Statistical Shape Analysis for Applications in Image Segmentation

by

Pradyot Prakash

Roll No: 130050008

under the guidance

of

Prof. Suyash Awate

Bachelors' of Technology Thesis (II)

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 28th April, 2017

Pradyot Prakash

Place: IIT Bombay, Mumbai

Roll No: 130050008

Acknowledgements

I am thankful to the people who have been instrumental in helping me out throughout this project. First and foremost, I express my sincere gratitude towards my supervisor Prof. Suyash Awate for his guidance. I would also thank Saurabh Shigwan for helping me out throughout the duration of this project. I am also thankful to my friends, family, and teachers who have been always there for me whenever I needed them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Researchers working in the field of image processing have tried to find succinct ways of representing images. The usual notation containing intensity values of each pixel is not interesting and does not exploit the structure within the image. The application of Fourier transform through frequency based analysis has been present for a long time. The Fourier transform and its neighbor, discrete cosine transform are universal bases used for image representation. However, they miss out on identifying the locality within the images. For instance, the Fourier transform of a box function gives us a sinc function but we can't pinpoint its exact coordinates. To alleviate this inefficiency, other kinds of bases such as contourlets and wavelets have been used. We see the application of PCA basis as another such method.

The work in the last semester through BTP (I) used the notion of dictionaries to learn meaningful shape contours. This report summarizes the work done there and extends that using the PCA basis applied to images for segmentation under the purview of Shape Analysis.

The novel idea behind this work is the introduction of a new similarity measure between shape contours and images. That forms an integral part of the optimization function to get a good fit.

Chapter 2

Shape Analysis

This section refers to method developed in [1] [3] and [2]. Shapes play an important role in medical image processing. Most body parts have a fixed shape and structure. Defects and diseases can often be identified by analysis of the structural changes in them. This motivates one strong reason for the study of shapes.

2.1 Shape representation

One primary issue that arises while dealing with shapes is how to represent them. A shape is a continuous curve and we choose some k points on it to get a discrete sampling. If the shape is in a d dimensional space, then we essentially get $d \times k$ dimensional shape space. Here we work with images for which $d = 2$ or $d = 3$. Let's denote a 2D shape, $\mathbf{S}$ using $\mathbf{S} = \left[\begin{pmatrix} x_1^{(1)} \\ x_1^{(2)} \end{pmatrix}, \begin{pmatrix} x_2^{(1)} \\ x_2^{(2)} \end{pmatrix} \dots \begin{pmatrix} x_k^{(1)} \\ x_k^{(2)} \end{pmatrix} \right]$ where $\begin{pmatrix} x_2^{(j)} \\ x_2^{(j)} \end{pmatrix}$ are points defining the shape boundary. Note that these are all points on the same curve and not points on different curves. Similarly, for 3D images, the corresponding points are vectors in $\mathbb{R}^3$.

To study shapes, we convert these set of points to a vector by concatenating the points. The vector representing the previous curve takes the form

$$\mathbf{X_i} = [x_1^{(1)}, x_1^{(2)}, x_2^{(1)}, x_2^{(2)}, \dots, x_k^{(1)}, x_k^{(2)}]^T$$

.

One issue resolved, there are more. The difficulty in coming up with a general notion of shape is difficult because of 4 factors:

1. Translation: Moving the image does not alter its shape
2. Scaling: Changing the scale by the same factor along all the axes also maintains the shape
3. Rotation: By changing the orientation of the image, the shape remains unaltered
4. Reflection: Reflection about a place or axis also keeps the shape same. We won't be dealing with reflections as part of this thesis

Hence, before moving on to analyzing shapes and their similarities, we need to remove the first three factors mentioned above. If this were not so then the notion of distance between two shapes won't be meaningful. We need to get them to a common coordinate frame and that is done step by step as follows:

1. Translation: To get away with this, we shift the centroid of the shape to the origin. This gives us a consistent notion of placement of a shape in the coordinate axis — at the origin. The centroid, μ of a shape S is

$$\mu = \frac{1}{k} \sum_{i=1}^{k} \begin{pmatrix} x_i^{(1)} \\ x_i^{(2)} \end{pmatrix}$$

   To remove translation, $\bar{\mathbf{x}}_\mathbf{i} = \mathbf{x}_\mathbf{i} - \mu$. This is then further converted to the combined vector representation $\bar{\mathbf{X}}_\mathbf{i}$.

2. Scaling: We set the variance of $\bar{\mathbf{X}}_\mathbf{i}$ to 1 by dividing it by its norm to consistently get a unit norm shape
3. Rotation: As discussed above, we get the same shape on rotation. So if we have two shapes, $\mathbf{S_1}$ and $\mathbf{S_2}$, the distance between them would have been defined between

them as,

$$d(\mathbf{S_1}, \mathbf{S_2}) = \|\mathbf{S_1} - \mathbf{S_2}\|_F^2$$

This is essentially measuring how far away the component points of the shapes are far away from each other. However, note that we can possibly find a smaller distance between them by rotating the shapes and aligning them to each other using some distance metric. This idea is captured by what is called the Procrustes distance. We find an orthogonal rotation matrix, $\mathbf{\Omega}$, which finds the nearest shape for a given shape such that,

$$\mathbf{\Omega^{opt}} = \arg \min_{\mathbf{\Omega} \in \mathbb{R}^{d \times d}} \|\mathbf{\Omega S_1} - \mathbf{S_2}\|_F^2$$

$$\mathbf{\Omega}^T \mathbf{\Omega} = \mathbb{I}, \det(\mathbf{\Omega}) = 1$$

This finds the "nearest" shape in the squared error sense. We will use this to align one shape with other. This above problem has a closed form solution given by $\mathbf{\Omega^{opt}} = \mathbf{U}\mathbf{V}^T$ where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular matrices of $\mathbf{M} = \mathbf{S_2}\mathbf{S_1^T}$. If $\det(\mathbf{\Omega^{opt}}) = -1$ then we flip the sign of one of the left singular values. MATLAB does this internally through the function `procrustes()`.

By removing translation, we are setting the centroid of the shape to origin. This is equivalent to imposing a linear constraint of the form $x_1^{(1)} + x_1^{(2)} + x_2^{(1)} + x_2^{(2)} + \cdots + x_k^{(1)} + x_k^{(2)} = 0$ which is a hyperplane in a kd dimensional space. By removing scale, we are essentially setting the norm of $\mathbf{\bar{X}_i}$ to 1 and hence projecting it on a unit norm hypersphere in a kd dimensional space. The combination of both these constraints is similar to the intersection of the hyperplane with the hypersphere. This space is referred to as the preshape space (PSS).

2.2 Invariance of the preshape space

One important consideration is that we want the rotation operation to preserve the preshape invariance. This indeed turns out to be true under the Frobenius norm condition.

2.2.1 Translation invariance

Suppose we have a shape matrix, $\mathbf{S}_{d\times k}$. Let $\mathbf{S} = [\mathbf{s_1}, \mathbf{s_2}, .., \mathbf{s_k}]$, where $\mathbf{s_i}$ are the columns of $\mathbf{S}$. Assume that $\mathbf{S} \in PSS$. Hence, we have $\sum_i \mathbf{s_i} = \mathbf{0}$. By pre-multiplying $\mathbf{S}$ by any arbitrary matrix $\mathbf{R} \in \mathbb{R}^{d\times d}$, we get $\mathbf{RS} = \mathbf{R}[\mathbf{s_1}, \mathbf{s_2}, .., \mathbf{s_k}] = [\mathbf{Rs_1}, \mathbf{Rs_2}, .., \mathbf{Rs_k}]$. Sum of columns of $\mathbf{RS}$ gives $\sum_i \mathbf{Rs_i} = \mathbf{R}(\sum_i \mathbf{s_i}) = \mathbf{R0} = \mathbf{0}$. Hence, $\mathbf{RS}$ has zero mean.

2.2.2 Scale invariance

We know that a rotation matrix is given by $\mathbf{R^T R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. For the matrix $\mathbf{RS}$, Frobenius norm is $\|\mathbf{RS}\|_F = \sqrt{trace((\mathbf{RS})^T\mathbf{RS})} = \sqrt{trace(\mathbf{S^T R^T RS})} = \sqrt{trace(\mathbf{S^T S})} = \mathbf{S}_F$. Hence, is $\mathbf{S}$ is a shape matrix, then $\mathbf{RS}$ also has norm 1.

2.2.3 Translation and scale invariance

From the previous two results, we see that if $\mathbf{R}$ is a rotation matrix then the matrix $\mathbf{RS} \in PSS$. Hence, a rotation matrix preserves the translation and scale invariance.

Chapter 3

Manifolds

A manifold $\mathcal{M}$ of dimension d is a topological space such that each point $x \in \mathcal{M}$ has a neighborhood which can be continuously transformed into a Euclidean space of the same dimension. More formally, the neighborhood is homeomorphic to the Euclidean space, $\mathbb{R}^d$. With the additional property of being able to perform differential calculus on the manifold, it becomes a differential manifold.

3.1 Riemannian manifolds

Every differential manifold has a tangent space associated with it. The tangent space, $T_x\mathcal{M}$ defined at $x \in \mathcal{M}$ is the vector space containing all the tangent vectors to $\mathcal{M}$ at the point x. With the additional constraint of having an inner product on the tangent space, $T_x\mathcal{M}$, we get the Riemannian manifold.

3.1.1 The notion of distance

Let $v \in T_x\mathcal{M}$ be a tangent vector to $\mathcal{M}$ at x. There exists a unique smooth curve, a geodesic,

$$\gamma_v : [0, 1] \to \mathcal{M}$$

satisfying $\gamma_v(0) = x$ with initial tangent vector in the direction of v, $\gamma_v'(0) = v$.

Suppose x and y are two points on $\mathcal{M}$. The distance function $d : \mathcal{M} \times \mathcal{M} \to [0, \infty)$ is defined as,

$$d(x, y) = \inf_{\gamma} \{x, y \in \gamma\}$$

In simpler terms, the distance between them is the minimum length of all the curves that start at x and end at y.

3.1.2 Exponential and logarithm maps

We can simply add or subtract two vectors to get the third one. However, if we apply the same notion over here, we may get a point which does not lie on the manifold. Hence it becomes important to define the operations of addition and subtraction carefully. There are two important functions which do this — $\exp$ and $\log$.

The exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ is defined as,

$$\exp_x(v) = \gamma_v(1)$$

This signifies that point on $\mathcal{M}$ which lies in the direction of v and is one unit distance away from x. Let's give this an informal interpretation. The constant e is defined to be the limit $\lim_{x\to 0}(1 + x)^{\frac{1}{x}}$ from calculus. This can be understood as adding some small quantity to 1 and see what it is, followed by adding a small quantity to this new entity and repeating the operating infinite number of times. The exponential map is analogous. Since directly adding two points on a manifold may push it away from it, we add small increments to a point several times until we have reached a point which a 1 unit away from it in a particular direction. These small increments help ensure that we are still on the manifold and the operations are meaningful. The inverse of the exponential map is the logarithm map,

$$\log_x : \mathcal{M} \to T_x\mathcal{M}$$

Under some assumptions of the global existence of the $\exp$ and the $\log$ maps, we get the following two important results:

1. $d(x, y) = ||\log_x(y)||_x$ where $||v||_x$ is the length of the vector $v \in T_x\mathcal{M}$

2. $d^2(x, .)$ is a smooth function for all $x \in \mathcal{M}$

3.2 Shape analysis and Riemannian manifolds

Riemannian manifolds naturally occur while studying shapes. Note that we project the shapes into the PSS. This sets the norm of the shape matrix to 1 and puts it on a hypersphere. The PSS is a subset of this hypersphere because of its intersection with a plane. Hence, the normal notion of Euclidean distance does not make sense on a hypersphere and we exploit its Riemannian manifold nature. To get a distance measure, we need a geodesic and in this case it turns out to be the great circle joining two points. The exponential and logarithmic maps for a hypersphere is defined as:

$$\exp_{\mathbf{a}}(\mathbf{v}) = \cos(|\mathbf{v}|)\mathbf{a} + \sin(|\mathbf{v}|)\frac{\mathbf{v}}{|\mathbf{v}|}$$

$$\log_{\mathbf{a}}(\mathbf{v}) = \frac{\mathbf{u}\cos^{-1}(\mathbf{a^T v})}{\sqrt{\mathbf{u^T u}}}$$

where $\mathbf{u} = \mathbf{v} - (\mathbf{a^T v})\mathbf{a}$.

Chapter 4

Learning on Riemannian manifold

Suppose we have a set of points $\mathbf{X_1}, \mathbf{X_2}, .., \mathbf{X_n}, \mathbf{X_i} \in PSS^{dk}\ \forall i$ where PSS^{dk} is the preshape space of shapes originally in $\mathbb{R}^{d\times k}$.

4.1 Dictionary learning

The usual definition of a dictionary searches for a set A, whose linear combination gives us the best set of points X. However, this linear combination makes no sense on a sphere. Hence, we want to look for a generalization of $\mathbf{X_i} = \mathbf{AW_i}$ since our spherical manifold does not support a global vector space structure. Riemannian manifolds help us in this regard by providing a tangent space, $T_x\mathcal{M}$, at point a $x \in M$ (Riemannian manifold) which allows us to extract global information using the exponential and logarithm maps.

Unlike the Euclidean case where we could exploit the vector structure we do not have that freedom here. The notion of "origin" is not present here since the tangent space is locally defined. The point x can be interpreted as the origin for $T_x\mathcal{M}$. Since we want to model the linear reconstruction nature of dictionaries for $\mathcal{M}$, we impose an affine constraint on the coefficients. So if,

$$\mathbf{X_i} = w_{1i}\mathbf{A_1} + w_{2i}\mathbf{A_2} + \cdots + w_{mi}\mathbf{A_m}$$

then by setting

$$w_{1i} + w_{2i} + \cdots + w_{mi} = 1$$

for each i gives us an origin independent environment.

[5] shows that

$$\min_{\mathbf{A},\mathbf{W}} \sum_{i=1}^{n} \left\| \sum_{j=1}^{m} W_{ij} \log_{x_i}(\mathbf{A_j}) \right\|_{x_i}^2 + \lambda \left\| \mathbf{W} \right\|_1$$

$$\sum_{j=1}^{m} W_{ij} = 1, \forall i = 1, 2, \ldots, n$$

is a good optimization problem for learning the dictionaries on $\mathcal{M}$.

For unit spheres, on putting the appropriate $\log$ function, this boils down to,

$$\min_{\mathbf{A},\mathbf{W}} \sum_{i=1}^{n} \left\| \sum_{j=1}^{m} W_{ij} \cos^{-1}(\langle \mathbf{X_i}, \mathbf{A_j} \rangle) \frac{\mathbf{u_{ij}}}{|\mathbf{u_{ij}}|} \right\|_{x_i}^2 + \lambda \left\| \mathbf{W} \right\|_1$$

$$\mathbf{u_{ij}} = \mathbf{A_j} - \langle \mathbf{X_i}, \mathbf{A_j} \rangle \mathbf{X_i}$$

$$\sum_{j=1}^{m} W_{ij} = 1, \forall i = 1, 2, \ldots, n$$

where $\langle\ ,\ \rangle$ is the vector dot product.

4.2 Learning the PCA basis

4.2.1 Euclidean PCA

The Principal Component Analysis tries to find an orthogonal basis, $\mathbf{A} = [\mathbf{A_1}, \mathbf{A_2}, .., \mathbf{A_p}]$, $\mathbf{A^T A} = \mathbf{I}$ which captures the maximum variance in the data. PCA tries to reconstruct the data using a linear combination of the basis vectors. It minimizes the following objective,

$$\min_{\mathbf{A},\mathbf{W_i}} \sum_{i}^{m} \left\| \mathbf{X_i} - \mathbf{A W_i} \right\|_F^2$$

where $\mathbf{X_i}, \mathbf{W_i} \in \mathbb{R}^p$.

The directions of maximum variance turn out to be the eigenvectors of the covariance matrix, $\mathbf{C} = \sum_i^m \bar{\mathbf{X}}_\mathbf{i} \bar{\mathbf{X}}_\mathbf{i}^\mathbf{T}/(m-1)$ where $\bar{\mathbf{X}}_\mathbf{i} = \mathbf{X_i} - \mu$, $\mu = \sum_\mathbf{i}^\mathbf{m} \mathbf{X_i}/m$. The eigenvectors are normalized to norm 1 and established as the columns of $\mathbf{A}$. Once we have $\mathbf{A}$, the coefficients $\mathbf{W_i}$ are simply $\mathbf{W_i} = \mathbf{A^T X_i}$.

4.2.2 Riemannian PCA

PCA works great for points in the Euclidean space because the Euclidean distance or the straight line distance is used. However, if we have to learn a similar basis on a hypersphere, we will have to modify the approach slightly. We need to counter two issues - (i) finding the mean vector and (ii) learning the directions of variance.

The tangent space associated with points in the Riemannian manifold come into picture here. If we somehow are able to get the points to the tangent space, we can apply the Euclidean operations there and borrow the concepts from the previous part. The algorithm is highlighted in 1.

Algorithm 1

1: Input: $\mathbf{X_1}, \mathbf{X_2}, \ldots \mathbf{X_m}$
2: set $\mu = \mathbf{X_1}$
3: **repeat**
4: $\quad \mathbf{X_i^{log}} = \log_\mu(\hat{\mathbf{X}}_\mathbf{i}) \; \forall \; i$ where $\hat{\mathbf{X}}_\mathbf{i}$ is aligned with μ
5: $\quad \mu_\mathbf{new} = \exp_\mu(\frac{1}{n} \sum_i \mathbf{X_i^{log}})$
6: $\quad \mu = \mu_{new}$
7: **until** μ converges
8: $\mathbf{C} = \sum_i \mathbf{X_i^{log}} (\mathbf{X_i^{log}})^\mathbf{T}$
9: Find the eigen decomposition of $\mathbf{C}$ as $\mathbf{CV_i} = \lambda_\mathbf{i} \mathbf{V_i}$

Hence, a point $\mathbf{X} = \exp_\mu(\mathbf{VW})$ where $\mathbf{W} = \mathbf{V^T} \log_\mu(\mathbf{X})$.

Chapter 5

Learning PCA basis from images

For the real world scenario, there aren't contours but rather images. We want to learn an image segmentation model from images. This is not so trivial because we do not have a set of landmark points defining the boundary of the contour and only have the segmentation with us.

To address this, a hierarchical model [4] is defined with the inclusion of some hidden variables. We model the top layer as a Gaussian with mean μ and covariance $\mathbf{C}$. From this, a hidden layer of another Gaussian is derived (mean $\mathbf{z_1}$, covariance $\mathbf{C_1}$) which models the observed data points, $\mathbf{y_{1i}}$. The β's used in the figure 5.1 are the smoothness parameters of a MRF. While modeling, we need to specify the number of landmark points that the model has to look for. This information is inherent in the number of dimensions of μ.

For this approach to work, we need to inundate the boundary of the segmentation using a few marker points, $\mathbf{x_{1ij}}$'s. The number of such points may vary from image to image and need not be equal. We align each $\mathbf{x_{1ij}}$ with the closest sampled from the $\mathbf{z_1}, \mathbf{C_1}$ distribution. Appropriate distance metrics are defined to make a MAP estimate to learn the best set of parameters. A Riemannian PCA-based approach is used to model the likelihood.

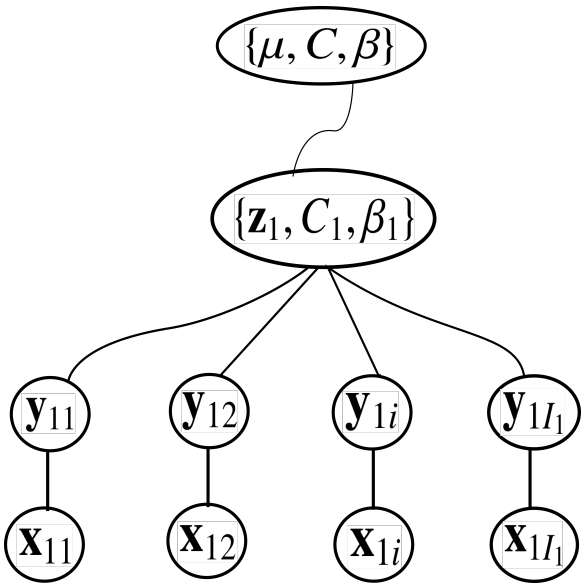


Figure 5.1: Hierarchical model to learn shapes

Chapter 6

Finding the segmentation

Now that we have the basics in place, the only task that remains is to segment an image. Over here, we are trying to fit a shape model over a section of the image and hope that it actually corresponds to the region that needs to be segmented out. This involves solving an optimization problem. But before that, we need to define a similarity measure between a shape contour and an image.

6.1 Binarizing a contour

A simple way to define a similarity measure is by first converting the contour into an image and then comparing the two images somehow. A binarization function `Bin`(c, h) takes a contour, c, and the size of the image, h, and gives a binary image, I, such that the region inside the contour has intensity 1 and the outside region has intensity 0. Now that we have two images, it is easy to define a distance measure between them.

6.2 The similarity measure

Suppose we have an image with a region that we want to segment out. Also, with each pixel within the image, let us associate a probability (rather a probability density) of the

pixel intensity lying inside the region of interest or outside it. Let us denote the respective probabilities with $P_{in}(I(p))$ and $P_{out}(I(p))$ for a pixel p in an image I.

Now, we define our similarity measure between an image I of size $h \times h$ and a contour c as,

$$\texttt{Sim}(c, I) = \frac{1}{h^2} \sum_{p \in I} \texttt{Bin}(c, h). * \log \frac{P_{in}(I(p))}{P_{out}(I(p))}$$

where `Bin`(., .) is as defined above.

6.2.1 Rationale behind `Sim`

For a pixel p which lies within the segmented region, we would expect that $P_{in}(I(p)) > P_{out}(I(p))$, and vice versa for a point lying outside. Hence, for the inside points $\log \frac{P_{in}(I(p))}{P_{out}(I(p))} > 0$. Hence, the similarity measure essentially finds the sum of the $\log$ probability ratios for the pixels which lie inside the binarized contour. Suppose that the contour c covered the entire image. In this case, we have a sum over the $\log$ probability ratios of the entire image. Since the actual segmentation lies somewhere inside the image, we would have a mix of positive and negative values. This would bring down the objective value. On the other hand, if the contour coincides with the correct segmentation, then the sum spans only the positive values of the $\log$ ratio which corresponds to the maximum sum we can get. If this optimal contour is perturbed slightly, we have some negative values coming inside the contour which reduces the value of the summation. If this contour had no overlap with any pixel inside the actual segmentation, the entire sum will be negative. These observations suggest that we try to maximize the sum.

This idea makes sense as the `Bin`(., .) term tries to maximize the spread and the log term tries to make the spread smaller. This kind of behavior leads to a maxima, perhaps a local one. This term is divided by the number of pixels, h^2, for normalization.

6.2.2 Finding the probabilities

To find the inside and the outside probabilities, we assume two separate Gaussian distributions over the inside and outside regions. This gives us 4 parameters, μ_{in}, μ_{out}, σ_{in} and σ_{out} to optimize for.

$$P_{in}(x) = \frac{1}{\sqrt{2\pi\sigma_{in}^2}} \exp^{-(x-\mu_{in})^2/2\sigma_{in}^2}$$
$$P_{out}(x) = \frac{1}{\sqrt{2\pi\sigma_{out}^2}} \exp^{-(x-\mu_{out})^2/2\sigma_{out}^2}$$

6.3 Objective function

Suppose we have the Riemmanian PCA model learned according to the previous mentioned approach. What remains is to use that model to segment out the image. Let $\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_p}$ be the eigenvectors in $T_\mu\mathcal{M}$, the tangent space of μ. We look for coefficients in $T_\mu\mathcal{M}$ and take their linear combination to get a point $\mathbf{G} = \sum_i \mathbf{V_i} w_i$ in $T_\mu\mathcal{M}$. This is projected on to the sphere using the $\exp_\mu$ map as $\mathbf{J} = \exp_\mu(\mathbf{G})$. $\mathbf{J}$ is reshaped to get a shape matrix, $\mathbf{S}$. $\mathbf{S}$ is then rotated and scaled appropriately. This is further translated to get the final contour. Maximizing `Sim`(., .) is equivalent to minimizing the negative of the function. This finally gives us our objective function as defined below:

$$\min_{s,\mathbf{R},\mathbf{t},w_i} -\texttt{Sim}(\texttt{Bin}(s\mathbf{R}\texttt{reshape}(\exp_\mu(\sum_i \mathbf{V_i} w_i) + \mathbf{t}), h), I) + \tau \sum_i |\frac{w_i}{\sqrt{\lambda_i}}| \qquad (6.1)$$

Here the second term corresponds to the regularization and sparsity we impose on the eigenvalues λ_i's. We also introduce a hyperparameter τ to control the relative importance of the two terms.

6.4 Optimization algorithm

The iterative optimization algorithm takes the form as given in algorithm 2.

Algorithm 2

Input: image I

initialize contour c, μ_{in}, μ_{out}, σ_{in} and σ_{out}

repeat

    find optimal c based on the above defined objective

    make a hard partition based on c to demarcate the inside (I) and the outside (O) regions

    perform MLE for I and O separately to learn the respective set of parameters

until convergence

Chapter 7

Experiments

Several sets of experiments were tried out to fit the model to the data and learn the various parameters involved in the optimization function.

7.1 Ellipses

The first set of experiment was performed on a simple dataset containing ellipses. The ellipses were manually fabricated by fixing the major axis and the length of the minor axis was varied to get 31 data points. Each ellipse has been traced by 36 contour points placed on their boundary at 10-degree intervals. All the ellipses were projected onto the preshape space and a Riemannian PCA model was learned using algorithm 1. The PCA model has one primary mode of variation with `sqrt`(principal eigenvalue) = 0.14.

The `Bin`(., .) function was implemented by first setting the intensity of the pixels at the landmark points to 1. This was followed by setting the intensities of the pixels on the line joining two consecutive points to 1. Now that we have a closed loop of 1's, we call MATLAB's `imfill()` to fill in the inside region.

The ellipse contours were used to generate images on which we want to test out the segmentation. The images were of size 512 x 512 and the unit norm ellipses were appropriately scaled. Since this not a realistic scenario as real images have a fuzzy segmentation

and the region of interest lies in a fuzzy region, IID gaussian noise was added to these ellipses to get target images as shown in figure 7.1. One such ellipse has been zoomed in and is shown in figure 7.2.

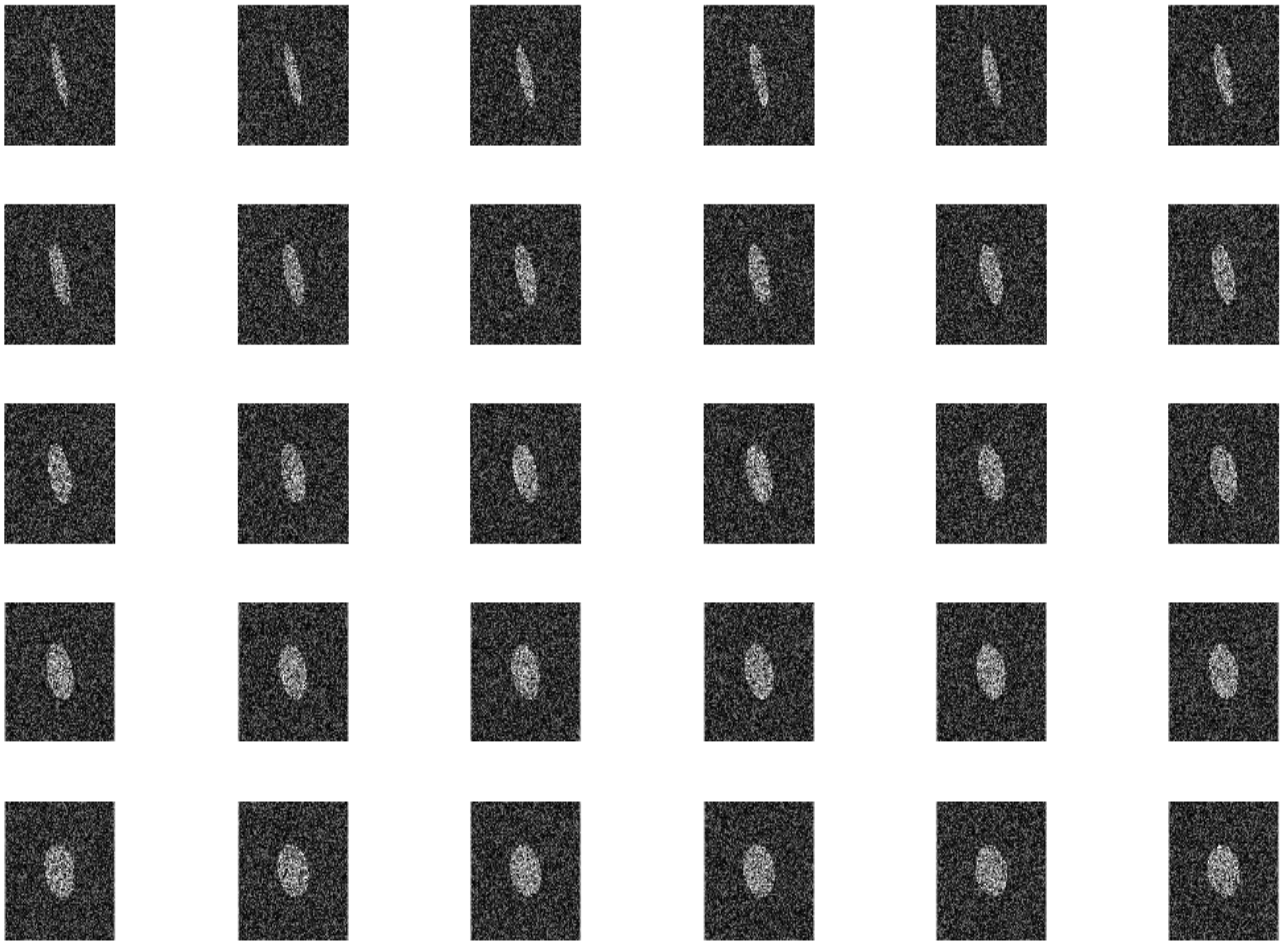


Figure 7.1: The ellipses we want to segment out

This is followed by fitting the PCA model to these images to learn the segmentation. The optimization algorithm is an iterative approach as highlighted in algorithm 3. Since

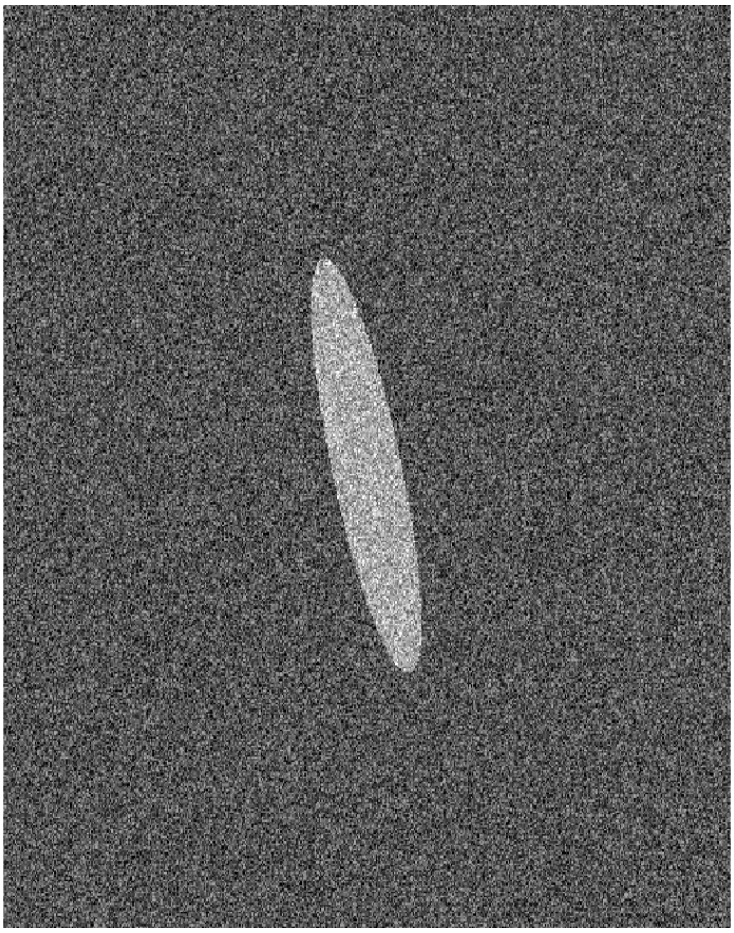


Figure 7.2: One example ellipse

there is just one mode of variation, only one eigencoefficient needs to be fitted. Also, since this a 2D image, we can get the rotation matrix using just a single parameter which models the rotation around the origin. Scale adds up one parameter and 2 parameters are added for translation in the 2D plane.

The initialization for w_i is done close to 0, θ close to 0, $\mathbf{t}$ such that the contour is initially aligned with the image and s between 0.8 to 1.2 times the actual scale. This is a good guess because in practice we have an estimate of the actual image size and can get a rough guess of where we need to initialize. In each step of algorithm 3, the MATLAB function `fminsearch()` was used. This is a non-gradient-based optimization technique and is well suited to our needs. This is because if we resorted to gradient-based approaches then

Algorithm 3

Input: image I, initial guesses for scale s, translation $\mathbf{t}$, rotation θ, eigencoefficients w_i, μ_{in}, μ_{out}, σ_{in} and σ_{out}
repeat
    repeat
        find optimal θ using the optimization problem in equation 6.1
        find optimal θ using the optimization problem in equation 6.1
        find optimal θ using the optimization problem in equation 6.1
        find optimal θ using the optimization problem in equation 6.1
    until convergence
    compute μ_{in}, μ_{out}, σ_{in} and σ_{out}
until convergence

unpredictable things might happen owing to the `Bin(., .)` function used for binarization. The fitted shapes are shown in figure 7.3. The absolute difference of the intensities between the (noiseless) target image and the learned images have been shown in figure 7.4.

We see that overall most of the ellipses have been properly learned except a couple of them where the fits are poor. This was because the scales for these ellipses were incorrectly learned and that hampered the optimization of θ and w_i properly. However, with a different initialization, this gets fixed. The means and variances for the inside and outside were learned really well — $\mu_{in} \approx 1$, $\mu_{out} \approx 0$ and $\sigma_{in} \approx \sigma_{out} \approx \sigma_{optimal}$.

7.1.1 Tuning τ

The optimal τ was found out to be close to 0. Setting it to any higher value, w_i is penalized strongly and the coefficients are fitted close to 0.

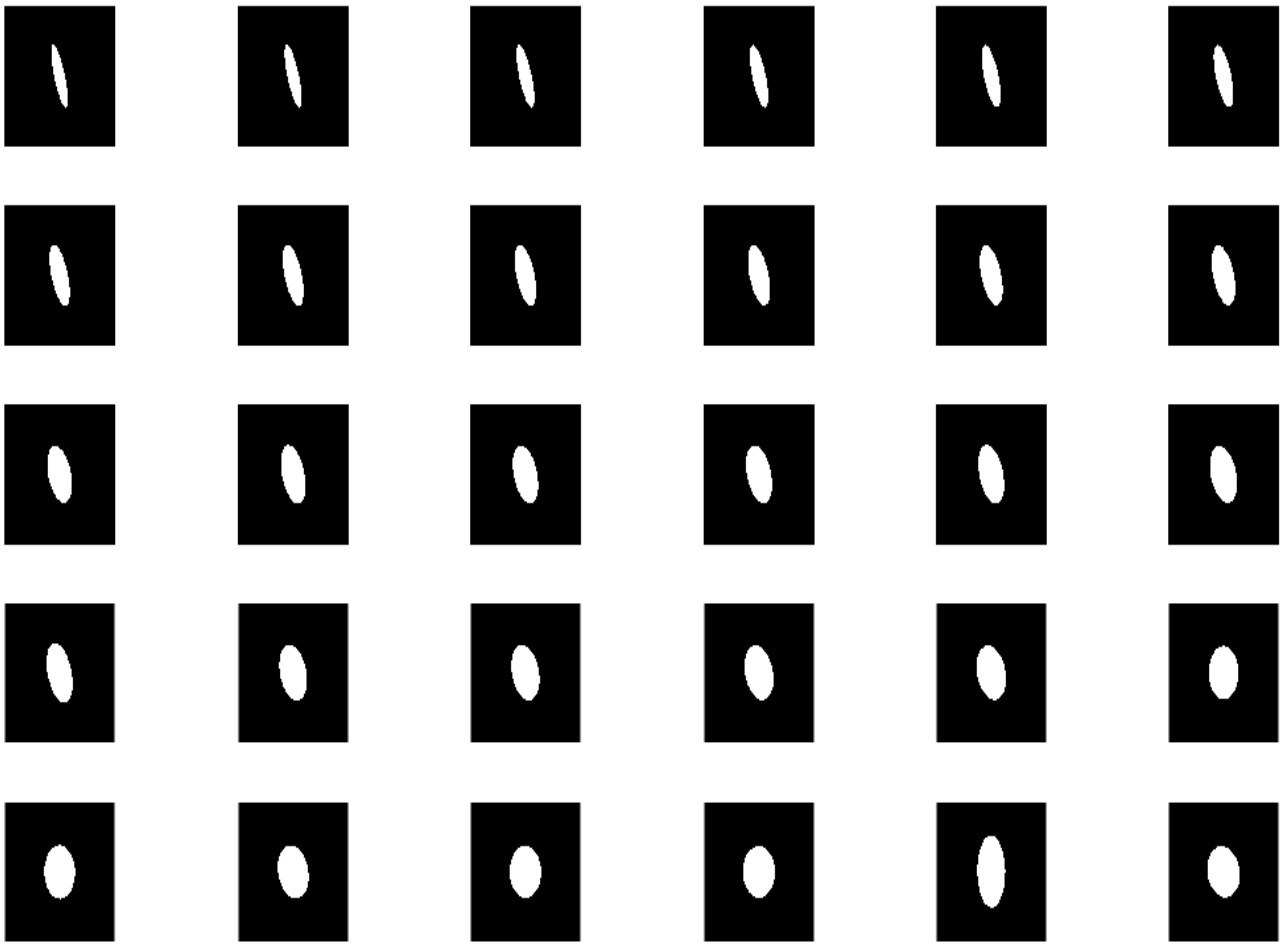


Figure 7.3: Binarized fitted ellipse contours

7.1.2 Numerical issues

A lot of numerical issues were encountered in this approach. This can be observed by looking at figure 7.5. The first row corresponds to two kinds of images—non-smoothened and smoothened images. These images are 256 x 256 in size. The second row plots the variation of the objective function with w_i. It is easy to observe that there are a lot of local

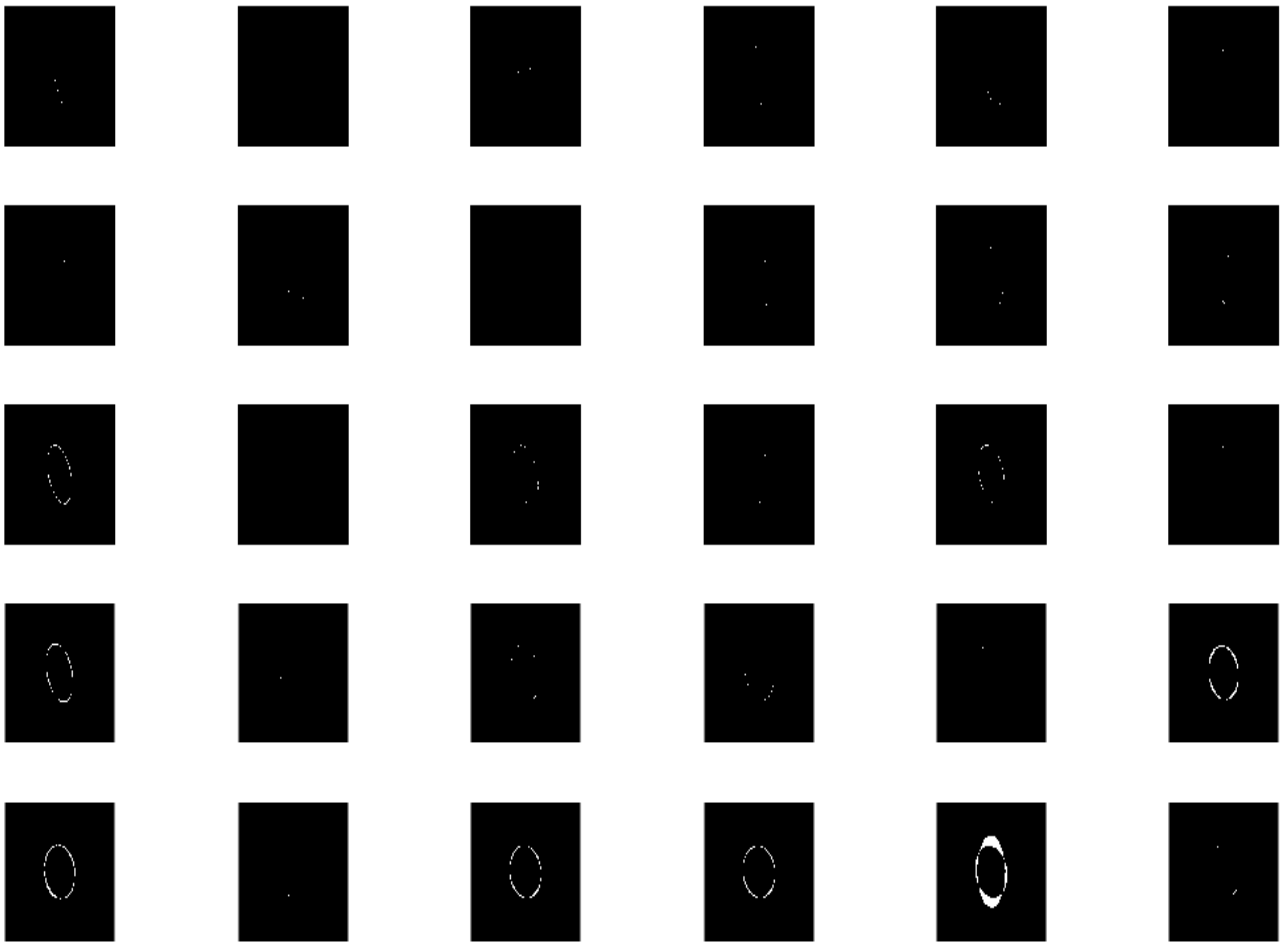
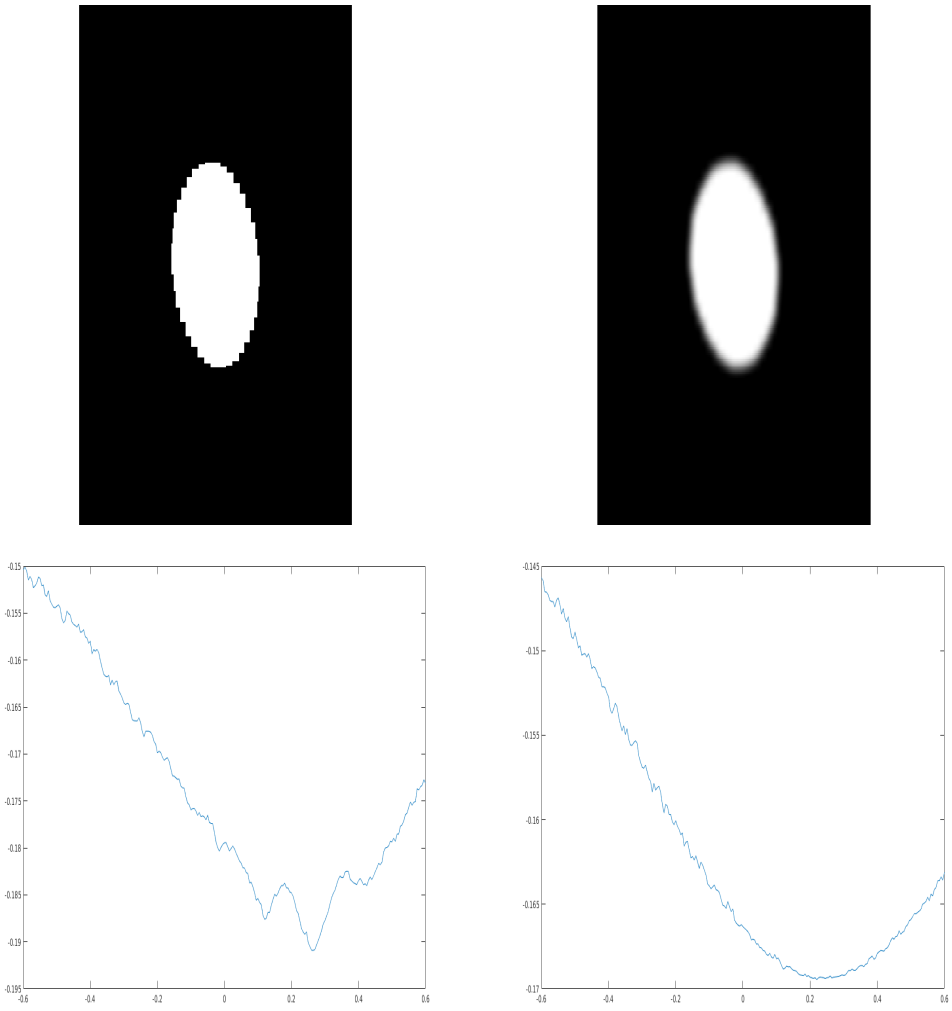


Figure 7.4: Absolute difference of intensities of uncorrupted data and fitted ellipses

minimas in the first case and this makes it extremely sensitive to initialization. A very obvious approach looking at the function plots is to employ simulated annealing. But that is an overkill and on further analysis, image resolution was the culprit.

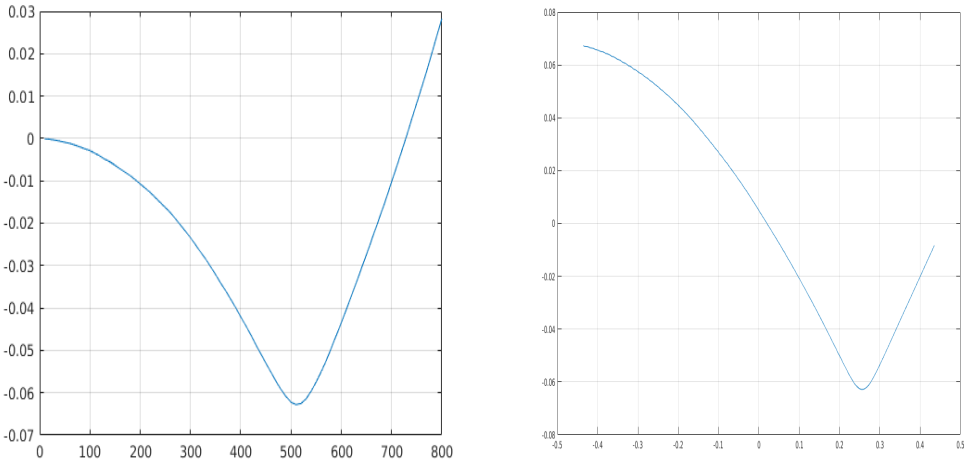
This raggedness can be attributed to the `Bin`(., .) function. The mathematics assumes that everything is continuous but while dealing with images, everything is discrete. So for instance, while rotating an object, a 1-degree rotation may lead to no rotation at all because

the resolution may not be good enough. However, once the image is smoothened, the function also becomes smoother and is easier to deal with. On applying heavier smoothing on the original shape, the function becomes even smoother (noise is added to this smoothened image). Furthermore, as the image resolution is increased, all these artifacts go away. Figure 7.6 shows the plot of objective functions with respect to s and w_i keeping the other parameters set to their optimal. The minimas in these plots correspond to the optimal minimas.



(a) Non-smoothened (b) Smoothened

Figure 7.5: Objective functions for non-smoothened and smoothened ellipses



(a) With s (b) With w_i

Figure 7.6: Objective functions for a smoothened ellipse as a function of s and w_i

7.1.3 Results after smoothening

This section builds on the previous section and considers smooth ellipses to learn the fits. On top of this, the mask to sum the log ratio values used in `Sim`(., .) is also smooth on the edge. This helps to assign a decreasing weight to pixels near the boundary. Figure 7.7 shows one such ellipse and figure 7.8 shows the absolute difference for this case.

It is clear that here reconstruction errors are higher. This is because here $\mu_{in} < 1$, and this leads to a cascading effect on the other parameters and they are not able to learn as good as in the previous case.

7.2 Cap bone

This section deals with learning the fits on 3D images. The images here are of the Capitate bone that is found in our wrists. Unlike the previous case, we have 100 x 100 x 100 images here, to begin with. So we can employ the ideas from Chapter 5 to learn a PCA model.

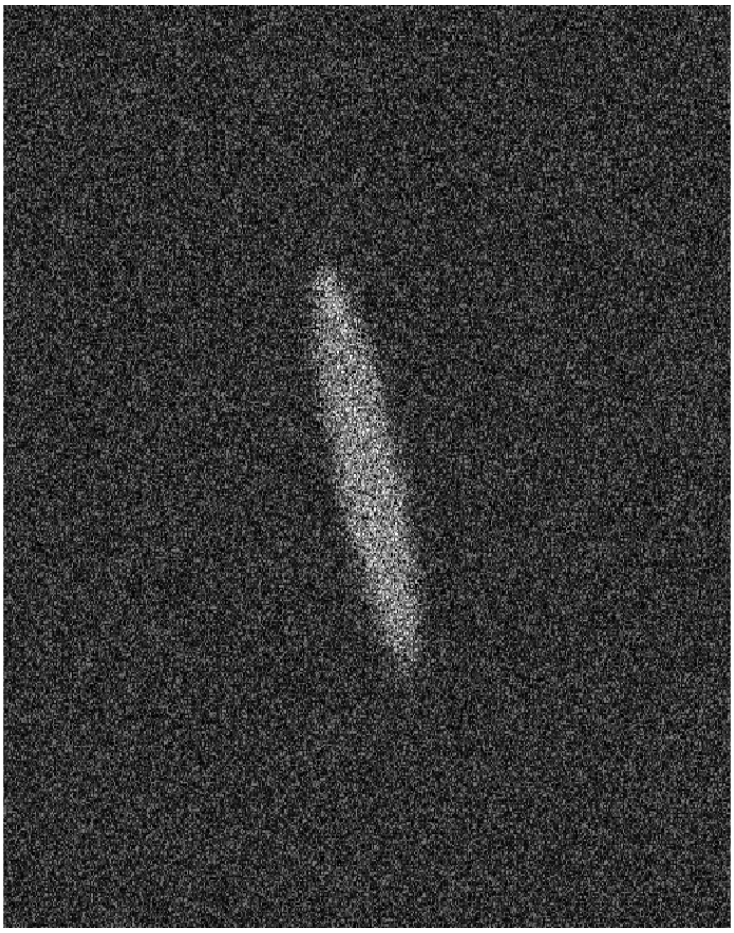


Figure 7.7: An example ellipse after smoothing

This model uses 482 landmark points to define a shape.

Two extra measures need to be taken for 3D shapes:

- `Bin`(., .) will not work as used for 2D images. Just by connecting the consecutive points, we will not be able to make a closed shell and the `imfill()` won't work. To solve this, we need to fill all the triangles through which the hull of the shape is defined. This requires iterating over all the triangular meshes and setting their interior intensities to 1. After this is done, we get a closed shell and `imfill()` can be applied on it.

- The rotation matrix is no longer defined in terms of a single rotation parameter but

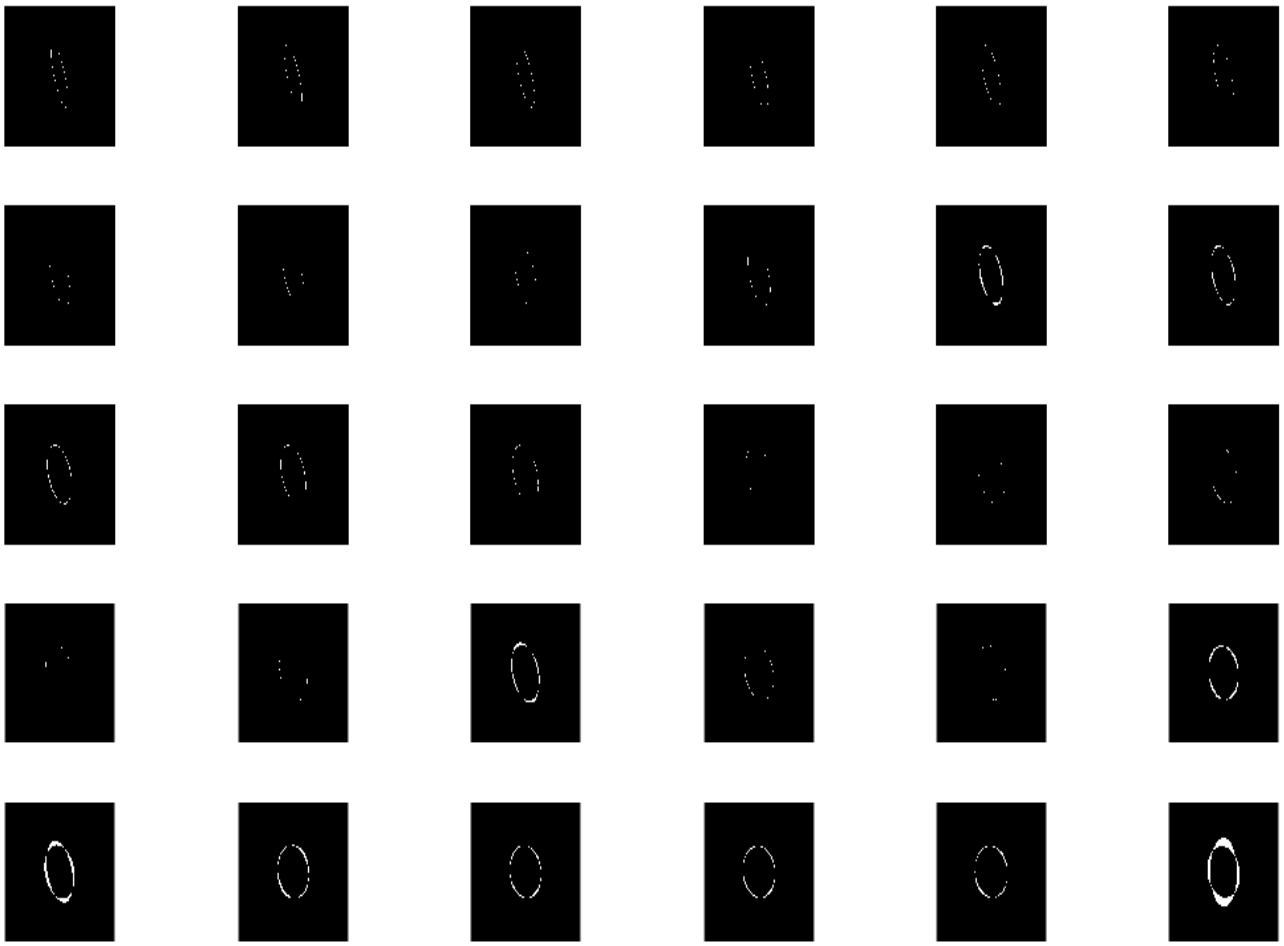


Figure 7.8: Absolute difference of intensities of data and fitted ellipses

has 3 degrees of freedom, hence 3 parameters. Imposing a non-linear constraint of 9 variables through $\mathbf{R^T R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$ can lead to lot of numerical errors. To circumvent this, we calculate $\mathbf{R} = \exp(\mathbf{M})$ as the matrix exponential of a skew-symmetric matrix,

$$\mathbf{M} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

With this structure of $\mathbf{M}, \mathbf{R}$ corresponds to rotation around an axis parallel to $[a, b, c]^T$ by an angle $\sqrt{a^2 + b^2 + c^2}$.

Having resolved these, we still are left with important issues. The `sqrt`(top 2 eigenvalues) = (0.0411, 0307) are very small in this case. Owing to this, the optimal coefficients would be small and close to each other. This presents numerical difficulties for the search algorithm. Also, the binarization operation significantly slows down here because (i) we have a larger number of pixels in the image to assign intensities to and (ii) because we have to iterate over the triangle meshes and fill them with 1.

The

Chapter 8

Conclusion

The ideas mentioned in this paper revolve around shape analysis and efficient segmentation of images. Dictionaries and Riemannian PCA based concepts have been mentioned. Although dictionary based segmentation methods were not implemented and tested, they hold potential and could be explored in a future work. The use of a new similarity measure between an image and a contour has been explored and that has guided the experiments. In addition to theoretical concepts, numerical optimization has been a major bottleneck which took a significant amount of time to solve.

We observe good results for segmentation using this measure and this certainly shows potential of being applicable to a large number of 3D images.

Chapter 9

Future Work

The next steps involving making the model robust with respect to 3D images so as to improve the fitting procedure. Furthermore, analysis and comparison with the segmentation provided by ShapeWorks needs to be done.

Another extension of the project would be to implement the same ideas using a Riemannian dictionary based approach and comparing the results.

Bibliography

[1] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[2] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 285–339, 1991.

[3] D. G. Kendall. A survey of the statistical theory of shape. *Statistical Science*, pages 87–99, 1989.

[4] S. J. Shigwan and S. P. Awate. Hierarchical generative modeling and monte-carlo EM in riemannian shape space for hypothesis testing. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016 - 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part III*, pages 191–200, 2016.

[5] Y. Xie, J. Ho, and B. Vemuri. On a nonlinear generalization of sparse coding and dictionary learning.