the quaternion eigenvectors are not as expected. The quaternion eigenvectors could be possibly wrong. Well, we can not eliminate the implementation issue either. More research should be done regarding this problem.

Figure 6.6: Variation of mean volume of 24 synthetic tubes from QPCA

As is known from Table 6.1, the quaternion results are consistent with those in
real domain when the sample size is 6, 7, 12, etc. Therefore, we also select six samples from the 24 synthetic tubes, and did RPCA (Figure 6.7) and QPCA (Figure 6.8) again. The representations generated from the first, third and fourth real and quaternion eigenvectors are the same but in different directions. That is, the $\mathbf{v} - 10\mathbf{u}_1$ from RPCA is the same as $\mathbf{v} + 10\mathbf{u}_1$ from QPCA, and so on. But the second component from QPCA has abnormal performance, which is difficult to interpret.

Figure 6.7: Variation of mean volume of 6 synthetic tubes from RPCA
Figure 6.8: Variation of mean volume of 6 synthetic tubes from QPCA

6.3 Experiments of Real-world Dataset

6.3.1 Sub-Volume Selection

Given a group of volumes within a cardiac cycle per individual, we pick the fourth time step (the peak value of blood flow velocity during the systole phase of heart), and two adjacent time steps, the third and sixth, both of which have a relatively large speed. We apply our model to the three sub datasets separately, and try to make some interesting discoveries.
6.3.2 Registration and Reorientation

This step is to align the volumes from different individuals and make them have the same number of voxels. After registration, all tMIP volumes with have $128 \times 128 \times 52$ voxels. Then after reorientation, all vector volumes also have the same size, namely $128 \times 128 \times 52 \times 3$ in real domain and $128 \times 128 \times 52$ in quaternion domain.

6.3.3 Masking

The main action of this step is to truncate the dataset, in order to get a cleaner region and reduce the number of voxels in the subsequent calculation. We find when we drop 90% of voxels, we could still get a relatively complete coverage of main thoracic vessels. Therefore, we set the threshold in order to keep the top 10% of voxels. The dimension of the dataset is reduced from $6 \times 2555905$ to $6 \times 255591$ in real domain, and to $6 \times 85197$ in quaternion domain. The dimension is still so large that the memory is not enough to calculate PCA using the direct method. Therefore, the gram method is used for both RPCA and QPCA.

6.3.4 RPCA and QPCA

Similar to the experiments done on the synthetic tubes, we will present a comprehensive comparison of the performance of RPCA and QPCA on the real-world dataset. The proportion of variance explained and the reconstruction error for the 4\textsuperscript{th} time-step volume are shown in the following sub sections, and these two measures for the other two time steps are also presented for comparison.

**Explained Variance**

For the 4\textsuperscript{th} time step, the proportion of variance explained by the first principal component is $\lambda_1/\sum_{i=1}^{5} \lambda_i$, which is about 57% for both RPCA and QPCA (see Figure 6.9a). QPCA slightly outperforms RPCA in that the proportion of variance explained by the second component is 28% in QPCA, 1% bigger than that in RPCA (27 %). In both RPCA and QPCA, the top three components explain 100% of variance of the dataset. This shows that 4D MRI of thoracic vessels can be compressed largely. Similarly, the variance explained in RPCA and QPCA for thoracic vessels in the 3\textsuperscript{rd} and 6\textsuperscript{th} time steps are shown in 6.9b and 6.9c.

**Reconstruction Error**

Figure 6.10 shows the reconstruction error for RPCA and QPCA in the 4\textsuperscript{th}, 3\textsuperscript{rd} and 6\textsuperscript{th} time steps, respectively. For the synthetic dataset, QPCA slightly outperforms RPCA, which can be seen from Figure 6.10a and 6.10c. From Figure 6.10a 6.10c 6.10e, the reconstruction error is decreased to nearly 0 at the 3\textsuperscript{rd} components, so we know the valid number of components is 3, and others can be regarded as not significant. And we also discover the same phenomenon.
Explained Variance in RPCA and QPCA for Thoracic Vessels in the 4th Time Step

Explained Variance in RPCA and QPCA for Thoracic Vessels in the 3rd Time Step

Explained Variance in RPCA and QPCA for Thoracic Vessels in the 6th Time Step

Figure 6.9: Explained variance in RPCA and QPCA for thoracic vessels

as that in the synthetic dataset: the reconstruction error of the quaternion representation does not continue to decrease infinitely close to 0 as the number of components increases, but increases rapidly.

### 6.3.5 Interpretation

For the five volumes of the real world thoracic vessels in the 4th time step, let us visualize the representations generated using the top three components (visualized using ParaView). First, we use streamlines to visualize the variations. The RPCA results are shown in Figure 6.11 and the QPCA results are shown in Figure 6.12. However, it is difficult to identify the changes using streamlines caused by the clutter when no obvious patterns are discovered.

We further focus on a fixed slice across the main arc of thoracic vessels and use arrow glyphs to visualize the changes. Figure 6.13 shows the synthesized volumes using the first three components in RPCA. Similarly, the volumes in the middle column is the mean volume $\mathbf{v}$ of the five thoracic vessels. All three components show slightly different locations in ascending aorta and descending aorta where the velocity magnitude changes. Figure 6.14 shows the synthesized volumes using the first three components in QPCA. The observation is similar to that of RPCA, but neither the amount of change in velocity magnitude nor the locations of change are the same. I do not see any obvious relationship between the two representations, possible due to the complexity of this dataset.
We also visualize the variations of the third and sixth time steps, both of which have similar findings as the fourth time step. These figures can be found in the appendix 2 and 3.
Figure 6.11: Variation of mean volume of five thoracic vessels from RPCA using streamlines
Figure 6.12: Variation of mean volume of five thoracic vessels from QPCA using streamlines

(a) $\nabla - 10u_1$  
(b) $\nabla$  
(c) $\nabla + 10u_1$

(d) $\nabla - 10u_2$  
(e) $\nabla$  
(f) $\nabla + 10u_2$

(g) $\nabla - 10u_3$  
(h) $\nabla$  
(i) $\nabla + 10u_3$
Figure 6.13: Variation of mean volume of 5 thoracic vessels from RPCA
Figure 6.14: Variation of mean volume of 5 thoracic vessels from QPCA
Chapter 7

Conclusion

CVD is the major cause of mortality that affects millions of people every year [34]. Early diagnosis is crucial for CVD patients to reduce the probability of emergency which is highly possibly leading to death [34]. New technologies, such as 4D flow MRI, has provided new tools to help physicians diagnose CVDs.

Our research attempts to explore this new data to decode information. A medical atlas is commonly used as a reference for physicians to learn the anatomy or patterns inside human body. Specifically, we build a hemodynamic atlas of the velocity vector of blood flow based on 4D flow MRI. By searching for the best practice in creating atlases, we finally choose the PCA method, which has been applied successfully in many fields to reveal the main patterns of a population, such as eigenfaces and statistical shape models. Furthermore, in order to handle vector dataset, we find some applications with good performance using complex numbers for 2D vector and quaternion numbers for 3D vectors in computer vision and computer graphics tasks. Before these methods are applied to the real dataset, we reproduce the implementation of CPCA and QPCA using Mathematica, and illustrate that both CPCA and QPCA are more efficient in terms of the number of eigenvalues. But the efficiency is highly dependent on the properties of the dataset. Therefore, we apply both traditional RPCA and QPCA to our dataset to further explore whether QPCA method outputs different eigenvectors which could express more information compared to RPCA.

In order to evaluate our algorithm with what to expect, synthetic tubes with less number of voxels, simulating the blood flows in the vessels, are generated. Although QPCA does not outperform RPCA in expressing the patterns of the synthetic dataset, they both represent the dataset well, and some interesting patterns are found. However, we also find the potential problems using quaternion representation by conducting many experiments with different number of samples, such as the unexpected large reconstruction error in QPCA.

Finally, these methods are applied to the real world dataset. QPCA performs slightly better than RPCA in that the first few eigenvalues take a larger proportion than those of RPCA. The interpretation of the variations of the real dataset is difficult. One of the main issues is the limit memory to calculate PCA of covariance matrix, which is solved by generalizing the gram matrix method from real domain to complex domain and quaternion domain.

Based on the performance of both synthetic dataset and real thoracic vessels, RPCA has a good representation of the dataset, while QPCA method seems not
stable nor mature in building the atlas of blood flow velocity vectors. QPCA has to be proven significantly better than RPCA, or QPCA should not be used because RPCA is conceptually simple and easy. More research of QPCA and more effort to interpret the components is needed before QPCA could be applied in this field.

7.1 Main Findings and Implications

Our research can be divided into three phases: proof-of-principle implemented in Mathematica, experiments on synthetic data and experiments on real-world data implemented in Python. Here is a summary of main findings during these steps. In the following conclusions, when we mention CPCA (compared to RPCA), we mean the dataset is composed of 2D vector data. Similarly, QPCA (compared to RPCA), is for the dataset of 3D vector data.

Comparison of RPCA, CPCA and QPCA

- CPCA outperforms RPCA for 2D vector dataset in that complex representation is more condensed and it needs fewer components to achieve better results. The performance is different dependent on the properties of the dataset.

- CPCA is rotation invariant (RPCA is not), which is a good property when the mapping relationship among different images is known and the dimensions of them are the same. Thus, the alignment process may be avoided.

- QPCA for 3D vector dataset has similar performance as CPCA for 2D vector dataset. QPCA significantly outperforms RPCA when the relationship between the three components of a vector is very complicated and appears to be random. However, we haven’t found an accurate definition of the properties of the dataset which is more suitable to use QPCA.

- QPCA is not rotation invariant. This conclusion is different from that of some published papers, such as [23].

Interpretation of variations

- For both the synthetic and real datasets, the reconstruction error of QPCA is slightly smaller than RPCA using the same number of eigenvectors. And the reconstruction error of QPCA increases rapidly when the number of components is bigger than the valid number of eigenvalues.

- For the synthetic dataset, the number of samples may influence the reconstruction of eigenvectors. When the number of samples is 6, 7, 12, 13, 16, 17 and 22, the three components of a velocity vector are as expected in the three imaginary parts \((v_xi + v_yj + v_zk)\). In other cases, the quaternion eigenvectors are not as expected. The variations from both RPCA and QPCA could represent the locations of vortices and helices. And RPCA seems better than QPCA in that more real eigenvectors contain meaningful and interpretable information.
• For the real-world thoracic dataset, the variations of RPCA and QPCA mainly represent the change of velocity magnitude. No other obvious patterns (such as vortex or helix) are identified.

7.2 Future Work

The present study has explored RPCA and QPCA methods in building an atlas of 4D flow MRI. RPCA for concatenated components of vector data can be regarded as a baseline. Although we have some interesting findings regarding QPCA, more research is needed before this method can be used in real world.

• Consult doctors or physicians about the variations and visualization: whether the visualization could help them identify abnormal patterns of blood flow and improve their work efficiency.

• Simulate more data with different properties and study what kind of data is best suitable using QPCA. Put more effort to interpret the relationship of the four components of quaternion, and how to restore the blood flow velocity from quaternion eigenvector accurately.

• Collect more real data, especially from patients with different CVDs, to find the abnormal patterns for each kind of disease.

• Explore possible kernel methods to realize the rotation invariance of QPCA, in order to skip the alignment step.

Alternative methods could also be considered to help diagnose CVD, such as deep learning algorithms. As deep learning nowadays has taken leaps forward in many fields, it also makes possible the automatic and intelligent diagnosis with high accuracy comparable to human experts. With massive data and pervasive evidence, it can be used to make diagnosis of CVD.


[12] William Rowan Hamilton. *Lectures on Quaternions: Containing a Systematic Statement of a New Mathematical Method; of which the Principles Were Communicated in 1843 to the Royal Irish Academy; and which Has Since Formed the Subject of Successive Courses of Lectures, Delivered in


[42] Chi Yuan, Xiaoping Yu, and Ziyue Luo. 3d point cloud matching based on principal component analysis and iterative closest point algorithm. In
Appendix A

Appendix

A.1 GUI

The GUI is composed of a rendering panel that visualizes the volumes and a parameter panel. The parameter panel of the GUI includes four parts, show in Figure A.1 and A.2:

- **Basic** parameter panel:
  - *Files*: Click on the file menu to choose the files. These files, including the mean volume, the 1\textsuperscript{st} component, the 2\textsuperscript{nd} component, the 3\textsuperscript{rd} component and the 4\textsuperscript{th} component) should be uploaded in order, since the rendering will search the specified component in this order.
  - *Opacity*: Set the opacity of the rendering, which is visualized by maximum intensity projection (MIP) raycasting.
  - *View*: Choose whether to show or hide the outline and clipping plane.

- **Others** parameter panel:
  - *Filters*: This parameter is to imitate Paraview, where filters are defined as pipeline modules or algorithms that have inputs and outputs. Two filters are supported, including streamline and glyph. And you can interactively choose the location of the source of the filters by moving the source in the rendering panel or input the coordinates in this panel.
  - *Integration Type / Seeds*: When you choose Stream Tracer as the filter, this parameter will appear. You can choose the integration direction of streamlines, Backward, Forward or Both. The source of stream tracer is chosen in Seed Type, where either Plane Source or High Resolution Line Source is selected. Once the seed type is selected, you can set the positional parameters of the source in the following two parameters.
  - *Plane Parameters*: Set the Origin, Point 1 and Point 2 of the plane source to decide the position of the plane. Input Resolution value to decide the number of streamlines.
  - *High Resolution Line Source*: Set the Point 1 and Point 2 of the line source to decide the position of the line. Input Resolution value
to decide the number of streamlines.

- **Animation 1** panel:
  
  - **Select Components**: Select the component which will be rendered. Only one component can be selected per time.
  
  - **Input Parameter**: Input the parameter used to obtain the rendered volume. The formula of the rendered volume $v$, given the mean vector volume $\bar{v}$, the component $u_j$, $j \in [1, 4]$ and parameter $P$ is as below:

  \[
  v = \bar{v} + Pu_j
  \]  

- **Animation 2** panel:

  - **Input Parameter**: Input the parameter range to obtain a group of rendered volumes, which will be rendered one by one from the lowest value to the largest value. This is activated by clicking the *Apply* button under the parameter input region.

\[\text{Figure A.1: GUI panels}\]

### A.2 Visualization of 3\textsuperscript{rd} Time Step of Real Vessels using Glyphs

The visualization of 3\textsuperscript{rd} time step of real vessels using glyphs is shown in Figure A.3 and A.4.
Figure A.2: Three additional parameter panels of the GUI

(a) Others panel  (b) Animation1 panel  (c) Animation2 panel

Figure A.3: Variations of five thoracic vessels in the 3rd time step from RPCA

(a) \( \nabla - 10u_1 \)  (b) \( \nabla \)  (c) \( \nabla + 10u_1 \)

(d) \( \nabla - 10u_2 \)  (e) \( \nabla \)  (f) \( \nabla + 10u_2 \)

(g) \( \nabla - 10u_3 \)  (h) \( \nabla \)  (i) \( \nabla + 10u_3 \)

Figure A.3: Variations of five thoracic vessels in the 3rd time step from RPCA

A.3 Visualization of 6th Time Step of Real Vessels using Glyphs

The visualization of 6th time step of real vessels using glyphs is shown in Figure A.5 and A.6.
Figure A.4: Variations of five thoracic vessels in the 3rd time step from QPCA

A.4 Parameters of the Real Vessels

The key parameters of the real-world dataset are listed in the table A.1 and A.2. Size is the total number of voxels in each volume. Origin is the coordinate of the origin point of the 3D volume, spacing refers to the distances between a voxel and its adjacent voxels in the up/down, front/back, left/right directions. Transformation is the rotation matrix of the volume caused by the rotation of detector ring in the scanner.

<table>
<thead>
<tr>
<th>Individual</th>
<th># time frames</th>
<th>the peak time frame (systole)</th>
<th>Size per volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19</td>
<td>4</td>
<td>$144 \times 144 \times 50$</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>4</td>
<td>$128 \times 128 \times 52$</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>4</td>
<td>$144 \times 144 \times 52$</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>4</td>
<td>$96 \times 96 \times 54$</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>$112 \times 112 \times 58$</td>
</tr>
</tbody>
</table>

Table A.1: Key parameters of thoracic aortas (1)
Figure A.5: Variations of five thoracic vessels in the 6th time step from RPCA

Table A.2: Key parameters of thoracic aortas (2)
Figure A.6: Variations of five thoracic vessels in the 6th time step from QPCA