Generative algorithms for Myocardial-Scar-Tissue structures

V. Bayer

August 2023

Contents

Chapter 1

Motivation

The incorporation of engineering methodologies, such as simulations, machine learning and software-development into the modern healthcare system offers a lot of potential in several areas. On the one hand, algorithms aid diagnosis ([22]) and drug-discovery ([9]), on the other hand, the incorporation of computer models has lead to the advent of in-silico clincal trials ([23]) and allows for a powerful analysis of diseases, treatments, drugs and further.

By can leveraging medical imaging technology, numerical models and machine learning one can an enable an in-depth analysis of blood-flow ([12]), tissue structure ([7]) and underlying physical processes is enabled ([17], [14]). For example, [20] examine machine learning algorithms to determine electrical patterns in the Atria, [25] examine mechanics and electrophysiology related o tachycardia, [21] examine graph neural networks for blood pressure pulses and [4] simulate cardiac stimulation by pacemakers.

It is possible to differentiate between a patient-specific ([13]) and population based approaches ([1]). In the former, one aims to aid diagnosis and treatment of a specific individual. The latter uses models to analyze populations, so called *virtual humans*.

This approach fits well into the field of generative machine learning. Generative algorithms have developed strongly in the last decade ([19]). Applications range from image generation ([3]) to prompt-conditioned language generation ([10]). In the setting of *virtual humans*, the data does not correspond to 2D images or language, but to 3D-objects with organic geometries.

Working with generalized *virtual humans* addresses several needs. By simulating drugs on computer models, animal testing can be reduced, enabling faster, cheaper and ethical testing of drugs. Furthermore, in-silico models enable the representation of otherwise under-represented patient-groups (such as children with ischemia) and circumvent privacy policies concerning data-sharing of patients.

This work is supported by ELEM Biotech, a biomedical simulation spin-off from the Barcelona Super-computing Center. ELEM offers simulations of virtual cardiovascular and respiratory systems. These services have found applications in drug-delivery via airways [2], aortic valve geometries and their effects [16], examining cardio-toxicity of drugs [1] and more.Fm

1.1 Problem Formulation

Aim of this work is to generate myocardial scar tissue structures as they occur after a heart attack. The generated data can then be used to gain insights with in-silico methods.

The problem can be classified as a machine-learning problem for 3D shapes. For this type of problem, several methodologies, as well as different ways of representing a 3D shapes, exist ([8]). Goal is generating the organic shape of a myocardial scar within a human heart geometry. Our approach obtains a set of plausible scars, which can be filtered to represent relevant features and evaluate the resulting bio-physical properties via simulation. The developed software is incorporated into ELEM's simulation software and enables the examination of user-defined ischemic cardio-myopathies. The main questions addressed in this work are:

- What is an adequate shape representation for a myocardial scar?
- What kind of features best define a myocardial scar?
- Which algorithm is best suited for shape generation?
- Which features does a user care about how can the generative process consider them?

This work starts by explaining the used software in section 2. Then an introduction into cardiology and the examined disease is provided in section 3. This is followed by an examination of common 3D-shape representations in section **??**. Section **??** provides an extensive overview of machine learning, intuitions, libraries and mathematical models. It ends with the examination of some specific algorithms which are potentially suitable for shape generations.

These sections provide all necessary background information.

Section **??** explains the data-acquisition process from MRI-scans to volumetric 3D data. Section **??** examines the chosen data representation. Section **??** provides a visual analysis of the data and occurring patterns. Furthermore it outlines the generative process, choice of algorithm, challenges and solutions. This section ends by providing a set of generations. Section **??** looks at future directions, such as automized evaluations.

Chapter 2

Software

2.1 The Visualization Toolkit

The Visualization Toolkit (VTK) is a C++ class library which offers several objects for the manipulation of data. It was originally written as part of a Textbook called "The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics" in the 20-th century ([18]).

Now it is widely used in data visualization and 3D computer graphics. It can work with scalars, vectors, tensors, volumetric methods as well as implicit fields and unstructured grids. Methods contained in VTK are triangulation, contouring, mesh smoothing and more. Its main features are

- data objects, such as finite element meshes, or unstructured grids for data processing and analysis
- data arrays, which define discrete field values over a mesh. E.g. pressure, temperature, stress tensor
- filters for data processing and analysis which do tasks such as smoothing, decimation, surface reconstruction, and implicit modeling
- advanced rendering techniques
- scripting capabilities in C++ as well as Python
- parallelizability of some functionalities for high performance computing

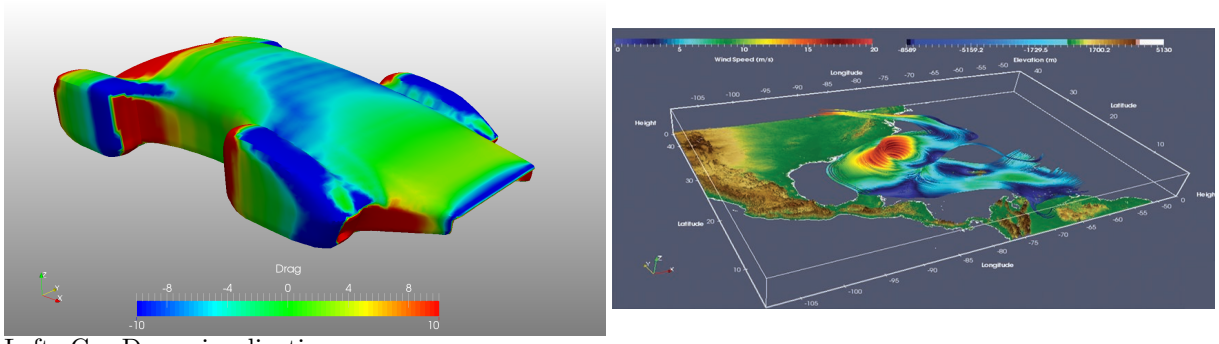
Within the context of this work, VTK methods are used extensively to manipulate data and create visualizations within Paraview.

2.2 Paraview

Paraview is an open-source platform for interactive data visualization focused on simulation and high performance computing. It is built on top of the Visualization Toolkit (VTK) Library. Among its functionalities is the analysis of extremely large data-sets, rendering and simulation. Due to its memory parallel processing capabilities it is used to visualize tera-scale data as they might occur during supercomputer-based CFD-simulations. Furthermore, it runs on windows, mac-OS as-well as on linux. It is a multi-platform visualization application with an extensive range of use-cases. Examples are the analysis of climate-data [15].

The 3D-data used in this work corresponds to a human heart geometry and ischemic scar tissue structures. The reference mesh consists of 3 million cells and 600.000 vertices. As such, para-views capability of handling large data-sets effectively is required. One can differentiate

Figure 2.1: Exemplary Applications of Paraview



Left: Car Drag visualization
www.simscale.com/wp-content/uploads/2014/07/futuristic-car$_d$*rag.png*
Right : *WeatherSpeedvisualization*
www.kitware.com/integration-of-paraview-catalyst-with-regional-earth-system-model/

between the left ventricle (blue) and the right ventricle (red). Figure **??** and figure **??** display a heart shape. This work examines the shape of ischemic scars within the left ventricle. These are obtained by processing MRI-scans. Figure 2.4 and figure 2.5 display such a scar shape. The hole displayed in figure 2.5 is not physical, instead it is due to the processing, which leaves out the basal region. This is explained with more detail in section 6

Figure 2.2: Front View of a Heart

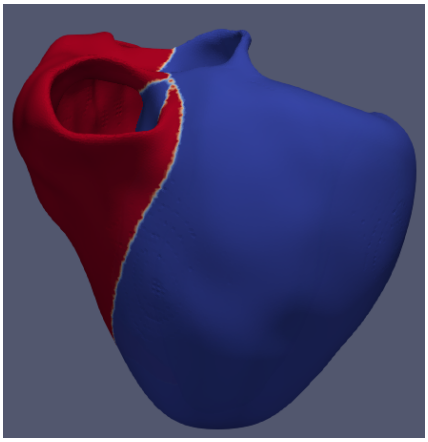


Figure 2.3: Back View of a Heart

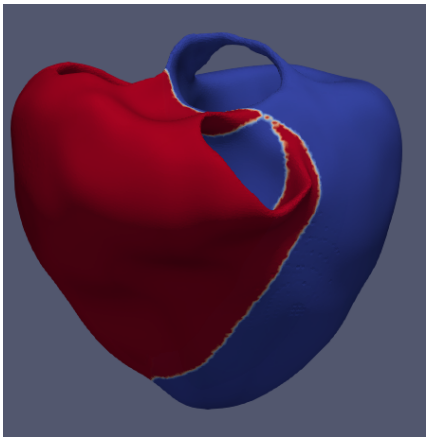


Figure 2.4: View of a scar from the front

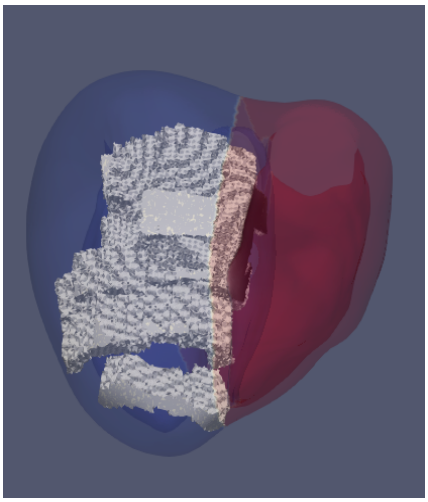
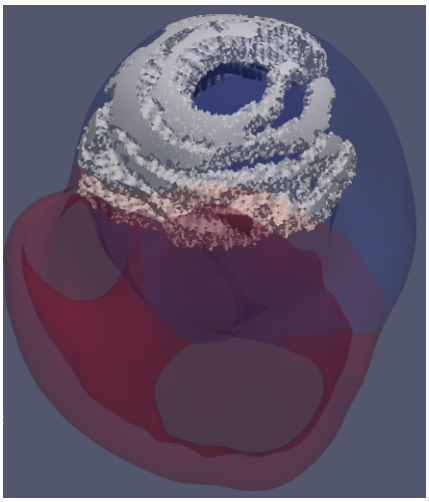


Figure 2.5: View of a scar from the below



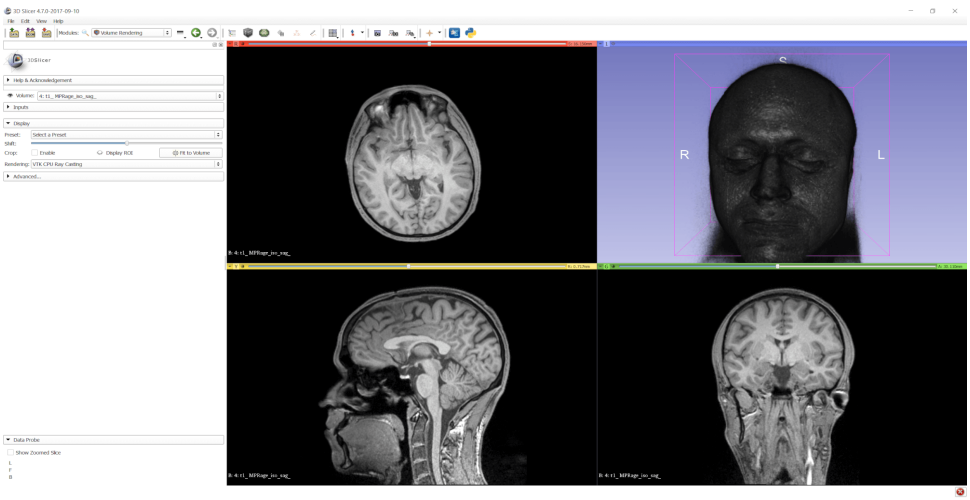
2.3 3D Slicer

3D Slicer is a comprehensive, open-source platform for medical image computing, widely used in clinical and research settings. It supports visualization, processing, segmentation, and analysis of various imaging modalities, including MRI, CT, Ultrasound, and PET. Its flexibility allows users to handle multidimensional datasets with precision, making it essential for tasks like treatment planning, diagnostic assessments, and quantitative analysis.

Key features of 3D Slicer include contrast enhancement and thresholding, which aid in the accurate segmentation of anatomical structures. Segmentation, a critical step in analyzing medical images, involves isolating regions of interest, such as tumors or ischemic scars. While 3D Slicer provides both manual and semi-automated tools, manual segmentation can be prone to user-dependent variability. Figure 2.7b demonstrates this issue, where the brown mask representing an ischemic scar fails to fully align with the endocardium, the innermost layer of the myocardium. This misalignment contradicts the known physical properties of ischemic scars, which typically adhere closely to the endocardium. Such errors, though common in manual segmentation, highlight the need for meticulous attention when using these tools in clinical or research applications.

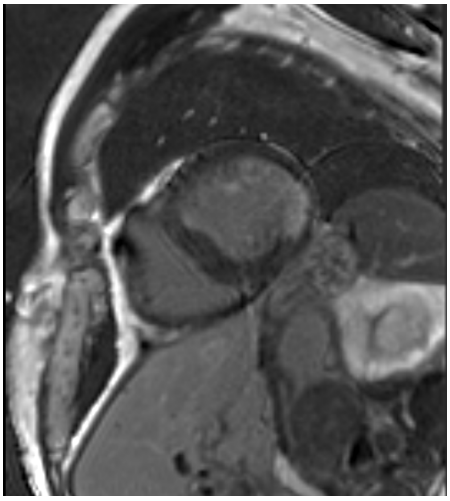
Despite these challenges, 3D Slicer remains a powerful tool due to its wide-ranging functionality and its strong user and developer community. Ongoing advancements aim to improve segmentation accuracy and enhance automation, further reducing the risk of errors and expanding its use in various medical imaging fields.

Figure 2.6: Slicer interface

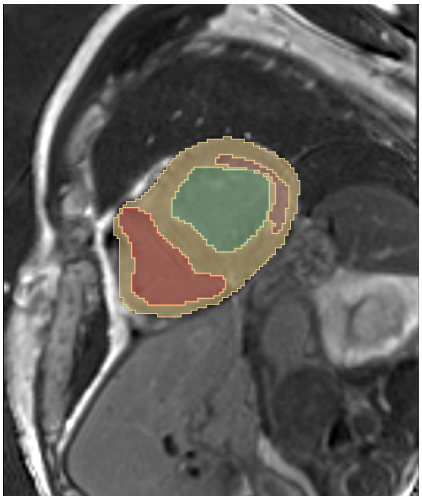


https://www.andreasjakl.com/wp-content/uploads/2017/10/mri-adjusted-brightness-contrast.png

Figure 2.7: MRI-scan and Segmentation



(a) MRI-slice



(b) Segmentation

Left: MRI scan of a human heart
Right: Segmented scan of a human heart (yellow = myocardium, green = left chamber, red = right chamber, brown = ischemic scar)

Chapter 3

Basics of Cardiology

In the study of heart, clinicians often differentiate between the left and right ventricle/atrium ([24]). The atrium is responsible for receiving blood, while the ventricles are responsible of the distribution to lungs and body tissue. Size-wise the ventricles are much bigger, the biggest part being the left ventricle. For the purposes of this work the atrium structure is neglected.

Concerning the left and right ventricle (subsequently referred to as LV and RV), although they operate with equal quantities of blood and under equilibrium conditions, they differ strongly in their size, shape and function. The left ventricle, which distributes oxygenated blood, through the aorta, into the body tissue, presents the more powerful, thicker, myocardial muscle.

A likely reason is that LV operates against high pressure which is present in the circulatory pathways throughout the body tissue, whereas the resistance in the pulmonary circulation is much lower. Therefore, while both sides generate the same flow, the RV flow is created against lower pressure, requiring the smaller muscle.

During the examination of ischemic scars only the left ventricle is considered. This is generally accepted since scarring occurs predominantly within the myocardium surrounding the Left-Ventricle [5].

3.1 Cardiac Function

Function-wise, the left ventricle receives oxygen-rich blood from the left atrium and pumps it to the aorta. From here, the blood is distributed to other parts of the body. The oxygen is used for cellular respiration and enables the synthesis of ATP. After this, the de-oxygenated blood is directed to the right atrium via the Vena Cava. Furthermore, the aorta branches into the left and right coronary artery, blood vessels, which transport oxygenated blood to the heart muscle itself. They are responsible for coronary circulation. Inhibition of the blood flow in these regions can lead to insufficient alimentation of the heart and cause infarcts.

The right ventricle receives de-oxygenated blood from the right atrium and pumps it to the pulmonary artery. Following this, the blood is re-oxygenated in the lungs and directed to the left atrium through the pulmonary veins. This closes the circulation.

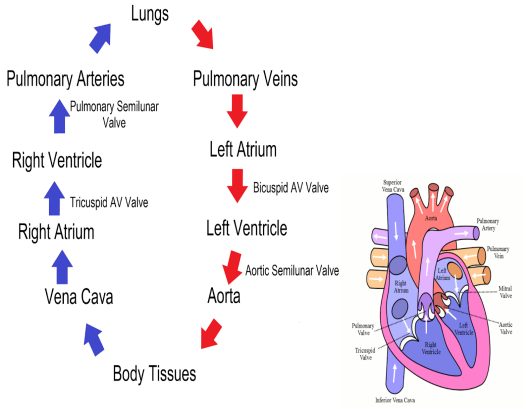


Figure 3.1: Blood Circulation
https://www.zenflowchart.com/blog/blood-circulation-in-heart-flowchart

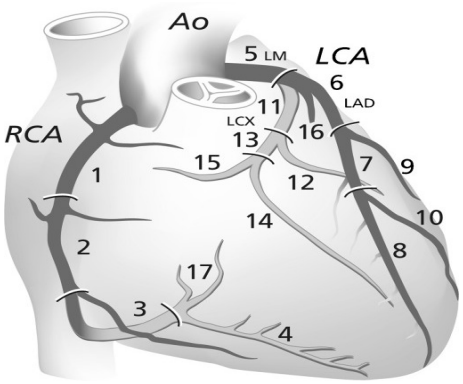


Figure 3.2: Coronaries
https://www.heartfoundation.org.nz/images/heart-healthcare/coronary-artery-disease.png

The alimentation of the heart itself is carried out by the coronary arteries displayed in Fig. 3.2. Blockage of these arteries leads to a heart infarct and subsequent scarring.

Within this work we are interested in the 3D form of scar tissue created by heart attacks. Scarring in the right ventricle, although possible, is a very rare condition and will not be considered in the scope of this work.

3.2 Left Ventricular Segmentation and Coronary Territories

Coronaries can be associated in an approximate manner to certain heart regions. These regions are used to segment the heart. The standard for myocardial segmentation has been established by the American Heart Association. It defines several regions in the left ventricle. These regions are used by clinicians to specify locations uniformly across different heart geometries. We refer to the segmentation standard as specifying *AHA*-regions. Fig. 3.3 visualizes the standardized segmentation applied to a heart. Fig. 3.3 visualizes a left ventricle with colored AHA-regions in the heart- and the disk-representation.

Figure 3.3: Left Ventricular Segmentation and Coronary Territories

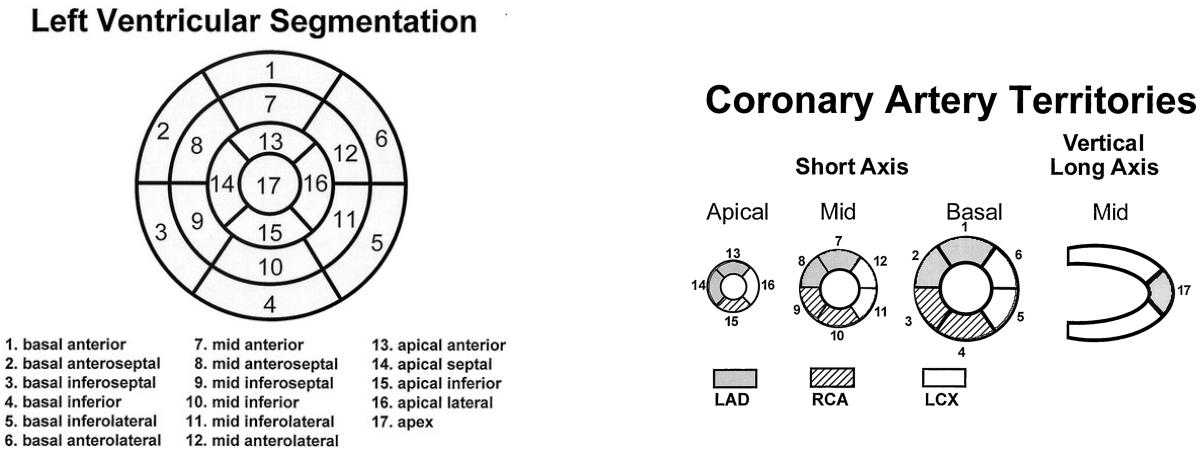
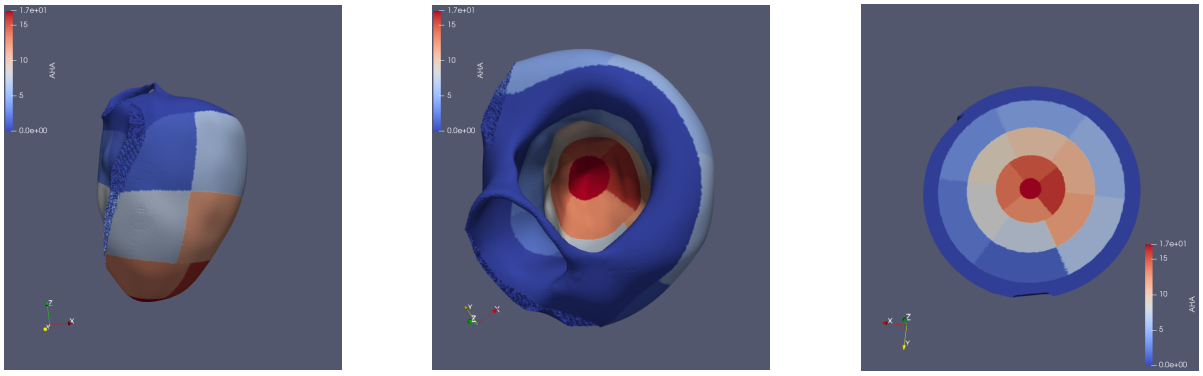


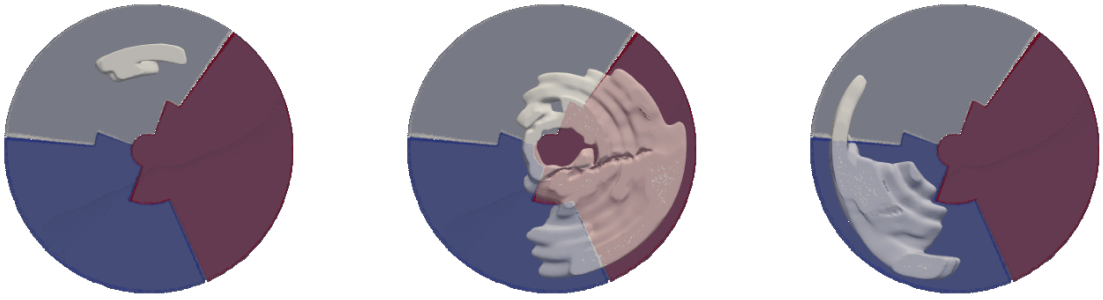
Figure 3.4: Heart with Colored AHA Regions



In these images, ParaView is used to visualize the left ventricle. Its areas are colored to indicate the respective AHA region, which is then mapped (flattened) onto a disk. The heart apex corresponds to the disk center, and the base to the outer border.

Figure 3.2 visualizes two coronaries (RCA and LCA) and their traversal through the AHA-regions. Since the region is related to a coronary artery, it is reasonable to assume that blockages of it result as scarring in associated regions. Figure 3.5 displays such scars in respective regions. Our dataset indicates that there is a correlation between shape and affected coronary artery/region.

Figure 3.5: Scars in coronal regions (white: LCX, blue: RCA, red: LCA)



(a) Scars in LCX region (b) Scars in LAD region (c) Scars in RCA region

3.3 Ischemic and non-ischemic Cardiomyopathy

Cardiomyopathy terms diseases which affect the myocardium, leading to abnormal behaviour of the heart, for example causing enlargement, thickness or stiffness. This effect can lead to symptoms of heart failure, inefficient blood flow and pathologies in the electrical activation pattern causing arrhythmia's. Literature differentiates between *non-ischemic* and *ischemic* heart disease (IHD). *Ischemic* cardiomyopathy is related to coronary artery disease. All other cardiomyopathyies are named *non-ischemic*. IHD is by far more common and will be the focus of this work. Coronary artery disease usually reduces blood flow to the heart and the resulting insufficient nutrition in affected areas causes cell death (necrosis) in response to stress. The affected region forms scar tissue, which has different mechanical and electrical properties than the surrounding healthy tissue. A result can be arrhythmia, weakened mechanical contraction and an overall decrease in heart functioning. It is known, that after an infarction and the resulting scarring, the heart adopts different techniques to mitigate the impaired functioning. These adoptions, called *hyper-enhancements*, depend on the type of the scar. Common adaption strategies are hypertrophy, where the heart-muscle enlarges in an attempt to increase its mechanical pumping ability and the coronaries growing new branches to supply the damaged cells with nutrients and prevent total necrosis. Ischaemic cardiomyopathy is differentiated between *transmural* and *subendocardial* infarction, as well as *segmental extend* of the scar-tissue. These differentiations are important and can be used as measures for the seriousness of the IHD. They are correlated to different outcomes of the hyper-enhancements.

Fig. 3.7 displays the segmental intensity of an *acute* ischemic scar. It is called *acute* in contrast to *chronic*, because parts of the damaged tissue can recover due to re-vascularization. For *chronic* infarcts this is not the case. The heart can re-recruit damaged heart cells and regain their functioning. The extend to which this is possible depends on the *transmurality* of the scar. As mentioned in [6], a scar with less than 50% transmurality has good chances of recovering, above 70% however it is very unlikely. Therefore, it is common to examine the segmental intensity, together with the segmental transmuarlity of scar tissue in order to evaluate the likelihood of recovery.

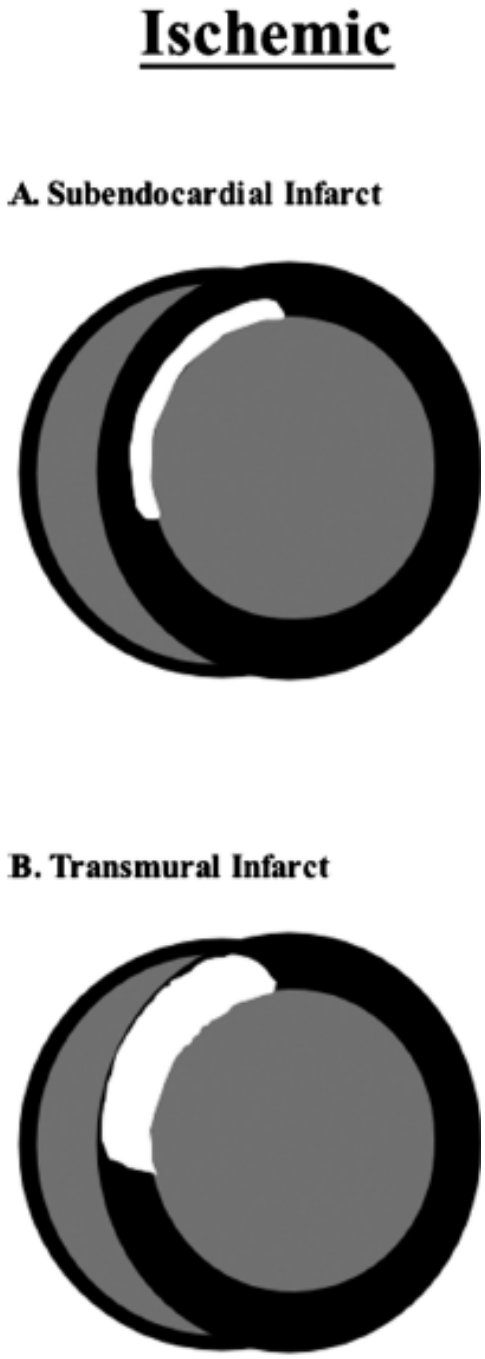


Figure 3.6: Transmurality of Infarction

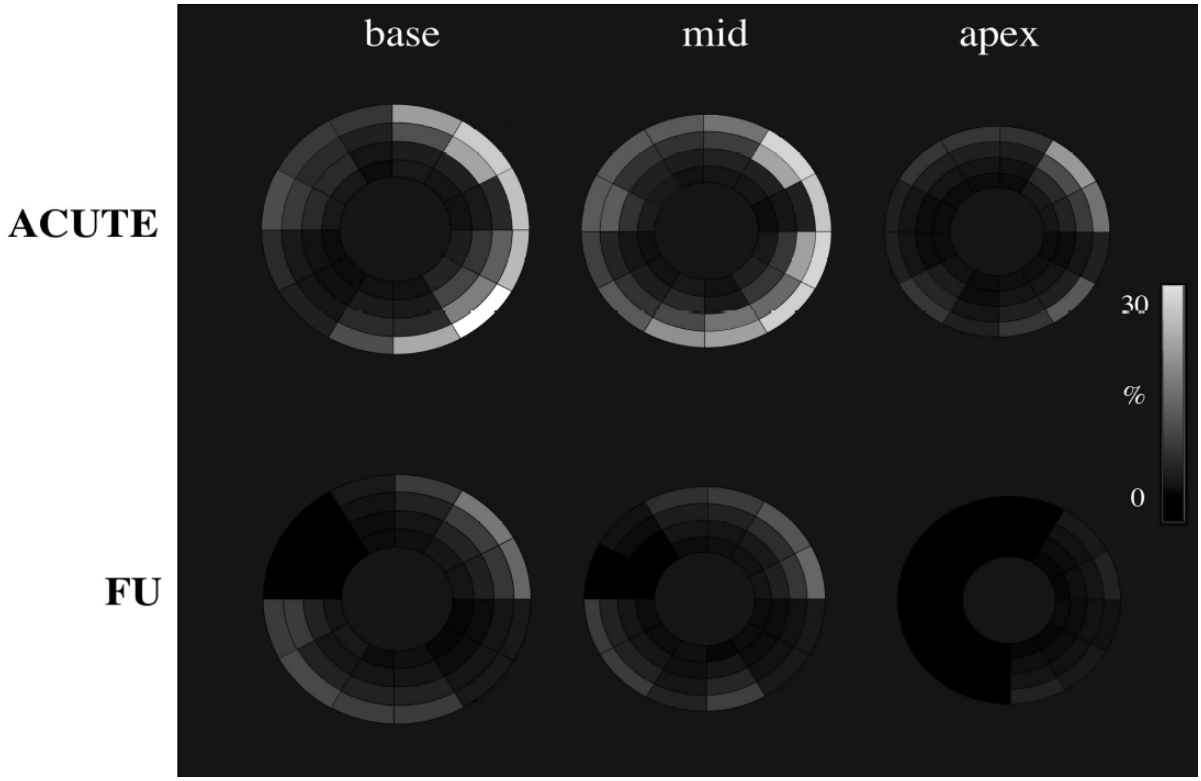


Figure 3.7: Segmental Intensity

3.4 Significance of Contribution

The disease IHD, also referred to as coronary artery disease (CAD) and atherosclerotic cardiovascular disease (ACD), is one of the leading causes of death worldwide. Approximately 1.72% of the world's population is affected, resulting in around nine million deaths annually [11]. According to the WHO, 16% of the annual deaths are caused by IHD. In Fig. 3.8, heart diseases represent the leading cause of death with a significant margin.

We deliver a framework for the customized generation of ischemic scar models, which can be leveraged with simulations of the cardiovascular system to study treatments through drugs, as well as the biophysical effects of specific scar types. Furthermore, privacy concerns are circumvented by the usage of artificial generation. sc

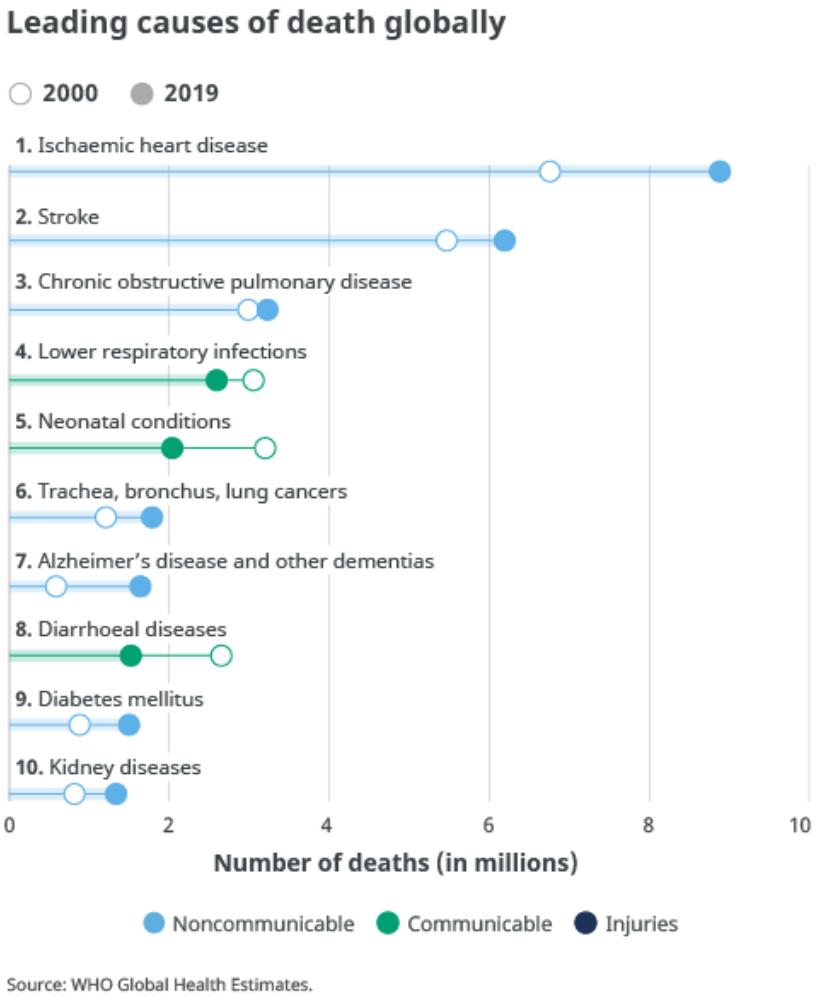


Figure 3.8: WHO Estimates

Considering these numbers, it is safe to say that heart disease continues to be among the most important medical issues. Although the heart is one of the most studied organs, many challenges and mysteries remain. The recent advance of engineering methodologies into the field of human biology can help to reach a deeper understanding of heart function, heart diseases, causes of death, and treatments. Never before has it been possible to examine the heart at such a low ratio of risk/cost to insight.

Chapter 4

3D-shape representations

Several different representations exist for 3D shapes, each with different advantages and disadvantages. In the following, these representations are presented. The specific characteristics and fields of application are outlined.

4.1 Voxels

Voxels can be understood as an analog of pixels onto 3D-space. They share many properties. For example, a representation of a 3D-shape by voxels corresponds to discretizing the space with a 3D-grid and labeling each element with the corresponding voxel-value. Much like pixel-values, these can represent a colour, a texture, or any other label.

- Advantages:
  - representation of multiple shapes within an equally-sized and equally discretized volume in a voxel-based manner
  - has a point-to-point correspondence between voxels
  - fine discretization enables high accuracy
- Disadvantages:
  - inefficient way of representing 3D shapes if one also needs to voxelize the space around it $\rightarrow$ high memory requirement
  - computer software is usually optimized to render polygons instead of voxels.
  - usually not differentiable
- Domain: natural data-format for medical images e.g. MRI-scans

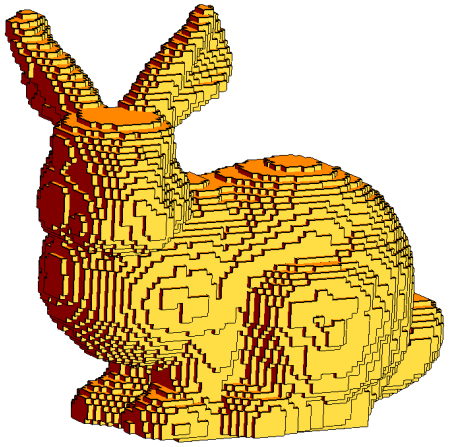


Figure 4.1: Voxel representation of a rabbit
https://cse.iitkgp.ac.in/ pb/research/3dpoly/bunny$_2$*.PNG*

4.2 Point Clouds

A common representation of 3D shapes is Point Clouds. Point Clouds are discrete sets of points, where each point is labeled with a position in 3D-space. This is among the simplest and most memory efficient representations. The point density can be balanced according to the needs. Points can also be labeled.

Advantages:

- efficient in terms of memory
- high (selective) density enables high accuracy
- very intuitive - shape manipulation is very easy

Disadvantages:

- no inherent surface information → unsuitable for rendering, simulation etc.
- point cloud to a polygon is very computationally expensive.
- no point-to-point correspondence -between points
- usually not differentiable

Domain: raw data-format obtained by scanning physical space e.g. Lidar

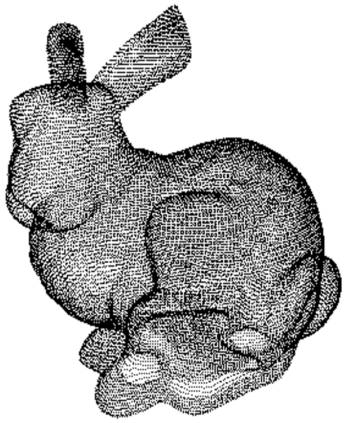


Figure 4.2: Point Cloud representation of a rabbit
https://www.researchgate.net/publication/228945815_Reconstructing_Implicit_Surfaces_with_Boundaries

4.3 Meshes

Meshes, or Polygons, are the standard format for many algorithms in computer science and engineering. They consist of a set of points in 3D space connected in a graph structure. While being among the most useful shape representations, it is also the most information-heavy with significant memory requirements. beginitemize

Advantages:

- contains surface information → suitable for rendering
- very suitable for numerical solvers.
- high (selective) density and variable polygons enables high accuracy
- used extensively in computer graphics, gaming, animations and engineering
- extensively studied → shape manipulation is easy

Disadvantages:

- complex representation
- can exhibit edge intersections and holes
- requires storage of point and topology information
- constricted to a fixed topology.
- no inherent point-to-point correspondence between vertices
- usually not differentiable

Domain: standard in engineering, simulations

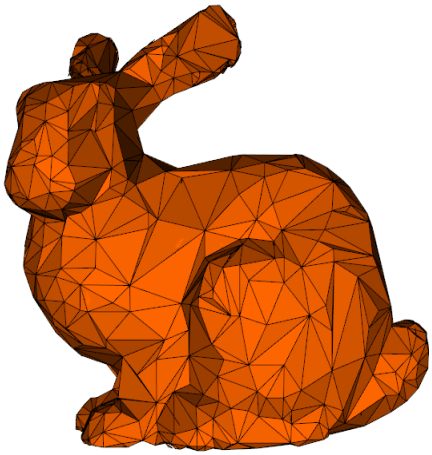


Figure 4.3: Mesh representation of a rabbit
https://i.stack.imgur.com/pFtBn.png

4.4 Implicit representations

Implicit shape representations do not attempt to represent the shape explicitly, e.g. as one would do with a field of occupancy, point clouds or meshes. Instead, the shape information is contained implicitly in some function over 3D space. A very common type of implicit shape representations is through the use of (un-)signed distance functions. Here, the 3D-shape is represented by the level of the distance function. For example, values of zero might correspond to the surface, negative values to the inner volume and positive values to the outer volume. The value in $3D$ space represents the nearest distance to the boundary. In principle it can also correspond to some other measure.

Advantages:

- information efficient - stores a function instead of a volume
- differentiable, the function can be continuous
- surface information as zero level set
- extraction of mesh-representations possible
- has point to point correspondence
- very popular in computer graphics.

Disadvantages:

- less intuitive → shape manipulation can be difficult
- shape only contained implicitly → requires post-processing
- usually obtained from another shape representation → requires pre-processing

Domain: computer graphics

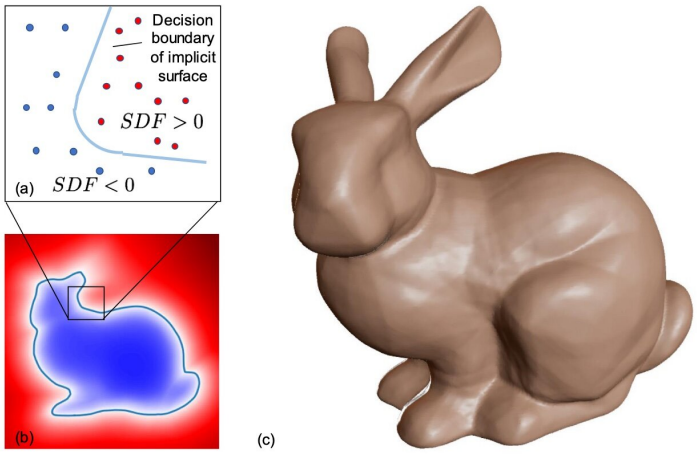


Figure 4.4: implicit representation of a rabbit
https://arxiv.org/pdf/1901.05103.pdf

4.5 Statistical Shape Models

Statistical Shape Models are a popular method in computational anatomy which represents shapes as vectors of landmarks. Between the landmarks of different shapes point-to-point correspondance can be established. The shape distribution is then be defined by the data-vectors via the euclidean mean and covariance. Liner interpolation between the corresponding landmark coordinates is possible. By employing linear methods, such as PCA, the shape variablitiy can be examined and new shapes can be created.

Advantages:

- information efficient - landmarks to represent shape
- linear, interpolation between landmarks possible, differentiable
- ubiquitous and well studied
- shape manipulation is easy

Disadvantages:

- requires landmarks with point-to-point correspondence → only works for equal topology
- does not capture nonlinear deformations

Domain: medical engineering

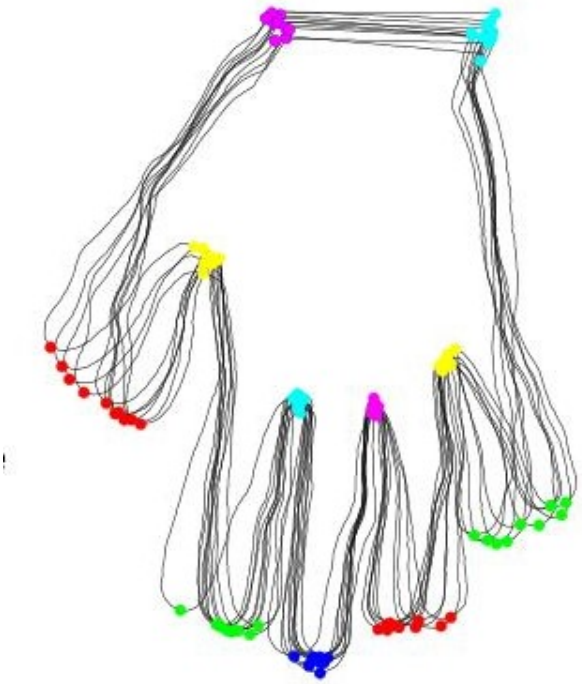


Figure 4.5: statistical shape model of a hand
https://www.slideserve.com/albert/information-theory-and-automatic-model-building

Chapter 5

Machine Learning basics

All of machine learning aims to create a model which minimizes/maximizes a loss/objective by choosing the best model parameters. This objective is designed in such a way that the resulting model captures underlying data dynamics. Usually, it corresponds to a likelihood-measure describing how well a parameterization describes available data and/or a loss-function which measures deviations from desirable properties. The extremum and the corresponding model-parameters yields the an optimal parameters within the set of possible parameterizations. Since the set of all possible model-parameterizations is prohibitively large, assumptions must be made to restrict the search-space to a feasible size. One such assumption might be that the data comes from a multi-variate Gaussian distribution, that it follows a linear line (or is separable by such) or that any optimal model is smooth. More assumptions exist.

Generally, two approaches are distinguished. The Fisherian and Bayesian approach. The first bases itself on the concept of long-term frequency of events and aims to develop a model which estimates parameters according to available data, while the latter bases itself on the concept of probability distributions and aims to develop a model which infers a posterior distribution of parameters. In simple terms, Bayesian methods use models whose parameters are themselves probabilistic. A prior distribution is assumed which is updated to form a posterior distribution. Fisherian models use deterministic, but unknown, parameters. In both cases these parameters are updated to (explicitly) optimize a metric and (implicitly) model the available data.

5.1 supervised/unsupervised/semi-supervised ML

An important distinction that can be made in machine learning is between supervised learning, unsupervised learning and semi-supervised learning. These terms correspond to learning from a labeled dataset $D = \{(x_i, y_i)\}_{i=0}^{N}$, extracting patterns from an unlabeled dataset $D = \{(x_i)\}_{i=0}^{N}$ and combining both methodologies respectively. Supervised learning is the more common and better studied method. In contrast to unsupervised learning it can be applied directly for making predictions by modelling the conditional probability $p(y|x)$. Furthermore it comes with rigorous mathematical guarantees. Unsupervised learning on the other hand aims towards useful pattern extraction. However, due to the lack of information on y it can not be used (directly) for prediction tasks. Furthermore, mathematically rigorous definitions of training-error and generalization-error are located in a supervised training setting. Extending these concepts to unsupervised learning may be possible but it is not guaranteed.

5.1.1 supervised machine learning

In the setting of supervised machine learning, a data-set D is available which can be described as

$$D = \{(x_1, y_1), ..., (x_n, y_n)\}, \tag{5.1}$$

with $x \in \mathcal{R}^d$ and $y \in \mathcal{R}^l$. The data-points are drawn from some unknown distribution $(x, y) \sim p(X, Y)$. Goal is finding a function $h_\theta(x)$ so that

$$y = h_\theta(x) \text{ for } (x, y) \sim p(X, Y). \tag{5.2}$$

This requires definition of a hypothesis class $\mathcal{H}$ which contains all examined hypothesis functions $h_\theta \in \mathcal{H}$. Ideally, the $\mathcal{H}$ would contain all possible hypothesis. In practice, assumptions must the made to restrict the search-space to a feasible size.

Given this class, one tries to find the hypothesis function $h_\theta \in \mathcal{H}$ which is optimal (most likely) for describing the data. This involves finding the function which minimizes (or maximises) some performance measure on the training data $D_{train} \subseteq D$ with the goal of generalizing to unseen data. These performance measures are defined as the training error and test error respectively.

The training error can be defined as

$$L_{training} = \frac{1}{|D_{train}|} \sum_{(x_i, y_i) \in D_{train}} (y_i - h_\theta(x_i)^2). \tag{5.3}$$

The test error can be defined as

$$\epsilon_{test} = \frac{1}{|D_{test}|} \sum_{(x_i, y_i) \in D_{test}} (y_i - h_\theta(x_i)^2). \tag{5.4}$$

If $D_{test} \sim p(X, Y)$ is drawn from the same distribution as the data D, then ϵ_{test} is an unbiased estimator of the generalization error, i.e.

$$\mathbb{E}(\epsilon_{test}) = \epsilon = \mathbb{E}_{(x,y)\sim p(X,Y)}[y - h_\theta(x)]. \tag{5.5}$$

The true ϵ is unknown, but, by the *weak law of large numbers*, its empirically estimated version ϵ_{test} converges to ϵ as $|D_{test}| \to \infty$.

The final objective of machine learning is to find

$$\hat{h}_{opt} = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{(x,y)\sim p(X,Y)}[y - h_\theta(x)] \approx \underset{\theta}{\operatorname{argmin}} \sum_{(x_i, y_i) \sim p(X,Y)} [y_i - h_\theta(x_i)]. \tag{5.6}$$

Consider now $\hat{h}_{opt}$ and let

$$\epsilon = 2\sqrt{\frac{\log(\frac{1}{\delta}) + \log(|\mathcal{H}|)}{2n}} \tag{5.7}$$

By the Chernoff bound

$$\Pr_{T\sim p(X,Y)}[|\frac{1}{n} \sum_{(x,y)\in T} (y - \hat{h}_{opt}(x))^2 - \mathbb{E}_{(x,y)\sim p(X,Y)}[(y - \hat{h}_{opt}(x))^2| \geq \epsilon] \leq e^{-2n\epsilon^2} \tag{5.8}$$

For simplicity write

$$\Pr[|err(h) - \hat{err}(h)| \geq \epsilon] \leq e^{-2n\epsilon^2} \tag{5.9}$$

By union bounds

$$\Pr[\exists h \in \mathcal{H} : err(h) - \hat{err}(h)| \geq \epsilon] \leq |\mathcal{H}| e^{-2n\epsilon^2} \tag{5.10}$$

Consider an arbitrary value δ so that

$$|\mathcal{H}|e^{-2n\epsilon^2} \leq \delta \tag{5.11}$$

which holds if

$$n \geq \frac{\ln 2|\mathcal{H}| + \frac{1}{\delta}}{2\epsilon^2} \tag{5.12}$$

so that, by conversion of the union bound,

$$h \in \mathcal{H} : \Pr[|err(h) - e\hat{r}r(h)| \leq \epsilon] \geq 1 - \delta \tag{5.13}$$

where

$$\epsilon = \sqrt{\frac{\ln 2|\mathcal{H}| + \frac{1}{\delta}}{2n}}. \tag{5.14}$$

This provides some solid mathematical guarantees on supervised learning. In essence, if the training data-size n is large enough and it comes from the same distribution $p(x, y)$ as the test-data, then the generalization error is bounded and tends towards zero with probability 1.

Two common algorithms for finding optimal parameters in a supervised machine learning setting are *Maximum Likelihood Estimation* (Fisherian) and *Maximum A-Posterior* (Bayesian). For relatively simple model parameterizations both minimize a convex function. Very popular are also neural network approaches, where the parameterization is performed by a neural net. The general approach is similar, with the main difference being in the model-parameterization, update scheme and objective function.

It instructive to approach supervised deep learning algorithms with neural nets by considering first these two types of machine learning algorithms.

5.1.2 unsupervised machine learning and information theory

The setting of unsupervised machine learning deals with discovering patterns in data which are previously not know. In essence, there is some Data

$$D = \{x_1, , ..., x_n\}$$

without connected observations or labels. Any algorithm which extracts patterns from label-less data belongs to unsupervised learning. These algorithms can not directly be used for predictions and classifications. Instead they aim to cluster the data-points into a specified amount of groups (Gaussian Mixture Model, K nearest neighbours), reduce dimensionality (Principal Components) and/or discover meaningful patterns (Apriori, Frequent Pattern, Eclat).

All unsupervised learning can be seen as an information representation problem. It involves finding the best (minimal) information representation for the relevant property. Available data is often noisy and contains irrelevant information. The extraction of all relevant information (and/or removal of irrelevant information) for the given task corresponds to finding an optimal representation. A common term for this is feature-extraction. This description, via features, is then be used, for example to make predictions or compress data. Finding this representations is often connected to dimensional augmentations. Kernel-method might construct additional polynomial dimensions to enable classification with linear methods, while other methods might use dimensionality reductions, such as PCA, to remove noise. It is no understatement to claim that finding suitable representations is a big, if not the most important, task of machine learning and often half the solution. Any representation need to contain all the relevant information for the required task, and possibly not more.

In contrast to supervised learning, unsupervised learning can not be defined in any simple

way which makes them much harder to evaluate. In fact, in many unsupervised learning tasks, the objective can not be formulated in a way representing the actual goal. As an example, consider the K nearest neighbours algorithm which aims to group data into k cluster centers. In the latter, the objective does not represent the actual goal. The goal is to discover *meaningful* patterns which reveal something about the data, not clustering into k groups. The discovered clustering might not always be useful. Meaningful clustering depends on the right number of groups. As An Impossibility Theorem for Clustering states, it is impossible to to give an axiomatically consistent definition of the "right" clustering.

However, for a deeper understanding, the problem of unsupervised learning can be related to compression theory. Both are about finding patterns in data and representing information in terms of these patterns. This link between the two allows to develop some intuitive concepts.

Consider the problem of compressing data $X = [X_1, X_2]$ with some compressor $C(X)$, then

$$|C(X_1, X_2)| \leq |C(X_1)| + |C(X_2)|.$$

In essence, the compression of $X = (X_1, X_2)$ is smaller than the compression of X_1 and X_2 summed. The gap between the two corresponds to the mutual information, information on X_2 which can be described by X_1 and vice versa.

Consider an optimal (purely theoretical and un-computable) compressor $K(X)$ of minimal length and maximal compression, then

$$|K(X)| \leq |C(X)| + |K(C)|,$$

where $K(C)$ corresponds to the theoretical information magnitude required to define the compressor K. Finding this optimal compressor is essentially the unsupervised learning problem. Some program space, defined by a parameterization, is searched for an optimal compression of the information. Clustering is a popular unsupervised learning approach, which essentially compresses information into a number of groups.

In Information theory, important concepts, related to infomration compression, are that of shannon entropy and mutual information.

Shannon entropy is a measure of the uncertainty or unpredictability of a random variable. For a discrete random variable X with possible outcomes $\{x_1, x_2, \ldots, x_n\}$ and a corresponding probability distribution $P(X = x_i) = p_i$, the Shannon entropy $H(X)$ is defined as:

$$H(p) = -\mathbb{E}_n p(x) \log p(x) \tag{5.15}$$

By the source coding theorem, an i.i.d random variable X with density $p(x)$, which occurs N times, can be compressed into $N * H(X)$ bits, where the log is computed with base two. As example, consider long sequences of k symbols, each representing a random variable with a distribution defined by the frequency of occurrence. Then the amount of bits to encode this sequence is $\sum_k N_k H(X_k)$.

In line with the Shannon Entropy, the relative-entropy is interesting. It is also also called Kullback-Leibler divergence and an ubiquitous quantity in machine learning, defined as

$$D_{KL}(p, q) = -\mathbb{E}_n 1(x) \log \frac{p(x)}{q(x)}. \tag{5.16}$$

The kullback-leibler divergence has a geometrical interpretation as the statistical divergence between two distributions. It is asymmetric and thus not a distance, but intimately related to the Fisher Information metric on Information manifold. Informally speaking, it is an quadratic form approximation of the geodesic distance between two distributions.

The cross-entropy unifies the shannon entropy and Kl-divergence and allows for another interpretation. It is defined as

$$H(p, q) = -\mathbb{E}_n p(x) \log q(x) \tag{5.17}$$

and measures the information required to identify a variable drawn from $q(x)$ if $p(x)$ is used to describe it. It can be related to the relative entropy and Shannon entropy via

$$H(p,q) = H(p) + D_{KL}(p,q). \tag{5.18}$$

Informally speaking, the amount of information required to describe p with q is the amount of information required to describe p plus the relative entropy $D_{KL}(p,q)$.

Another important quantity is Mutual Information (MI), which quantifies the amount of information obtained about one random variable through another random variable. It measures the reduction in uncertainty (entropy) about one variable when knowing the value of the other.

For two random variables X and Y, the mutual information $I(X;Y)$ is given by:

$$I(X;Y) = H(X) - H(X|Y). \tag{5.19}$$

The conditional entropy $H(X|Y)$ can be expressed via

$$H(X|Y) = H(X,Y) - H(Y) \tag{5.20}$$

Alternatively, mutual information can be expressed as

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right) = D_{KL}(P(x,y)||P(x)P(y)). \tag{5.21}$$

Mutual Information is often used to determine statistical similarities between data X and Y, for example in image registration.

Generally, one aims to find a description Y of the data X which reduces the entropy $H(Y)$ and maximizes the mutual information $I(X;Y)$.

Many works cite approach unsupervised learning from an information theoretic perspective.

5.1.3 semi-supervised machine learning

Semi-supervised machine learning combines both, supervised and unsupervised, methods, for example by reducing the dimensionality or grouping the data, before performing a classification or prediction. Examples may be clustering data to a subset of labeled data by discovering similarities, performing dimensionality reductions prior to a supervised learning setup or assigning high-confidence model predictions of unlabeled data-points as labels.

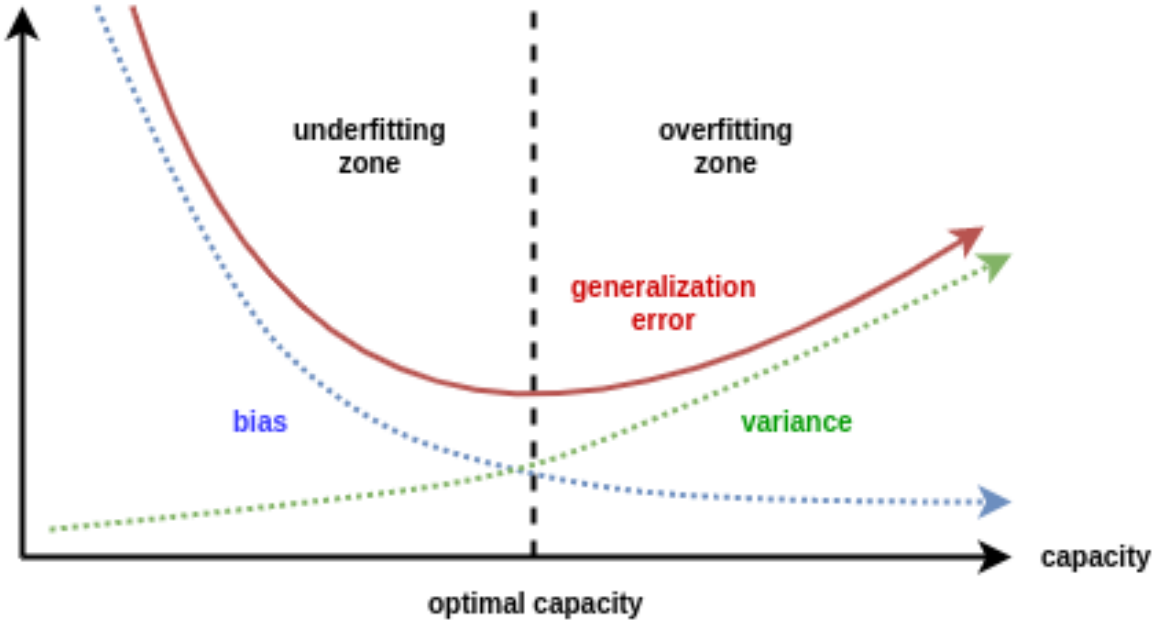
5.1.4 Limits: the Bias-Variance Trade-off

Assume a Dataset $D = \{(x_1,y_1),(x_2,y_2),...(x_n,y_n)\}$. Let $h_\theta(x)$ be a parameterization of a model, let $y(x)$ be the outcome observations corresponding to the data-sample $x \in D$, i.e. $y_1 = y(x_1)$. Let $\bar{y}(x) = \inf_y yp(y|x)dy$ be the expected label given some feature vector x. For noiseless data $y(x) = \bar{y}(x)$ Let

$$\hat{h}_\theta(x) = \mathbb{E}[h_\theta(x), D] = \int_D \int_x h_\theta(x)p(x|D_{train})dx\partial D_{train}$$

denote the expected model given the Data D, with $D_{train} \in D$. Here $p(x|D_{train})$ denotes the probability of drawing x given the data D_{train}, while $p(D_{train})$ denotes the probability of drawing the training-data D_{train} from D. Furthermore denote by $\mathbb{E}[(h_\theta(x) - y_s)^2]$ the expectation of the

Figure 5.1: The Bias Variance Tradeoff

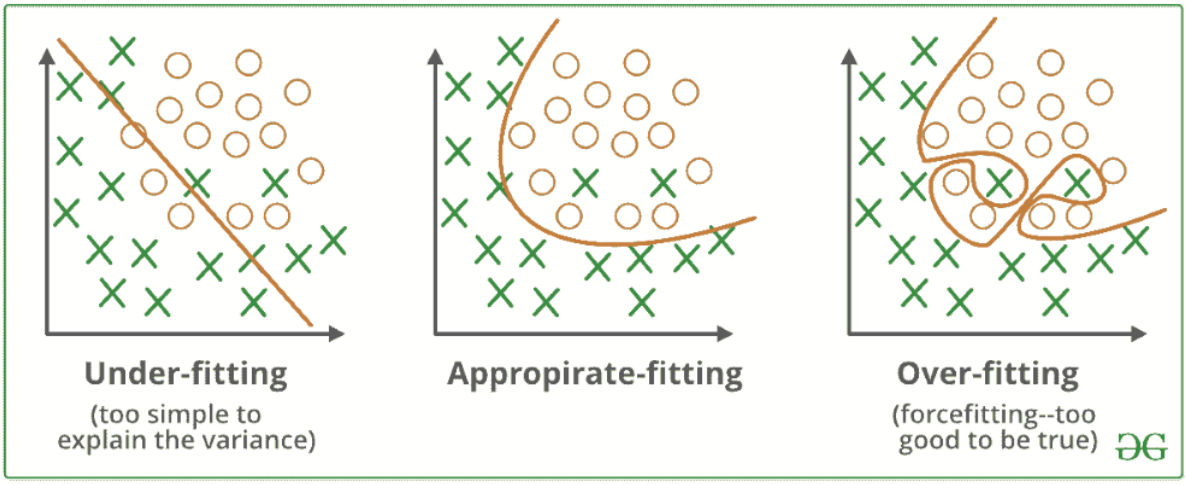


error given a sample $x \in D$ Then the generalization error E becomes

$$\begin{aligned}
\mathrm{E} &= \mathbb{E}[(h_\theta(x) - y)^2] \\
&= \mathbb{E}[h_\theta(x) - \hat{h}_\theta(x) + \hat{h}_\theta(x) - y)^2] \\
&= \mathbb{E}[(h_\theta(x) - \hat{h}_\theta(x))^2] + \mathbb{E}[\hat{h}_\theta(x) - y)^2] + 2\mathbb{E}[(h_\theta(x) - \hat{h}_\theta(x))(\hat{h}_\theta(x) - y)()] \\
&= \mathbb{E}[(h_\theta(x)) - \hat{h}_\theta(x))^2] + \mathbb{E}[(h_\theta(x)) - y] \\
&= \mathbb{E}[(h_\theta(x)) - \hat{h}_\theta(x))^2] + \mathbb{E}[\hat{h}_\theta(x) - \bar{y})^2] + \mathbb{E}[(\bar{y} - y)^2] + 2\mathbb{E}[(\hat{h}_\theta(x) - y)(y - y)] \\
&= \mathbb{E}[(h_\theta(x)) - \hat{h}_\theta(x))^2] + \mathbb{E}[\hat{h}_\theta(x) - \bar{y})^2] + \mathbb{E}[(\bar{y} - y)^2] \\
&= \underbrace{\mathbb{E}[(h_\theta(x)) - \hat{h}_\theta(x))^2]}_{\text{Variance}} + \underbrace{(\hat{h}_\theta(x) - \bar{y})^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[(\bar{y} - y)^2]}_{\text{Irreducible Error}}
\end{aligned} \tag{5.22}$$

This decomposition of the generalization error into three terms reflects the Bias-Variance trade-off in machine learning. Figure 5.1 displays the dependency of bias and variance on the model complexity. Figure 5.2 displays a classification that is overfitted vs underfitted to the data. When the model fits the training data well, i.e. the bias is low, the variance increases and vice versa. The variance captures how much the model would change if it were trained on a different training set. It reflects the degree to which the model is overspecialized. The Bias-Variance trade-off highlights, that while complex models can achieve better training errors, the simplest model generalizes best. This is in accordance with an information theoretical point of view and Occam's razor.

Figure 5.2: Overfitting and Underfitting



https://www.geeksforgeeks.org/regularization-in-machine-learning/

5.2 MLE - Maximum Likelihood Estimation

Consider a model, parameterized by θ, so that,

$$z = h(\theta) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and θ deterministic. Then $y \sim \mathcal{N}(h(\theta), \sigma^2)$.

The probability density function of this Gaussian is defined as

$$p(y, h(\theta)) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(y - h(\theta))^2}{2\sigma^2}. \tag{5.23}$$

Given data-samples (x_i) with $i = 1, ..., N$, Maximum Likelihood Estimation amounts to finding the model which results in the maximum joint probability of

$$\prod_{i=1}^{N} p(x_i | h(\theta)). \tag{5.24}$$

For $N = \infty$, the strong law of large numbers states

$$p(x|, h(\theta)) = \prod_{i=1}^{N} p(x_i, h(\theta))). \tag{5.25}$$

The Problem can be cast into an optimization problem:

$$\begin{aligned}
\theta_{opt}(x) &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p(x_i, h(\theta)) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \ln(P(x_i, h(\theta))) \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} \ln(P(x_i, h(\theta)))) \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} [\ln(\frac{1}{\sqrt{(2\pi\sigma^2)}}) + \frac{-(x_i - h(\theta))^2}{2\sigma^2}] \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} \frac{-(x_i - h(\theta))^2}{2\sigma^2} \\
&= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} (x_i - h(\theta))^2 * \text{const.}
\end{aligned}$$

Often const. $= \frac{1}{N}$. The $h_\theta(x)$, which minimizes this function, maximises $\prod_{i=1}^{N} p(y_i|h_\theta(x_i))$. Given a sufficient amount of data-samples, this approximation of the data-density is exact, assuming that a Gaussian parameterization is suitable. The proposed method uses data-samples (y_i, x_i) to extract the most likely parameterization of $y = h_\theta(x)$. Note, that

$$\sum_{=1}^{N} (x - h(\theta))^2 * \frac{1}{N}$$

is often referred to as *mean squared error*. Minimizing the mean squared error returns optimal model parameters if the data distribution is Gaussian. It can be proven that the estimator $h(\theta_{opt})$, which maximizes $p(x|h(\theta)$, is an unbiased minimum variance estimator (UMVE) for $x = h(\theta) + \epsilon$. Summarizing, Maximum Likelihood Estimation amounts to computing

$$\theta_{opt}(x) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} (x_i - h(\theta))^2 * \text{const.} \tag{5.26}$$

5.3 MAP - Maximum A-Posterior

Consider a model, parameterized by θ, so that,

$$y = h(\theta) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $\theta \sim \mathcal{N}(0, \tau^2)$. Contrary to MLE we assume a probabilistic model parameterization.

Then, by definition, the density function of this Gaussian can be written as

$$p(\theta) = \frac{1}{\sqrt{2\pi\tau^2}} \exp \frac{\theta^2}{2\tau^2}.$$

Furthermore, given a dataset x_i, note that

$$p(x, h(\theta)) = p(\theta|x_1, .., x_N)p(x_1, .., x_N) = p(x_1, .., x_N|\theta)p(\theta) = \prod_{i=1}^{N} [p(x_i|\theta)]p(\theta),$$

where $p(x_1, .., x_N) = \text{const.}$ is determined by the data.

Therefore

$$\begin{aligned}
\theta_{opt} &= \underset{\theta}{\operatorname{argmax}}\, p(\theta|x_1, .., x_N) \\
&= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} [p(x_i|\theta)] p(\theta) \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} [\ln(p(x_i|\theta)] - \ln(p(\theta)) \\
&= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} [\frac{-(y_i - h_\theta(x_i))^2}{2\sigma^2}] - \frac{\theta^2}{2\tau^2} \\
&= \underset{\theta}{\operatorname{argmin}} - \frac{1}{N} \sum_{i=1}^{N} [-(y_i - h_\theta(x_i))^2] - \lambda\theta^2,
\end{aligned}$$

with $\lambda = \frac{\sigma^2}{N\tau^2}$.

Note that the first term is equal to the mean squared error, as obtained during MLE. The second term in this objective function can be seen as a regularizer which constrains the model-parameters θ to be as small as possible. This is in accordance with Occam's razor. The Loss obtained through Maximum A-Posteriori coincides with a Maximum-Likelihood approach with regularization.

5.3.1 Connections between MLE and MAP

Assume a probability-distribution $p(x, \theta)$ as we have done in the MLE and MAP sections. Then, by the rules of probability

$$p(x, \theta) = p(x|\theta)p(\theta). \tag{5.27}$$

Since, in MLE we consider a fixed parameter θ, the term $p(\theta)$ is irrelevant and the MLE objective

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(x|\theta)$$

is equivalent to

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(x, \theta),$$

under the MLE assumption that $p(\theta)$ is uniform, i.e. all θ are equally likely.

Furthermore,

$$p(x, \theta) = p(x|\theta)p(\theta) = p(\theta|x)p(x). \tag{5.28}$$

Since $p(x)$ does not depend on θ, the MAP objective

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|x)$$

is also equivalent to

$$\theta_{opt} = \underset{\theta}{\operatorname{argmax}}\, p(x, \theta),$$

however, this time the parameters θ are also drawn from a non-uniform distribution $p(\theta)$ which is also optimized.

Note, that the loss-function resulting via MAP corresponds to the loss-function resulting via MLE with an additional term. This methodology which introduces additional penalty-terms to the MLE-loss is called model regularization. For example, one might penalize the square of the parameter-magnitude, so that the algorithm explores low parameter spaces more. This can

help to avoid over-fitting by balancing small training errors with complex (and strongly biased) models. Several methods exist, common ones are the Ridge regularization and the Lasso regularization. Maximum likelihood estimation with Ridge regularization is equivalent to Maximum A-posterior estimation, when the prior $p(\theta)$ is assumed to be a Gaussian. Regularization uses prior assumptions to derive terms which penalize deviations. As such, it is not surprising that MLE with regression is equivalent to MAP with an imposed prior. Penalizing model-complexity can help with over-fitting, penalizing distribution differences (as done with the KL-divergence) favours certain parameterizations.

Summarizing, note that MLE and MAP are different, although similar methodologies for maximizing the likelihood of $p(x, \theta)$ given some observed data x. The sections on MLE and MAP considered data-availability in pairs (x_i, y_i) but this is not a strict requirement on the methodologies.

5.4 Expectation Maximization

We presented two popular ways of looking at data-distributions and finding a corresponding model. Both methods aim towards maximizing $p(x, h(\theta)) = p(x, \theta)$ given the data $\{x\}$ and model $h(\theta)$ by minimizing a loss-function $L(\theta)$.

The difficulty of the MLE and MAP depends strongly on the chosen model. For simple models, such as linear models $h_(\theta) = \theta^T * x + b$, a closed-form solution for $\nabla L(\theta) = 0$ is available and the likelihood can be evaluated at the extrema to find the best. In fact, for linear models, the Loss-function is found to be convex, yielding one unique optimal point. For more complex models the loss-landscape might be non-convex and closed form solutions are often not available.

Furthermore, an additional complication is that $p(x, \theta)$ might involve a third, unmeasured variable z so that

$$p(x, \theta) = \int_z p(x, z, \theta) dz. \tag{5.29}$$

If $p(x, \theta)$ can be evaluated, this indicates that the integral is tractable.

In more complicated cases $p(x, \theta)$ can not be evaluated, indicating that the integral is not tractable.

One can also write the log-likelihood as

$$\begin{aligned}
\log(p(x, \theta)) &= \log \int_z (\frac{p(x, z, \theta)}{q(z)} q(z)) \\
&\geq \int_z q(z) \log \frac{p(x, z, \theta)}{q(z)} \\
&= \mathbb{E}_{z \sim q(z)} (\log \frac{p(x, z, \theta)}{q(z)}) \\
&= \mathbb{E}_{z \sim q(z)} (\log(p(x, z, \theta)) - \mathbb{E}_{z \sim q(z)} (\log q(z)) \\
&= \text{ELBO}(p(x, z, \theta), q(z))
\end{aligned} \tag{5.30}$$

where the inequality comes from the Jensen Inequality and $\mathbb{E}_{z \sim q(z)} (\log q(z)) = -H(q(z))$ is the negative entropy of $q(z)$. While the integral itself might not be tractable, the lower bound, defined by the ELBO often is.

Note, furthermore

$$\log(p(x, \theta)) - \text{ELBO}(p(x, z, \theta), q(z))$$

$$\log(p(x, \theta)) - \int q(z) \log \frac{p(x, z, \theta)}{q(z)} dz$$

$$= \int q(z) \log(p(x,\theta)))dz - \int q(z) \log \frac{p(z|x,\theta)p(x,\theta)}{q(z)} dz$$
$$= \int q(z) \log \frac{q(z)}{p(z|y,\theta)} dz$$
$$= D_{KL}(q(z), p(z|y,\theta))dz \tag{5.31}$$

Thus we have two relationships

$$\log(p(x,\theta) \geq \text{ELBO}(p(x,z,\theta), q(z)) = \mathbb{E}_{z\sim q(z)}(\log(p(x,z,\theta)) - \mathbb{E}_{z\sim q(z)}(\log(q(z)) \tag{5.32}$$

and

$$D_{KL}(q(z), p(z|x,\theta)) = \log(p(x,\theta)) - \text{ELBO}(p(x,z,\theta), q(z))dz \tag{5.33}$$

In the Expectation Maximization algorithm, one starts by minimizing the gap defined by equation 5.33 by finding the $q(z)$ closest to $p(z|y,\theta)$. In essence, by computing the expected distribution $q(z)$. This is equivalent to maximizing the ELBO with respect to z at a fixed θ_t. If $p(z|x,\theta_t)$ can be evaluated, then one can directly set

$$q(z) = p(z|x,\theta_t).$$

This step is called the Expectation-step because it requires finding the expectation of z given x and θ_t. Then one can compute the log-likelihood under $q(z) = p(z|x,\theta_t)$ (sometimes also called Q-function)

$$Q(\theta,\theta_t) = \mathbb{E}_{z\sim p(z|x,\theta_t)}[\log(p(x,z|\theta)] = \mathbb{E}_{z\sim q(z)}[\log(p(x,z|\theta)] \tag{5.34}$$
$$= \frac{1}{n}\sum_{i=1}^{n} \int_z p(z|x_i;\theta_t) \log p(x_i,z|\theta)dz = \frac{1}{n}\sum_{i=1}^{n} \int_z q(z) \log p(x_i,z|\theta)dz.$$

The Q-function is the cross-entropy between $p(z|x_i;\theta_t)$ and $p(x_i|z,\theta)$. It quantifies how well $p(x_i,z|\theta)$ describes $p(z|x_i;\theta_t)$ and is also the expected value of the log-likelihood function of θ, with respect to the current conditional distribution $p(z|x,\theta_t)$. In essence, it quantifies how well the data coincides if latent-samples are drawn from the current conditional, .

In the next step, one aims to find θ_{t+1} which maximizes $Q(\theta,\theta_t)$. This step is called the Maximization-step because one maximizes the likelihood of the model given the data x and (the estimated) z.

Repeated iterations of the Expectation and Maximization step are expected to converge. However, due to a sensitivity to starting parameters, not always to global optima. The EM-algorithm can also be used for MAP. The direct evaluation of $p(z|x,\theta_t)$ is not always possible. In the case of a VAE, the ELBO is the objective function to be maximized, however, since $p(z|y,\theta_t)$ can not be evaluated analytically, it is parameterized by a prior. In Gaussian Mixture Models $p(z|y,\theta_t)$ can be evaluated, allowing for direct iteration of the EM-algorithm.

5.4.1 From EM to Gaussian Mixture Models and VAE

Gaussian Mixture Models (GMMs) are a popular unsupervised learning model which can be optimized with the EM algorithm to find a mixture of K Gaussian's that describe the data

optimally. We will see that the problem can be cast into a distribution $p(x)$ which involves a third, unknown variable so that

$$p(x) = \int p(x, z)dz. \tag{5.35}$$

This is the same setting as for Expectation Maximization, making the EM-algorithm a popular (but not the only) choice for Gaussian Mixture Models.

Consider the probability that x is drawn from the k-th Gaussian as

$$p(x|\mu_k, \Sigma_k) = \frac{1}{\pi^{D/2}|\Sigma_k|^{0.5}} \exp(-\frac{1}{2}(x - \mu_k)\Sigma^{-1}(x - \mu_k)), \tag{5.36}$$

where $x, \mu_k \in \mathbb{R}^D$ and $\Sigma_k \in \mathbb{R}^{DxD}$. We can write

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\mu_k, \Sigma_k) = p(x|\mu, \Sigma, \pi) \tag{5.37}$$

where π_k is the probability that x is drawn from the k-th Gaussian and $\sum_{i=1}^{K} \pi_i = 1$. Equation 5.37 corresponds to the probability of drawing x from the mixture of Gaussians, with each Gaussian being weighted by the probability which quantifies how likely it is that x belongs to the k-th Gaussian generate. We want to maximize

$$\log p(x) = \sum_{i=1}^{N} \log p(x_i|\mu, \Sigma, \pi) = \sum_{i=1}^{N} \log \sum_{z_i=1}^{K} p(x_i|z_i, \mu, \Sigma)p(z_i|\pi). \tag{5.38}$$

Here we have introduced a hidden variable z_i which can take values between 1 and K and whose meaning is the association of each x_i with one of the K clusters. The sum within the log makes this difficult to optimize. It is due to the integration of the probability distribution of z_i. If we knew z_i for every i, then

$$\log p(x) = \sum_{i=1}^{N} \log p(x_i|z_i, \mu, \Sigma) + \log p(z_i|\pi) \tag{5.39}$$

and the optimal parameters for π_k, μ_k and Σ_k would be

$$\mu_k = \frac{\sum_{i=1}^{N} I_{z_i=k} x_i}{\sum_{i=1}^{N} I_{z_i=k}} \tag{5.40}$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} I_{z_i=k}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} I_{z_i=k}} \tag{5.41}$$

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} I_{z_i=k}, \tag{5.42}$$

where $I_{z_i=k} = 1$ if $z_i = k$ and zero else. Although it is not a-priori known which data-sample belongs to which Gaussian, note that

$$p(z_i = k|x_i) = \frac{p(x_i|z_i = k)p(z_i = k)}{\sum_{k'=1}^{K} p(x_i|z_i = k')p(z_i = k')} = \frac{p(x_i|z_i = k)\pi_k}{\sum_{k'=1}^{K} p(x_i|z_i = k')\pi_{k'}}, \tag{5.43}$$

can be evaluated given π and equation 5.36.

Denote by θ the parameter sets π, μ and Σ. Let us consider the likelihood again $\log p_\theta(x) =$

$\log \int p_\theta(x,z)dz$ which we would like to maximize but can not directly due to the integral in the log (in equation 5.39 this corresponds to the sum).

For intractable problems of this kind, we often construct the ELBO, as in equation 5.32,

$$\log p_\theta(x) = \log \int p_\theta(x,z)dz \geq \int q(z) \log \frac{p_\theta(x,z)}{q(z)} dz = \text{ELBO}(q(z), p_\theta(x,z)).$$

Equivalently, we also obtain the gap (see equation 5.33).

$$\log(p_\theta(x)) = \text{ELBO}(p_\theta(x,z), q(z)) - D_{KL}(q(z), p_\theta(z|x))dz$$

The EM-algorithm starts by minimizing the gap from equation 5.33 by setting $q(z) = p_\theta(z|x)$ at fixed $\theta = \theta_t$. This is equivalent to maximizing the ELBO with respect to z. Since

$$\gamma_k^i = p(z_i = k|x_i, \theta_t)$$

can be evaluated via equation 5.43 this can be done analytically. This step is called the Expectation-step. It calculates the expected values for z_i. Each x_i to the k-th Gaussians with probability γ_k^i. This is similar to the assignment step in k-means.

Once we know $p(z_i = k|x_i, \theta_t)$ we can compute the log-likelihood (or Q-function from equation 5.34) via

$$\begin{aligned}
Q(\theta, \theta_t) &= \mathbb{E}_{z\sim p(z|x,\theta_t)}[\log(p(x,z|\theta)] \\
&= \mathbb{E}_{z\sim q(z)}[\log(p(x,z|\theta)] \\
&= \frac{1}{N}\sum_{i=1}^{N} \int_z p(z|x_i;\theta_t) \log p(x_i,z|\theta)dz. \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} p(z_i = k|x_i;\theta_t) \log p(x_i, z_i = k|\theta). \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_k^i \log(p(x_i|\mu_k,\Sigma_k)p(z_i = k|\theta)). \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_k^i \log p(x_i|\mu_k,\Sigma_k) + \gamma_k^i \log \pi_k.
\end{aligned}$$

This can be maximized analytically to

$$\mu_k = \frac{\sum_{i=1}^N \gamma_k^i x_i}{N_k} \tag{5.44}$$

$$\Sigma_k = \frac{\sum_{i=1} \gamma_k^i (x_i - \mu_k)(x_i - \mu_k)^T}{N_k} \tag{5.45}$$

$$\pi_k = \frac{N_k}{N} \tag{5.46}$$

where $N_k = \sum_{i=1}^N \gamma_k^i$ and $\theta_t = (\mu_t, \Sigma_t)$.

The Maximization step, maximizes the log-likelihood from equation 5.34 with respect to $\theta = (\mu, \Sigma)$. This is equivalent to maximizing the ELBO with respect to θ. As such, the Gaussian Mixture Model can be optimized with a particular case of the EM-algorithm where $p(z|x,\theta)$ can be evaluated, allowing for an analytical minimization of $D_{KL}(q(z), p(z|x,\theta))$, which, by equation

5.33, is equivalent to maximizing the ELBO with respect to z. The unobserved variables z correspond to the assignment of data-points x to one of the K-clusters. It determines from which cluster the data-sample x is drawn. The Gaussian Mixture is in the same dimensional space as x.

In a variational auto-encoder setting, no expression for $p(z|x,\theta)$ can be evaluated, so no analytical minimization of 5.33 is possible by setting $q(z) = p(z|x,\theta)$. Instead, one assumes a prior parametric description $p(z|x,\theta)$ and maximizes the ELBO with numerical gradient methods. Often this prior distribution is the isotropic Gaussian. However, for some data this can be too limiting. However, the procedure is similar in the sense that the same type of objective function is maximized, however with gradient descent methods instead of explicit iterative expectation maximization.

5.5 VAEs

Variational auto-encoders are an extension of auto-encoders. The latter considers a function

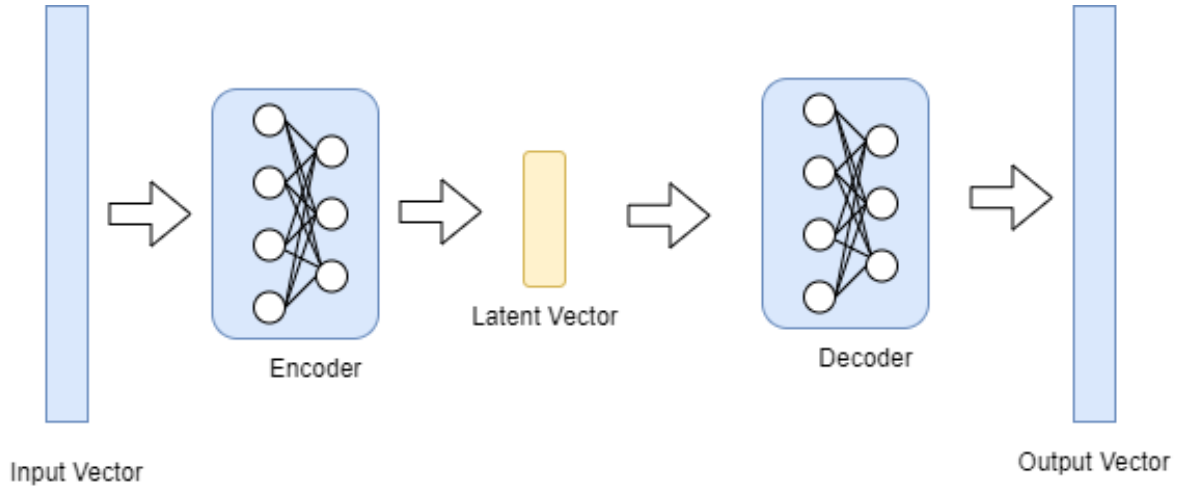
$$z = g_\phi(x)$$

(the Encoder. parameterized by a neural network) to compress data and another function

$$x = f_\theta(z)$$

(the Decoder. parameterized by a neural network) to decompress data. The dimensional space into which data-samples are mapped is called the latent space. Usually a dimensionality reduction is performed. Encoder and Decoder are trained so that the encodings lead to decodings

Figure 5.3: An Auto-Encoder



which are as close to the originals as possible. The information in the input vector is compressed into the latent space and retrieved by the decoder. Model-optimization is performed via back-propagation, so that the reconstruction loss

$$\sum_{i=1}^{N} x_i - f_\theta(z_i) = \sum_{i=1}^{N} x_i - f_\theta(g_\phi(x_i)) \tag{5.47}$$

is maximized. Here x_i corresponds to the original data-sample and z_i to the corresponding low-dimensional encoding. Thus the auto-encoder enables the definition of a mapping from latent-space to output space and vice-versa. The latent-space is unrestricted and the mapping

deterministic.

The main difference between an auto-encoder and the variational auto-encoder, is that the latter considers the latent vector z as coming from an imposed prior distribution with density $p(z)$, leading to variational inference. This is advantageous, because in the unrestricted auto-encoder case, the encoding z can have arbitrary locations. Values close to each other can have totally different reconstructions and some values might not correspond to any sensible decoding. In the variational auto-encoder this problem is reduced by constraining the latent-space to a distribution. Therefore, instead of learning a set of numerical latent representations, one learns a representation in form of a distribution, thereby creating a continuous latent-space. Each value has a certain probability of occurring during training and is mapped to the best corresponding decoding. Often, a prior distribution $p(z)$ is assumed as an isotropic Gaussian. One aims to learn a mapping from the output space to the prior distribution with the encoder $z = g_\phi(x)$, implicitly defining $q_\phi(z|x)$. Consider

$$x = f_\theta(z)$$

with $z \sim \mathcal{N}(0, I)$. Then

$$p_\theta(x|z) \sim \mathcal{N}(f_\theta(z), I)$$

and

$$p_\theta(x) = \int p_\theta(x, z)dz \geq \int_z q_\phi(z|x) \log \frac{p_\theta(x, z)}{q_\phi(z|x)} dz = \text{ELBO}(p_\theta(x, z), q_\phi(z|x)).$$

Note, that

$$\begin{aligned}
\text{ELBO}(p_\theta(x, z), q_\phi(z|x)) &= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z|x)) - \mathbb{E}_{z\sim q_\phi(z|x)}(\log(q_\phi(z|x)) \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z|x)p_\theta(z)) - \mathbb{E}_{z\sim q_\phi(z|x)}(\log(q_\phi(z|x)) \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z|x)) + \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(z))) - \mathbb{E}_{z\sim q_\phi(z|x)}(\log(q_\phi(z|x)) \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(x|z)) - \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p_\theta(z)),
\end{aligned} \tag{5.48}$$

which is the common expression for the objective employed in VAE. Usually $p_\theta(z)$ can not be evaluated directly. Instead it is parameterized via prior assumptions (often as isotropic Gaussian) and $q_\phi(z|x)$ is updated with gradient methods. We aim to learn $q_\phi(z|x)$ with the Encoder and $p_\theta(x|z)$ with the Decoder.

The objective is

$$\mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(x|z)) - \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z)), \tag{5.49}$$

where $p(z)$ is a prior distribution and $q_\phi(z|x)$ and $p_\theta(x|z)$ are parameterized by neural networks.

While in Gaussian Mixture Models $\mathbb{E}_{z\sim q(z)}(\log(p_\theta(x|z))$ can be evaluated directly, for a VAE this is not always the case, so instead Monte-Carlo Approximations might be used, so that

$$\mathbb{E}_{z\sim q_\phi(z|x)} \log(p_\theta(x|z) \approx \sum_{i=0}^{N} \log(p_\theta(x|z_i)). \tag{5.50}$$

where $z_i \sim q_\phi(z|x)$ and $p_\theta(x|z_i) =$. If one assumes $p_\theta(x|z) = \mathcal{N}(x; \mu(z), \sigma(z)^2 I)$, then the maximum likelihood estimator θ_{opt} is given by (see section 5.2)

$$\theta_{opt} = \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} (x_i - f_\theta(z_i))^2.$$

If the data distribution does not align well with the Gaussian assumption, then other choices should be used as objective function.

If the encoder perfectly maps different inputs x to the isotropic gaussian, so that $q_\phi(z|x) = \mathcal{N}(0, I)$, all information is lost since different inputs x lead to the same distribution of z. This event, which happens when the $\mathrm{D}_{\mathrm{KL}}(x)$-term is zero for multiple inputs x, is also called mode-collapse, and is undesirable. Instead, what is desired during training is that $\sum_{i=0}^{N} q_\phi(z|x_i)$ is a multi-modal distribution of several overlapping distributions (as many as training examples, each with different means and variances). The overlap is enforced by the regularizing term $\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z))$, which encourages $q_\phi(z|x_i)$ to follow the prior $p(z)$ for all x_i. This regularizing term is weighted by a hyper-parameter β enabling deviations and leading to the objective

$$\mathbb{E}_{z\sim q_\phi(z|x)}(\log(p_\theta(x|z)) - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x), p(z)). \tag{5.51}$$

Assuming that the latent-space is a distribution from which values are sampled and decoded optimally during training creates a latent-space with smooth transitions. If the prior $p(x)$ is the isotropic Gaussian, the actual encoder distribution $q_\phi(z|x_i)$ should deviate from $p(z)$ for all, but one sample x_i. Choosing the right prior is non-trivial but very important. The wrong prior can destroy all applicability of the VAE.

For neural network optimization the loss-function must be differentiable. However, if we consider equation 5.52 with the mean squared error loss, we obtain

$$\sum_{i=1}^{N}(x_i - f_\theta(z_i))^2 - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(z|x_i), p(z)), \tag{5.52}$$

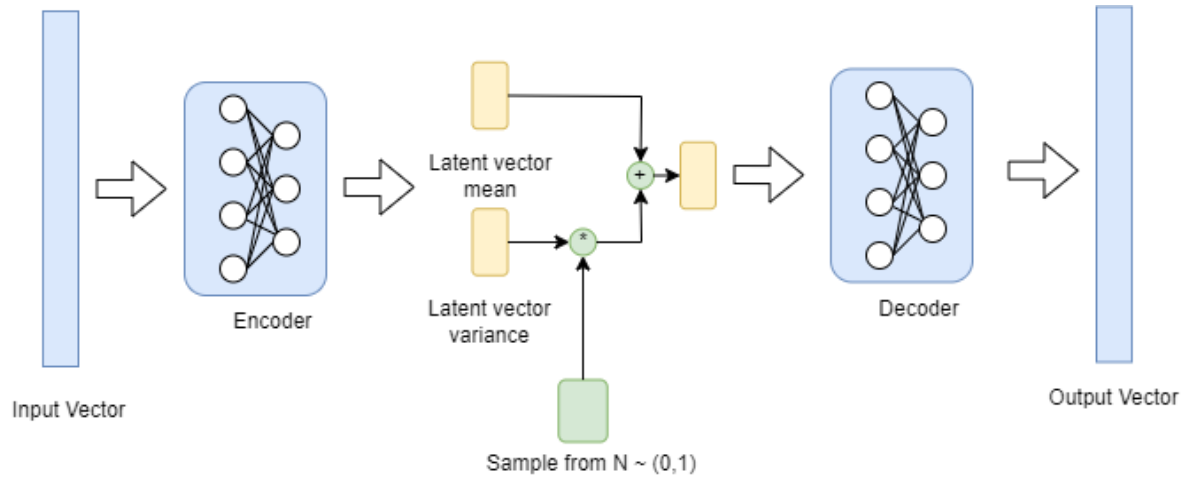
where z is a random variable and thus not directly differentiable. This is solved with a re-parametrization. Consider the Encoder $z = g_\phi(x)$. We want z to be a Gaussian with mean $\mu(x) = g_{\mu,\phi}(x)$ and covariance $\Sigma(x) = g_{\mu,\phi}(x)$, so that the decoder $f_\theta(g_\phi(x))$ remains differentiable with respect to θ. Consider

$$z = \mu(x) + \Sigma(x)\epsilon \sim \mathcal{N}(\mu(x), \Sigma(x)),$$

if $\epsilon \sim \mathcal{N}(0), I)$. Due to this construction, also called the reparameterization trick (it also exists for other distributions) we can sample a random variable in differentiable way with respect to $\mu(x)$ and $\Sigma(x)$. Often (but not always), $\Sigma(x) = \mu(x)I$ is chosen to be a diagonal covariance matrix. This encourages dis-entangled latent features with orthogonal directions.

VAE's have several drawbacks. Among those is a blurry reconstruction ability since high resolution information gets lost in the compression. Other issues are uninformative latent spaces due to posterior-collapse, a phenomenon in which some latent dimensions are filled with random noise and ignored completely during generation, as well as unrealistic data distributions, when the prior does not fit the data well. Furthermore, there exists a trade-off between good generation and good reconstruction. Generally, an increased KL-divergence loss causes a greater regularization of the latent space and therefore allows for a greater variety of "realistic" samples, however it can come at the cost of efficient compression, leading to a lower reconstruction quality. Additionally, the isotropic Gaussian prior can be too limiting for many problems and it can be difficult to find suitable priors.

Figure 5.4: The Variational Auto-Encoder



5.6 PCA

Principal Component Analysis is a popular linear algorithm for systems of the form

$$x = f(z) = Wz \tag{5.53}$$

or, in the probabilistic PCA setting

$$x = f(z) + \epsilon = Wz + \sigma^2\epsilon, \tag{5.54}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and $z \in \mathcal{N}(0, I)$ so that

$$p(x|z) = \mathcal{N}(Wz, WW^T) \text{ and } p(x) = \mathcal{N}(Wz, WW^T)$$

in the former case and

$$p(x|z) = \mathcal{N}(Wz, WW^T) \text{ and } p(x) = \mathcal{N}(Wz, WW^T + \sigma^2 I)$$

in the latter.

PCA can be motivated from a variety of directions. For simplicity, let us start with equation 5.53 and extend it to 5.54.

Consider white Gaussian noise $z \sim \mathcal{N}(0, I)$ and a mean-centered data-distribution $p(x) = \mathcal{N}(0, \Sigma)$. Let $x, z \in \mathcal{R}^n$. Given the data-set matrix $X = (x_1, ..., x_N) \in \mathcal{R}^{N\times n}$, we can construct the singular value decomposition

$$X = U\Delta V^T, \tag{5.55}$$

with $U \in \mathcal{R}^N$, $\Delta \in \mathcal{R}^{N\times N}$ and $V \in \mathcal{R}^n$ (in the complex case we need to consider the conjugate transpose). We can construct the linear covariance matrices

$$XX^T = V\Delta^2 V^T = VDV^T \in \mathcal{R}^{n\times n}$$

and

$$X^T X = U\Delta^2 U^T = UDU^T \in \mathcal{R}^{N\times N}.$$

In the real case U and V are guaranteed to be real orthogonal matrices so that $\text{rank}(X^T X) = \text{rank}(XX^T)$. Both covariance matrices have different dimensionality but the same non-zero eigenvalues and equal rank. This equivalent nature of the two covariance matrices and their

corresponding distributions has deep consequences in machine learning (especially for Gaussian Processes) and is referred to as the data-space and feature-space duality.

Consider the question, which linear transformation of $z \sim \mathcal{N}(0, I)$ yields the distribution $p(x) = \mathcal{N}(0, XX^T)$. The answer is

$$x = Wz = UD^{1/2}z \tag{5.56}$$

as becomes obvious from equation 5.53. Note that U is the matrix of eigenvectors of the linear covariance matrix XX^T and that $D^{1/2}$ is the square-root of its eigenvalue matrix. Thus the linear transformation relating the data-distribution to white noise is defined by eigen-directions and eigenvalues of the covariance matrix. The eigenvalues are called the principal components, the eigenvectors are called principal directions.

The main application of PCA is in dimensionality reduction. Until now a dimensionality preserving linear map waas derived. For dimensionality reduction consider a system as in equation 5.54 with $x \in \mathbb{R}^n$, $z \in \mathbb{R}^d$ and consequently $W \in \mathbb{R}^{n \times d}$ with $d < n$. Consider the distribution $\mathcal{N}(0, WW^T)$ and note that WW^T has rank d but dimension n. Such cases are called singular distributions (some eigenvalues are zero). Singularity indicates the existence of directions in which no variability occurs. Given a dataset X, we can perform the PCA decomposition and yield equation 5.56. By keeping only the d biggest eigenvectors ($D_- \in \mathcal{R}^{d \times d}$) and corresponding eigenvalues ($U_- \in \mathcal{R}^{n \times d}$) , a dimensionality reduction is performed. Note, that then

$$W = U_- D_-^{1/2}$$

$$\Sigma = WW^T = U_- D_- U^T \in \mathbb{R}^{n \times n}$$

is a singular matrix with rank d. Along the directions corresponding to the omitted eigenvectors no variability occurs. Since singular distributions are hard to work with, a common practice is adding noise. In essence, we consider

$$x = Wz + \sigma^2 \epsilon$$

with $W \in \mathbb{R}^{n \times d}$, $z \in \mathbb{R}^d$ and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. From this equation, which corresponds to the probabilistic PCA setting (see equation 5.54), we obtain

$$p(x|z) \sim \mathcal{N}(Wz, WW^T)$$

leading to

$$p(x) \sim \mathcal{N}(Wz, \Sigma)$$

with with $\Sigma = WW^T + \sigma^2 I \approx XX^T$. Consider the mean centered version $p(x) \sim \mathcal{N}0, \Sigma)$ and perform MLE,

$$\Sigma_{opt} = \text{argmax}_\Sigma \prod_{i=1}^{N} \log(\frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp \frac{-x_i \Sigma^{-1} x_i^T}{2}$$

$$= \text{argmin}_\Sigma \sum i = 1^N \log(\frac{1}{\sqrt{(2\pi)^n |\Sigma|}}) + \frac{-x_i \Sigma^{-1} x_i^T}{2}$$

$$= \text{argmin}_\Sigma -\frac{N}{2} \log((2\pi)^n) - \frac{N}{2} \log(|\Sigma|) - \frac{1}{2} \sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T$$

$$= \text{argmin}_\Sigma -\frac{N}{2} \log(|\Sigma|) - \frac{1}{2} \sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T$$

$$= \text{argmin}_\Sigma -\frac{N}{2}\log(|\Sigma|) - \frac{1}{2}\sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T. \tag{5.57}$$

Consider

$$\frac{\partial}{\partial \Sigma} - \frac{N}{2}\log(|\Sigma|) = -\frac{N}{2}\Sigma^{-1}$$

and

$$\frac{\partial}{\partial \Sigma} - \frac{1}{2}\sum_{i=1}^{N} x_i \Sigma^{-1} x_i^T = +\frac{1}{2}\Sigma^{-1}\sum_{i=1}^{N} x_i x_i^T \Sigma^{-1}.$$

so that the extremum is reached if

$$\frac{N}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}\sum_{i=1}^{N} x_i^T x_i \Sigma^{-1} = 0.$$

Consider $\sum_{i=1}^{N} x_i x_i^T = XX^T = UDU^T$, according to the singular value decomposition (see equation 5.55) so that

$$\frac{N}{2}\Sigma^{-1} + \frac{N}{2}\Sigma^{-1} U D U^T \Sigma^{-1} = 0.$$

and thus

$$\frac{N}{2} + \frac{N}{2}\Sigma^{-1} U D U^T = 0 \rightarrow U D U^T = \Sigma$$

We only wan to keep the d biggest eigenvalues and eigen-directions so that

$$WW^T + \sigma^2 I \approx \Sigma = UDU^T \approx U_- D_- U_-^T + \sigma^2 I.$$

Let, $W = U_-(D_- - \sigma^2 I)^{1/2}$ so that

$$WW^T = U_- D_- U_-^T - \sigma^2 I \approx \Sigma.$$

D_- should contain the d biggest eigenvalues and $\sigma = \frac{1}{n-d}\sum_{j=d+1}^{n} \lambda_j$. Thus, the dimensionality reduction results in

$$x = U_-(D_- - \sigma^2 I)^{1/2} z + \sigma^2 \epsilon.$$

If we omit ϵ, this corresponds to a linear projection of the n dimensional data X onto a subspace of d dimension.

We have shown how PCA can be considered a linear transformation of white noise to a Gaussian distribution. Thus given a data-distribution we can find the transformation W which produces our data given white noise as an input. We have also extended this point of view to a dimensionality reduction setting. Any linear transformation of d-dimensional white noise creates a d dimensional subspace in the n-dimensional feature space. Considering noise contributions in its orthogonal directions solves the this issue and defines a regular distribution in the whole space. We have shown how solving for the optimal transformation and noise parameters traces back to the eigen decomposition of the linear covariance matrix defined by X. Essentially, one keeps the d biggest directions and eigenvalues (which have the biggest contribution to the covariance matrix) and uses the others to construct the noise parameters. For this we have performed maximum likelihood estimation with a parameterized Σ matrix.

We have also hinted at an equivalence between the matrices XX^T and $X^T X$. While living in different spaces, the image of both matrices has equal dimensions. In some sense, both distributions are equivalent.

PCA is a very well understood and popular method. Being completely linear allows for closed

form solutions, good interpretability and analysis. Linearity is at the same time its strength and main limitation. As a linear transformation of white-noise we can not expect to create a model for nonlinear distributions. It has been shown paper that the variational auto-encoder with a diagonally parameterized covariance matrix pursues the principal directions of PCA. Given these results, we might consider PCA as the linear alternative of VAEs and VAEs as its nonlinear counterpart. If our data-distribution is indeed linear, both yield the same results. If our data-distribution is non-linear, PCA will fail to capture it while the VAE still can.

5.6.1 Extension to Gaussian Processes

Note, that the model

$$x = Wz + \epsilon$$

leads to

$$p(x) = \int p(x|z)p(z)dz$$

since z is white noise. If $p(z)$ is Gaussian, then so are $p(x|z)$ and $p(x)$. Remember that XX^T and X^TX define equivalent distributions in different dimensions. Consider now an alternative model.

$$x = W\Phi(z) + \epsilon, \tag{5.58}$$

where $\Phi(z)$ is a kernel (some map into higher, potentially infinite dimensions), z deterministic and $w \sim \mathcal{N}(0, I)$. This leads to

$$p(x) = \int p(x|w)p(w)dw.$$

If $p(w)$ is Gaussian, then so are $p(x|w)$ and $p(x)$. Assume one is interested in

$$p(x) \sim \mathcal{N}(W\Phi(z), \Sigma),$$

where

$$\Sigma = WW^T + \sigma^2 I$$

and has available a set of N data-samples (x_i, z_i). This direct consideration of $p(x)$ as a Gaussian distribution without any concretizing $\Phi(x)$ or W is what is referred to as an integration over the space of all functions. However, the covariance matrix still needs to be parametrized, which limits this space of all functions to more concrete, but still very broad types. In PCA, the prior was over z and we would just consider $\Sigma \approx XX^T \in \mathcal{R}^{n\times n}$. We can do this because

$$\text{Var}(x) = \sum_{i=1}^{N} \mathbb{E}(Wz_i - \mu)\mathbb{E}(Wz_i - \mu)^T = \sum_{i=1}^{N} \mathbb{E}(Wz_i)\mathbb{E}(Wz_i)^T = W\sum_{i=1}^{N} \mathbb{E}(z_i)\mathbb{E}(z_i)^T W^T = WW^T.$$

However, for equation 5.58 we can not do this since the map $\Phi(z)$ is not known (not even its dimensionality). We have

$$\text{Var}(x) = \sum_{i=1}^{N} \mathbb{E}(W\Phi(z_i))\mathbb{E}(W\Phi(z_i))^T = \sum_{i=1}^{N} \Phi(z_i)\mathbb{E}(W)\mathbb{E}(W)^T\Phi(z_i)^T$$

A way of alleviating the issue of an unknown covariance matrix with unspecified feature dimensions, is by considering the covariance $\Sigma = X^TX \in \mathcal{R}^{N\times N}$ which is limited in dimensionality only by the amount of data-samples, but defines a distribution in the feature-space dimensions, as illustrated o the linear case. One chooses a non-linear parameterization of the kernel covariance matrix $\Sigma = K(Z, Z)$, such as the radial basis function kernel. This limits the form of corresponding functions $\Phi(z)$. The maximum likelihood estimation from equation 5.57 with

respect to the kernel parameters can be performed for non-linearly parameterized matrices. For some kernel covariances it can the solved analytically. This is the concept behind Gaussian processes. Given the covariance matrix which maximizes our likelihood

$$p(x) = \int p(x|w)p(w)dw = \mathcal{N}(0, K),$$

we can consider

$$\begin{bmatrix} p(x) \\ p(x^*) \end{bmatrix} = \mathcal{N}(\begin{bmatrix} \mu_X, \mu_x^* \end{bmatrix}, \begin{bmatrix} K, K(Z, z^*) \\ K(z^*, Z), k(z^*, z^*) \end{bmatrix}. \tag{5.59}$$

The conditional mean and variance of a multi-variate Gaussian is

$$\mathbb{E}[x^*|X] = \mu_X + K(z^*, Z)K^{-1}(X) \tag{5.60}$$

and

$$\text{Var}(x^*|X) = k(z^*, z^*) - K(z^*, Z)K^{-1}K(Z, z^*). \tag{5.61}$$

Usually we consider mean centered data, so that $\mu_X = 0$. Note, that in this framework, we assume an available data-set of pairs x_i, z_i and z_i is not associated to a prior distribution.

In an alternative setting, the data consists only of x_i and the maximum likelihood is optimized also with respect to z_i. These type of models are called Gaussian process latent variable models. The dimension of z is a hyperparameter, making this model suitable for dimensionality reduction. Similar to the auto-encoder, the latent-space is unrestricted, so that smoothness and continuity of the latent distribution is not necessarily given. Similar to the variational auto-encoder, a natural extension to bayesian variational gaussian process latent variable models exists. However, within the community, Gaussian Processes are used mainly for regression tasks.

5.7 Summary

We have examined machine learning basics in section **??**, extended the log-likelihood to models with unobserved variables for which we derived a lower bound. We show how one can maximize this bound iteratively in the case of Gaussian Mixture Models and how it is maximized with gradient methods in the case of variational auto-encoders.

We then examined linear models where the unobserved variable follows an isotropic Gaussian and an empirical data-distribution is available. The results are that for linear models, a linear covariance matrix can be constructed in the feature space, and the linear transformation is found by an eigen-space decomposition. We show how the feature-space is equivalent to the data-space in the linear case. This eigen-decomposition is extended to dimensionality reduction techniques, where the unobserved variable distribution has lower dimensions. We show that the linear transformation defines a singular distribution in the higher dimensional space defining a linear projection of the data onto a subspace. The singularity issue is alleviated by adding random noise. Then we solve for the optimal noise-variance and linear transformation by maximum likelihood estimation with respect to the covariance parameters.

We extent the maximum likelihood estimation with respect to the covariance parameters by considering linear models with a (potentially infinite dimensional) kernel $\Phi(x)$. Considering a prior distribution offer the model-parameters instead of the unobserved variable, we examine how the data-space and feature-space duality can be used to directly optimize for the log-likelihood, integrating over all possible models. We provide an intuition connecting non-linear

covariance parametrization to the kernel. Furthermore, we extend the kernel framework to latent variable models where the unobserved variable is part of the optimizable model parameters and hint towards variational latent variable models, where a prior is assumed over the model parameters and the unobserved variables.

5.8 The ELBO

The lower bound obtained in equation 5.30 is an ubiquitous quantity in machine learning, especially in variational inference. It is often used as optimization objective when equation **??** is not directly tractable. Summarizing the results from paper, we examine its decomposition into meaningful mathematical formulas. This provides insights into the components which are considered in a optimization process.

We show the forms

$$\log(p(x,\theta) \geq \text{ELBO}(p(x,z,\theta),q(z)) = \mathbb{E}_{z\sim q(z)}(\log(p(x,z,\theta)) - \mathbb{E}_{z\sim q(z)}(\log(q(z))$$

in equation **??** and

$$\text{ELBO}(p(x,z,\theta),q(z)) = \log(p(x,\theta)) - D_{KL}(q(z),p(z|x,\theta)dz$$

via equation **??**.

A reformulation of the ELBO can introduce the mutual information explicitly. Let us decompose the formulation of equation 5.48 into its empirical counterpart and use $q(z|x)$ instead of only $q(z)$. This is possible since the derivation of the ELBO does not limit our choice of $q(z)$ as long as it is a distribution of the hidden variable z. We will also remove the explicit parameters θ to simplify notation. We write

$$\text{ELBO}(p(x,z),q(z|x)) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{z_n\sim q(z_n|x_n)}(\log(p(x_n|z_n)) - \text{D}_{\text{KL}}(q(z_n|x_n)),p(z_n),$$

where we have n observations of data-samples x_n and latent-samples z_n respectively. Each observation is made with equal probability so that we say $q(x_n) = p(x_n) = \frac{1}{N}$. Furthermore, we know that $p(z)$ does not depend on the observations since it is the imposed prior, i.e. $p(z|x_n) = p(z)$. However, $q(z_n|x_n)$ does, since it corresponds to the learned map from input-space to latent-space. Examine

$$\text{D}_{\text{KL}}(q(z_n|x_n)),p(z_n)) = \frac{1}{N}\sum_{n=1}^{N}q(z_n|x_n)\frac{q(z_n|x_n)}{p(z_n))} = \sum_{n=1}^{N}q(z_n,x_n)\log\frac{q(z_n,x_n)}{p(z_n)p(x_n)}$$

$$= \sum_{n=1}^{N}q(z_n|x_n)q(x_n)(\log\frac{q(x_n|z_n)q(z_n)}{p(z_n)p(x_n)}) = \sum_{n=1}^{N}q(z_n|x_n)q(x_n)(\log\frac{q(z_n)}{p(z_n)} + \log\frac{q(x_n|z_n)}{p(x_n)})$$

$$= \text{D}_{\text{KL}}(q(z_n),p(z_n)) + \sum_{n=1}^{N}q(x_n|z_n)q(z_n)\log q(x_n|z_n) - \sum_{n=1}^{N}q(x_n|z_n)q(z_n)\log p(x_n)$$

$$= \text{D}_{\text{KL}}(q(z_n),p(z_n)) + \sum_{n=1}^{N}q(x_n,z_n)\log q(x_n|z_n) - \sum_{n=1}^{N}p(x_n)\log p(x_n).$$

$$= \text{D}_{\text{KL}}(q(z_n),p(z_n)) - H(q(x_n|z_n)) + H(q(x_n)).$$

$$= \text{D}_{\text{KL}}(q(z_n),p(z_n)) + I_q(x_n,z_n).$$

Here the second term is the negative of the conditional entropy $H(q(x_n|z))$, while the third term is the entropy if $H(p(x_n)) = H(q(x_n)) = \log N$. The mutual information of $q(x_n)$ and $q(z_n)$ is $I(x_n, z_n) = H(q(x_n)) - H(q(x_n|z))$ Thus we can write

$$\text{ELBO}(p(x,z), q(z|x)) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{z_n \sim q(z_n|x_n)}(\log(p(x_n|z_n)) - \text{D}_{\text{KL}}(q(z_n), p(z_n)) - I_q(x_n, z_n). \tag{5.62}$$

Through this composition, it becomes apparent, how maximizing the ELBO is related to maximizing the mutual information between x_n and z_n.

Further decomposition into a mutual information $I_q(x,z)$, an entropy $H(q(z))$ and cross-entropy $H(q(z), p(z))$ term is possible. Consider

$$\begin{aligned}\text{ELBO}(p(x,z), q(z|x)) &= \mathbb{E}_{z\sim q(z|x)}(\log(p(x|z)) - \text{D}_{\text{KL}}(q(z), p(z)) - I(x,z) \\ &= \mathbb{E}_{z\sim q(z|x)}(\log(p(x|z)) - I_q(x,z) + H(q(z)) - H(q(z), p(z))\end{aligned} \tag{5.63}$$

Note, that the prior $p(z)$ appears only in the last term, highlighting the role of good priors.

Through the decompositon of the ELBO, we can see that its maximization correpsonds to the minimization of mutual information between x and z (informally recoverable information of x in z is redundant), a maximization of the entropy of $q(z)$ (informally $q(z)$ has minimal structure) and minimization of the cross-entropy between $q(z)$ and $p(z)$ (informally $p(z)$ is close to $q(z)$).

5.9 The KL divergence

The KL-divergence is a popular quantity that reappears in many generative machine learning algorithms. Here we provide a geometrical intuition on its meaning and highlight some of its limitations and strengths. In machine learning, one aims to maximize a likelihood $p(x,\theta)$ or log-likelihood $\log(p(x,\theta))$ by finding the optimal parameters θ. The gradient of the likelihood with respect to the parameters is defined as

$$s(x,\theta) = \frac{\partial \log(p(x,\theta))}{\partial \theta} \tag{5.64}$$

and called the score of the model. It can be shown, that under regularity conditions, the score of the true model $\tilde{\theta}$ is zero. In general, the score is indicative of the model-parameter sensitivity. The notion of a score-based gradient has been exploited successfully in generative modeling, under the name of score matching. Score-matching does not find a model, but enables sampling nonetheless. It considers only the gradient field and is useful for model-parametrization which involve an intractable normalization constant (such as energy-based models).

For each θ, it is possible to define the Fisher Information Matrix as the covariance of the score,

$$\begin{aligned}\mathbf{I}(\theta) &= \mathbb{E}\left[\left(\frac{\partial \log p(x,\theta)}{\partial \theta}\right)\left(\frac{\partial \log p(x,\theta)}{\partial \theta}\right)^T\right] \\ &= -\mathbb{E}\left[\frac{\partial^2 \log p(x|\theta)}{\partial\theta\partial\theta^T}\right] = \int_x \frac{\partial}{\partial\theta}\log(p(x,\theta)^2)p(x,\theta)dx.\end{aligned} \tag{5.65}$$

Given the previous examples, this should create some intuition of the underlying geometry which is defined by this matrix. Note, that this covariance matrix varies smoothly, depending on the location. Considering the distance induced by a covariance matrix, it can be shown that

$\mathbf{I}(\theta)$ is a Riemannian metric which provides the tangent space of θ with an inner product and can be used to define distances between distributions. The geodesic distance is the path-integral of distances induced in each point by the local covariance matrix. The idea is similar to the Mahalanobis distance, but, due to the interpretation in terms of model parameter sensitivity (contrary to data-sensitivity), we will not use the inverse.

Consider an infinitesimal displacement

$$\frac{d\theta}{dt} = \lim_{\Delta \to 0} = \theta + \Delta$$

and a path $\gamma(t)$ which is related to the displacement by

$$d\theta = \frac{d\gamma(t)}{dt} dt.$$

The infinitesimal distance from θ to $d\theta$ is

$$ds^2 = d\theta^T I(\theta) d\theta = \frac{d\gamma(t)}{dt} I(\theta) \frac{d\gamma(t)}{dt} dt. \tag{5.66}$$

This is a the infinitesimal squared distance in Riemannian geometry with the fisher information matrix as metric tensor. When $I(\theta)$ is large, indicating a high likelihood sensitivity of a parameter, then the distance in that direction is also large. If we move in directions of high sensitivity (i.e. corresponding to high information content in that direction) we traverse a larger distance. On the other hand, moving in directions with low sensitivity does not get us very far. If we want to minimize the likelihood, we would prefer to move the shortest path from θ_1 to θ_2 (i.e. update the most informative parameters).

The distance between θ_1 and θ_2 becomes

$$d_G(\theta_1, \theta_2) = \int_0^1 \sqrt{\frac{d\gamma(t)}{dt}^T I(\gamma(t)) \frac{d\gamma(t)}{dt}}\, dt = \int_0^1 ds, \tag{5.67}$$

where $\gamma(0) = \theta_1$ and $\gamma(1) = \theta_2$.

By this geodesic distance, the similarity of distributions can be quantified. Distributions (parametrized by θ) are close if the information content is similar and distant else. The geodesic distance is symmetric.

Since direct computation might be intractable, often quadratic form approximations are used, so called divergences. A famous case is the KL-divergence which is defined as

$$D_{KL}(p(x|\theta) \| q(x)) = \int p(x) \log \frac{p(x)}{q(x)} \tag{5.68}$$

By relation to the Shannon-entropy, it measures an information-theoretical distance between two distributions. For small displacements, it can be shown that

$$D_{KL}(p(x|\theta) \| p(x|\theta + d\theta)) \approx \frac{1}{2}\, d\theta^T I(\theta) d\theta.$$

Employing this quadratic approximation of the riemannian distance solves computational issues but destroys distance properties such as symmetry. However, an information theoretical interpretation is possible. Some algorithms, such as the variational autoencoder, use the KL-divergence as a loss-term. Considering the notion of asymmetric distances between distributions, this corresponds to its minimization. Minimizing the KL-divergence can also be

considered equivalent to maximum likelihood estimation. Consider standard maximum likelihood estimation

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log p_\theta(x_i) \tag{5.69}$$

and the discrete KL-divergence between the empirical data-distribution $x_i \sim q(x)$ and parameterized model distribution $p_\theta(x)$

$$D_{KL}(q(x)|p_\theta(x)) = \sum_{i=1}^{N} \log \frac{1}{p_\theta(x_i)} = -\sum_{i=0}^{N} \log p_\theta(x_i). \tag{5.70}$$

Minimizing equation 5.70 is equivalent to maximizing equation 5.69. However, the KL-divergence is not symmetric so that this equivalence does not hold for $D_{KL}(p_\theta(x)|q(x))$, although, if the zero is reachable, both have the same optimum. Furthermore, the KL-divergence requires both distributions to live in the same space and have significant overlap. If $q(x) = 0$ where $p(x) \neq 0$ the Kl-divergence is not defined. If $p(x) = 0$ where $q(x) \neq 0$ the KL-divergence is zero. This can create differentiability problems and lead to unstable training. Thus, some algorithm replace it by other methods to quantify similarities between distributions, such as the Wasserstein distance. An example is the Wasserstein-GAN which stabilizes training by replacing optimization with respect to a divergence with optimization with respect to the Wasserstein distance.

Chapter 6

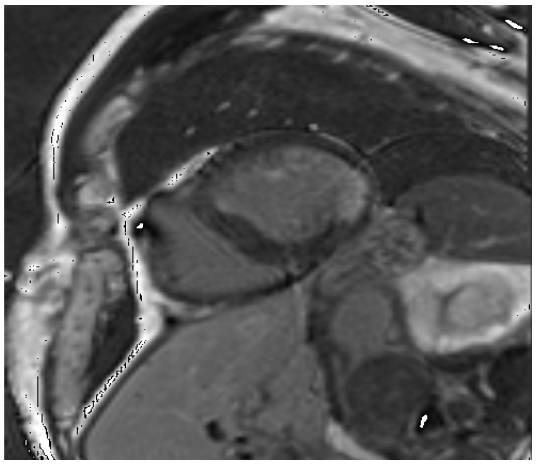
Data acquisition

In this chapter the data acquisition and processing is explained. We start by patient data obtained from hospitals. These are segmented manually. The contour lines obtained from these segmentation's are used to fit a heart template geometry via B-spline based mesh morphing. The result is a heart which contains patient specific geometric features. Knowledge of the segmentation masks, also enable the assignment of cell-values to the mesh, for example to differentiate whether a cell corresponds to healthy myocardium of ischemic scar tissue. Since the number of cells and points of each patient specific mesh coincides with the heart template geometry, the cell values can be mapped back. This representation serves as a normalized visualization of the scar-geometry. In essence, by visualizing it in the template geometry, the patient specific heart geometry is abstracted away. Each patient case is represented by the same heart geometry, however, with different scars.

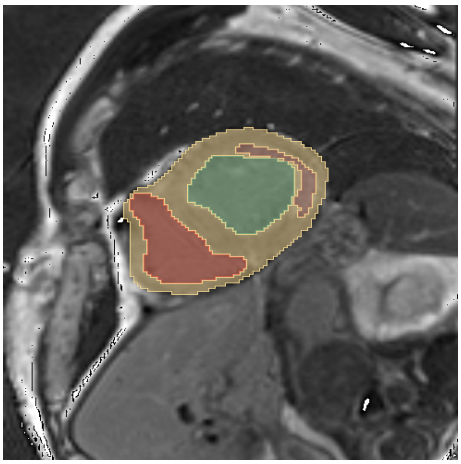
6.1 Segmentation

This patient data consists of several (usually ca. 5-10) horizontal slices of MRI images. Figure 6.1a shows such a MRI scan in the right image. The segmentation-masks are obtained by manual segmentation. An example is displayed in figure 6.1b. These segmentation masks differentiate between healthy myocardium (yellow), left-ventricle (green), right ventricle (red) and ischemic scar tissue (brown). The contour lines obtained from these masks on different vertical levels are displayed in figure 6.1c.

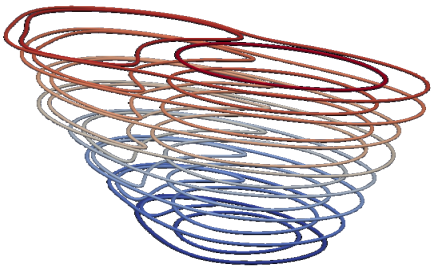
Figure 6.1: Fitting the left ventricle



(a) MRI scan



(b) MRI scar with masks



(c) Contour lines

For each patient several of these masks are obtained (ca. 10), each corresponding to a slice at different height . Contour lines of the corresponding heart geometry are determined from the mask at relatively low vertical resolution is. These contour lines are then used to fit a template mesh to the scanned heart geometry.

6.2 Fitting the patient heart

The segmentation masks, which are obtained by a process explained in section 6.1, are used to obtain contour lines of the heart geometry at the respective heights. The contour corresponding to the healthy myocardium mask outlines the outer heart geometry and corresponds to the endocardium, the left and right ventricle masks correspond to the epicardium. These three contours define the patient-specific heart-wall geometry at the respective height. To obtain a 3D-model of the corresponding heart *mesh-morphing* is used. An available template-geometry is fitted to the contour lines, so that the epicardium and endocardium correspond to the contour lines. Figure 6.2a displays the template heart which is fitted to the contour lines in figure 6.1c. Outcomes at specific time-steps are visualized in figure 6.2b and 6.2c respectively.

The concrete methodology utilizes basis splines and an energy (or cost) function, which balances the mesh-curvature and line fitting bending to avoid physically unrealistic gradients in the heart walls.

Consider a three-dimensional domain

$$\Omega = \{(x, y, z) | 0 \leq x < X, 0 \leq y < Y, 0 \leq z < Z\} \tag{6.1}$$

with volume $V = X * Y * Z$ in within which the contours are located. This domain corresponds to the image volume. Let the contour domain be

$$N = \{(x, y, z) | 0 \leq x < X, 0 \leq y < Y, 0 \leq z < Z\}. \tag{6.2}$$

where $\forall x, y, z \in N$ are elements of the obtained contour-lines.

Let $\Phi_{i,j,k}$ be a set of $n_x \times n_y \times n_z$ control point coordinates with uniform spacing, Let Φ denote the corresponding matrix.

Let $T_(x, y, z)$ be a function that corresponds to the contour line coordinates and consider its approximation by

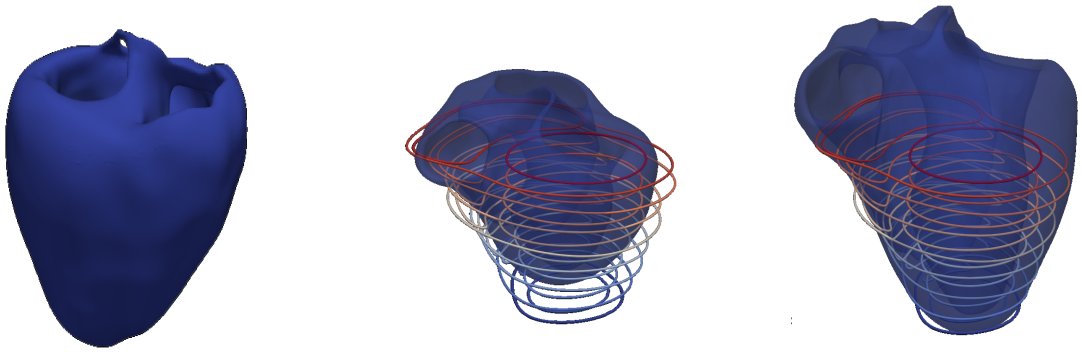
$$\tilde{T}(x, y, z) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_l(u) B_m(v) B_n(w) \Phi_{i+l, j+m, k+n}$$

with $i = \lfloor x/n_x \rfloor - 1$, $j = \lfloor y/n_y \rfloor - 1$, $k = \lfloor z/n_z \rfloor - 1$, and $u = \lfloor x/n_x \rfloor - x/n_x$, $v = \lfloor y/n_y \rfloor - y/n_y$, $w = \lfloor z/n_z \rfloor - z/n_z$ Where B_l represents the l-th basis function of the B-splines:

$$B_0(u) = (1 - u)^3/6$$

$$B_1(u) = (3u^3 - 6u^2 + 4)/6$$

Figure 6.2: Fitting the left ventricle



(a) Template heart (b) Early timestep (c) Late timestep

$$B_2(u) = (-3u^3 + 3u^2 + 3u + 1)/6$$
$$B_3(u) = u^3/6$$

The number of control points of n_x,n_y,n_z are the parameters of the B-spline transformation and correspond to the resolution of the control point mesh Φ. Choosing the right amount of control points is a design choice which balances the trade-off between the computational complexity and the accuracy of the approximation. A multi-resolution approach is used to compute $\tilde{T}(x, y, z)$ as

$$\tilde{T}(x, y, z) = \sum_{l=0}^{L} \tilde{T}_l(x, y, z).$$

This enables a separation of the coarser deformations from the finer high-resolution ones and results in a more computationally efficient approach. First the low-resolution forms are approximated at lower computational cost (due to the low resolution) and then, in an iterative manner, the resolution is incremented to approximate regions which require high resolution. This hierarchical approach enables much faster convergence, since the low-resolution details do not need to be approximated with a high resolution mesh. To penalize smoothness of the mesh-deformation a penalty term C is computed:

$$C_{\text{smooth}} = \frac{1}{V} \int_o^X \int_0^Y \int_0^Z [\frac{\partial^2 \tilde{T}}{\partial^2 x^2} + \frac{\partial^2 \tilde{T}}{\partial y^2} + \frac{\partial^2 \tilde{T}}{\partial z^2}] dxdydz.$$

The penalty of line fitting is computed as

$$C_{\text{fit}} = \frac{1}{V}(T_(x, y, z) - \tilde{T}(x, y, z))^2$$

for all $(x, z, z) \in N$ and zero elsewhere. Both together form the optimization penalty:

$$C = C_{\text{fit}} + C_{\text{smooth}}$$

The algorithm to fit a mesh to the contour lines can be described by the following Pseudo-code

procedure FIT MESH TO CONTOURS
  initialize control points $\Phi_{l=0}$ by minimizing C_{fit}
  repeat
    $\nabla C \leftarrow \frac{\partial C(\Phi_l)}{\partial \Phi_l}$
    while $||\nabla C|| > \epsilon$ **do**
      $\Phi_l \leftarrow \Phi_l + \mu \frac{\nabla C}{||\nabla C||}$
      $\nabla C \leftarrow \frac{\partial C(\Phi_l)}{\partial \Phi_l}$
    end while
    increase control point resolution Φ_l to Φ_{l+1}
  until finest resolution is reached
end procedure

6.3 Heart shape normalization

This section explains how the patient specific heart and scar geometry is normalized to enable comparison of scars. The sections **??**, **??** and **??** explain how a mesh geometry with the corresponding cell values is obtained. Section **??** mentions the fitting of a template mesh to a set contour lines obtained from the segmentation (see section **??**). The patient heart is a deformation of this template mesh and has the same number of nodes and faces. Furthermore, there is point-to-point correspondence between the original and the deformed mesh. This means, that the cell-values which are obtained by the procedure in section **??** can be mapped back to

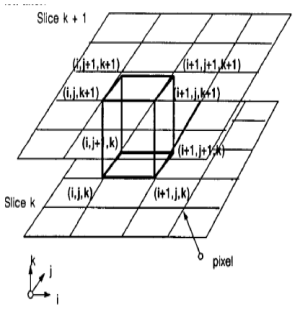
the template by exploiting the point to point (and therefore also cell to cell) correspondence between meshes. The scar geometry is mapped back to the template heart and is represented independently of the patient specific heart geometry. This has the advantage, that parameters like size, and orientation become comparable. If one would compare scar-geometries as they occur in patient specific hearts (as opposed to a template heart), the absolute size of a scar says very little about its actual size relative to the heart. Scars in the heart of a small person (who likely has a smaller heart) would not compare well to scars in the heart of a very big person. The representation in a uniform template heart solves this problem.

6.4 Iso-surface extraction

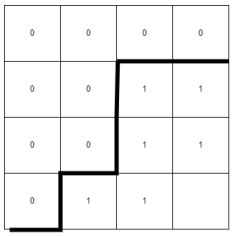
Implicit shape representations define a field of level-set values in 3D-space. The shape boundary is represented by a specific level set. For example, the zero-level set in the case of signed distance functions. In a binary representation with values zero and one, the shape boundary is defined by the transition.

The marching cubes algorithm is a powerful method for extracting iso-surfaces from volumetric data. This section presents a detailed explanation of the algorithm, encompassing both binary (discrete) and continuous scalar fields. Each point in the field is associated with a scalar value that describes a property of the corresponding voxel, such as occupancy (in the binary case) or distance to a surface (in the continuous case). The algorithm extracts an iso-surface by identifying where this scalar value crosses a particular threshold, typically zero. Figure 6.3a displays the cube which moves over the domain. Figure 6.3b displays a discrete field and its iso-surface. Figure 6.3c displays a continuous field and its iso-surface.

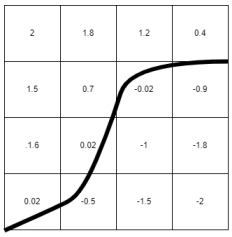
Figure 6.3: Marching cubes



(a) the cube



(b) a discrete field



(c) a continuous field

1. **Cube Configuration:** The scalar field is divided into cubes formed by eight neighboring voxels. For each cube, an 8-bit index is generated based on the scalar values at its vertices. In a binary field, the value at each vertex is either 0 (outside the object) or 1 (inside the object). In a continuous field, the value can vary, with negative values typically indicating inside the object and positive values indicating outside.

   For both cases, the index is computed by treating each vertex value as a bit in an 8-bit integer, where the bit is set if the value exceeds the threshold (commonly zero). This index determines which of the 256 possible cube configurations applies.

   Figure 6.4b displays the 8-bit index for each cube.

2. **Lookup Table:** A precomputed lookup table maps each cube configuration index to a set of edges where the iso-surface intersects the cube. This table is essential for efficiently determining the surface geometry without recalculating the intersections for each cube

configuration from scratch. Each 8-bit index points to its respective cube-configuration. Figure 6.4a displays some of the possible iso-surface configurations.

3. **Edge Interpolation:** The precise position where the iso-surface intersects an edge of the cube is determined by linear interpolation between the scalar values at the edge's vertices.

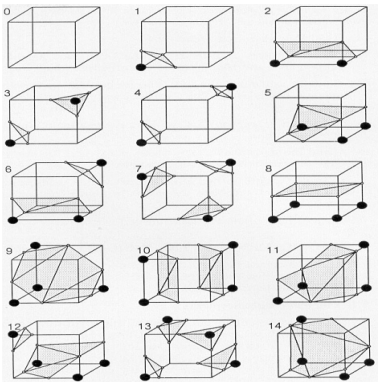
   - In the **binary case**, the scalar values are 0 or 1, so the intersection is typically approximated at the midpoint of the edge where a transition occurs between 0 and 1.
   - In the **continuous case** (e.g., signed distance function), the intersection is more accurately determined by interpolating between the actual scalar values:

$$p = \frac{|f(v_1)|}{|f(v_1)| + |f(v_2)|} \times v_2 + \frac{|f(v_2)|}{|f(v_1)| + |f(v_2)|} \times v_1$$

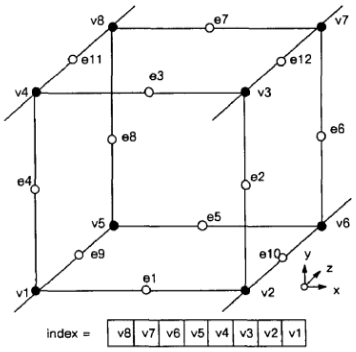
     where $f(v_1)$ and $f(v_2)$ are the scalar values at the vertices v_1 and v_2, respectively. This interpolation allows the extraction of a smooth surface even when the underlying grid is relatively coarse.

4. **Surface Construction:** Using the interpolated intersection points, the algorithm constructs surface triangles within the current cube. These triangles are generated according to the lookup table, which specifies how the surface should be connected across the edges based on the cube configuration.

Figure 6.4: Look-up table and 8-bit index



(a) Look-up table



(b) 8-bit index for a cube

The marching cubes algorithm is versatile and can be applied to a variety of domains. In medical imaging, for example, it is used to reconstruct surfaces of organs from volumetric scans. In computer graphics, it is employed to generate meshes from defined scalar fields. The choice between binary and continuous scalar fields depends on the application's specific needs for surface detail and smoothness.

In this work, iso-surface extraction is applied to the binary volumetric field obtained during the data-acquisition and to a computed signed distance function which is used during the generative process.

6.5 Acquisition Results and Processing Artifacts

This section explained the acquisition of the heart and scar geometries. Following the procedure explained in section 6.2, a mesh of a patient specific heart geometry is obtained from segmentation masks of MRI scans. This procedure involves fitting a template mesh to contour lines of

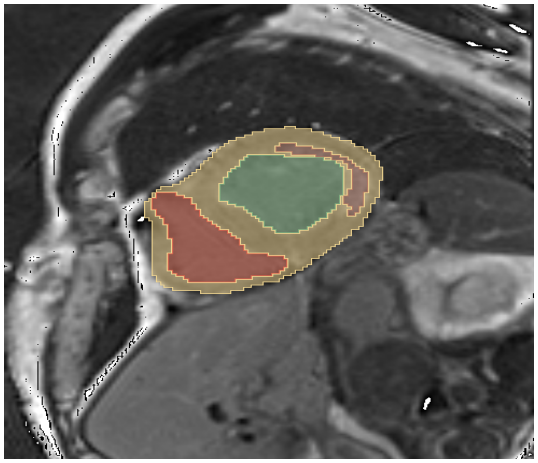
segmentation masks.

By overlapping the respective segmentation masks with the heart-geometry, one can infer information about the attributes of the cell values. In essence, it is possible to distinguish between cells which belong to healthy myocardium, left or right ventricle and cells which belong to ischemic scar tissue by examining cell location and mask contours. This procedure is used to fill cell values with scalars, zero for healthy myocardium, one for ischemic scar tissue. The cell-values can be extrapolated to vertex values and define an (discrete and non-continuous) implicit field. By considering the segmentation mask visualized in figure6.5a, and the corresponding heat, visualized in figure 6.5b, cell values, corresponding to scar-tissue and healthy-tissue are assigned to the mesh by processing the closest contours. This results in a discrete field. By using the iso-surface extraction algorithms explained in section 6.4 the scar surface mesh can be obtained. Figure 6.5c and figure 6.5d display such a scar.

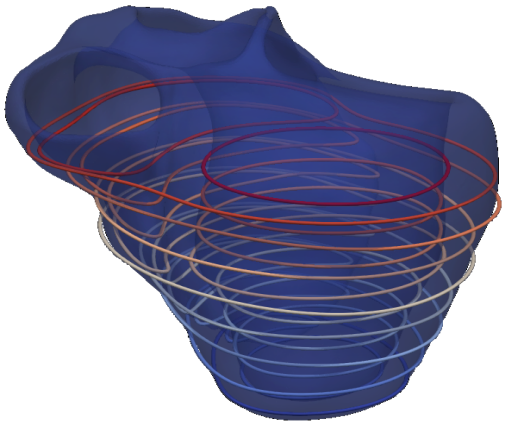
A disadvantage of this processing methodology is that the vertical resolution of the MRI scans and, therefore, also the segmentation masks is low. Only ca. 10 horizontal slices exist. The current approach identifies the nearest mask for each cell value and assigns the corresponding value. The result is nonphysical processing artifacts. For example, the cell-values (zero and one) have a block-like structure (due to the low vertical resolution), and sometimes the scar is represented by seemingly separated "blocks". In reality, the tissue structure is most likely smooth and compact. Furthermore, MRI-scans are usually not available at the base. Scar tissue there is not captured, although it is most likely there. In figure 6.5c and figure 6.5d the disjoint, block like, structure of the obtained scar is visualized. Furthermore, figure 6.5d, does not display scar tissue at the base. However, it is likely that the physical scar tissue does cover the base. Another example of such a case is visualized in figure 2.5. In section 7, this issue is clearly visualized by nonphysical holes in the scar.

For the purpose of this work, the obtained scars with processing artifacts will be considered ground-truth. These processing artifacts need to be considered when evaluating the representative power of the obtained 3D-scars.

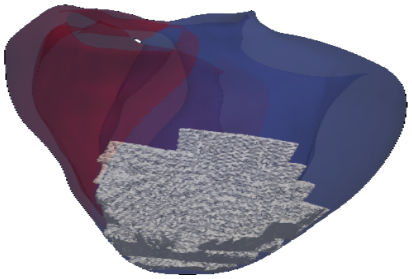
Figure 6.5: Assignment of cell values based on the Segmentation Mask



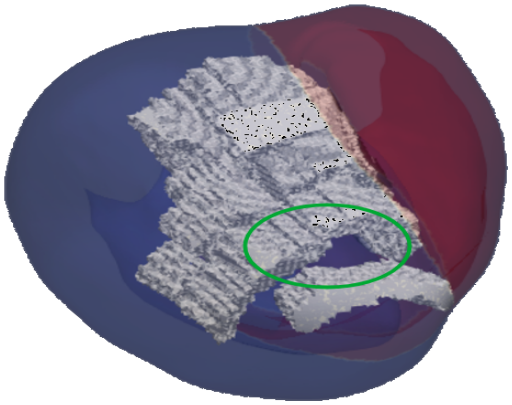
(a) segmentation mask with scar mask



(b) Heart with contour lines



(c) Obtained Scar



(d) Obtained Scar (view from below)

Chapter 7

Signed Distance Representation in a Disk

In clinical practice the left ventricle of the heart is represented by a Standardized Myocardial Segmentation and Nomenclature provided by the American Heart Associated. This representation flattens the heart into a disk, with the apex at the center and the base on the border. Then 17 regions are distinguished and numbered. The regions enable talking in a standardized way about locations in the heart. For example, clinicians define scar-shapes by the degree of occupancy within each standardized region. Furthermore, the region-wise discretization also enables an assignment of respective coronary alimentation.

In a crude way, each region has a corresponding coronary which provides its oxygenation. Although, this generalization is common in clinical practice, it is not exact. The exact geometry of the coronaries, and as such, the regions they oxygenate, are different from patient to patient.

Within the context of ischemic scars, the right ventricle is often ignored. This is due to the fact of its very thin walls. A formation of a scar in such a low volume is highly unlikely. The majority of myocardium corresponds to the left ventricle. We therefore concern ourselves only with the left ventricle.

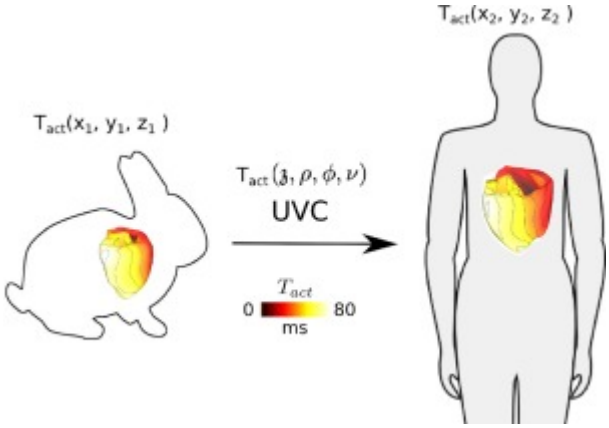
To approach the clinical setting, we choose to map the left ventricle to a cylinder. The height is representative of the transmurality at the respective position.

The mapping into the disk requires the usage of standardized ventricular coordinates. Each point in the template heart is assigned four standardized coordinates, ρ, Φ, θ and γ. These four coordinates enable an assignment of each point of the left ventricle to a three-dimensional cylinder.

7.1 Universal ventricular coordinates

In this section a procedure for mapping sets of cardiac ventricles to topologically equivalent representations is presented. This mapping is used to compare quantities of interest between different heart geometries. It is based on the knowledge, that mammalian hearts share a common structure with generally four-chambers and two ventricles. Figure 7.1 visualizes a mapping from a rabbit heart to a human heart in universal ventricular coordinates. Mammalian hearts express electrical heterogeneity with respect to trans-mural, apicobasal and left-right gradients cite Universal ventricular coordinates!. Describing such tensors on a generic heart is important for comparison and quantification across geometries. The segmentation of AHA-regions, is a similar approach, in the sense that it describes various heart geometries in a generalized frame. cite Universal ventricular coordinates! highlights the low-resolution of this standardized mapping by AHA-regions and provides a more detailed way of computing a generalized representation of mammalian hearts. A representation in 4-coordinates is presented. These coordinates are significant within clinical and experimental studies and are well suited for a

Figure 7.1: Mammalian hearts in universal coordinates



standardized representations of a heart. Within this work, these coordinates are used to map the heart into a cylindrical representation, in line with the AHA-segmentation.

- ρ represents the distance between the endocardium and epicardium, i.e. the transmurality.
- Φ represents the circumferential distance from the long axis of right ventricle and left ventricle respectively.
- r corresponds to the distance traveled along the long axis of the ventricles from apex to base, i.e. the apicobasal coordinate.
- γ is used to distinguish between right and left ventricle. The computation is explained in xxx.

Here, a short summary is provided. Given the surfaces for the heart base, epicardium, endocardium of both ventricles and septum, the coordinates of each point within the volumentric mesh are obtained by solving the Laplace equations. Dirichlet boundary conditions are applied onto each surface.

For the transmural coordinate, the epicardium gets $\rho = 1$, the endocarium gets $\rho = 0$.

For the apicobasal coordinate, the base gets $r = 0$ and the apex gets $r = 1$.

For the rotational coordinate, the left ventricle endocardium gets $\Phi = \pm\pi$ at the wall where it combines with the right ventricle. $\Phi = \pm\frac{\pi}{2}$ at the outer junctions of left and right ventricle $\Phi = 0$ at the middle of the septum, $\Phi = \pm\frac{\pi}{2.5}$ at the inner junctions of left and right ventricle.

To seperate left and right ventricle, the left ventricle endocardium gets $\gamma = 0$, the right ventricle endocardium gets $\gamma = 1$, the septum gets $\gamma = 0.5$.

Figure 7.2: Boundary conditions for the transmural coordinate

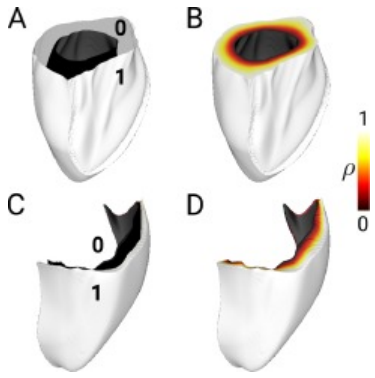


Figure 7.3: Boundary conditions for the apicobasal coordinate

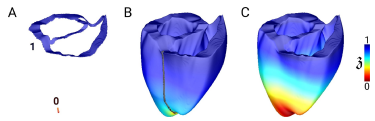


Figure 7.4: Boundary conditions for the rotational coordinate

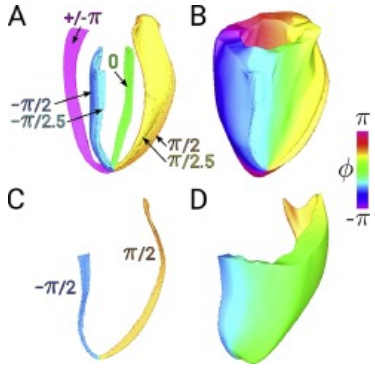
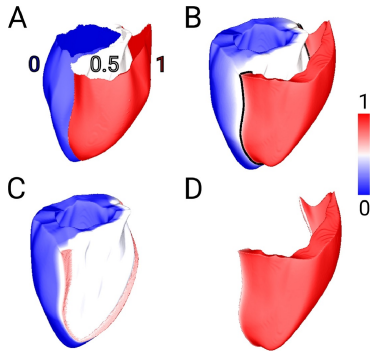


Figure 7.5: Boundary conditions for the left and right ventricle



The Laplace equation is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

where

$$y_b(x) = f(x) \quad \forall x \in \partial\Omega$$

correspond to the boundary conditions on the boundary $\partial\Omega$.

The problem consists of finding a solution f, so that the specified boundary conditions are satisfied. The Laplace equations appear in heat-flow. Therefore, one can formulate the problem as fixing the temperature (value) at the specified boundaries and letting heat flow until a stationary temperature (value) distribution is achieved. This is the solution to the problem.

The heat flow is computed numerically on a discretization of the domain, e.g. by the finite element method. We will not go deeper into the numerical solving procedure.

Once the universal ventricular coordinates are computed, each point of the left ventricle, together with the respective scalar value is mapped to a cylinder with coordinates r for the radius, rho for the height and Φ for the angular coordinate. The end-result is a cylindrical mesh with the same connectivity information as the heart mesh.

Figure 7.6: Representation as heart

Figure 7.7: Representation as disk

7.2 Signed distance representation

Within disk mesh, a signed distance function is computed which labels each node with its minimal distance to the scar boundary. Points inside the boundary correspond to negative values while points outside correspond to positive values. Since all scars have a closed surface, a signed distance function is well defined and yields an implicit representation of the shape. The signed distance function is described by a function $d(p, S)$, where p is a point in (usually three dimensional) space. The function is defined in the whole domain. Its values are so that $d(p, S) < 0$ if p

is inside the shape and $d(p, S) > 0$ if p is outside the shape. Values of p which are positioned exactly on the surface boundary correspond to $d(p, S) = 0$. The signed distance function requires the shape-boundary to be closed. This allows for the distinction between inner and outer space. An advantage of a signed distance representation is the low requirement on information-storage. Whereas a mesh or point-cloud representations require storing each point or voxel value, the signed distance function requires only the storage of a function. Depending on the shape this function can have a very simple form. This makes the representation very memory efficient. However, the shape is only represented implicitly. Recovering the shape-surface requires usage of other algorithms. Very often the iso-surface extraction methods, such as marching cubes (see section 6.4) is used.

Practically, computing the signed distance function requires two objects. A set of points in space, as well as surface-information of the shape boundary. These objects are then used to assign to each point the corresponding minimal distance to the surface. In our case, the shape is encoded by voxel-values in a disk-volume. Similar to a signed distance function the representation is implicit, however in a discrete manner. Points in the shape have value one and outside they have the value zero. By using the *Extract-Surface* function provided by *paraview* and *vtk* we can convert the implicit scar representation into a surface-mesh representation. We then consider the set of $3D$-points within the disc. For each point we require the signed distance information.

The mathematics rely on the computation of Point-Plane distances and a boolean function which differentiates between inner and outer points. This is used to assign to each point a signed distance to the nearest polygon. One can differentiate between the computation of the distance and the boolean function. These employ a point-plane distance algorithm and a winding-number or ray-casting algorithm respectively.

7.2.1 Point-Plane Distance algorithm

Let

$$P_i = (x_i, y_i, z_i),\ i = 0, 1, ...N - 1$$

be a set of N three-dimensional coordinates corresponding to a set of discrete points within the disc. Let

$$\vec{a_j} = (x_a, y_a, z_a)_j,$$
$$\vec{b_j} = (x_b, y_b, z_b)_j,$$
$$\vec{c_j} = (x_c, y_c, z_c)_j$$

with

$$j = 0, 1, ...M - 1$$

be a set of vertices corresponding to a surface mesh with M triangular faces.

The normalized normal vector of the plane can be computed as

$$\vec{n} = \frac{(a_j \times b_j)(a_j \times c_j)}{||(a_j \times b_j)(a_j \times v_j)||}.$$

To find the projection of a point

$$\vec{p} = (x_p, y_p, z_p)$$

onto the plane consider the decomposition

$$\vec{p} = \alpha\vec{n} + \beta\vec{n}^{\perp}.$$

$\alpha\vec{n}$ corresponds to the distance of $\vec{p}$ to the plane, $\beta\vec{n}^{\perp}$ corresponds to the coordinates of the projection. Define the plane equation

$$\vec{n} \cdot \vec{x} + d = 0.$$

Compute d for the plane using $\vec{a_j}$, $\vec{b_j}$ or $\vec{c_j}$ as $\vec{x}$.
Then the minimal distance of $\vec{p}$ to this plane is given by

$$\vec{n} \cdot \vec{p} + d = \beta\vec{n}^{\perp}.$$

Thus the projected coordinate is

$$\vec{p}_{proj.} = \alpha\vec{n} = \vec{p} - \beta\vec{n}^{\perp} = \vec{p} - (\vec{n} \cdot \vec{p} + d) = \vec{p} - (\vec{n} \cdot \vec{p} - \vec{n}\vec{a_j}).$$

This distance is always positive.

7.2.2 Winding Number Algorithm

The winding surface algorithm computes the solid angle to point $\vec{p}$ for every triangle of the surface mesh defined by

$$\vec{a_j} = (x_a, y_a, z_a)_j,$$
$$\vec{b_j} = (x_b, y_b, z_b)_j,$$
$$\vec{c_j} = (x_c, y_c, z_c)_j$$

and sums them.

The solid angle is computed as

$$\Omega_i = 2 \cdot \text{atan2}\left(\vec{n}_i \cdot \vec{ap}_i, \|\vec{ap}_i\|\|\vec{bp}_i\|\|\vec{cp}_i\| + (\vec{bp}_i \cdot \vec{cp}_i)\|\vec{ap}_i\| + (\vec{cp}_i \cdot \vec{ap})_i\|\vec{bp}_i\| + (\vec{ap}_i \cdot \vec{bp}_i)\|\vec{cp}_i\|\right),$$

where $\vec{ap}_i$,$\vec{ap}_i$,$\vec{ap}_i$ corresponds to the distance between $\vec{a}_i$,$\vec{b}_i$ and $\vec{c}_i$ respectively to $\vec{p}_{proj.}$. $\vec{n}$ corresponds to the normal vector of the triangle and is computed via

$$\vec{n} = \vec{ab} \times \vec{ac}.$$

The winding number for point $\vec{p}$ is computed as the sum of all solid angles and is

$$w = \sum_{i=0}^{T} \Omega_i,$$

where T is the number of triangles within the surface mesh. If $\vec{p}_{proj.}$ is within the surface, then $w \approx 4\pi$, else $w \approx 0$. The winding number algorithm is quite robust but computationally expensive, although it can be parallelized heavily. For complex meshed, with the possibility of self-intersections this algorithm is recommended.

7.2.3 Ray Casting Algorithm

A method to determine whether $\vec{p}$ is inside or outside the surface mesh is via a ray-casting algorithm. This method is less robust than the winding-number algorithm, but often good enough, easier to implement and requires less computation.

Consider the point $\vec{p}$ and define a ray

$$\vec{y}_{ray} = \vec{p} + \gamma\vec{r}$$

. Consider the three vertices $\vec{a_j}, \vec{b_j}, \vec{c_j}$ of triangle j and the three corresponding edges

$$\vec{e_1} = \vec{a_j} - \vec{b_j},$$

$$\vec{e_2} = \vec{a_j} - \vec{c_j})$$

and

$$\vec{e_3} = \vec{b_j} - \vec{c_j}).$$

To find the intersections of the ray with each edge by solving the linear system of equations

$$\vec{y}_{\text{ray}} = \vec{a}_j + u\vec{e}_1 + v\vec{e}_2, \quad \text{where } u \geq 0,\ v \geq 0,\ u + v \leq 1$$

.

One can solve for u,v and γ.
If γ is positive and u, v satisfy $u \geq 0$, $v \geq 0$, $u + v \leq 1$, then the ray intersects the triangle.
If the number of intersections is odd, then the $\vec{p}_{proj.}$ is not within the triangle.

7.2.4 Computation in VTK

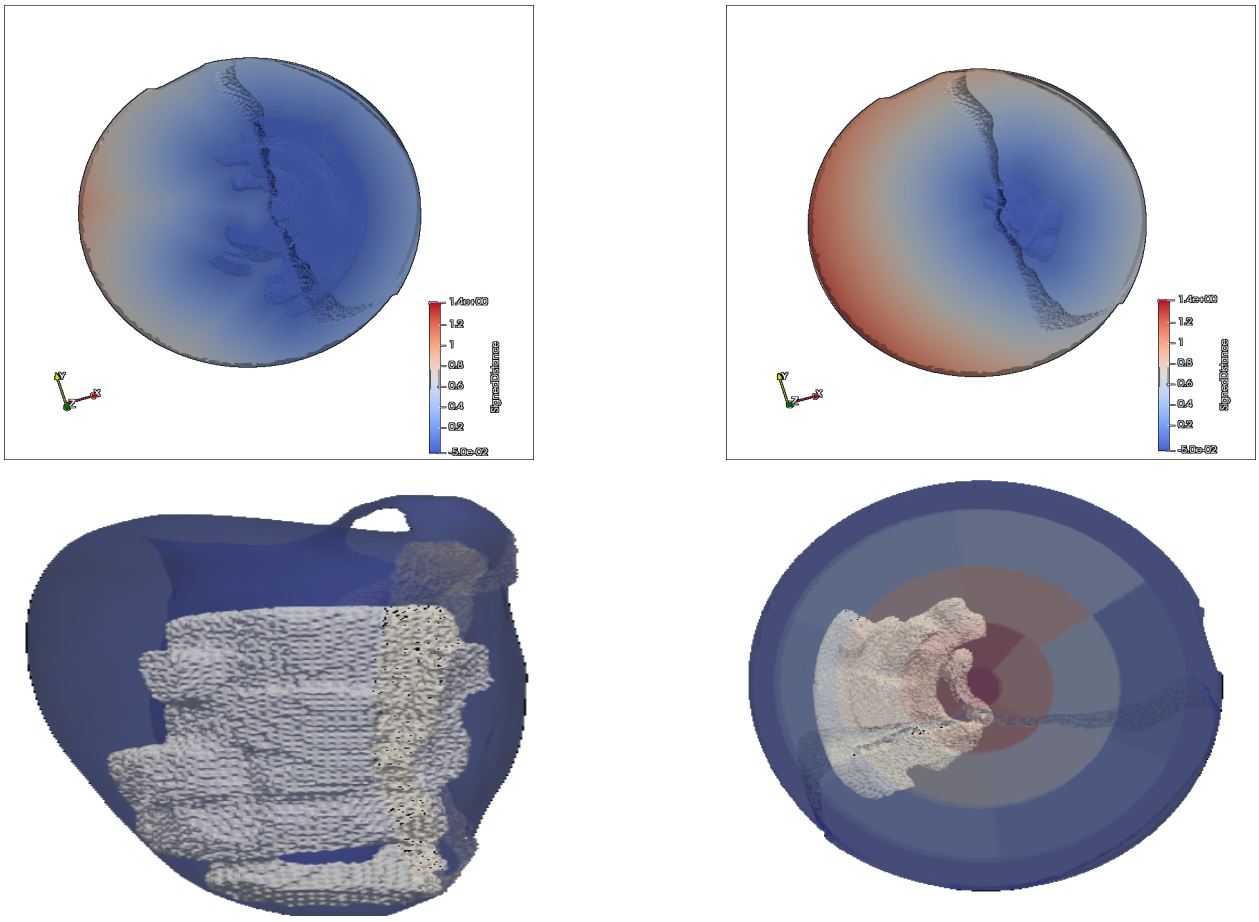
The *vtk*-library provides a *FindClosestPoint()*- function. This function accepts a surface-mesh-input as well a $3D$ coordinate. It returns the *closest point* on the surface as well as the corresponding distance. Additionally, the *selector*-class has a function *IsInsideSurface()*, which is used to determine the sign of the returned distance. It employs the ray casting algorithm.

Since the disk contains 600000 points, the computational efficiency becomes relevant. Running the *FindClosestPoint()* and *IsInsideSurface()* in a python *for-loop* takes a significant amount of time. Therefore it is better to choose to write the code in C++ and connect it to python with the *c-types* library. Additionally, using *vtks* parellizability and running the for-loop in several threads with *vtkSMPTools* results in a significant speed-up. Since the *selector*-class with the *IsInsideSurface()* -method is not thread-safe, it must be initialized again for each point. Contrasting this, the *CellLocator* -class can be initialized only once for each shape. The Pseudo-code to compute the Signed Distance Function becomes:

function ComputeSignedDistance(SignedDistanceVector, pointSet, Point_number, surface, transformed)
    Initialize `cellLocator`(surface)
    numberOfPoints ← Get number of points from surface
    grains ← `Point_number` / 28 + 1
    vtkSMPTools::**For** (0, Point_number, grains, [](int startPtId, int endPtId) *-Parallel computation: Divides range (0, Point_number) into 'grains' of size endPtId - startPtId*
        Init `selector`(surface)
          `double`* p: a pointer to the current point coordinates
          `double` closestp[3]: array to store the closest point's coordinates
          `double` dist2: distance squared from the current point to the closest point on surface
          `vtkIdType` cellID: identifier for the closest cell
          `int` subID: identifier for the sub-cell of the closest cell
          `vtkNew<vtkGenericCell>` genericCellPtr: pointer to a generic cell object
        For `int`ptId from startPtId to endPtId:
          p ← pointSet[ptId] - Point coordinates at index ptId in `pointSet`
          If `selector`.IsInsideSurface(p)
            sign ← -1
          else
            sign ← 1
          `cellLocator`.FindClosestPoint(p, closestp, genericCellPtr, cellId, subId, dist2)
          SignedDistanceVector[ptId] ← sign × $\sqrt{\text{dist2}}$
    End parallel For
end function

This computation returns a vector with 600000 values, one signed distance for each point within the disk. Figure 7.8 displays the iso-surface obtained from the implicit signed distance function representation in the template heart and disk respectively (from left to right). Recovering the scar-shape involves iso-surface extraction with the marching cubes algorithm (see section 6.4). The iso-surface is defined on the zero level set of the signed distance field.

Figure 7.8: Zero level set of a signed distance field



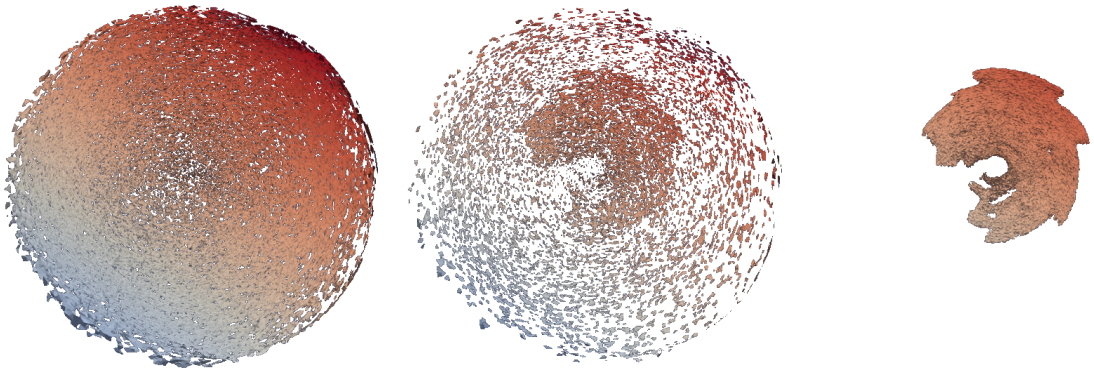
The scar shape, as recovered from the implicit representation explained in chapter **??**, is represented well. No important features are lost.

7.3 Dimensionality reduction with basis splines

The scar is represented as a signed distance function in the $3D$-domain of a disk, i.e. $sdf = f(x, y, z)$. The signed distance valued are available on discrete points as a vector of ca. 600000 sdf values. Initial tests with an (non-variational) auto-encoder show that this high dimensionality prevents model-fitting. As soon as more than one data-sample is considered, the model fails to converge. Figure 7.9 displays the training progress on a single shape. Each of the 600000 discrete points is updated individually.

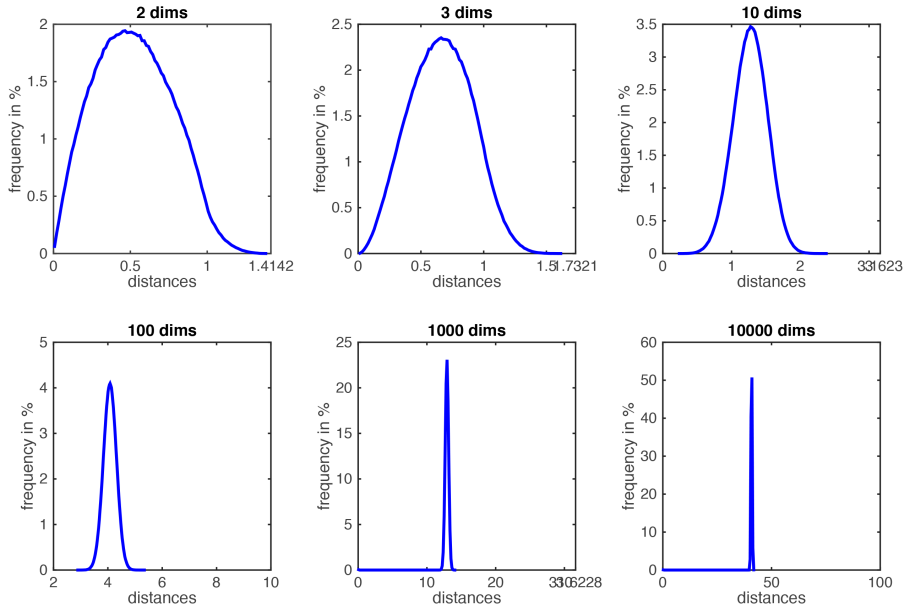
Figure 7.9: Overfitting a variational auto-encoder to the high dimensional signed distance

(a) Epoch 0 (b) Epoch 2000 (c) Epoch 5000



The curse of dimensionality provides a scaling law of the training error with data-dimensionality. This scaling law is exponential, indicating that the 600000 dimensional input vectors make convergence difficult. Consider the auto-encoder with a 600000 dimensional vector as input and output respectively. Assume two samples in the training data. To evaluate the model we compute the eucliden distance between input and output pair. However, as the dimensionality grows, it can be shown that this distance is equal. Figure 7.10 displays the euclidean distance between points drawn uniformly from space. With increasing dimensions, the distances converge to higher and more similar values.

Figure 7.10: Distances in high dimensions



For training on a single data-sample this is less problematic since the model does not need to find a compromise between different samples. For two samples, it needs to distinguish between the to, but creating a hyperplane that separates in high dimensions is difficult since nearly all planes have the same distance to both points.

The underlying signed distance function is a continuous function. This continuous function

can be represented as a linear combination of weighted basis functions. This methodology of representing lines, surfaces, or $3D$-functions in terms of continuous basis function coefficients is called B-spline representation. The continuous function is mapped to few coefficients. This representation is advantageous because it alleviates problem of high-dimensions and incorporates a prior on the function smoothness.

By mapping the function to coefficients of basis splines, the dimensionality can be reduced by a factor of 100. Figure 7.15, displays the over-fitting process on the basis-spline representation. Contrary to the full-dimensional signed distance representation, it is possible to over-fit an auto-encoder to all 44 data-samples.

7.3.1 Bezier functions

Bezier functions can be used to represent lines in space as linear combination of basis functions. Consider a $1D$-function and discrete d control points $p_i = x_i$ for $i = 0, ..., d-1$. Consider the parameterization of the Bernstein polynomials, where $u \in [0, 1]$ and d corresponds to the degree

$$B_{i,d}(u) = \binom{d}{i}(1-u)^{d-i}u^i.$$

If $d = 4$, then the cubic bernstein polynomials are defined as

$$B_0(u) = (1-u)^3,$$

$$B_3(u) = 3u(1-u)^2,$$

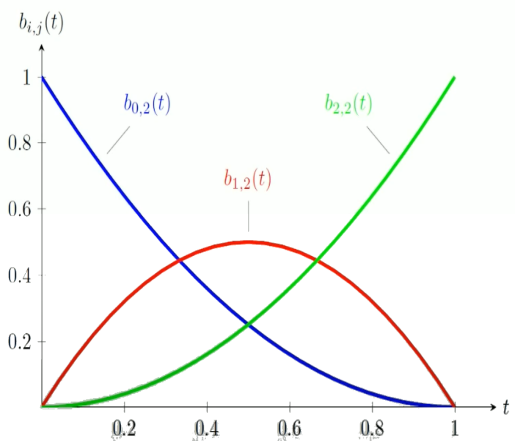
$$B_2(u) = 3u^2(1-u),$$

and

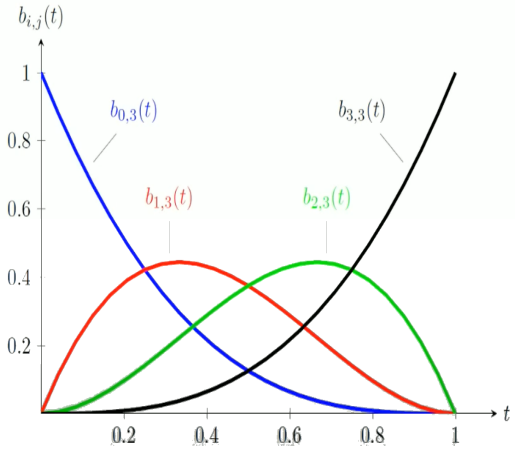
$$B_3(u) = u^3.$$

Figure 7.11a and 7.11b respectively display quadratic and cubic bernstein polynomials. A function is approximated by assigning weights to the control points of each polynomial and summing them. The resulting curve is called a Bezier-curve. There are as many Bernstein polynomials as there are control points. Furthermore, each Bernstein polynomial influences each control point. For each control point there is one bernstein polynomial. The construction of a degree d Bezier-curve requires $d+1$ control points.

Figure 7.11: Bernstein polynomials



(a) quadratic Bernstein polynomials



(b) cubic Bernstein polynomials

Note, that, for cubic bernstein polynomials, we can write

$$B(u) = [B_0, ..., B_3](u) = [u^3, u^2, u, 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = u^T M.$$

The bernstein polynomials are used as a basis to define a bezier-curve as

$$\mathbf{f}(u) = \sum_{i=0}^{d} \mathbf{c}_i \, B_{i,d}(u)$$

which can be written in matrix form

$$f(u) = Bc = u^T M c,$$

where $c = [c_0, c_1, ..., c_d]^T$, $u = [1, u^2, ..., u^d]$ and M is a matrix of scalars. Remember that d is the degree of the basis functions. For most practical purposed $d = 3$ is sufficient. A cubic Bezier curve can be written as

$$f(u) = (1-u)^3 c_0 - 3t(1-t)^2 c_1 + 3t^2(1-t)c_2 + t^3 c_3$$

This provides a description for how a cubic $1D$ function can be written as linear combination of basis functions.

Assume there is a function $r(u)$ which is provided as vector of discrete values at n points, i.e, $r \in \mathbb{R}^n$.

This function can be approximated by

$$f(u) = \sum_{i=0}^{d} c_i B_{i,d}(u) = Bc$$

by determining the coefficients $c = [c_0, c_1, ..., c_d]^T$ which minimize the mean squared error

$$E(c) = \sum_{j=0}^{n} \left(r[j] - \sum_{i=0}^{d} c_i B_{i,d} \right)^2 \tag{7.1}$$

or in matrix notation

$$E(c) = (r - Bc)^2.$$

Note, that in the current framework, the number of control points determines the degree of the polynomial. Furthermore, each polynomial has global influence. $B_0(u)$ affects the whole range of u, as does any other basis function. Similarly, the same concept can be applied to $2D$-surfaces. Herefore, consider m_x control points in x-directions, m_y control points in y-direction and a total of $m = m_x * m_i$ control points. Then,

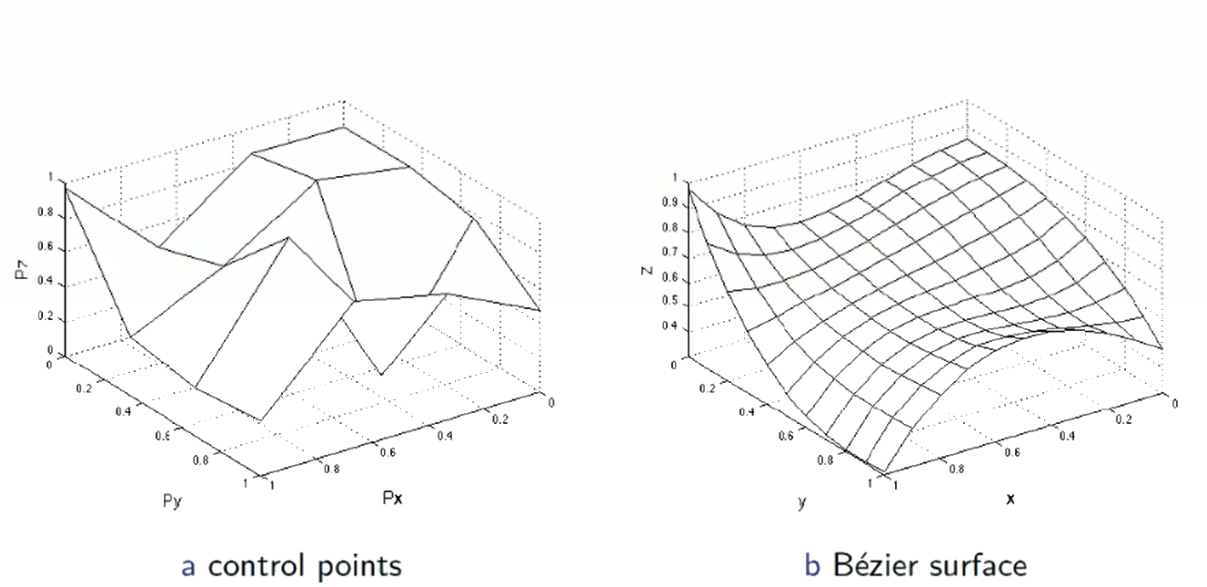
$$B_i(u) = B_i^x(u) * B_i^y(u)$$

and the surface is approximated by

$$f(u) = \sum_{i=0}^{m-1} c_i B_i(u) = Bc$$

by minimizing the loss of equation 7.1 Figure 7.12 displays control-points and a constructed $2D$-surface respectively.

Figure 7.12: A Bezier surface



a control points b Bézier surface

7.3.2 Basis splines

The basis-spline approach approximates functions by the linear combination of section-wise defined basis functions. In essence, each basis function only has a finite range of influence. This has the advantage of increased control and removes the problematic of increasing basis function degree with increasing control points. However, this introduces continuity issues and requires a specific, recursive definition of the basis functions.

A k degree B-spline curve defined by $n+1$ control points will consist of $n-k+1$ Bezier curves. For example, a cubic B-spline defined by 6 control points $P_0, ..., P_6$ consists of 3 Bezier curves. The support of each basis-function is defined through a knot vector $k \in m+d+1$, where m represents the number of control points and d the degree of the basis functions. The knot vector contains values in the range $[u_{min}, u_{max}$ in a non-decreasing manner. Typically, as in section 7.3.1, $u_{min} = 0$ and $u_{max} = 1$, but other ranges are also possible. The first and last element have multiplicity $d+1$, meaning, that the value u_{min} is repeated in index 0 to $d-1$ and u_{max} in index $m+d+1$ to $m+1$. All other values might have equal difference to each other and multiplicity one. This is not necessary but makes application easier. Summarized, a knot vector $k = [u_0, u_1, ...u_(m+d+1)]$ is used to determine the local support of each basis function B_i with $i \in [0, .., m-1]$. Often u_0 to u_{d-1} have value u_{min} and u_{m+d+1} to u_{m+1} have value u_{max}. A special set of basis functions is used to fulfill continuity on all control points. These functions are calculated recursively via the Cox-de-Boor formula. These basis-functions are non-negative and have local support.

One starts with a base case (degree 0) functions defined as

$$B_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

and the recursively constructs higher degree functions via

$$B_{i,d}(u) = \frac{u - u_i}{u_{i+d} - u_i} N_{i,d-1}(u) + \frac{u_{i+d+1} - u}{u_{i+d+1} - u_{i+1}} N_{i+1,d-1}(u),$$

where d denotes the degree. Note, that if $u \notin [u_i, u_{i+1}]$, then $N_{i,0}(u) = 0$. This ensures that the basis function of the i-th control point only has local support. Through this, the knot-vector influences the support region of each basis function $B_{i,d}(u)$. At the boundaries, values

are repreated for continuity. Figure 7.13 displays a set of basis functions with local support and their knot values. For each control point we obtain one Bezier function, i.e. $i \in [0, m-1]$ As for Bezier functions, the curve in this basis is defined as

$$f(u) = \sum_{i=0}^{d} c_i B_{i,d}(u) = Bc$$

Again, a function $r(u)$ which is provided as vector of discrete values at n points, i.e, $r \in \mathbb{R}^n$ can be approximated by

$$f(u) = \sum_{i=0}^{m-1} c_i B_{i,d}(u) = Bc$$

by determining the coefficients $c = [c_0, c_1, ..., c_d]^T$ which minimize the mean squared error

$$E(c) = \sum_{j=0}^{n} \left(r[j] - \sum_{i=0}^{m-1} c_i B_{i,d} \right)^2$$

or in matrix notation

$$E(c) = (r - Bc)^2.$$

The main difference to the Bezier function approach lies in the flexibility gained by being able to choose an arbitrary number of control points $m \geq 4$ to represent curves as combinations of solely cubic basis functions. The local support property introduces a sparse structure into the B-matrix, which can be exploited for computational efficiency. Furthermore, local regions can be manipulated without creating global effects.

7.3.3 Application to the signed distance function

Sections 7.3.1 and 7.3.2 provide some mathematical background for basis function representations of $1D$ functions. The extension to multiple dimensions is achieved by defining multiple $1D$ basis functions in basis directions, i.e. for $3D$ functions one defines one basis-spline function for each x, y, z in cartesian or r, θ, z in cylindrical coordinates. Each of these basis-functions requires a certain set of control points. Assume for example m_r, m_θ, m_z control-points in each direction. The $3D$-domain is thus discretized into $m = m_r * m_\theta * m_z$ control points whose values define the m-dimensional vector c. The basis in $3D$ is defined as a product of the basis-spline functions, each constructed via the Cox-de-Boor recursion (for brevity the index d for basis degree is omitted), to

$$B_i(u) = B_i^r(u) * B_i^\theta(u) * B_i^z(u)$$

and the function as a linear combination

$$f(u) = \sum_{i=0}^{m-1} c_i B_i(u) = Bc.$$

Given a vector s of dimension n, a discretization of the disk domain is chosen. This discretization defines the number of control points m. The basis functions and the matrix B are computed. B has dimension $m \times n$, where $B_{i,j} = B_i(r_j, \theta_j, z_j)$. Then, the mean squared error can be constructed

$$E(c) = (s - Bc)^2.$$

Figure 7.13: Basis functions and knot values

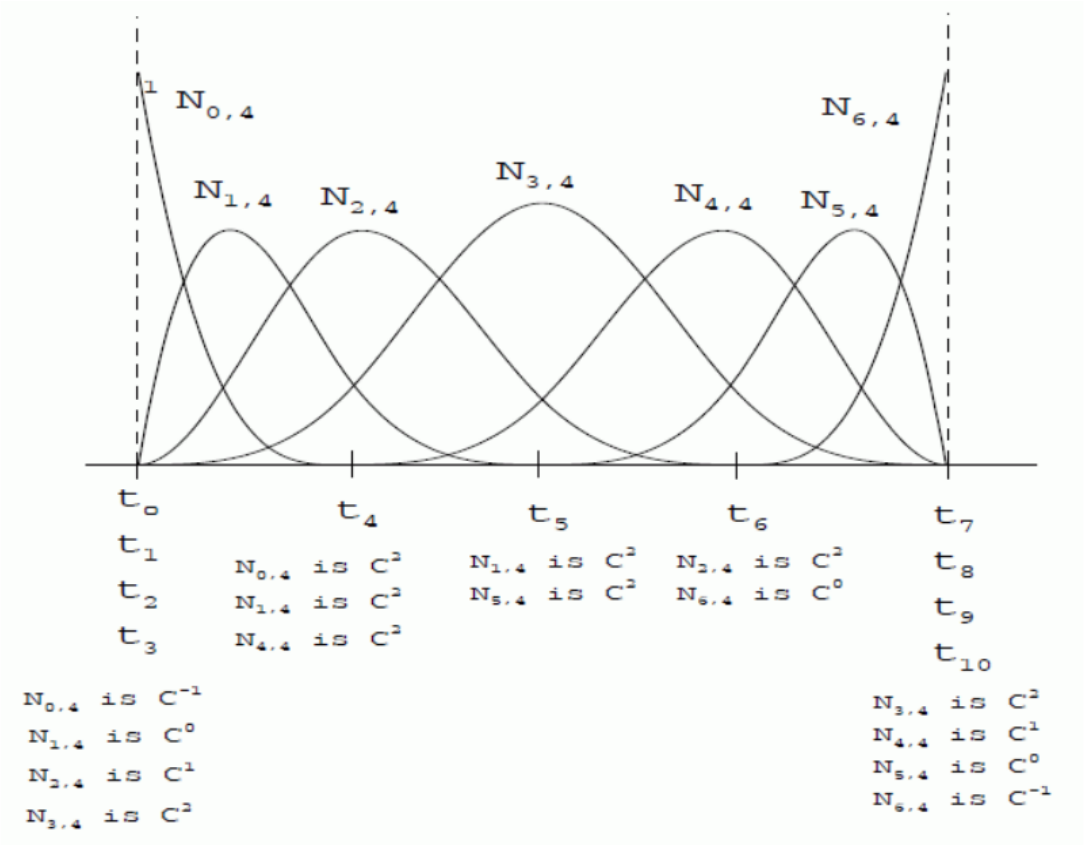


Figure 1.10: *An order four B-spline basis functions with uniform knot vector*

Minimizing with respect to $c \in \mathbb{R}^m$ yields the optimal weights at each control points. The analytical solution for the optimal parameter values is

$$c = (B^T B)^{-1} B^T s.$$

Once c has been obtained, *sdf* can be reconstructed by

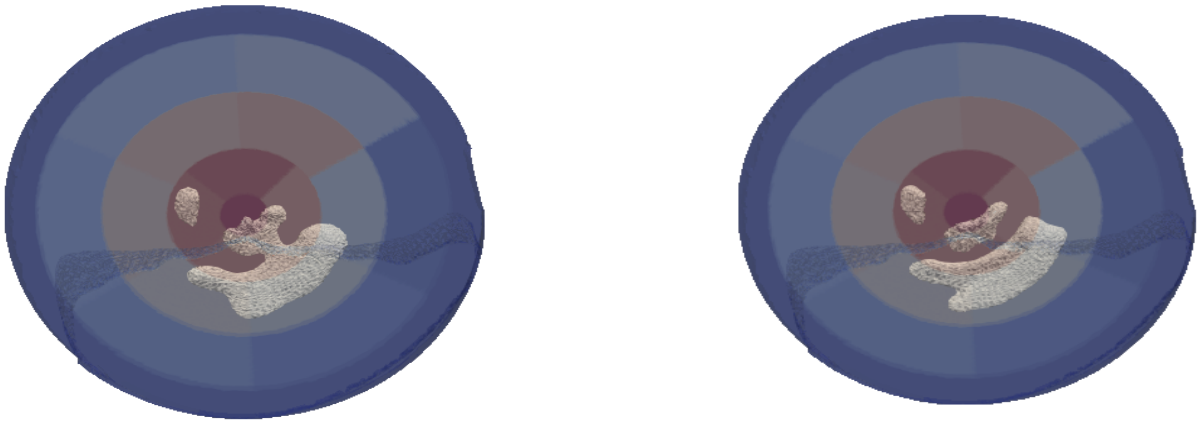
$$s = Bc.$$

The s vector has been mapped to the c-vector by projection onto the basis splines. To evaluate the required number of control points in each direction, we visually expect the reconstructions and compare the results to the originals. Figure 7.14 displays a scar projected onto basis splines with varying number of control points. Less control points correspond to a stronger dimensionality reduction. Furthermore, the basis spline projection creates a smooth surface, which alleviates some of the processing issues explaines in section 6.5. However, it can also have undesired effects. Figure 7.14a displays a bridge between two usually disconnected parts. While the separation is most likely due to the processing, the bridge between the two disjoint parts is not plausible physically. We choose a resolution which reconstructs the signed distance function nearly lossless for all scars and consider processing artifacts as ground truth.

7.3.4 Results

During computation, we store the B and $(B^T B)^{-1}$ and matrices in a sparse format. For our purposes we choose 30, 50 and 5 control points in θ, ρ, z coordinates respectively. This corresponds to a relatively exact reconstruction of the original signed distance function, including

Figure 7.14: Projection onto basis splines

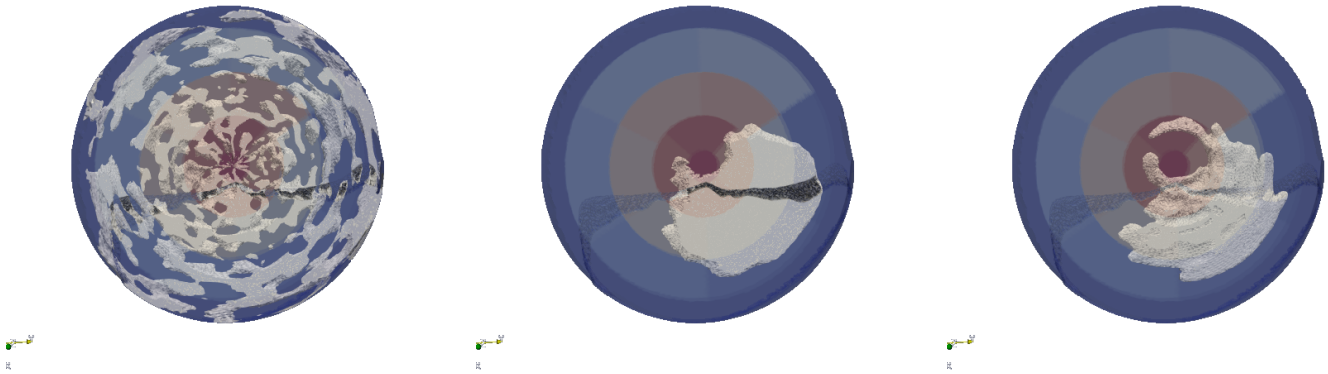


(a) low resolution (b) high resolution

all processing artifacts. This projection reduces dimensionality by a factor of 100 and enables over-fitting to the whole dataset. Figure 7.15 visualizes model output snapshots during the training process. Whereas training on the signed distance function (see figure 7.9) adapts every discrete point individually, training on the basis spline representation is more coarse grain and enforces function smoothness.

Figure 7.15: Overfitting a variational auto-encoder to the b-spline coefficients

(a) Epoch 0 (b) Epoch 2000 (c) Epoch 5000



Chapter 8

Data and Generations

8.1 Naive PCA

Principal component analysis is a very popular, linear methodology to quantify the main variability directions of data. These main variability directions can be used to create new samples which adhere to the same distribution. For it to be meaningful point-to-point correspondence is required, This is available for the signed-distance functions. Due to the different topologies of the scar meshes, it can not be obtained trivially. The point-to-point correspondence of the signed distance function enables linear interpolation between data-samples. However, it is not given that the result is a plausible scar, only that it is a signed distance function drawn from the same distribution.

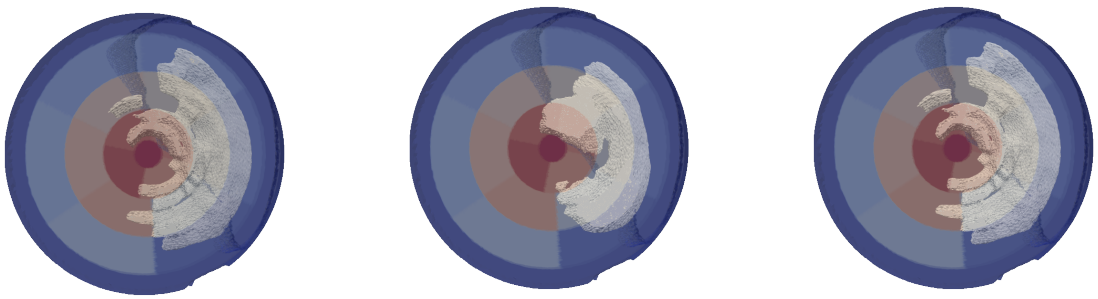
Fig. 8.1 displays a scar shape and its reconstruction using 8 and 43 principal components. As expected, using 8 principal components keeps the main form but disposes of small variability's, Using 43 components represents the shape nearly lossless.

Figure 8.1: Principal component reconstructions

(a) Original scar shape

(b) Reconstruction with 8 principal directions

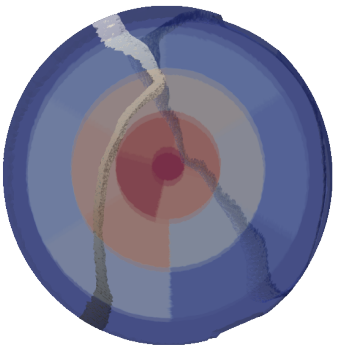
(c) Reconstruction with 43 principal directions



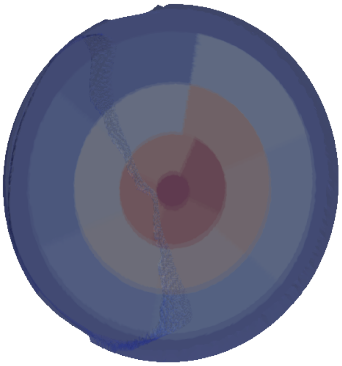
We use the 43 principal components to create new scars. Not all generated shapes are plausible. In fact many are not. Fig 8.2a shows a completely nonphysical shape and Fig. 8.2b corresponds to a purely positive signed distance function from which no shape can be recovered. This indicates, that the linear representation of the shape space in inaccurate.

Figure 8.2: Principal component generations

(a) Generation 1



(b) Generation 2



8.2 Naive VAE

The model is trained with a Kl penalty term of 0.0001 on 44 shapes represented by a 7500 dimensional vector corresponding to basis spline coefficients. For the reconstruction loss we employ the mean squared error of the signed distance functions, corresponding to the model input and model output

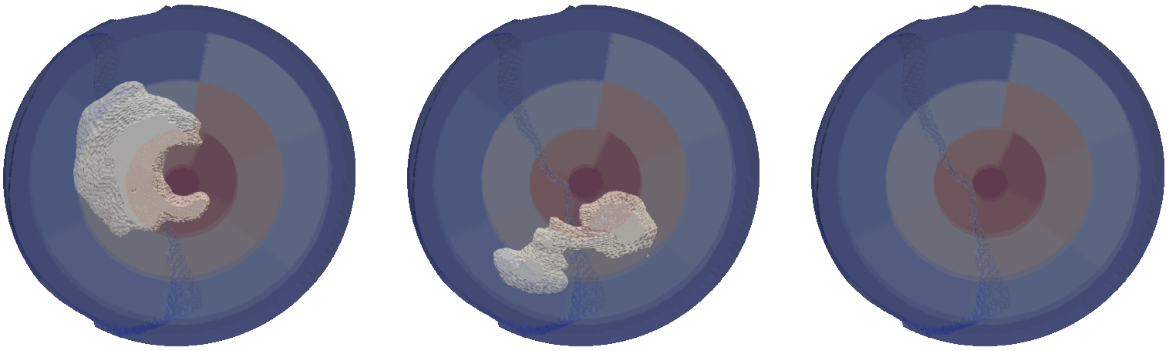
$$\text{Reconstruction}_{\text{Loss}}(\text{output}_{\text{generation}},\ \text{input}_{\text{original}}) = (\text{output}_{\text{generation}} -\ \text{input}_{\text{original}})^2 \tag{8.1}$$

After the training, the encoder is disposed of. Latent variables are sampled from the Gaussian distribution and decoded. The decoder outputs correspond to generated basis spline coefficients, which are mapped to an implicit signed distance functions. This function should represent plausible scar shapes by ISO-surface extraction at the zero-level-set.

Figure 8.4 displays some of the generated shapes. Fig. 8.3a displays a shape which, by only considering the visual aspect, might be considered plausible. However, figure 8.3b and especially figure 8.3c are questionable generations. The scars in the training data revolve around the disk center and are either circular, half-mood shaped or elliptical. Figure 8.3b does not display these features. Figure 8.3c does represent any scar shape at all. It corresponds to a purely positive signed distance output.

Figure 8.3: Generations (VAE trained on basis spline coefficients)

(a) a Vae generation (b) a Vae generation (c) a Vae generation



It is likely, that these issues are due to the low data-sample size. Consider the extreme example, where the only two training-samples are the shapes displayed in figure 8.4a and 8.4b. Note, that while the shapes are similar, they are opposed in position. Consider training a variational auto-encoder on these two shapes. Due to the KL-divergence term, the encoder attempts to fit the latent-distributions as close the isotropic Gaussian as possible. Since we have two samples, the corresponding Gaussian latent-distribution should have at least two peaks. Sometimes, a latent variable will be decoded which has equal probability of belonging to either shape. The decoder will aim to generate a sample which minimizes the reconstruction loss to both shapes. Since we employ the mean-squared error of the input and output signed distance function, the optimal reconstruction which can correspond to both shapes is a purely positive signed distance function. This is a non-physical shape.

Figure 8.5 displays the isotropic gaussian and the latent distribution which correspond to each of the data-samples. Assume a dataset of only two shapes. The latent which is in the middle, corresponds to a generation which minimizes the reconstruction loss to both shapes. If the data-samples correspond to figure 8.4a and 8.4b, then this generation is a purely positive signed distance function. If the data-samples correspond to figure 8.4a and 8.4c, then the this generation is a plausible shape which can be understood as some sort of nonlinear interpolation between the two.

The best reconstructor minimizes the overlap of these distributions so that the overlapping values do not occur and become meaningless for the decoder. This is not desired for generation. All values in the isometric Gaussian should correspond to plausible generations. The example on three shapes illustrates how the lack of overlap in the training data can lead to un-physical shapes for some latent codes.

Figure 8.4: Data-sample examples

(a) Sample 1 (b) Sample 2 (c) Sample 3

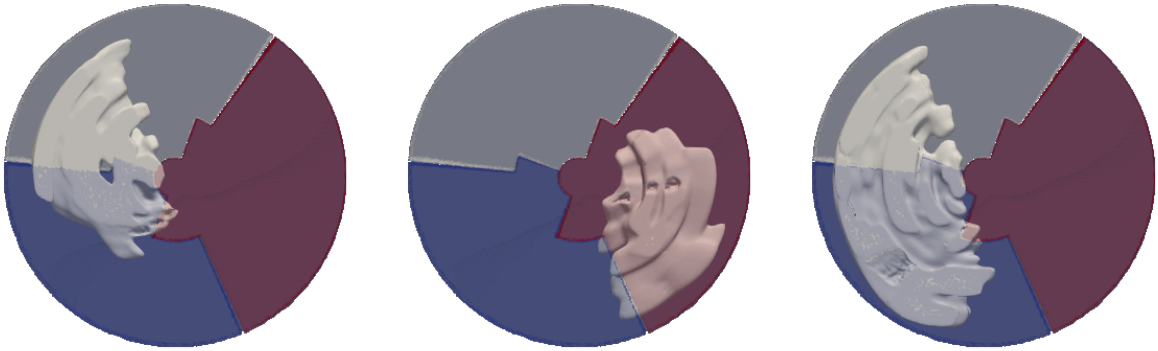
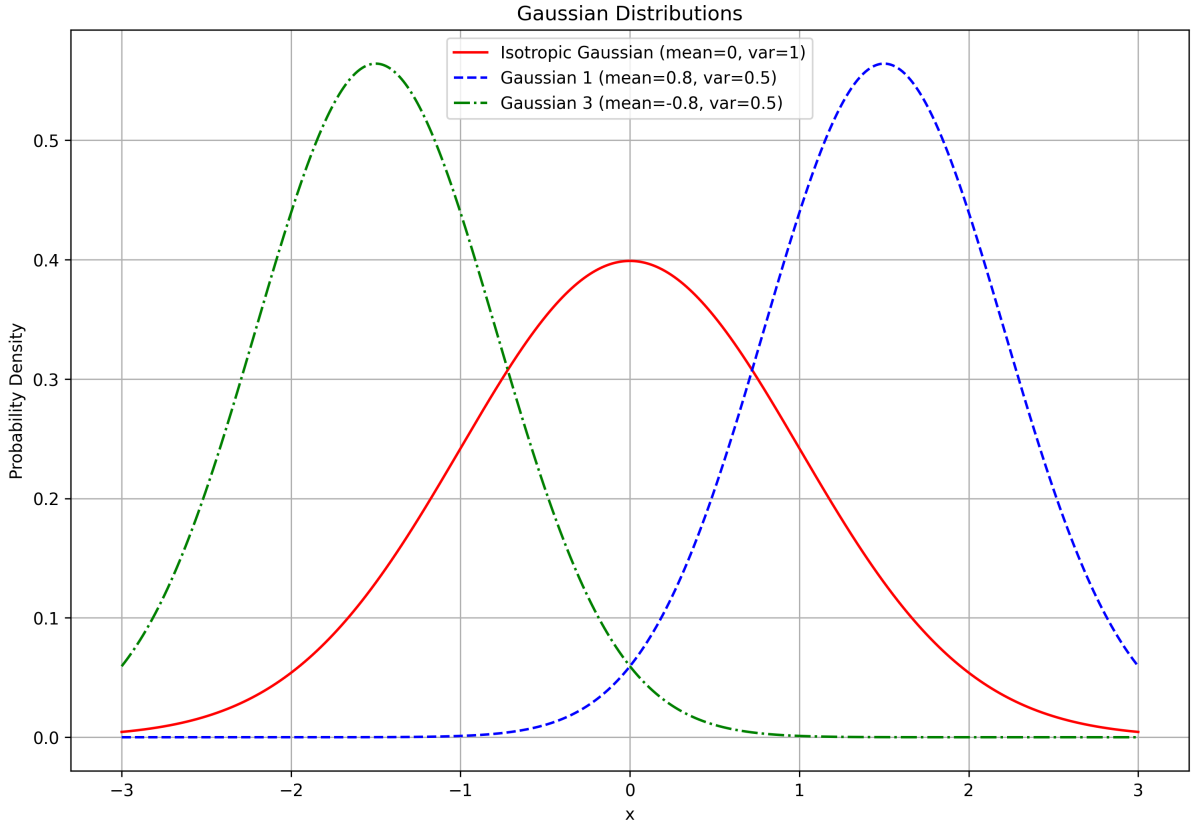


Figure 8.5: Latent space visualizations



8.3 Shape-position disentanglement

In our case, only 44 samples are available. This low data density leads to problems with lacking overlap as explained in section 8.2. As a solution we perform a pre-processing step which aligns the shapes and increases the respective overlap. In doing so we consider shape and position as independent from each other. Most likely this is not true. The affected coronary is correlated with the location, size and shape of the scar. However, this assumption alleviates the limitations posed by low data availability and low spatial density .

We transform the shapes as to increase the overlap. In a first step, we align the center of gravity and principal inertial axis of the shapes via translation and rotation. In a second step we perform a diffeomorphic mapping, which allows for some bending and scaling. The transformation parameters are choosen so that they minimize an energy-function of the target-shape and deforming shape. This energy-function considers a distance between shapes and a penalty term for strong bending and scaling. Possible functions for the shape distance are the Hausdorff-distance, the Wasserstein distance and the mutual information.

8.3.1 Mutual Information and Cross Correlation

Consider two shapes $X = \{x_i\}$ and $Y = \{y_i\}$. Consider the distribution

$$p(x) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(x|x_i, \Sigma_i).$$

Recall that

$$\mathcal{N}(x|x_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}}\exp(-\frac{1}{2}(x - x_i)^T\Sigma_i^{-1}(x - x_i)).$$

Thus

$$p(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}}\exp(-\frac{1}{2}(x - x_i)^T\Sigma_i^{-1}(x - x_i)).$$

Let us use a Gaussian Kernel

$$K_h(x - x_i) = \frac{1}{\sqrt{2\pi}h}\exp\frac{-||x - x_i||^2}{2h^2}$$

so that

$$p(x) = \frac{1}{N}\sum_{i=1}^{N}K_h(x - x_i).$$

This corresponds to assuming a diagonal covariance matrix $\Sigma_i = h^2 I$. High values for h mean the data is not trusted and low values mean the contrary. This is similar to a RBF-kernel, with the difference of the normalization constant $\frac{1}{\sqrt{2\pi}h}$. This defines the two probability densities $p(x)$ and $q(y)$ respectively. We can now consider the shape X and Y as being drawn from their respective distributions. The mutual information is defined as

$$I(X;Y) = H(X) - H(X|Y), \tag{8.2}$$

where

$$H(X) = \sum_{i}^{N}p(x_i)\log(p(x_i))$$

and $H(X|Y) = H(X,Y) - H(Y)$. We can assume that $p(x)$ is independent from $p(y)$ so that $p(x,y) = p(x)p(y)$. This defined the "distance" between both shapes in terms of mutual information. Mutual Information is not symmetric and thus not a real distance. It quantifies the

statistical similarity. This approach is robust to noise and is suited to different modalities since it considers the statistics of each shape, but not directly the shape itself. It is used ubiquitously in image registration. However, it can be computationally expensive.

Given the density functions one can also use cross correlation as similarity measure. It is defined as

$$C(p,q) = -\sum_{i=1}^{N} p(x_i)\log(q(x_i))dx. \tag{8.3}$$

It quantifies the "cost" of expressing $p(x)$ using $q(x)$, i.e. how well $q(x)$ can encode $p(x)$.

In image registration, mutual information is used for multi-modal data. Since it measures a statistical information divergence it is suited for images where pixel intensities vary. Cross-entropy on the other hand requires equal intensities since it directly compares the two distribution at direct pixel positions. Certain tissue structures appear bright in MRI scans but dark in CT scans. Normalization can deal with the intensity range but not with modality-specific differences. Thus mutual information is the preferred choice for a similarity loss.

8.3.2 Hausdorff distance

The Hausdorff Distance measures the distance between to subsets of a metric space. It corresponds to the biggest smallest distance between the elements of the respective subsets.

Let $(\mathcal{X}, d)$ be a metric space, and let $A, B \subseteq \mathcal{X}$. The *Hausdorff distance* $d_H(A,B)$ is defined as

$$d_H(A,B) = \max\left\{\sup_{a\in A} d(a,B),\ \sup_{b\in B} d(A,b)\right\}, \tag{8.4}$$

where $d(A,b) = \inf_{a\in A} d(a,b)$. A brute-force algorithm would be

```
1:  h ← 0
2:  for each point a_i of A do
3:      shortest ← ∞
4:      for each point b_j of B do
5:          d_ij ← d(a_i, b_j)
6:          if d_ij < shortest then
7:              shortest ← d_ij
8:          end if
9:      end for
10:     if shortest > h then
11:         h ← shortest
12:     end if
13: end for
```

https://cgm.cs.mcgill.ca/ godfried/teaching/cg-projects/98/normand/main.html

So for each point in A the smallest distance to B is computed and in the end, the maximum is picked out of all smallest distances. It is a relatively simple distance, straightforward to compute and easily interpreted as measuring the worst-case deviation between two sets. Due to this it is sensitive to outliers, if only one single element has a high minimal distance, the Hausdorff distance corresponds to it, since it focuses on extremes.

8.3.3 Wasserstein distance

The Wasserstein Distance is a very widely used metric which quantifies how difficult it is to deform one distribution $q(x)$ into another distribution $p(x)$. It is deeply related to optimal transport and can be interpreted as the minimal cost for shaping one form into another. Furthermore, it is symmetric and satisfies the triangle inequality. It is very widely used and can be a good alternative to other dissimilarity measures, such as the KL-divergence. In contrast to the information-theoretical dissimilarity (KL divergence) it measures the difference between distributions in a geometric/spatial sense.

Consider a space of probability measures $P(\Omega)$ and formalize Consider two probability densities $p(x)$ and $q(x)$ defining the distributions P and Q, so that $X \sim P$ and $Y \sim Q$ with $X, Y \in \mathbb{R}^d$. Consider the optimal transport map $T : \mathbb{R}^d \to \mathbb{R}^d$, where the distribution of $T(X)$ is called push-forward ($T_{\#}P$) of P, so that

$$T_{\#}P(A) = P(\{x : T(x) \in A\}) = P(T^{-1}(A)).$$

Here $T_{\#}P$ is a new probability distribution (measure) induced by P through the mapping T. $T_{\#}P(A)$ represents the probability that $T(X)$ in in A. $\{x : T(x) \in A\}$ is the set of all points, which mapped though T end up in A. This set of points can also be written as $T^{-1}(A))$.

The optimal transport distance is

$$\inf_T \int ||x - T(x)||^p dP(x)$$

so that $T_{\#}P(A) = Q$. If the minimizer $T*(\cdot)$ exists it is called optimal transport map. The map $T_t(x) = (1-t)x + tT*(x)$ is a geodesic connecting P to Q. In this formulation, the minimizer might not exist since the measure (equivalent to the mass) can not be splitted. A way to alleviate this is by a soft formulation which introduces a joint measure $dJ(x, y)$ and defines the Wasserstein distance as

$$W_p(P, Q) = (\inf_{J \in \mathcal{J}(P,Q)} \int_{x,y} ||x - y||^p dJ(x, y))^{\frac{1}{p}} \tag{8.5}$$

so that $\int_y J(x, y)dy = P$ and $\int_x J(x, y)dy = Q$. The minimizer $J^*(\cdot)$ is called the optimal transport plan. If $T : X \to X$, i.e. no splitting of mass occurs (input and output measure coincide), it is called the optimal transport map. For many applications the input and output measures are equal. Note that minimizing the Wasserstein distance returns the minimal cost, as well as a corresponding transport plan. This makes it a very attractive method for image morphing, image registration and further. For example, consider two greyscale images in $2D$. Then the optimal transport map is a $2D$ vector-field, where each point receives a $2D$-vector with the corresponding displacement.

In a discrete setting we have samples x_i and y_j which occur with probability p_i and q_j respectively. Equation 8.8 can be written as

$$W_p(P, Q) = \min_{J \in \mathcal{J}(P,Q)} \sum_i \sum_j ||x_i - y_j||^p J(x_i, y_j) \tag{8.6}$$

so that $\sum_j J(x_i, y_j) = p_i$, $\sum_i J(x_i, y_j) = q_j$ and $J(x_i, y_j) \geq 0$.

Interesting cases are $p = 1$ (Earth-Mover distance) and $p = 2$. For $p = 2$, it can be shown that there exists a unique convex function ϕ so that $J^*(\cdot) = \nabla\phi$. Furthermore, for $p = 2$ the metric space $\{P(\Omega), W_p\}$ has the structure of a Riemannian manifold as pointed out by add cite.

By converting the primal problem 8.8 into its dual, it can be shown that

$$W_p(P,Q) = (\sup_{\Phi,\Psi} \int_y \Phi(y)dQ(y) - \int_x \Psi(x)dP(x)), \tag{8.7}$$

where For $p = 1$, one can write

$$W_p(P,Q) = (\sup_{f} \int_x f(x)dQ(x) - \int_x f(x)dP(x)) \tag{8.8}$$

The Wasserstein distance can be used to define "typical" distributions from a set of distributions $P_1, P_2, ..., P_N$, also called the barycenter. This is the distribution P_{center}, so that

$P_{center} = (_P\sum_{i=1}^{N} W_p(P,P_i)$.(8.9) An intuition on its computation ban be found in cite.

Also interesting it the notion of distances between P_0 and P_1. Consider a map $c(t) : [0,1] \rightarrow P(\Omega)$ so that $c(0) = P_0$ and $c(1) = P_1$. Then

$$c(t) = (1-t)P_0 + tP_1$$

is the geodesic connecting P_0 and P_1. There exists a path $c(t)$ so that its length $L(c) = W_p(P_0, P_1)$ and

$$P_t = F_{t\#}J$$

where J is the optimal coupling and $F_t(x,y) = (1-t)x + ty$. This allows for the interpolation between two distributions following the Wasserstein geodesic. A Linear Optimal Transportation Framework for Quantifying and Visualizing Variations in Sets of Images present a method which projects these geodesics (nominally located in a curved, nonlinear space) to a linear embedding via linear optimal transport while preserving distances and angles. Then linear statistical analysis with PCA is used in the embedding space to quantify variations. However, they do not mention extensions to image generation.

8.3.4 Energy function

The approach is similar to diffeomorphic image registration. Analogously, we search for a transformation which maps one shape to another and constrain the type of transformation. We choose a loss, or energy-function which considers the chosen shape distance. In our case, we choose the Hausdorff distance since it is computationally cheap and easy to implement. However, it is not differentiable. waiiit did we really use Hausdorff? it is not differentiable, nor convex and thus not suited for gradient based optimization. Furthermore, we add a term which penalizes the type of transformation. In diffeomorphic image registration this penalty is constructed to be a gradient regularization (imposing a smoothness constraint), a elastic regularization (penalizing bending and ensuring physicality) and a diffusion regularization (penalizing scaling). These terms are chosen to ensure diffeomorphic properties.

We choose a transformation which bijectively maps points in the disk to points in the disk. It is defined in cylindrical coordinates, acting independently in each of the coordinates,

$$(\rho, \theta, z) \mapsto f(\rho, \theta, z) = (f_\rho(\rho), f_\theta(\theta), f_z(z))$$

The z coordinate is kept unchanged:

$$f_z(z) = z$$

and the transformations affecting the polar coordinates ($\rho' = f_\rho(\rho)$ and $\theta' = f_\theta(\theta)$) are defined implicitly by

$$2\pi\rho' - a\sin(2\pi\rho') - b\cos(2\pi\rho') + b = 2\pi\rho + a\sin(2\pi\rho) + b\cos(2\pi\rho) - b$$
$$\theta' - \mu\sin\theta' - \nu\cos\theta' = \theta + \mu\sin\theta + \nu\cos\theta + \alpha$$

Here, we have $\rho \in [0, 1]$ and θ is cyclic modulus 2π. This implicit definition is already bijective and smooth and thus diffeomorphic. However, we still employ penalty terms constrain the transformation.

The total energy-function results in

$$\mathcal{L}_{energy}(a, b) = \mathcal{L}_{similarity}(a, b) + \lambda\mathcal{L}_{regularisation}(a, b), \tag{8.10}$$

where $\mathcal{L}_{regularisation}$ is composed as the sum of the individual penalty terms, a, b, corresponds to the transformation parameters and $\mathcal{L}_{similarity}(a, b)$ measures similarity. Possible choices for it are, mutual information, cross correlation, Hausdorff distance, Wassersteinu distance and the mean squared error,

A possible term for gradient regularization is

$$\mathcal{L}_{\text{gradient}} = \int \left(\left(\frac{d\rho'}{d\rho} \right)^2 + \left(\frac{d\theta'}{d\theta} \right)^2 \right) d\rho d\theta,$$

which in discrete settings can be written as

$$\mathcal{L}_{\text{gradient}} = \sum_{i=1}^{N-1} \left(\left(\frac{\rho'_{i+1} - \rho'_i}{\rho_{i+1} - \rho_i} \right)^2 + \left(\frac{\theta'_{i+1} - \theta'_i}{\theta_{i+1} - \theta_i} \right)^2 \right) (\rho_{i+1} - \rho_i)(\theta_{i+1} - \theta_i).$$

A possible term for elastic regularization is

$$\mathcal{L}_{\text{elastic}} = R_{\text{elastic}} = \int \left(\left(\frac{d\rho'}{d\rho} - 1 \right)^2 + \left(\frac{d\theta'}{d\theta} - 1 \right)^2 \right) d\rho d\theta,$$

which in discrete settings can be written as

$$\mathcal{L}_{\text{elastic}} = \sum_{i=1}^{N-1} \left(\left(\frac{\rho'_{i+1} - \rho'_i}{\rho_{i+1} - \rho_i} - 1 \right)^2 + \left(\frac{\theta'_{i+1} - \theta'_i}{\theta_{i+1} - \theta_i} - 1 \right)^2 \right) (\rho_{i+1} - \rho_i)(\theta_{i+1} - \theta_i).$$

It penalizes deviations from the identity transform. A possible term for diffusion regularization is

$$\mathcal{L}_{\text{diffusion}} = R_{\text{diffusion}} = \int \left(\left(\frac{d^2\rho'}{d\rho^2} \right)^2 + \left(\frac{d^2\theta'}{d\theta^2} \right)^2 \right) d\rho d\theta,$$

which in discrete settings can be written as

$$\mathcal{L}_{\text{diffusion}} = \sum_{i=2}^{N-1} \left(\left(\frac{\rho'_{i+1} - 2\rho'_i + \rho'_{i-1}}{(\rho_{i+1} - \rho_i)^2} \right)^2 + \left(\frac{\theta'_{i+1} - 2\theta'_i + \theta'_{i-1}}{(\theta_{i+1} - \theta_i)^2} \right)^2 \right) (\rho_{i+1} - \rho_i)(\theta_{i+1} - \theta_i).$$

Each of these terms is weighted and summed to the similarity loss.

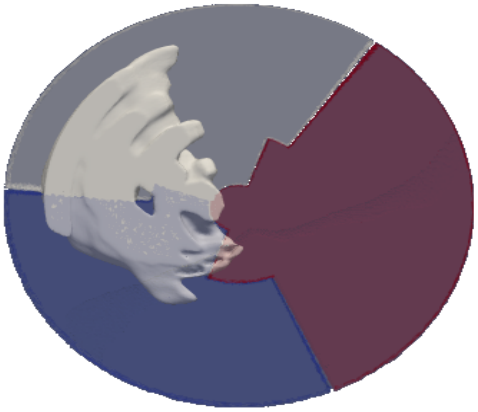
8.3.5 Transformations

We choose a reference shape and align all other shapes to it by choosing a transformation which minimizes the energy function. This is done via gradient based optimization.

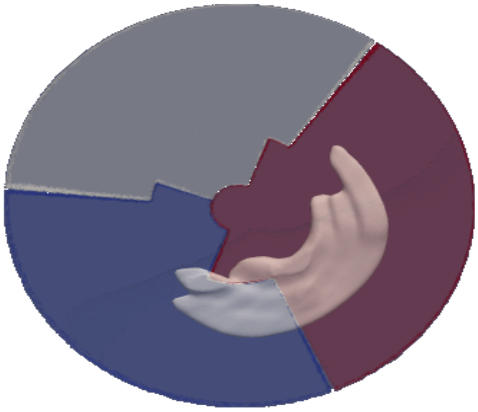
Figure 8.6a and 8.6b display the original shapes. Figure 8.6c and 8.6d display the transformed shapes. The main features are maintained, although the curvature of the scar in 8.6b is increased. The overlap of negative regions in the signed distance representation is increased. This enables the use of the mean squared error as reconstruction loss for plausible generations.

Figure 8.6: Normalization and shape

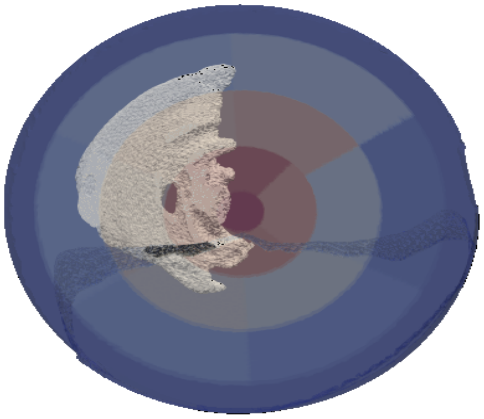
(a) Original shape 1



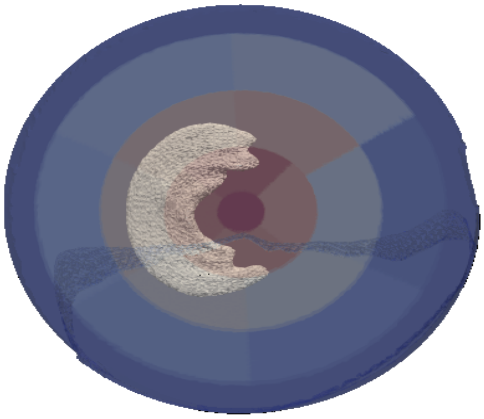
(b) Original shape 2



(c) Transformed shape 1

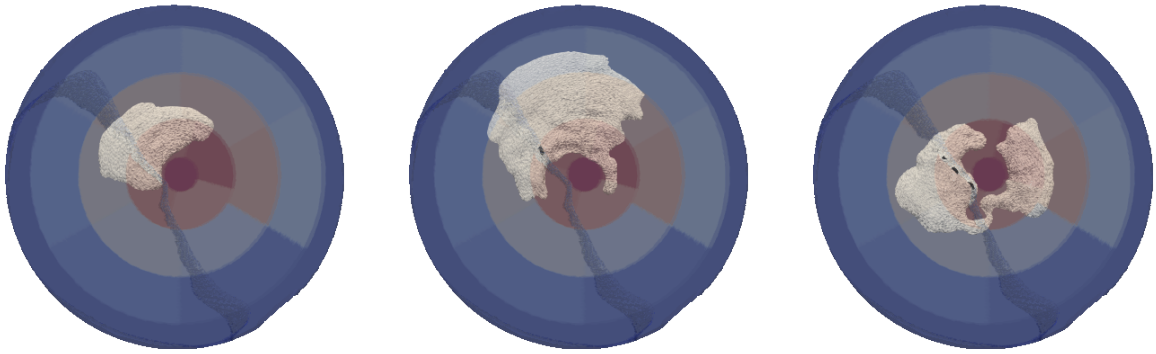


(d) Transformed shape 2



We retrain the variational auto-encoder on the position independent shapes obtained by the transformation. Figure 8.7 displays examples. Figure 8.7a and 8.7b can both be considered plausible scars. Figure 8.7c displays some nonphysical features. Our experiments show that the position normalization alleviates the issue of purely positive signed distance generations. However, the generations lack the more detailed features, such as the circular topology of some scars.

Figure 8.7: Generations on normalized shapes



(a) Generation 1 (b) Generation 2 (c) Generation 3

8.4 Manifold hypothesis

A common assumption is the manifold hypothesis add paper, which states that real world data tends to live on a low-dimensional manifold of some sorts. add paper highlight the reasonability of this assumption. Most manifolds relevant for physics and machine learning are locally linear (or at least locally isomorphic to Euclidean space). Figure 8.8 shows how such a manifold could look like.

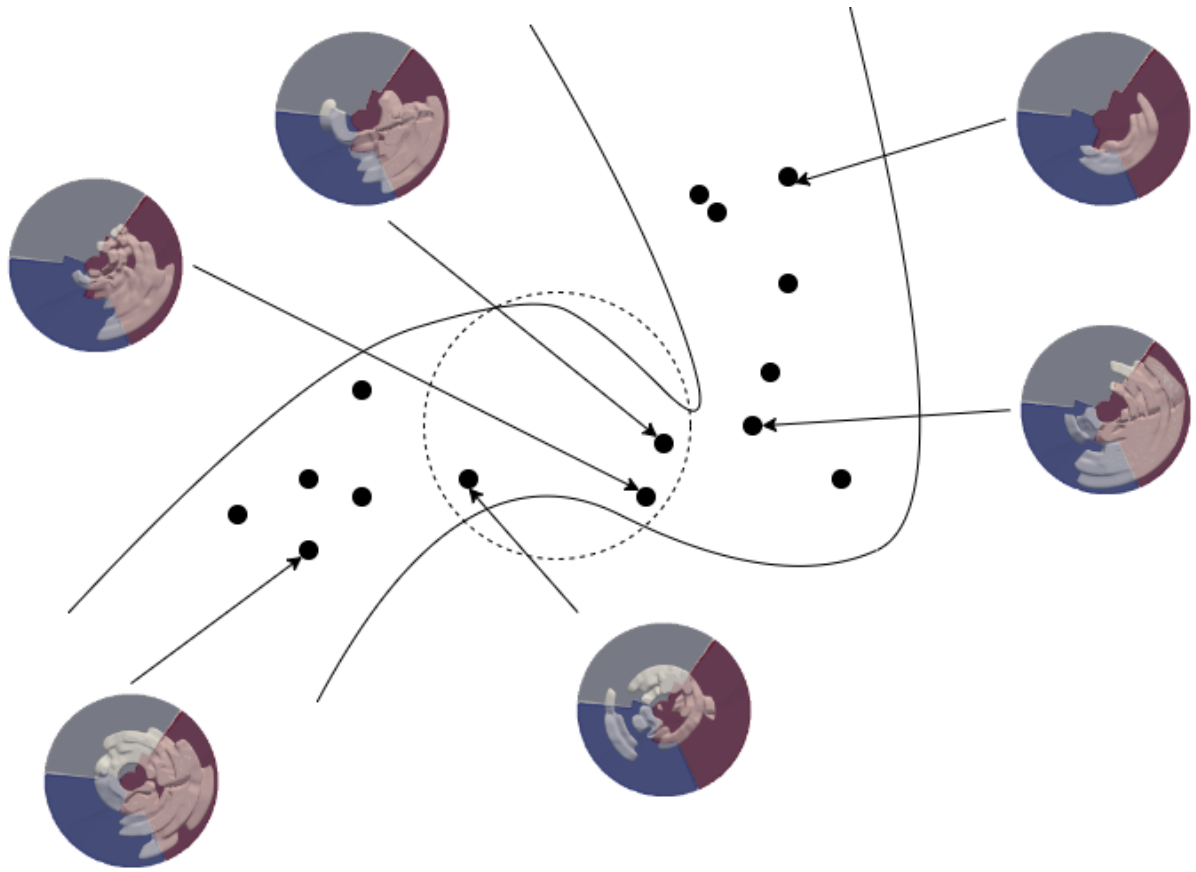


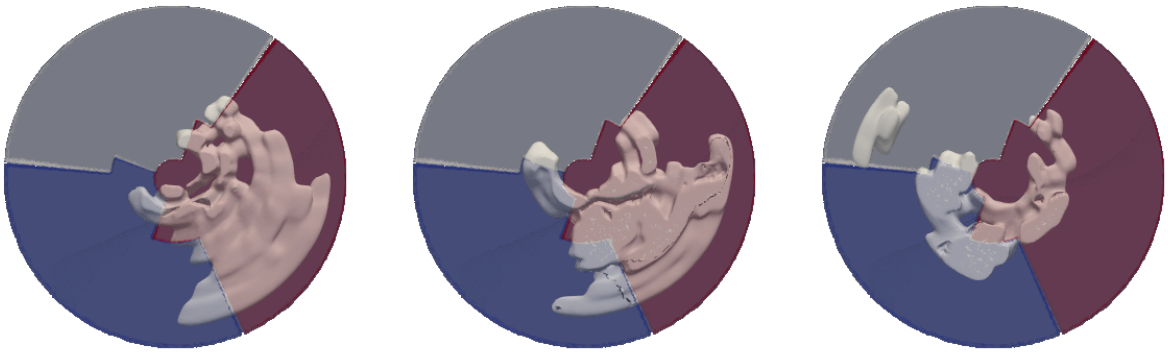
Figure 8.8: The shape manifold

We choose a set of similar shapes which might be located in a (linear) neighbourhood of each other. If the manifold hypothesis holds, we expect the data-space to be locally linear.

Then we should be able to perform PCA to model their data-distribution. Figure 8.11 displays a set of 3 shapes which might be in a neighbourhood of each other. The scar in figure 8.9c is more dissimilar than the other two, indicating a bigger geodesic distance.

Figure 8.9: Shapes from a neighbourhood

(a) Shape 1 (b) Shape 2 (c) Shape 3

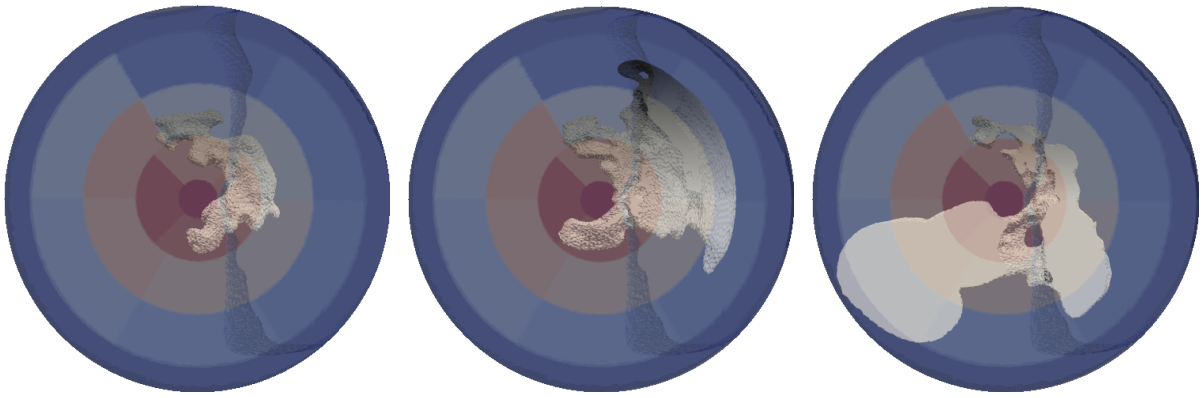


8.4.0.1 linear methods in neighbourhoods

To test the manifold hypothesis we choose samples which are similar to each other and employ PCA to sample from their distribution space. These samples are displayed in figure 8.11. If the manifold hypothesis hold, we expect these samples to be sensible, since the data-space should be linear locally. Figure **??** displays some generations. While the quality is not great, it is an improvement to naive PCA on the whole data-space. This indicates, that choosing similar neighbours can improve the generative quality. The generations displayed in figure 8.11a and 8.11b are good generations. The shape visualized in figure 8.11c displays a un-phyiscal part. This might be due to attempted interpolation between the relatively distant shapes of figure 8.9a,8.9b with the shape of figure. 8.9c. The disjoint part might create issues

While the generations are of better quality, supporting the hypothesis of linear neighborhoods, the samples we choose are not sufficiently close to assume a purely euclidean data distribution. In an extension we perform non-linear interpolation by training a variational auto-encoder and show that is is more robust to nonlinear distributions. An approach which has not been explored in this work is the usage of a kernel-PCA method to capture the nonlinear relationships.

Figure 8.10: Shapes from a neighbourhood

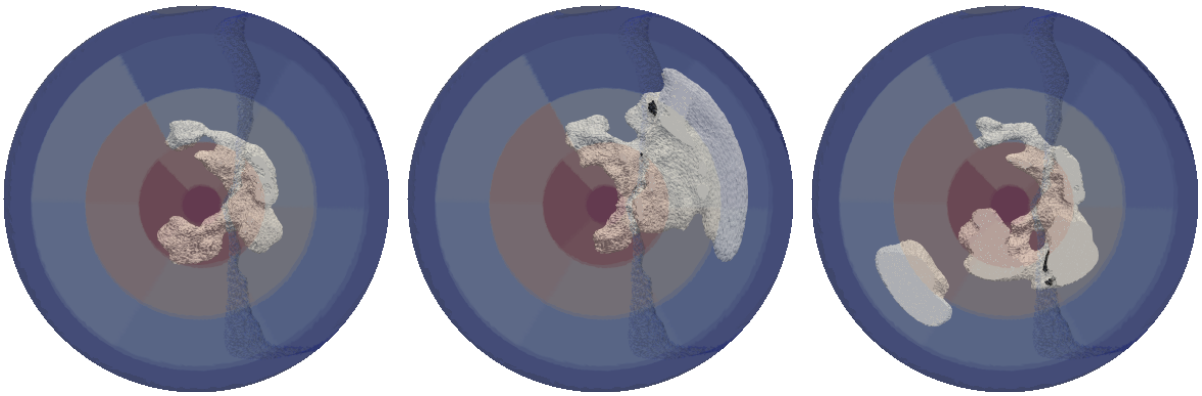


(a) generated shape 1 (b) generated shape 2 (c) generated shape 3

8.4.0.2 Non-linear methods in neighbourhoods

We train a variational autoencoder on the samples displayed in figure 8.11. Figure **??** displays the generations. Compared to those created by PCA in section **??** they are better. Furthermore, the results are significantly better than those created by a naive variational auto-encoder. This indicates, that by employing non-linear methods on near-linear neighborhoods of the data-manifold the quality of the generated samples is incremented.

Figure 8.11: Shapes from a neighbourhood



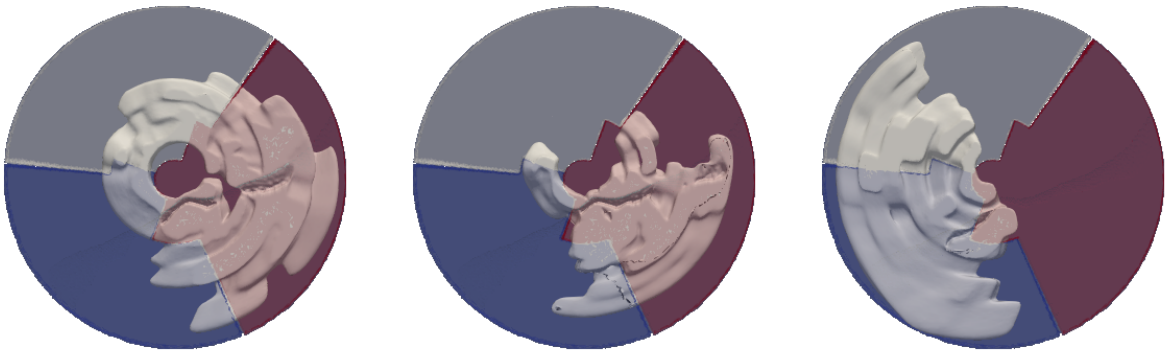
(a) generated shape 1 (b) generated shape 2 (c) generated shape 3

8.4.0.3 Conditioning by feature-neighbourhoods

Section presents the manifold hypothesis which is tested with PCA. The result is that there is validity to the hypothesis, although, due to the low-data density, the neighbourhoods might not be completely linear. Employing non-linear variational autoencoders on these neighbourhoods accounts for the non-linearity and produces better generations. The VAE trained on fewer samples generates examples which contain many features of the original shapes which are lost by training on all the data. The generations of a variational autoencoder (or pca) on only 3 data-samples are novel but highly similar to the. By choosing fewer data-samples with desired features it is possible to create interpolations (either linearly with PCA or nonlinearly with VAE) which maintain these. Through this, it is possible to generate scars of certain sizes and with specific shape. We provide experiment where we group the scars into three groups with visually striking characteristics. These groups are called the circular, longitudinal and half-moon groupings. Some shapes might belong to both groups, creating smooth transitions.

The groupings might be constructed as neighborhoods of the low-dimensional latent-space. It is not clear if any nonlinear interpolation between data-samples produces a valid scar. However, it is likely, that similar (neighboring) shapes produce physical generations by interpolation (see figure **??**). Distant shapes might be more prone to produce non-physical generations (see figure **??**).

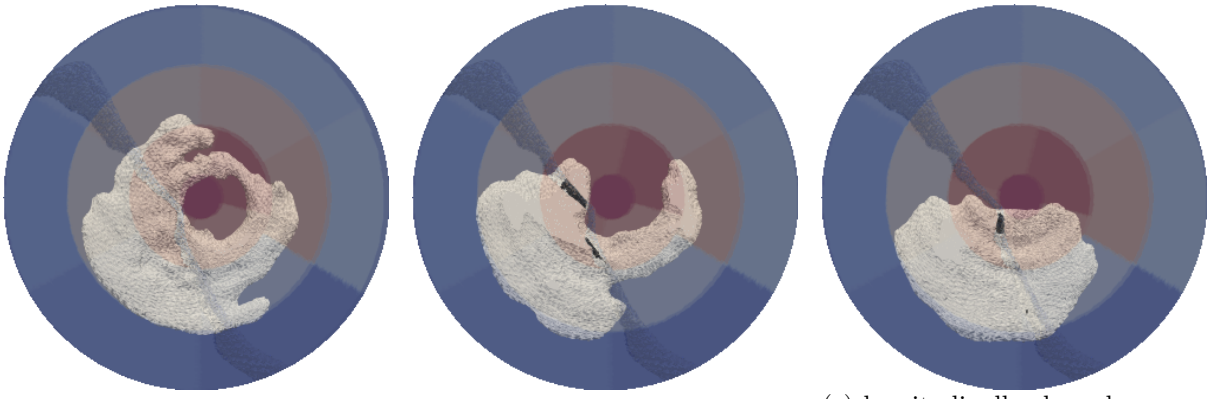
Figure 8.12: Feature based groups



(a) Circular shaped scar (b) Half-moon shaped scar (c) longitudinally shaped scar

We train three seperated variational auto-encoders on each group and examine the generations. By choosing adequate training sets so that the desired feature is not subject to variability, the interpolations maintain their representation. We propose using this mechanism for guided generations.

Figure 8.13: Feature based generations

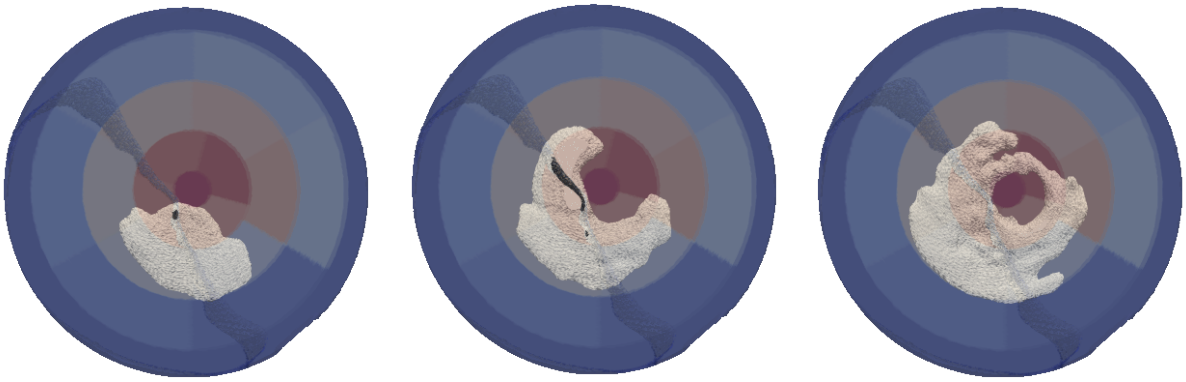


(a) Circular shaped generation (b) Half-moon shaped generation (c) longitudinally shaped generation

8.4.1 Generated Scars

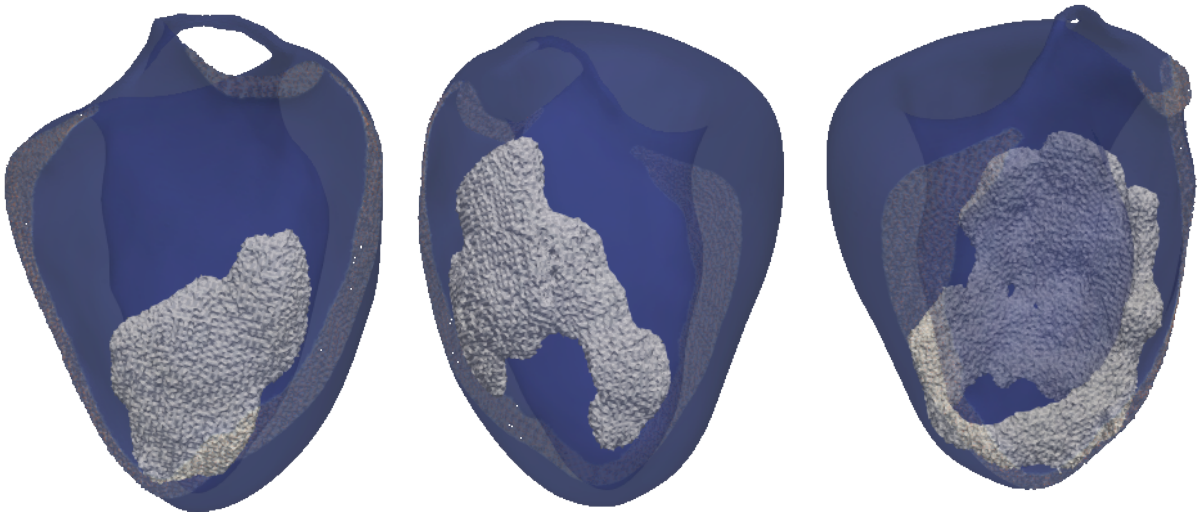
We examine the generations in the heart mesh.

Figure 8.14: Generations in disk

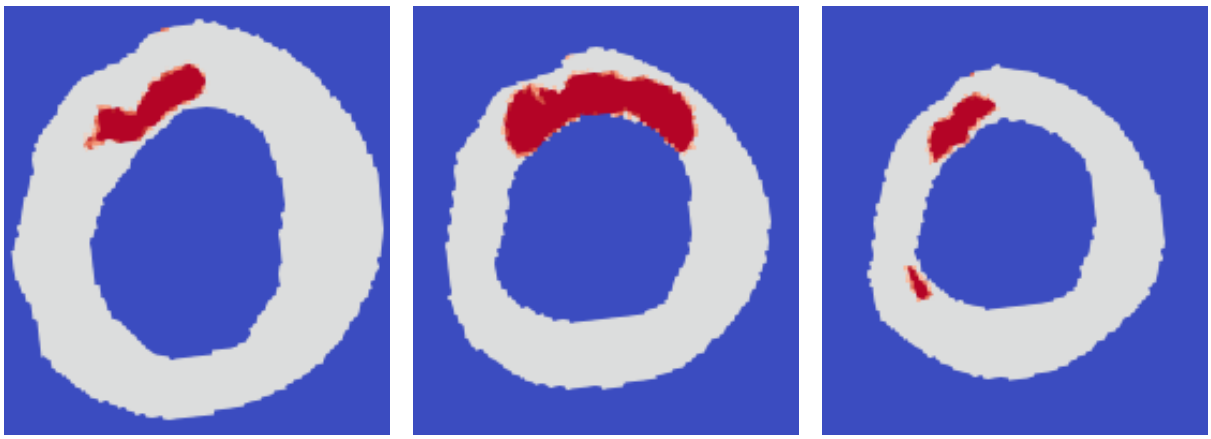


(a) Generation 1 in disk (b) Generation 2 in disk (c) Generation 3 in disk

Figure 8.15: Generations in heart



(a) Generation 1 in heart (b) Generation 2 in heart (c) Generation 3 in heart



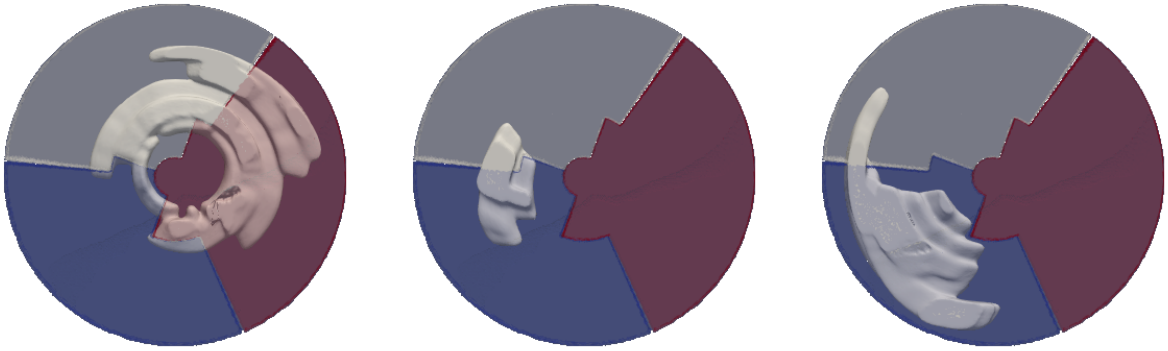
(a) Segmenation 1 (b) Segmenation 2 (c) Segmenation 3

8.5 Automated clustering

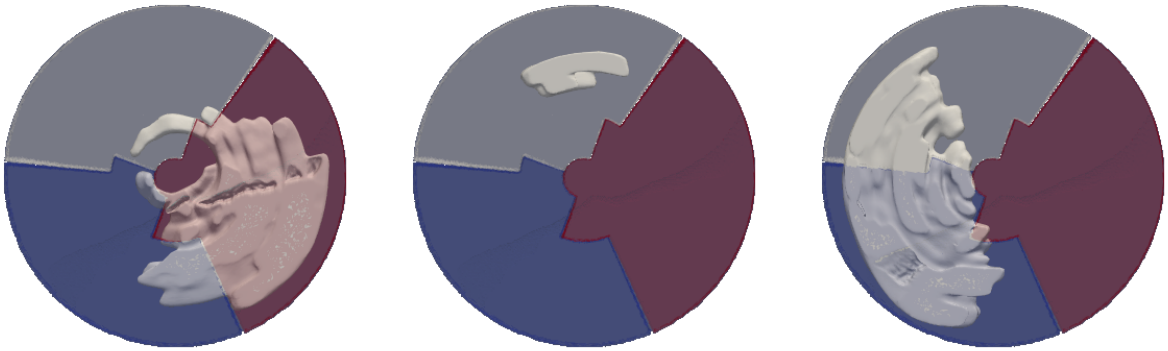
Since, manual grouping is not a feasible option for many generation in many neighbourhoods an automated grouping process should be proposed. Create this automated clustering is beyond the scope of this work. However, we offer some perspectives on how it might be obtained.

All clustering algorithms use a similarity measure to group similar elements together. We propose using the energy function from section as similarity. Given two shapes, we can compute their similarity. Thus assume for each shape a computed similarity vector $x_i \in \mathbb{R}^N$, where N is the number of data-samples. Then we can use K-means clustering or Gaussian Mixture Models to group the shapes by closeness. On each group one can use linear or nonlinear methods to interpolate and create new shapes. By enabling an overlap between groups (i.e. assigning certain shapes to multiple groups) smooth transitions can be enabled. Limited by time, we test the grouping on a simplified $1D$ distance vector containing the similarity to a chose reference shape. This clustering captures some similarities, but is not always accurate. However, it serves as a proof of concept for automated grouping.

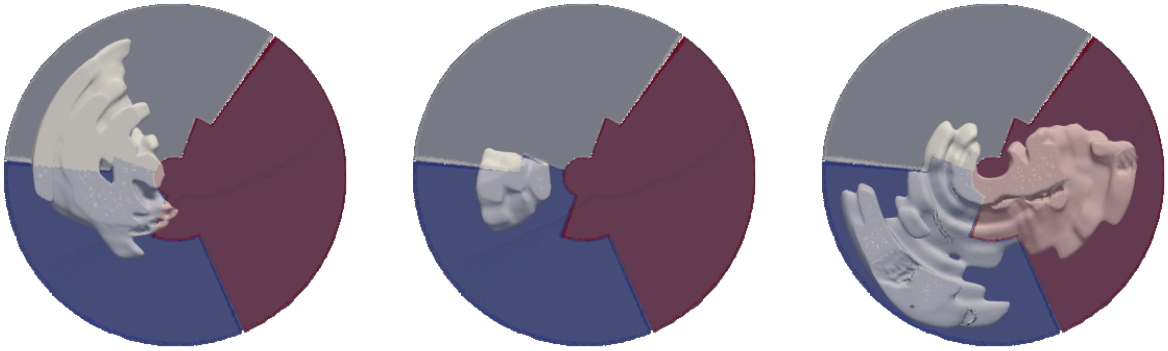
Figure 8.17: Automated Clustering into 3 groups



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3

8.6 Evaluation

Chapter 9

Conclusion

We provide a methodology which enables the generation of novel scar shapes. We examine shape representations and algorithms for their suitability and make our choice. We highlight the high-dimensionality of our representation and related problematics and enable learning the high-dimensional signed distance functions by a projection onto basis spline coefficients. We highlight the unsuitability of the isotropic gaussian prior for our dataset with low spatial density and provide a solution which considers shape and position as independent by performing a normalizing transformation. We examine measures for shape similarities based on information theory, metric spaces and optimal transport. Via shape normalization (mainly with respect to position) and clustering (with respect to shape similarity), we show how even linear methods can be employed for plausible generations. We provide a visual comparison to non-linear generations from a VAE, indicating that these are more robust to nonlinearity but equivalent if the distribution is linear. We show how clustering can be manipulated to represent desired features in generations. Observations indicate that smaller groupings lead to more details in the generations. Furthermore, we examine first steps towards an automated clustering. While our method is highly simplified due to time-reasons, it creates three groups with distinct features. However, some un-intuitive grouping occurs. This might be reduced by more expressive similarity vectors. We provide visualizations, indicating that the generations performed on automated clusters are qualitatively better than generations on the whole data-set. Evaluation methods are only mentioned theoretically. We also show that the generated scars maintain their plausible shape and that the clinically relevant features, such as growth from endocardium to epicardium are maintained. Our work considers the scars obtained from processing as ground-truth. However we point out several processing artifacts which need to be addressed. Among these are the low-vertical resolution of MRI scans (leading to a block-wise structures and disjoint parts) and missing scans in the apex (leading to shapes with nonphysical holes).

Summarizing, this work provides insights into the available data and challenges associated to generative algorithms. We provide solutions regarding high-dimensionality and low-quality generations and hypothesize that training in neighborhoods of few, but similar, shapes can be advantageous. Examination of the manifold hypothesis hints towards the applicability of linear methods, if the data were dense enough. However, nonlinear methods are more robust.

For the future we propose a combination of automated clustering techniques and evaluation methods to fill the data-space with shapes and their interpolations. The generated scars might be labeled and used to create a gallery. Our work does not create shapes with specific occupancies or transmuralities. However, by choosing adequate training groups, the generations can be guided to represent these features. By providing the desired feature-label, relevant scars can be displayed and the user can make his choice. This work considers shape and position as independent. However we also mention that this is a simplifying assumption and not true in general.

Additional research to correlate shape with position might be useful. Users familiar with the dataset and cardiomyopathy can do the placement manually by referring to their knowledge of realistic occupancies.

Bibliography

[1] Jazmin Aguado-Sierra, Constantine Butakoff, Renee Brigham, Apollo K. Baron, Guillaume Houzeaux, Jose M. Guerra, Francesc Carreras, David Filgueiras-Rama, Paul A. Iaizzo, Tinen L. Iles, and Mariano Vazquez. Hpc framework for in-silico trials on 3d virtual human cardiac population to assess drug-induced arrhythmic risk. *medRxiv preprint*, September 2022.

[2] Hadrien Calmet. *Large-scale CFD and micro-particle simulations in a large human airways under sniff condition and drug delivery application.* Phd dissertation, Universitat Politècnica de Catalunya, 2020.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Yunlong Duan, Zhen Zhou, Xiaogang Zhang, and Xin Liu. Modeling cardiac stimulation by a pacemaker, with accurate electrophysiological simulations. *Journal of Computational Cardiology*, 20:305–319, 2023.

[5] Mark K. Friedberg and Andrew N. Redington. Right versus left ventricular failure. *Circulation*, 129(9):1033–1044, 2014.

[6] Bernhard L. Gerber, Julie Darchis, Jean-Benoît le Polain de Waroux, Gabin Legros, Anne-Catherine Pouleur, David Vancraeynest, Agnès Pasquet, and Jean-Louis Vanoverschelde. Relationship between transmural extent of necrosis and quantitative recovery of regional strains after revascularization. *JACC: Cardiovascular Imaging*, 3(7):720–730, 2010.

[7] Mark Golob, Richard L. Moss, and Naomi C. Chesler. Cardiac tissue structure, properties, and performance: A materials science perspective. *Annals of Biomedical Engineering*, 42(10):2003–2013, 2014.

[8] Yulan Guo, Zhiwei Lei, Tianjia Wan, Xing Xu, and Jianbo Wang. A survey of deep learning-based 3d shape generation. *arXiv preprint arXiv:2002.04579*, 2020.

[9] Clemens Isert. *Quantum Mechanics-Based Geometric Deep Learning for Drug Discovery.* Phd dissertation, ETH Zurich, 2023.

[10] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b: A 7–billion-parameter language model engineered for superior performance and efficiency. *arXiv preprint arXiv:2310.06825*, 2023.

[11] Moien Ab Khan, Muhammad Jawad Hashim, Halla Mustafa, May Yousif Baniyas, Shaikha Khalid Buti Mohamad Al Suwaidi, Rana AlKatheeri, Fatmah Mohamed Khalfan Alblooshi, Meera Eisa Ali Hassan Almatrooshi, Mariam Eisa Hazeem Alzaabi, Reem Saif Al Darmaki, and Shamsa Nasser Ali Hussain Lootah. Global epidemiology of ischemic heart disease: Results from the global burden of disease study. *Cureus*, 12(7):e9349, July 2020.

[12] Sebastian Krittian, Stefan Schmitt, Christoph Lorenz, Michael A. Brockmann, and Tobias Preusser. Prediction of 3d cardiovascular hemodynamics using fluid-structure interaction models. *Journal of Computational Surgery*, 1(1):97–107, 2011.

[13] Pierre Lahoud, Reinhilde Jacobs, Seyed Ali Elahi, Maxime Ducret, Wout Lauwers, G. Harry van Lenthe, Raphaël Richert, and Mostafa EzEldeen. Developing advanced patient-specific in silico models: A new era in biomechanical analysis of tooth autotransplantation. *Journal of Endodontics*, 50(6):820–826, 2024.

[14] Hao Liu, João S. Soares, John Walmsley, David S. Li, Samarth Raut, Reza Avazmohammadi, Paul Iaizzo, Mark Palmer, Joseph H. Gorman III, Robert C. Gorman, and Michael S. Sacks. The impact of myocardial compressibility on organ-level simulations of the normal and infarcted heart. *Scientific Reports*, 11:13466, 2021.

[15] Thomas Maxwell, Boonthanome Nouanesengsy, Andy Bauer, Aashish Chaudhary, and John Patchett. Exploratory climate data visualization and analysis using dv3d and paraview in uv-cdat. *Kitware Source*, 04 2013.

[16] David Oks, Cristóbal Samaniego, Guillaume Houzeaux, Constantine Butakoff, and Mariano Vázquez. Fluid–structure interaction analysis of eccentricity and leaflet rigidity on thrombosis biomarkers in bioprosthetic aortic valve replacements. *International Journal for Numerical Methods in Biomedical Engineering*, 38(12):e3649, 2022.

[17] Santiago Sanchez, Eduardo V. Garcia, Jose Luis Serna, Eric W. Remme, and Thor Edvardsen. The impact of myocardial fibrosis on left ventricular function during acute coronary syndromes: insights from cardiovascular magnetic resonance imaging. *Journal of Cardiovascular Magnetic Resonance*, 17(1):1–10, 2015.

[18] William Schroeder, K. Martin, and William Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. 01 2006.

[19] Sandeep Singh Sengar, Affan Bin Hasan, Sanjay Kumar, and Fiona Carroll. Generative artificial intelligence: A systematic review and applications. *arXiv preprint arXiv:2405.11029v1*, 2024.

[20] Joshua Steyer, Lourdes Patricia Martínez Díaz, Laura Anna Unger, and Axel Loewe. Simulated excitation patterns in the atria and their corresponding electrograms. In *Functional Imaging and Modeling of the Heart*, 2023.

[21] Julian Suk, Christoph Brune, and Jelmer M. Wolterink. Se(3) symmetry lets graph neural networks learn arterial velocity estimation from small datasets. In *Functional Imaging and Modeling of the Heart (FIMH)*, pages 445–454. Springer Nature, 2023.

[22] Eric J. Topol, Francisco Lopez-Jimenez, and Andrea N. DeMaria. Artificial intelligence in the diagnosis and management of disease: Overview and recommendations. *Journal of the American College of Cardiology*, 73(17):2137–2152, 2019.

[23] Marco Viceconti, Francesca Pappalardo, Baudouin Rodriguez, Matthew Horner, M. Bischoff, and Lara Montoya. In silico trials: Verification, validation and uncertainty quantification of predictive models used in the regulatory evaluation of biomedical products. *Methods*, 94:146–160, 2016.

[24] Shinelle Whiteman, Yusuf Alimi, Mark Carrasco, Jerzy Gielecki, Anna Zurada, and Marios Loukas. Anatomy of the cardiac chambers: A review of the left ventricle. *Translational Research in Anatomy*, 23:100095, 2021.

[25] R Willems, EKPM Kruithof, KLPM Janssens, MJM Cluitmans, O van der Sluis, PHM Bovendeerd, and CV Verhoosel. Isogeometric-mechanics-driven electrophysiology simulations of ventricular tachycardia. *Lecture Notes in Computer Science*, 13958:97–106, 2023.