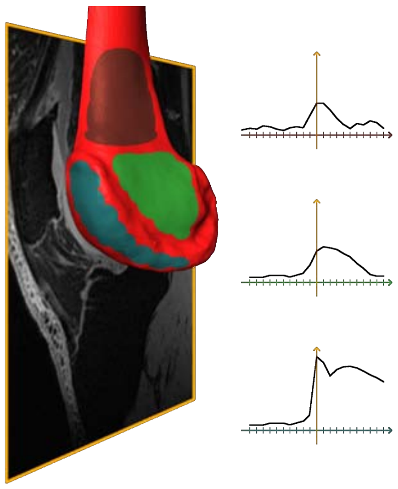
Université Libre de Bruxelles

Faculté des Sciences Appliquées

Improvement of Image Segmentation Based on Statistical Shape and Intensity Models



Mémoire de fin d'études présenté par Maximilien RENARD en vue de l'obtention du diplôme de Master en **Ingénieur Civil Biomédical** à finalité Informatique et imagerie biomédicales, **option Informatique et imagerie**

Directeur de mémoire:	Olivier DEBEIR	ULB Année académique **2010-2011**

Contents

Acknowledgements

I would like to thank my supervisor at ULB, Olivier Debeir, for his comments on my work, his support and the many proofreadings he did to help me.

This Master thesis wouldn't have been possible if I hadn't done my training during the summer of 2010 in the *Konrad Zuse Zentrum für Informationstechnick* in Berlin (also known as ZIB) where the whole staff of the Medical Planning group warmly welcomed me.

In particular, I would like to thank Stefan Zachow and Hans Lamecker who suggested and helped me to write the subject of the thesis. They allowed me to use the Amira® software that is developed at the ZIB and all the material that I used during my thesis. Stefan also proofread my work several times.

Carl Martin Grewe was the person with whom I had most contacts at the ZIB and who helped me to develop the first implementation of the NEM algorithm that we had made in R during my training. I am very grateful for his support and his help throughout the thesis.

On a personal note, I would like to thank my friend Élodie Romnée for making the title page template and gaving it to me. I would also like to thank all my friends (especially Louise Gonda, Frédéric Morel and Élodie Romnée), my family and my girlfriend for their support during all these years.

Abstract

The aim of the thesis is to propose and analyse a method for improving the segmentation process based on Statistical Shape Models (SSM) by the means of a model based on intensity profiles.

These profiles are sampled along the normals of each point of the surfaces in the training-set that were used to build the Statistical Shape Model.

Once the profiles are sampled, they are normalised and then the profiles of each data-set in the training-set is clustered separately using the Neighbourhood Expectation Maximisation (NEM) algorithm which unlike the regular EM algorithm performs a spatial regularisation by taking neighbourhood information into account which remove some noise, i.e. geometrical distance between profiles is added to the profile comparison introducing local smoothing of the obtained clusters.

Finally a fusion step is performed on the different clustering solutions to merge clusters that are similar. This allows to drastically diminish the number of different clusters and provide a statistical significance for the model.

The goal is to find zones where the profiles are similar on all of the surface in the training set. This would enable to enhance the segmentation process with the SSM. In fact, it could then deform the surface of the models in such a way that these zones find their matching profiles in the segmented image.

To determine the optimal number of clusters, the Overlap Separation Index (OSI) proposed by F. Chung was analysed. It turns out that the method isn't suited for the use on the posterior probabilities generated by the NEM algorithm due to the sharpness of the resulting clusters. For future work I proposed to compute the OSI on a fuzzy clustering initialised with the results of the NEM algorithm.

An heuristic was developed to estimate the optimal parameter for the spatial regularisation performed in the NEM algorithm. It proved to be rather successful as it provided valid values for the parameter sought.

Before studying the merging of the clusters itself, attention was paid to the measure that allows to compare two clusters and to finally decide if they should be merged or not. The Jaccard index was used and although it gives good results, its nature prevents to take into account big differences between the means of two clusters as it is only based on the surface delimited by the standard deviation around the mean.

The last step of the global algorithm was analysed too and it seems that another method should be found to merge clusters as the one proposed by F. Chung. In fact, merging clusters that are similar regarding their intensity values only, can dramatically expand them spatially producing clusters that cover a high percentage of the surface with small cluster memberships. As the goal is to find zones with very high of such membership values, that particular point should also be improved.

All in all, the method works quite well and the proposed improvements should allow to move towards the realisation of a powerful intensity model which could then lead to a better segmentation using Statistical Shape Models.

Introduction

Introduction on Image Segmentation For the segmentation of 3D image volumes, various methods do exist, covering a wide range of complexity. The simplest method is manual segmentation which is user-dependent, tedious, labour-intensive and thus time consuming. Semi-automatic methods reduce both the user interaction and the aforementioned drawbacks. Meanwhile, fully automated methods do exist which are based on shape priors. One possible class of shape priors are the so called Statistical Shape Models (SSM). Such SSMs were among others described by Cootes [1].

Statistical Shape Models The particularity of SSMs is that they are built in such a way that they are both general (i.e. they are capable of generating shapes that are not present in the training set) and specific, i.e. they only generate shapes that can be encountered in the real world.

The main drawback of this method is that it cannot be used to describe amorphous objects with high deformability. This is due to the necessity of finding landmarks, i.e. points that can be found on the structure of every data in the training set. These landmarks are used to establish a relationship between points on the various shapes of the training set and eventually align and scale them i.e. to perform registration. The surfaces are split in patches which have a disc topology. This allows point-to-point correspondence between the surfaces via a bijective mapping of their connectivity information.

The result of the creation of such a SSM as described in Section 1.1 is a mean shape $\overline{X}$ and a set of shape modes d_i and their associated weights w_i.

$$S(\boldsymbol{w}) = \overline{X} + \sum_{i=1}^{M} w_i d_i \tag{1}$$

Segmentation using Statistical Shape Models As described in chapter 1.2, the shape represented by the model has to be deformed to perform segmentation of an image volume not included in the training set. This is done according to cost functions and heuristic methods which determine whether or not a vertex of the surface model should be moved or not.

Cootes et al. proposed a method called Active Shape Models [1] which determines these cost functions for each point of the model by performing a PCA on the intensity profiles, which are sampled along the surface normals of any vertex for each 3D image volume in the data-set (see Fig. 1 and Fig. 2). This allows to establish a characteristic profile for each sample point of the shape model. It is then possible to compute the similarity between stored profiles and those that are found within the image data to be segmented, by using a suitable metric like the Mahalanobis distance, which takes the correlation between each sample point of two profiles into account. This allows to move each point (and thus the shape model boundary) along its normal to a new position where the distance between its average profile and the profile sampled at the new position is minimal. Once the distance has been minimised for each point, the parameters of the model are recomputed to best fit the model to the new positions of its sample points.

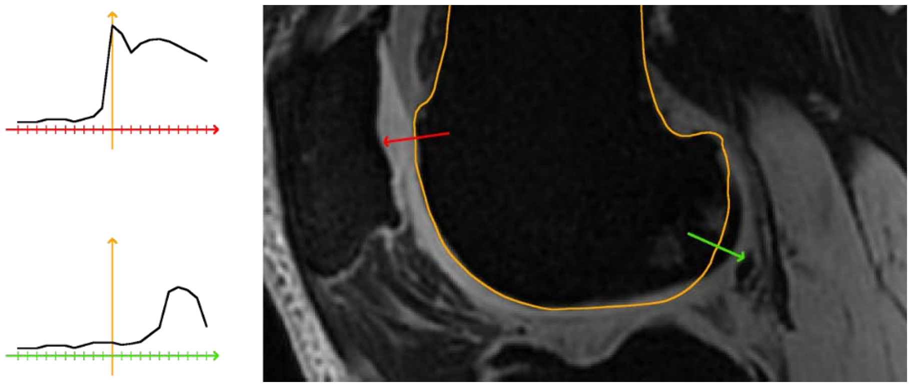


Figure 1: Left: Plots of two different intensity profiles (X-axis sample points, Y-axis intensity value). Right: MRI slice with the normals of the surface along which the two profiles were extracted.

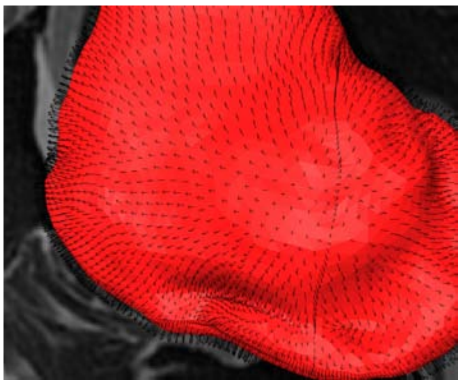


Figure 2: 3D surface (gold standard) obtained after manual segmentation. Normals to the surface at each vertex are displayed.

Limitations Using Active Shape Models allows to utilise a characteristic profile at each sample point to establish a cost function in an automatic fashion. However, the assumption is made that there is a correlation between the geometry of the shape and the intensity profiles i.e. corresponding points of the shape in different data-sets have a similar intensity profile. This is often not the case and introduces errors in the model.

It is however possible to evaluate the cost functions in a heuristically fashion which gives very good results [2]. Since they are basically a set of formulae used under certain conditions that can not be described as a single mathematical function, they are very difficult to optimise, i.e. computation of derivatives is not possible. Furthermore their efficiency is user-dependent and although they do work for the image volumes in the training set, there is no guarantee that they would give good results for new image data.

In practise this works well in many cases as shown with [3] and [4] in the MICCAI segmentation contests.

Goals of the Thesis The goal of this thesis is to propose and study a method for generating a profile intensity model which could be used to create cost functions in an automatic fashion for model-based segmentation.

The last limitation described above could be solved by using a method proposed by F. Chung et al. which consists of performing a neighbourhood-aware intensity profile clustering [5] to extract these typical profiles for regions of each shape of the training set (instead of for each point. See Fig. 3). The method is based on a Gaussian Mixture Model clustering optimised by a modified version of the Expectation-Maximisation algorithm which was extended to take the neighbourhood into account (it is thus called *Neighbourhood EM* or *NEM*). For each shape, the result is a smoothed, spatially regularised clustering of the intensity profiles. In order to establish an intensity profile model, the method is first run separately on each shape in the training set and then a fusion step takes place where similar clusters are merged and where all points of a reference shape are given a probability to belong to their respective classes.

This intensity model could then be used to determine cost functions automatically. Some regions of the shape are likely to have a high probability of belonging to a particular class. This information could be used to create a cost function for these regions by comparing the result of a Principal Component Analysis of the profiles in this class with the profile of the current point using a suited distance measure. In the process of matching the entire shape using a SSM, the regions with high class-membership probabilities could be more weighted than those where the variance in intensity profiles was higher which would more or less follow elastically (due to the bounded shape space).

The work presents the whole algorithm, discusses the key problems of its implementation and proposes solutions to these.

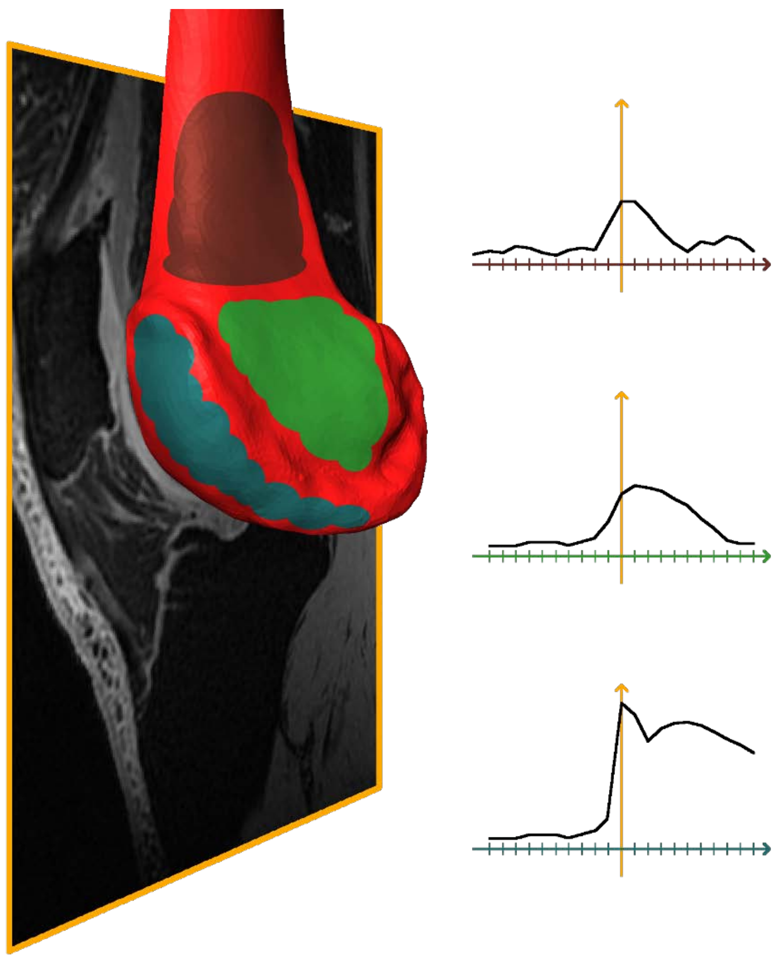


Figure 3: Left: 3D surface with 3 zones where the vertices have a high probability to belong to one class respectively. Right: Typical profiles of the 3 classes.

Part I

Backgrounds

Introduction

This part briefly presents various concepts that are mandatory for the understanding of the presented work.

The chapter 1.1 describes one of the existing types of models based on shape priors: Statistical Shape Models (SSM).

In chapter 1.2, it is explained how the SSMs are used to perform segmentation of the object in unknown images.

Finally, chapter 2 introduces the Estimation-Maximisation (EM) algorithm which will be modified as described in [6].

Chapter 1

Statistical Shape Models

1.1 Building a Statistical Shape Model

In order to build the model, a training set $\boldsymbol{X}$ made of M meshes of N vertices of the object of interest manually segmented in the M corresponding image data, is used. It can be represented as a $M \times N$ matrix where each row $\boldsymbol{x}_i$ (in this document, any element that is not a scalar will be written in bold) contains the coordinates of all vertices of mesh j in the training set:

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{1j} & \cdots & x_{1N} \\ x_{i1} & x_{ij} & \cdots & x_{iN} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{Mj} & \cdots & x_{MN} \end{pmatrix} \tag{1.1}$$

For $\mathbb{R}^n$ there are actually $n \times N$ values in a column, thus for $\mathbb{R}^3$ the matrix has an actual size of $M \times 3N$ and looks like:

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & y_{11} & z_{11} & \cdots & x_{1j} & y_{1j} & z_{1j} & \cdots & z_{1N} \\ x_{i1} & y_{i1} & z_{i1} & \cdots & x_{ij} & y_{ij} & z_{ij} & \cdots & z_{iN} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{M1} & y_{M1} & z_{M1} & \cdots & x_{Mj} & y_{Mj} & z_{Mj} & \cdots & z_{MN} \end{pmatrix} \tag{1.2}$$

However, the number of columns will be N in the following document. Once the shapes are aligned, a Principal Component Analysis (PCA) is run on X in order to extract the shape modes. For this purpose, a mean mesh $(1 \times N)$ is created:

$$\overline{\boldsymbol{X}} = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{x}_i \tag{1.3}$$

Then the data is recomputed in the mean deviation form $d\boldsymbol{X}$:

$$d\boldsymbol{x}_i = \boldsymbol{x}_i - \overline{\boldsymbol{X}} \qquad \forall i = 1, \ldots, M \tag{1.4}$$

The $M \times M$ covariance matrix $\boldsymbol{S}$ is computed:

$$\boldsymbol{S} = \frac{1}{M-1} d\boldsymbol{X} d\boldsymbol{X}^T \tag{1.5}$$

Then the matrix $\boldsymbol{V}$ that diagonalises $\boldsymbol{S}$ is computed:

$$\boldsymbol{V}\boldsymbol{S}\boldsymbol{V}^{-1} = \boldsymbol{\Lambda} \tag{1.6}$$

$\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues and $\boldsymbol{V}$ the orthogonal matrix ($\boldsymbol{V}^{-1} = \boldsymbol{V}^T$) of eigenvectors:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_i & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \lambda_M \end{pmatrix} \quad (1.7) \;, \quad \boldsymbol{V} = \begin{pmatrix} v_{11} & v_{12} & v_{1j} & \cdots & v_{1M} \\ v_{21} & v_{22} & v_{2j} & \cdots & v_{2M} \\ v_{i1} & v_{i2} & v_{ij} & \ddots & v_{iM} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ v_{M1} & v_{M2} & v_{Mj} & \cdots & v_{MM} \end{pmatrix} \quad (1.8)$$

The eigenvalues and their associated eigenvectors are ordered so that $\lambda_i \geqslant \lambda_{i+1}$.

An important note concerns the relationship between the "set size over element size" ratio and the number of modes. In fact, if M is strictly greater than N, there are $P = N$ shape-modes where in the other case, there are only $P = M - 1$ modes. This is illustrated on the Fig. 1.1.

In the specific case of the Statistical Shape Models, it is likely that there will be much more vertices (N) than meshes in the training set (M) thus leading to $P = M - 1$ modes..

The shape-modes $\boldsymbol{D} = \{\boldsymbol{d}_i\}_{i=1,\dots,P}$ are computed:

$$\boldsymbol{D} = \boldsymbol{V}\, \boldsymbol{dX} \tag{1.9}$$

Finally the training data of mesh l can be recomputed as $\hat{X}_l$ so that

$$\hat{\boldsymbol{X}}_l = \overline{\boldsymbol{X}} + \boldsymbol{w}_l \boldsymbol{D} \tag{1.10}$$

where $\boldsymbol{w}_l$ is the vector of size P of weights of the corresponding shape-modes.

Since the eigenvectors are ordered in descending eigenvalue order, the first one has the biggest variance where the last one has the smallest. This allows to select only some modes and still be able to describe the data sufficiently since each supplementary mode adds fewer information. It is possible to measure the percentage of variance explained E when selecting $p < P$ modes:

$$E = \frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{P} \lambda_i} \tag{1.11}$$

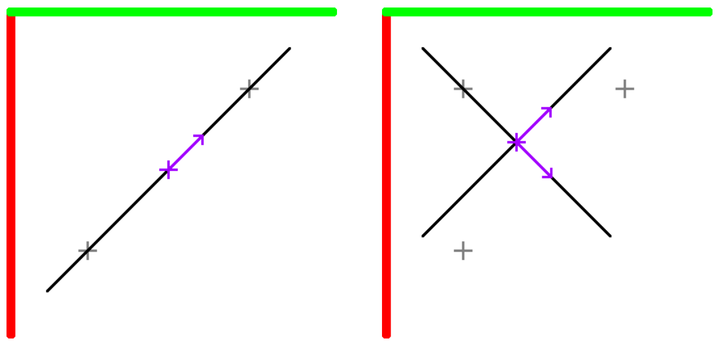


Figure 1.1: The left figure shows 2 points in $\mathbb{R}^2$ (thus $M = 2$ and $N = 2$). Running a PCA would choose the black line as the new axis, clearly showing that the two points would be described by the sum of the mean point (magenta cross) and a weighted contribution of the only mode (shown as the magenta arrow). The right figure shows 3 points in $\mathbb{R}^2$ (thus $M = 3$ and $N = 2$). Doing the same as above, it is shown that now, there as many modes as space dimensions.

1.2 Segmentation Using Statistical Shape Models

Once the model is built, it can be used to perform segmentation of shapes inherently present in medical image data. In case the training set is representative, a similar shape of a new object can be represented as:

$$S(\boldsymbol{w}, T) = T\left(\overline{\boldsymbol{X}} + \sum_{p=1}^{P} w_p \boldsymbol{d}_p\right) \tag{1.12}$$

As described above, $\boldsymbol{w}$ is the vector of the weights assigned to each mode of the model, $\boldsymbol{d}_p$ are the shape-modes and T is a linear transformation applied to the entire shape.

First, the average shape as described by the model ($w_p = 0 \, \forall p \in [1, P]$) is aligned in a suitable initial position within a given data-set [7]. This is the first iteration k of the segmentation process and is thus denoted $S(w^{(k)}, T^{(k)})|k = 0$.

Then, the shape is deformed to match image features such as high gradients, matching of a stored intensity profile with an actual one,... Typically this is done in a direction normal to the tangential plane of the respective point using a cost function. Points along the displacement direction are each given a cost that represents the probability to move the current point to them. The costs, for instance, can be the inverse of the gradient at a particular location of the profile. It is then possible to implement a heuristic method that determines, through the use of tests and conditions, whether or not the current point should be moved.

This deformation can be expressed for each point of the shape as a vector $\Delta \boldsymbol{x}_j \in \mathbb{R}^3$ such as the new shape is given by:

$$\hat{X}^{(k)} = S(w^{(k)}, T^{(k)}) + \Delta X \tag{1.13}$$

In order to represent the deformed shape as good as possible with respect to the shape space of the training set, the weights and the transformation of the deformable shape are recomputed by solving the optimisation problem:

$$(w^{(k+1)}, T^{(k+1)}) = \arg\min_{w,T} \left| \hat{X}^{(k)} - S(b, T) \right|^2 \tag{1.14}$$

The process is iterated ($k \leftarrow k + 1$) until shape convergence i.e. while:

$$\left| S(w^{(k)}, T^{(k)}) - S(w^{(k+1)}, T^{(k+1)}) \right| > 3N\epsilon \tag{1.15}$$

The model thus serves as a guide to constrain the resulting shape in the world of plausible shapes as described by the model [8] i.e. to constrain each w_i to its respective boundary values found in the training set to prevent the generation of unrealistic shapes (not covered by the given shape space).

In case the SSM is not able to fully cover the shape inherently being given by the image data, a 'free form' deformation step allows a final adjustment of the geometry to the image data [2].

Chapter 2

Clustering of Data Using the EM Algorithm

The algorithm described in [6] is based on a modified version of the Expectation-Maximisation (EM) optimisation algorithm applied to the Gaussian Mixture Models. The purpose of this chapter is to briefly explain how it works in order to make the explanation of the modifications easier.

This chapter rephrases parts of [6] and [9] in an attempt to unify it with my work and maybe to clarify some points.

2.1 Gaussian Mixture Models

A Gaussian Mixture Model is made of various components:

data $\mathcal{X}$: M independently and identically distributed (i.i.d.) observed variables $\boldsymbol{x}_i$ (for $i = 1, \ldots, M$) in $\mathbb{R}^d$ from a mixture of K normal distributions f_k.
This is referred to as the *incomplete data*.

mixture weights: K probabilities π_k that satisfy $0 < \pi_k < 1 \, \forall k \in [1, K]$ and $\sum_{k=1}^{K} \pi_k = 1)$

distribution parameters: K sets $\boldsymbol{\theta}_k$ of parameters consisting of the mean $\boldsymbol{\mu}_k$ and the covariance matrix $\boldsymbol{\Sigma}_k$ of each distribution f_k of the mixture.

Often, one other component is defined for clarity's sake:

mixture parameter: $\boldsymbol{\Phi} = (\pi_1, \pi_k, \ldots, \pi_K, \boldsymbol{\theta}_1, \boldsymbol{\theta}_k, \ldots, \boldsymbol{\theta}_K)$

Each observation $\boldsymbol{x}_i$ in the data thus follows the distribution:

$$f(\boldsymbol{x}_i|\boldsymbol{\Phi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.1}$$

and the density for all the samples is:

$$f(\mathcal{X}|\mathbf{\Phi}) = \prod_{j=1}^{N} f(\boldsymbol{x}_i|\mathbf{\Phi}) = \mathcal{L}(\mathbf{\Phi}|\mathcal{X}) \tag{2.2}$$

which is also called the *likelihood* $\mathcal{L}(\mathbf{\Phi}|\mathcal{X})$ of the parameters considering the data. This quantity should be maximised in order to describe the data $\mathcal{X}$ as good as possible.

While $\mathcal{X}$ being fixed, the only parameter which can be optimised to find the biggest likelihood is $\mathbf{\Phi}$:

$$\mathbf{\Phi}^* = \arg\max_{\mathbf{\Phi}} \mathcal{L}(\mathbf{\Phi}|\mathcal{X}). \tag{2.3}$$

In the case of the Gaussian Mixture Model, this is not immediate and methods exist that enable this calculation. One of these is the Expectation-Maximisation algorithm.

2.2 Expectation-Maximisation Algorithm

In addition to the incomplete data $\mathcal{X}$, another variable is defined:

unobserved data $\mathcal{Y}$: M i.i.d. unobserved variables $\boldsymbol{y}_i$ (for $i = 1, \cdots, M$). Together with $\mathcal{X}$, it forms the *complete data* $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$. This variable is also considered as i.i.d. since it is unknown, random and probably following an underlying distribution.

A joint density function $f(\mathcal{Z}|\mathbf{\Phi})$ is assumed such as:

$$f(\mathcal{Z}|\mathbf{\Phi}) = f(\mathcal{X}, \mathcal{Y} \mid \mathbf{\Phi}) = f(\mathcal{Y}|\mathcal{X}, \mathbf{\Phi}) f(\mathcal{X}|\mathbf{\Phi}). \tag{2.4}$$

It is thus possible to define a complete-data likelihood function:

$$\mathcal{L}(\mathbf{\Phi}|\mathcal{Z}) = \mathcal{L}(\mathbf{\Phi} \mid \mathcal{X}, \mathcal{Y}) = f(\mathcal{X}, \mathcal{Y} \mid \mathbf{\Phi}). \tag{2.5}$$

In order to make analytical resolution of the maximisation easier, the complete-data **log**-likelihood $L(\mathbf{\Phi}|\mathcal{Z})$ is used:

$$L(\mathbf{\Phi}|\mathcal{Z}) = \log\left(\mathcal{L}(\mathbf{\Phi}|\mathcal{Z})\right) = \log\left(\prod_{i=1}^{M} f(\boldsymbol{z}_i|\mathbf{\Phi})\right) = \sum_{i=1}^{M} \log f(\boldsymbol{z}_i|\mathbf{\Phi}). \tag{2.6}$$

The EM algorithm is an iterating process consisting of 2 steps.

The first one is called Expectation Step (or E-Step) because its goal is to find the expected value $\mathcal{Q}(\mathbf{\Phi})$ of $L(\mathbf{\Phi}|\mathcal{Z})$ with the current estimate of the parameter $\mathbf{\Phi}^{(s-1)}$:

$$\mathcal{Q}\left(\mathbf{\Phi}, \mathbf{\Phi}^{(s-1)}\right) = E\left[\log f(\mathcal{X}, \mathcal{Y} \mid \mathbf{\Phi}) \mid \mathcal{X}, \mathbf{\Phi}^{(s-1)}\right] \tag{2.7}$$

where $\mathcal{X}$ and $\mathbf{\Phi}^{(s-1)}$ are fixed for this step and $\mathcal{Y}$ follows the distribution $f\left(\boldsymbol{y} \mid \mathcal{X}, \mathbf{\Phi}^{(s-1)}\right)$. This allows to rewrite the right side of equation (2.7) as:

$$E\left[\log f(\mathcal{X},\mathcal{Y}|\boldsymbol{\Phi}) \mid \mathcal{X}, \boldsymbol{\Phi}^{(s-1)}\right] = \int_{\boldsymbol{y}\in\boldsymbol{\Upsilon}} f\left(\boldsymbol{y} \mid \mathcal{X}, \boldsymbol{\Phi}^{(s-1)}\right) \log f(\mathcal{X},\mathcal{Y} \mid \boldsymbol{\Phi}) d\boldsymbol{y} \tag{2.8}$$

where $\boldsymbol{\Upsilon}$ denotes the space of the values $\boldsymbol{y}$ can take.

It is now possible to perform the second step, the Maximisation-Step (or M-Step), where $\mathcal{Q}\left(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(s-1)}\right)$ is maximised:

$$\boldsymbol{\Phi}^{(s)} = \arg\max_{\boldsymbol{\Phi}} \mathcal{Q}\left(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(s-1)}\right). \tag{2.9}$$

The process is iterated as long as needed and, for each step, the log-likelihood monotonically increases:

$$L\left(\boldsymbol{\Phi}^{(s+1)}\right) \geq L\left(\boldsymbol{\Phi}^{(s)}\right). \tag{2.10}$$

2.3 Using the EM Algorithm to Estimate Mixture Parameters

For the case of the Gaussian Mixture Model, the equation (2.2) can be written as:

$$\log\left(\mathcal{L}(\boldsymbol{\Phi}|\mathcal{X})\right) = \log\prod_{i=1}^{M} f(\boldsymbol{x}_i|\boldsymbol{\Phi}) = \sum_{i=1}^{M}\log\left(\sum_{k=1}^{K}\pi_k f_k(\boldsymbol{x}_i|\theta_k)\right) \tag{2.11}$$

Because of the sum in the logarithm, this function is really difficult to optimise and that is where the "trick" of posing $\mathcal{Y}$ happens.

If $\mathcal{Y}$ follows a K-dimensional categorical distribution (which requires $y_{ik} \in [0,1]\ \forall k \in [1,K]$ and $\sum_{k=1}^{K} y_{ik} = 1$), the value of $\boldsymbol{y}_i$ indicates which one of the K density functions of the mixture "generated" the sample.

Using equations (2.4) and (2.6) comes:

$$\log(\mathcal{L}(\boldsymbol{\Phi} \mid \mathcal{X},\mathcal{Y})) = \log(f(\mathcal{X},\mathcal{Y} \mid \boldsymbol{\Phi})) = \sum_{i=1}^{M}\log(f(\boldsymbol{x}_i|\boldsymbol{y}_i)f(\boldsymbol{y}_i)) \tag{2.12}$$

$$= \sum_{i=1}^{M}\log(f_{\boldsymbol{y}_i}(\boldsymbol{x}_i|\boldsymbol{\theta}_{\boldsymbol{y}_i})\pi_{\boldsymbol{y}_i})) \tag{2.13}$$

Since $\pi_{\boldsymbol{y}_i}$ can be seen as the prior probabilities of each component of the mixture, it is possible to use the Bayes's rule to write:

$$f\left(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{\Theta}^{(s-1)}\right)=\frac{f(\boldsymbol{y}_i)f_{\boldsymbol{y}_i}\left(\boldsymbol{x}_i|\boldsymbol{\theta}_{\boldsymbol{y}_i}^{(s-1)}\right)}{f\left(\boldsymbol{x}_i|\boldsymbol{\Theta}^{(s-1)}\right)}=\frac{\pi_{\boldsymbol{y}_i}^{(s-1)}f_{\boldsymbol{y}_i}\left(\boldsymbol{x}_i|\boldsymbol{\theta}_{\boldsymbol{y}_i}^{(s-1)}\right)}{\sum_{k=1}^{K}\pi_k^{(s-1)}f_k\left(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{(s-1)}\right)} \tag{2.14}$$

and thus:

$$f\left(\boldsymbol{y}|\mathcal{X},\boldsymbol{\Theta}^{(s-1)}\right)=\prod_{i=1}^{M}f\left(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{\Theta}^{(s-1)}\right) \tag{2.15}$$

which is the marginal density function needed in equation (2.8).

Since $\boldsymbol{\Upsilon}$ can only take a discrete set of values, it comes:

$$\begin{aligned}\mathcal{Q}\left(\boldsymbol{\Phi},\boldsymbol{\Phi}^{(s-1)}\right)&=\sum_{\boldsymbol{y}\in\boldsymbol{\Upsilon}}f\left(\boldsymbol{y}\mid\mathcal{X},\boldsymbol{\Phi}^{(s-1)}\right)\log f(\mathcal{X},\mathcal{Y}\mid\boldsymbol{\Phi}) &(2.16)\\&=\sum_{\boldsymbol{y}\in\boldsymbol{\Upsilon}}\sum_{i=1}^{M}\log(\pi_{\boldsymbol{y}_i}f_{\boldsymbol{y}_i}(\boldsymbol{x}_i|\boldsymbol{\theta}_{\boldsymbol{y}_i}))\prod_{l=1}^{M}f\left(\boldsymbol{y}_l|\boldsymbol{x}_l,\boldsymbol{\Theta}^{(s-1)}\right) &(2.17)\end{aligned}$$

which after a few intermediates steps (described in [9]) gives:

$$\begin{aligned}\mathcal{Q}\left(\boldsymbol{\Phi},\boldsymbol{\Phi}^{(s-1)}\right)&=\sum_{k=1}^{K}\sum_{i=1}^{M}\log(\pi_k f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k))f\left(k|\boldsymbol{x}_i,\boldsymbol{\Theta}^{(s-1)}\right) &(2.18)\\&=\sum_{k=1}^{K}\sum_{i=1}^{M}\log(\pi_k)f\left(k|\boldsymbol{x}_i,\boldsymbol{\Theta}^{(s-1)}\right)\\&\quad+\sum_{k=1}^{K}\sum_{i=1}^{M}\log(f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k))f\left(k|\boldsymbol{x}_i,\boldsymbol{\Theta}^{(s-1)}\right) &(2.19)\end{aligned}$$

This is the function that we want to estimate for the current estimation of $\boldsymbol{\Phi}$ in the E-step and then maximise in the M-step.

The E-step computes the terms of (2.19) that do not depend on $\boldsymbol{\Phi}$ that is to say $f(k|\boldsymbol{x}_i,\boldsymbol{\Theta}^{(s-1)})$. This term is the probability for a sample $\boldsymbol{x}_i$ to belong to class k and will further be referred to as $c_{ik}^{(s)}$ given by:

$$c_{ik}^{(s)}=\frac{\pi_k^{(s-1)}f_k\left(\boldsymbol{x}_i|\boldsymbol{\mu}_k^{(s-1)},\boldsymbol{\Sigma}_k^{(s-1)}\right)}{f\left(\boldsymbol{x_i}|\boldsymbol{\Phi}^{(s-1)}\right)} \tag{2.20}$$

The M-step computes $\mathbf{\Phi}^{(s)}$ that maximises (2.19). As it can be observed, each term of (2.19) can be maximised separately since π_k and $\boldsymbol{\theta}_k$ do not depend on each other.

Using Lagrange multipliers (cf. [9]) the following expressions can be derived for the new estimation of the mixing coefficient, mean vector and covariance matrix of class k:

$$\pi_k^{(s)} = \frac{1}{N}\sum_{i=1}^{M} c_{ik}^{(s-1)} \tag{2.21}$$

$$\boldsymbol{\mu}_k^{(s)} = \frac{\sum\limits_{i=1}^{M} \boldsymbol{x}_i c_{ik}^{(s-1)}}{\sum\limits_{i=1}^{M} c_{ik}^{(s-1)}} \tag{2.22}$$

$$\mathbf{\Sigma}_k^{(s)} = \frac{\sum\limits_{i=1}^{M} c_{ik}^{(s-1)} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(s)}\right)\left(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(s)}\right)^T}{\sum\limits_{i=1}^{M} c_{ik}^{(s-1)}} \tag{2.23}$$

These values can be used as parameter estimates of the next E-step.

The process is iterated until the value of the log-likelihood converges.

2.4 Using the EM Algorithm as a Fuzzy Clustering Method

It has been shown (cf. [6], [10]) that optimising GMMs with the EM algorithm is equivalent to an iterative 2-steps optimisation of the following function:

$$D(\boldsymbol{c}, \mathbf{\Phi}) = \sum_{k=1}^{K}\sum_{i=1}^{M} c_{ik}\log(p_k f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k)) - \sum_{k=1}^{K}\sum_{i=1}^{M} c_{ik}\log(c_{ik}) \tag{2.24}$$

The function is optimised alternatively for the classification matrix $\boldsymbol{c}$ and mixture parameter set $\mathbf{\Phi}$.

As described in [6], the result of the optimisation of $\boldsymbol{c}$

$$\boldsymbol{c}^{(s)} = \arg\max_{\boldsymbol{c}} D\left(\boldsymbol{c}, \mathbf{\Phi}^{(s-1)}\right) \tag{2.25}$$

is equivalent to the E-Step in (2.20) thus:

$$c_{ik} = \frac{\pi_k^{(s-1)} f_k\left(\boldsymbol{x}_i|\boldsymbol{\mu}_k^{(s-1)}, \mathbf{\Sigma}_k^{(s-1)}\right)}{f\left(\boldsymbol{x_i}|\mathbf{\Phi}^{(s-1)}\right)} \tag{2.26}$$

The result of the optimisation of the mixture parameters set $\boldsymbol{\Phi}$

$$\boldsymbol{\Phi}^{(s)} = \arg\max_{\boldsymbol{\Phi}} D\left(\boldsymbol{c}^{(s)}, \boldsymbol{\Phi}\right) \tag{2.27}$$

is also equivalent to the former M-Step since in

$$D\left(\boldsymbol{c}^{(s)}, \boldsymbol{\Phi}\right) = \mathcal{Q}\left(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(s-1)}\right) - \sum_{k=1}^{K}\sum_{i=1}^{M} c_{ik}^{(s)} \log\left(c_{ik}^{(s)}\right) \tag{2.28}$$

the last term is independent of $\boldsymbol{\Phi}$.

The result is thus an optimised clustering solution consisting of K classes with their respective means $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_k$ as well as the mixing coefficients π_k. Moreover a fuzzy classification of the data is also obtained through the classes membership probabilities $\boldsymbol{c}_i$ of each data sample $\boldsymbol{x}_i$.

Part II

Methods

Chapter 3

Material

Sixty MRI image volumes of the proximal head of left and right femurs were used. In addition, each image was associated with the surface of its femur segmented by hand. This segmentation is considered as ground truth. This data will further be referred as the training-set.

To implement the algorithm and run tests, I was allowed to work on the source code of an advanced software called Amira®. It is developed at the Konrad Zuse Zentrum für Informationstechnik Berlin were I had the great opportunity to do a training during the summer of 2010.

This allowed me to implement the algorithm really quickly as Amira® offers a great variety of existing data-structures (meshes, matrices, vectors,...), efficient algorithms (singular value decomposition of matrices, volume rendering,...) and the associated visualisation tools.

Chapter 4

General Overview of the Algorithm

The algorithm to create a *Multimodal Prior Appearance Model* as described in [5] yields 2 steps:

1. The profiles extracted per vertex from each data-set are clustered separately using a *Gaussian Mixture Model* optimised by the *Neighbourhood Expectation-Maximisation* algorithm.
2. Some clustering solutions are possibly merged together to create a unique prototype of all profile classes found in the training data. If the set is large enough, the assumption can be made that it describes many profile clusters that could be encountered.

The following chapters describe respectively the first and the second step.

Chapter 5

NEM Clustering of Intensity Profiles

5.1 Data Extraction and Pre-Processing

A histogram normalisation should be performed on each image I in order to map the intensity range of the image to $[0; 255]$ as shown in Eq. 5.1.

$$I' = \frac{I - I_{min}}{I_{max} - I_{min}} \times 255 \tag{5.1}$$

It is possible though that this simple normalisation can be rather inefficient because the presence of noise in some images causes the intensity range to be very wide although most of the image information is located in a small intensity range (see Fig. 5.1). The normalisation would thus flatten most of their profiles too much (see Fig. 5.2).

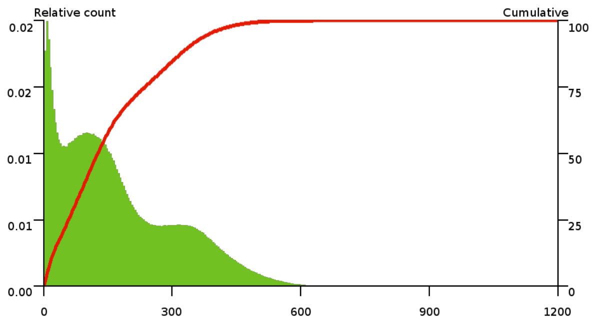


Figure 5.1: Original histogram of an image. Green: relative histogram of intensity values. Red: cumulative histogram. The presence of noise stretches the histogram although it is clear that all interesting values are located before 600.

I_{max} is thus chosen so that 99.5% of the pixels in the image has an intensity value lower or equal to it. After normalisation, all the pixels with a value greater

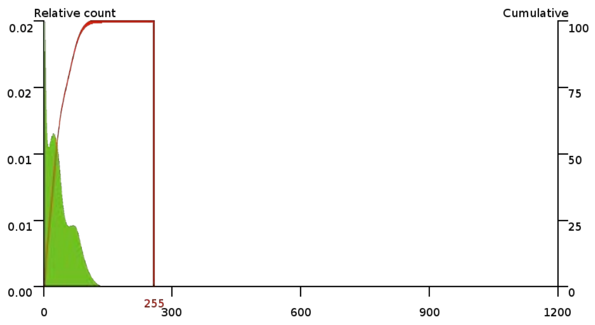


Figure 5.2: The normalisation of the histogram of Fig. 5.1 has failed since the interesting data has been compressed to much because of the noise in the original image.

than 255 are squashed to this limit. This removes the noise and ensures that there is no information loss due to this phenomenon.

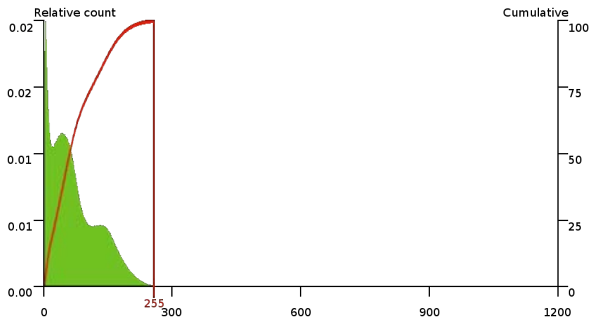


Figure 5.3: Taking the intensity value under which 99.5% of the pixels have their value, removes noise and allows to avoid loss of information.

As already described in the introduction, the intensity profiles are sampled along the normals of the vertices of the oriented surface associated with each image from the data-set. There are 3 main types of profiles that can be extracted:

Outward profiles: The profiles are sampled starting from the point on the surface along the normal at this point towards the outside.

Inward profiles: Same as above but the profiles are sampled in the other direction, pointing towards the inner part of the object.

Centred profiles: The centre of the profile is positioned on the surface point and it is sampled both in and outwards the object.

Of course, profiles sampling is not restricted to these 3 types. The choice depends on the object that is studied and on the image modality that was used to acquire the volume. For example, in the case of MR images of the femur

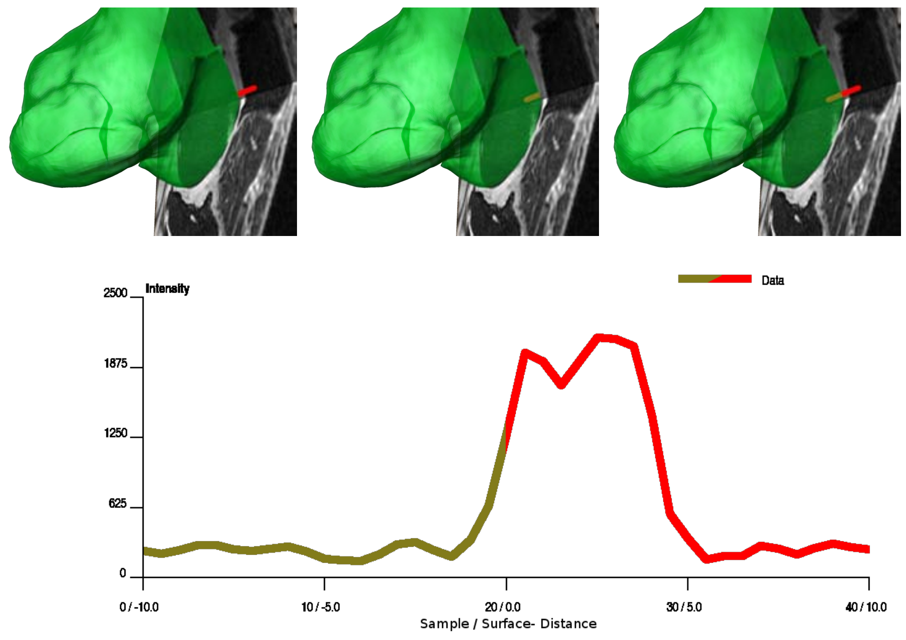


Figure 5.4: Left: Outward profiles. Middle: Inward profiles. Right: Centred profiles. Bottom: Corresponding profiles.

that were used, the inner part of the bones is almost completely and uniformly black and thus does not give any particular information so the profiles could have been sampled only outwards.

In order to generate a model with enough statistical significance, the data-set must contain more than one surface. F. Chung proposes to perform clustering on each image separately for 2 different reasons:

1. In doing so, it is not necessary to perform accurate registration between all image volumes. In fact, for the clustering of all images at once to make sense, the points where the profiles are extracted should correspond precisely.

2. In case an image is added to or removed from the training set, the whole clustering process has to be performed again. As it is very time and resource consuming, separate clusterings followed by clustering solutions fusion are more efficient in this case.

Some profile samples may be partially outside the image volume bounding box when the surface of the object is too close to the image boundaries (see Fig. 5.5). In [5], F. Chung describes a method for coping with such missing profiles. Since it wasn't really clear how this had to be done, those profiles were simply completely removed from the clustering.

The chosen approach for clustering the intensity profiles is a Gaussian Mixture Model optimised by a modified version of the Expectation-Maximisation

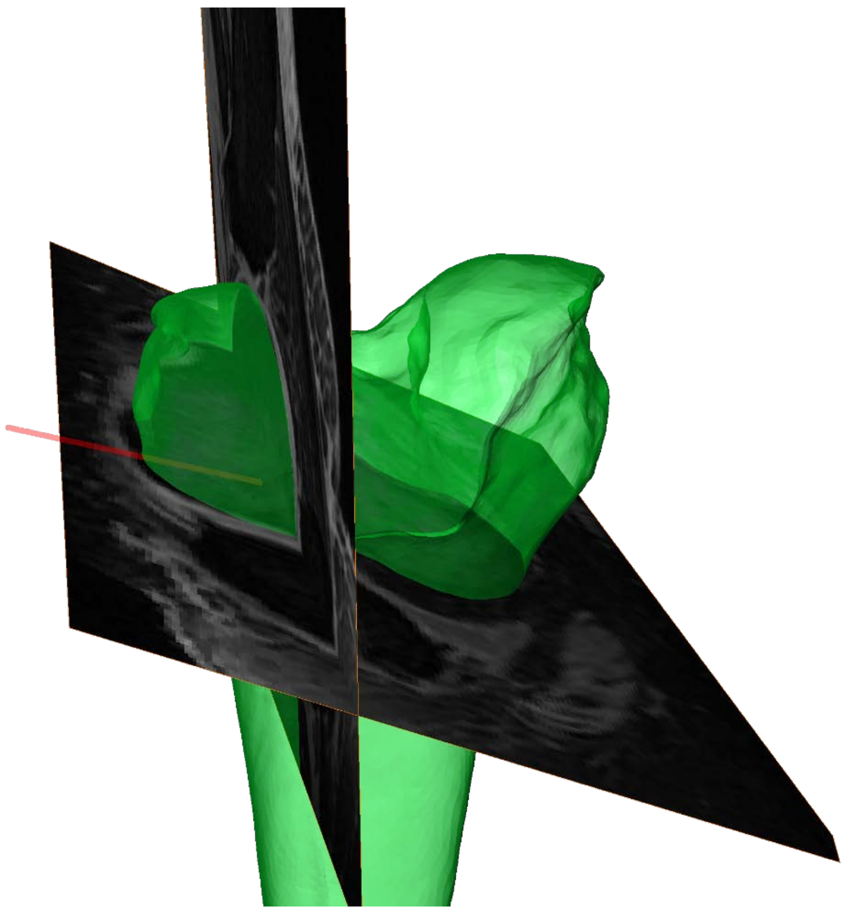


Figure 5.5: The figure shows an *incomplete* profile. The point where the profile is centred is too close to the image boundaries and some samples (red) are thus outside the image volume bounding box.

algorithm. The process of the said clustering with the regular EM algorithm is described in Chapter 2. The modified version is described hereunder.

5.2 Neighbourhood Expectation-Maximisation Algorithm

In order to obtain a spatially smoothed clustering, [6] proposes to add a new term G to the function D in (2.24):

$$G(\boldsymbol{c}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{M} c_{ik}c_{jk}v_{ij} \tag{5.2}$$

v_{ij} are the elements of the neighbourhood matrix $\mathcal{V}$. Its value must be positive otherwise it would penalise spatial regularity:

$$v_{ij} = \begin{cases} \alpha > 0 & \text{if } i \text{ and } j \text{ are neighbours} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

α can either be a constant value or depend on the pair of points. For example, the correlation between the two points could be used. This would allow to apply spatial regularisation only on profiles that are correlated.

The new function U to optimise is thus:

$$U(\boldsymbol{c}, \boldsymbol{\Phi}) = D(\boldsymbol{c}, \boldsymbol{\Phi}) + \beta G(\boldsymbol{c}) \tag{5.4}$$

where $D(\boldsymbol{c}, \boldsymbol{\Phi})$ is described in Eq. (2.24) and β is the parameter that controls the magnitude of the spatial regularisation. Its determination will be discussed in Section 8.

Since G does not depend on $\boldsymbol{\Phi}$, the M-Step remains unchanged and the updated E-Step yields:

$$c_{ik}^{(s)} = \frac{\pi_k^{(s-1)} f_k\left(\boldsymbol{x}_i | \boldsymbol{\mu}_k^{(s-1)}, \boldsymbol{\Sigma}_k^{(s-1)}\right) e^{\beta \sum_{j=1}^{K} c_{jk}^{(s)} v_{ij}}}{\sum_{l=1}^{K} \pi_l^{(s-1)} f_l\left(\boldsymbol{x}_i | \boldsymbol{\mu}_l^{(s-1)}, \boldsymbol{\Sigma}_l^{(s-1)}\right) e^{\beta \sum_{j=1}^{K} c_{jl}^{(s)} v_{ij}}} \tag{5.5}$$

As it can be seen, $c_{ik}^{(s)}$ depends on itself and thus the E-Step becomes iterative itself. As always, the iterative process stops when the value of $c_{ik}^{(s)}$ converges.

5.3 NEM Algorithm Initialisation

The Gaussian Mixture Model has to be initialised properly. The EM algorithm is in fact known to be sensitive to its initialisation. If it is not good enough, the algorithm could converge towards a local suboptimal solution.

For this purpose, several solution exists. The most common is an initialisation using the unsupervised K-Means clustering algorithm. However, due to its fuzzy nature, it could also be initialised with the fuzzy counterpart of the K-Means, the Fuzzy C-Means (FCM).

In the first case, after the clustering has converged, the hard partitioning P can be used to initialise the classes' membership matrix:

Let $1 < P_i \leq K$ be the class to which $\boldsymbol{x}_i$ was attributed,
Then $c_{iP_i} = 1.0$ and $c_{ij} = 0.0 \; \forall j \in [1, K], j \neq P_i$

and then, the M-Step can be performed one time to initialise the mixing coefficients, classes means and covariances matrices.

In the latter, the result of the clustering, $\boldsymbol{\gamma}$ the classes' membership matrix and $\boldsymbol{m}$ the classes' means, can directly be used to compute the parameter estimates for the E-Step:

$$\boldsymbol{c} = \boldsymbol{\gamma} \tag{5.6}$$

$$\pi_k^{(s)} = \frac{1}{N} \sum_{i=1}^{M} \gamma_{ik} \tag{5.7}$$

$$\boldsymbol{\mu}_k = \frac{\sum\limits_{i=1}^{M} \boldsymbol{x}_i \gamma_{ik}}{\sum\limits_{i=1}^{M} \gamma_{ik}} \tag{5.8}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum\limits_{i=1}^{M} \gamma_{ik} \left(\boldsymbol{x}_i - \boldsymbol{m}_k\right) \left(\boldsymbol{x}_i - \boldsymbol{m}_k\right)^T}{\sum\limits_{i=1}^{M} \gamma_{ik}} \tag{5.9}$$

Recently, Hu et al. proposed a new initialisation method conceived on purpose for the NEM algorithm (cf. [11]). The idea is to run the regular initialisation methods but on an "augmented" data-set $\boldsymbol{x}'$ where each sample $\boldsymbol{x}_i$ is adjoined the mean value of its neighbours ${}^{\mu}\boldsymbol{x}_i$:

$${}^{\mu}\boldsymbol{x}_i = \frac{\sum\limits_{j=1}^{M} v_{ij} \boldsymbol{x}_i}{\sum\limits_{j=1}^{M} v_{ij}} \tag{5.10}$$

$$\boldsymbol{x}'_i = [\boldsymbol{x}_i, \alpha\, {}^{\mu}\boldsymbol{x}_i] \tag{5.11}$$

where α is a parameter that controls the weight given to the neighbours.

In doing so, the samples that are neighbours and thus are likely to belong to the same class, will have a similar ${}^{\mu}\boldsymbol{x}_i$. This means that in term of distance, they will be closer to each other than profiles which have a different neighbours' mean.

It is also possible to use hierarchical clustering with different metrics and linkage criterions.

5.4 Determining the number of clusters

In unsupervised clustering, the selection of the number of clusters is a complex problem that has been studied for a long time. It is often impossible to determine it *a priori* and a common way to achieve the selection is to perform the clustering several times with an increasing number of clusters and then select the optimal solution. This selection is done by using a criterion which should be able to penalise over and under-fitting.

Over time many of such criteria were proposed among which the Bayesian Information Criterion (**BIC**) and the Akaike Information Criterion (**AIC**) (cf. [12] for a review of many criteria). They are often prone to over-fitting which in the case of the Intensity Profiles Model is absolutely to avoid as it will only make the fusion step harder.

In [5], F. Chung describes a new criterion named Overlap Separation Index (**OSI**). It only uses the posterior probabilities c_{ik} and penalises clusters overlap while encouraging their separation. It is computed as the ratio of the amount of overlap C_1 over the minimum separation between all clusters C_2. α_i and β_i are the clusters which have respectively the highest and second highest probabilities for profile i to belong to:

$$\alpha_i = \arg\max_k c_{ik} \qquad \forall k \in [1, K],\ i \in [1, M] \tag{5.12}$$

$$\beta_i = \arg\max_{k \neq \alpha_i} c_{ik} \qquad \forall k \in [1, K],\ i \in [1, M] \tag{5.13}$$

$$C_1 = \sum_{i=1}^{M} 2 \frac{c_{i\beta_i}}{c_{i\beta_i} + c_{i\alpha_i}} \tag{5.14}$$

$$\begin{aligned} S_{kl} = & \sum_{\forall i \in [1,M] | \alpha_i = k} 2 \frac{c_{il}}{c_{ik} + c_{il}} \\ & + \sum_{\forall i \in [1,M] | \beta_i = l} 2 \frac{c_{ik}}{c_{il} + c_{ik}} \qquad \forall k, l \in [1, K] \end{aligned} \tag{5.15}$$

$$C_2 = \min_{k,l} S_{kl} \qquad \forall k, l \in [1, K] \tag{5.16}$$

$$OSI = \frac{C_1}{C_2} \tag{5.17}$$

To select the optimal number of clusters, the clustering is run several times with an increasing amount of clusters. Only after that, the optimal solution can be selected by finding the one for which the OSI is the best.

Chapter 6

Fusion of the Clustering Solutions

6.1 Introduction

In order to obtain some statistical significance, the model has to be built on a large training set. There are three possible options :

1. Run the clustering on the whole training set at once i.e. all profiles extracted from all volumes are pooled,
2. Run the clustering on each data-set D_p $(p = 1, \ldots, P)$ of the training set individually and each clustering solution is then projected as is on a reference mesh,
3. Run the clustering on each data-set D_p $(p = 1, \ldots, P)$ of the training set individually and then perform a fusion step to merge similar clusters and reduce the complexity of the model before projecting the solutions on a reference mesh.

The first one would be the easiest to achieve since there is no fusion step afterwards. However, it also means that each data-set has to contain exactly the same number vertices associated with intensity profiles and that it should be possible for any point to find its corresponding points in every other data-set of the training set.

When clustering the complete data, the neighbourhood matrix should be adapted too. In fact, the corresponding points in two data-sets should be considered as neighbours and the neighbours of one of the points should be neighbours of the other one as shown on Fig. 6.1. However, it is not easy to find such correspondence especially in soft tissues and each time a new data-set is added to the training set, the whole clustering process has to be run again.

The second option solves the problem of the correspondence between data-sets, allows to use meshes with varying resolution and allows that each solution can have a different and adapted number of clusters. It is also possible to add a data-set to the training set and then simply reconstruct the intensity profiles

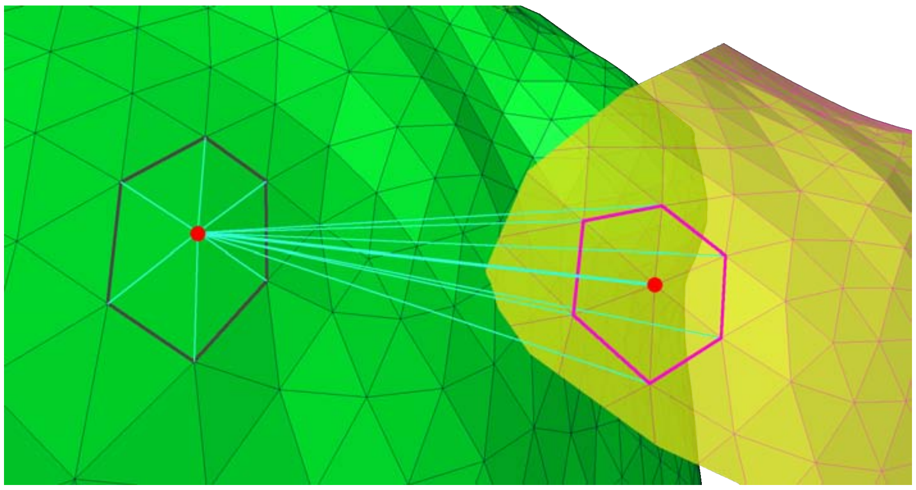


Figure 6.1: The figure shows two corresponding points (in red) on the surfaces of two different data-sets (in green and yellow). The neighbourhood relationship between the left point and its neighbours is shown by the cyan lines.

model quickly. It is very similar to the third solution and only differs from it, in that the complexity of the model sharply increases with the number of data-sets in the training set (in fact, if the training set is made of 60 data-sets and each data-set is clustered in 5 clusters, the total number of clusters in the resulting intensity profiles model is 300!).

As pointed out above, the third solution solves the problem of the complexity by performing a fusion of similar clusters. In fact, if all images in the data-set are of the same nature, there will probably be many clusters that are similar and that could therefore be merged.

The following sections addresses the *problématique* raised by this process:

- How can one measure the similarity between clusters?
- How should the merging graph be constructed such that it does not denature the clusters?
- How does one merge the clusters' means, variance matrices and membership matrices?
- How should these new values be projected on a reference mesh?

6.2 Similarity between two clusters

In order to decide whether two clusters should be merged or not, it is necessary to determine the similarity between them. In [5], F. Chung proposes to use the Jaccard Index $\mathcal{J}_{(k,l)}^{(p,q)}$ defined as the ratio of the intersection between two

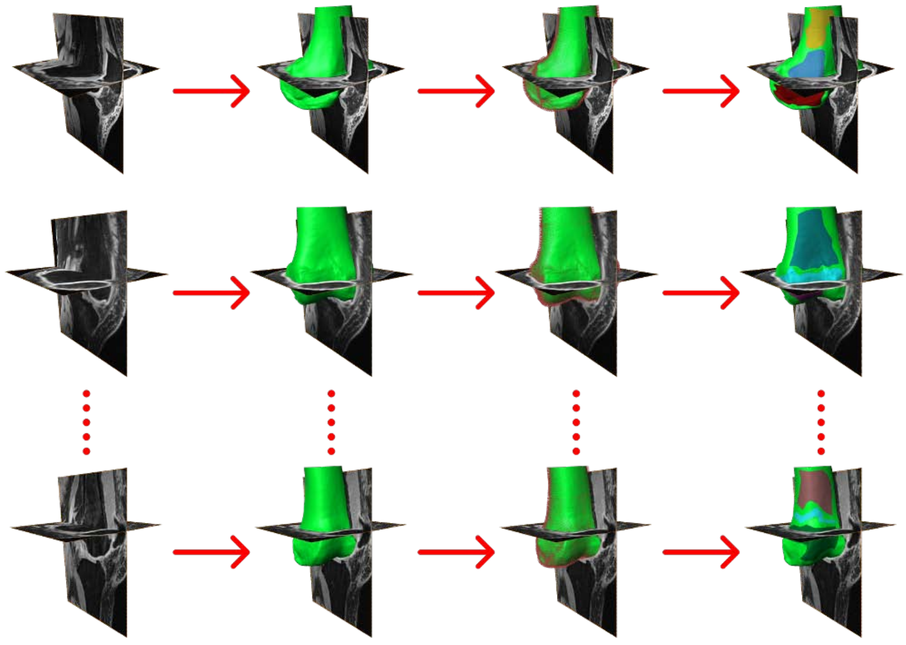


Figure 6.2: The figure shows the clustering process for three data-sets: 1st column: image volumes, 2nd column: segmentation of the structure of interest in the image volume, 3rd column: intensity profiles extraction and 4th column: NEM clustering of the intensity profiles.

clusters C_k^p and C_l^q over their union where q and p refer to data-sets from the training set and k and l to one of their clusters respectively:

$$\mathcal{J} = \frac{|C_k^p \cap C_l^q|}{|C_k^p \cup C_l^q|} \tag{6.1}$$

It is proposed to compute it using the surfaces delimited by the mean and standard deviation $\boldsymbol{\mu}_k^p \pm \boldsymbol{\sigma}_k^p$ (where $\boldsymbol{\sigma}_k^p = \left\{\sqrt{(\boldsymbol{\Sigma}_k^p)_{ii}}\right\}_{i=1,\ldots,N}$) as shown on Fig 6.3.

When both clusters are identical, $\mathcal{J}$ amounts 1.0 whereas it amounts 0.0 when they are disjoint.

6.3 Construction of the merging graph

To construct the merging graph, the Jaccard index is computed for each possible pair of clusters (C_k^p, C_l^q) for $k = 1, \ldots, K^p$, $l = 1, \ldots, K^q$ (K^i is the number of clusters associated with data-set i) and $p, q = 1, \ldots, P$. An example of the resulting table if given at Table 6.3.

The table is then thresholded in order to decide which clusters should be merged together. An example, is shown at Table. 6.3. In [5], F. Chung states that clusters from a same data-set should not be merged together. However, this is not possible since it often happens that the thresholded table contains

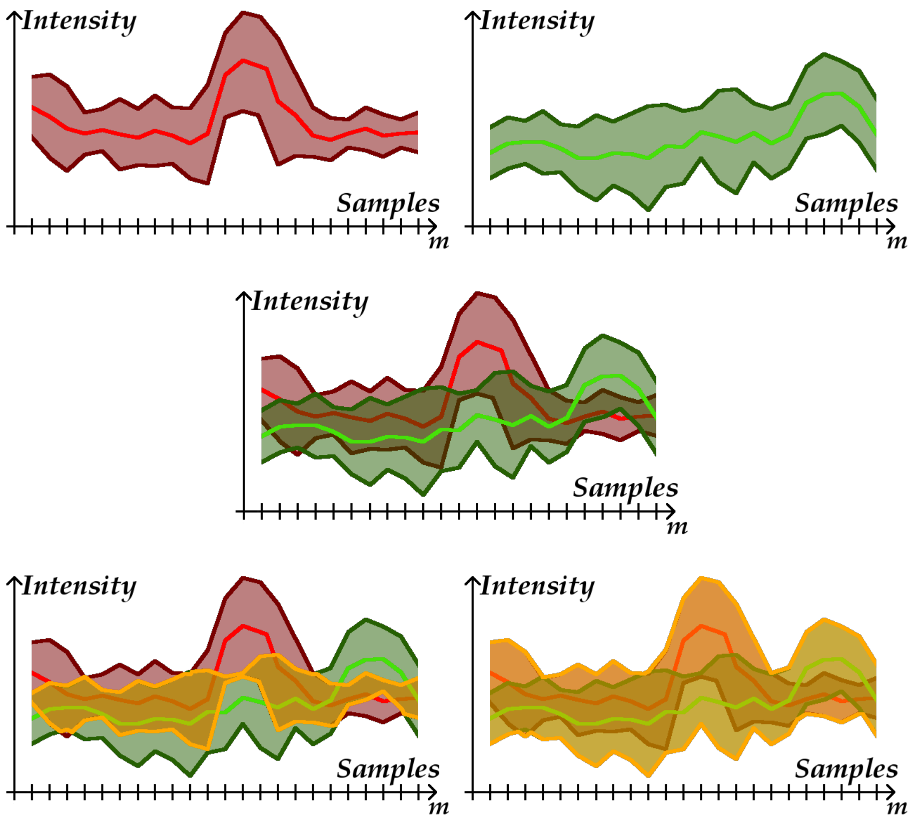


Figure 6.3: Top-Left: First cluster with mean (light red) and region spanned by standard deviation (dark red). Top-Right: Second cluster with mean (light green) and region spanned by standard deviation (dark green). Middle: Both clusters superimposed. Bottom-Left: Intersection between the two clusters. Bottom-right: Union between the two clusters.

such cases where one cluster A is to be merged with 2 other clusters B and C from another data-set. It would be then necessary to either not merge the A with one of B or C, or to create two new clusters : $A + B$ and $A + C$. Since this would not help to reduce the number of classes, it is probably not a good solution. In fact, it is also a hint that something could be wrong somewhere in the process:

Normalisation: if it has not been done properly, one data-set could have such higher profiles than the others, that these ones would all be considered as equivalent as shown on Fig. 6.4.

Model Order Selection: if the cluster count is too high, the clusters could only differ from each other by slight and possibly uninteresting differences. On the other hand, if the cluster count is too low, the clusters would have a high variance and thus an wide surface in the Jaccard index computation. It would be then possible for some pair of clusters to have wider intersection an a smaller union which could make the Jaccard index reach

		1					2			
		1	2	3	4	5	1	2	3	4
1	1	1	0.79	0.80	0.45	0.60	0.33	0.88	0.56	0.33
	2	0.79	1	0.78	0.47	0.52	0.32	0.47	0.56	0.33
	3	0.80	0.78	1	0.35	0.52	0.91	0.89	0.51	0.38
	4	0.45	0.47	0.35	1	0.37	0.20	0.18	0.33	0.16
	5	0.60	0.52	0.52	0.37	1	0.29	0.34	0.44	0.89
2	1	0.33	0.32	0.91	0.20	0.29	1	0.59	0.38	0.54
	2	0.88	0.47	0.89	0.18	0.34	0.59	1	0.59	0.61
	3	0.56	0.56	0.51	0.33	0.44	0.38	0.59	1	0.41
	4	0.33	0.33	0.38	0.16	0.89	0.54	0.61	0.41	1

Table 6.1: The table shows the Jaccard index computed for each pair of mode of two data-sets with respectively 4 and 5 classes.

the threshold.

Similarity criterion between clusters: it is of course also possible that the Jaccard index might not be the most appropriate criterion to compare clusters.

		1					2			
		1	2	3	4	5	1	2	3	4
1	1	1	0	1	0	0	0	1	0	0
	2	0	1	0	0	0	0	0	0	0
	3	1	0	1	0	0	1	1	0	0
	4	0	0	0	1	0	0	0	0	0
	5	0	0	0	0	1	0	0	0	1
2	1	0	0	1	0	0	1	0	0	0
	2	1	0	1	0	0	0	1	0	0
	3	0	0	0	0	0	0	0	1	0
	4	0	0	0	0	1	0	0	0	1

Table 6.2: The table shows the previous table thresholded: $\mathcal{J} > 0.8$.

The table 6.3 can also be represented as a graph (Fig. 6.5) where it is easier to see which clusters will be merged together and which ones won't. An important issue is the choice of how to handle clusters that are indirectly connected like shown with clusters $(1, 1)$ and $(2, 1)$. In fact, merging them all at once could lead to a cluster with a very high variance which is not desirable in our case.

There are thus two options:

1. Merge all the connected clusters together as shown left on Fig. 6.6.
2. Create two new clusters where only directly connected clusters are merged in, as shown right on Fig. 6.6.

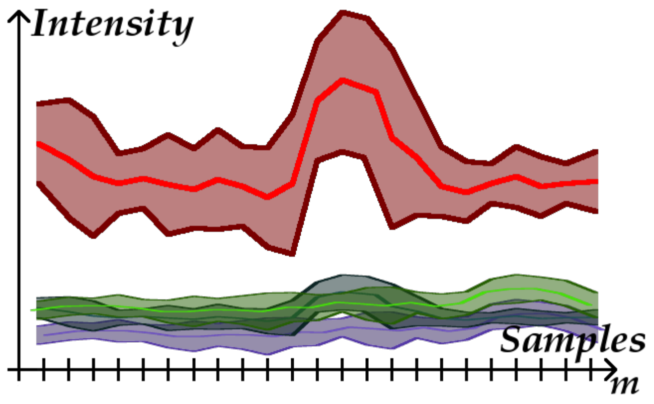


Figure 6.4: The figure shows clusters where the normalisation was not successful. Except the red cluster, all others have been too flattened and all look alike. These clusters will erroneously be merged together.

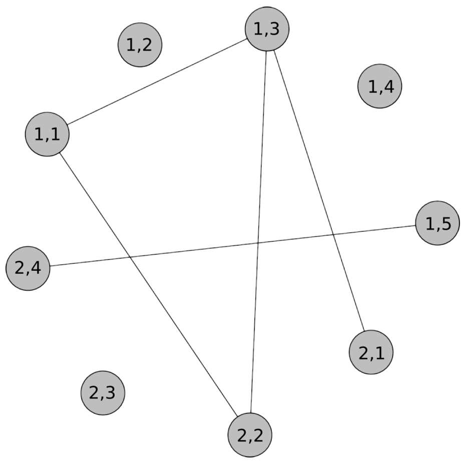


Figure 6.5: The figure shows the merging graph with links between the classes to be merged.

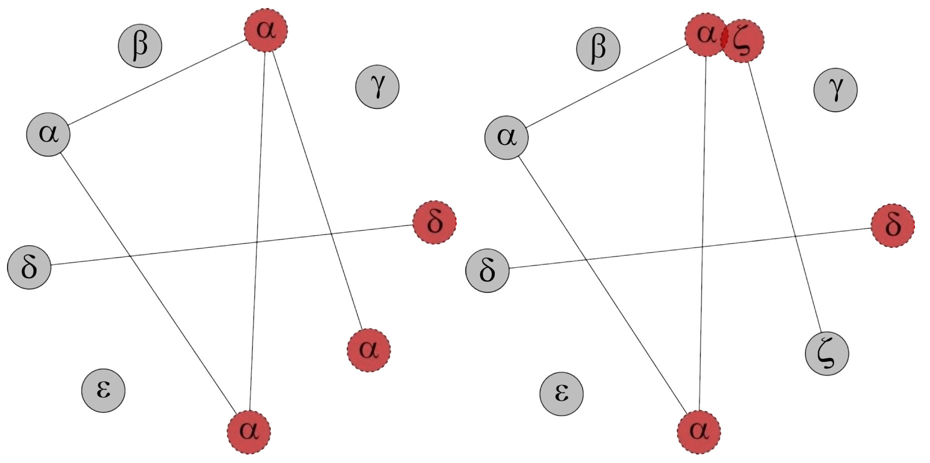


Figure 6.6: The figures show the merging graph after merging. In the left one, the chosen strategy was to merge all clusters together in a new cluster α even if they are only indirectly connected (for example $(1, 1)$ and $(2, 1)$). In the right one, the two new clusters α and ζ are the result of the fusion of the directly connected clusters in the graph.

6.4 Fusion of clusters

Once the decision has been taken to merge some clusters, the model has to be built. The clusters that are not merged with another one can simply be included in the model without any further processing. For the others, it is necessary to recompute their means ${}^{\mu}\boldsymbol{\mu}_{\kappa}$ and variance matrices ${}^{\mu}\boldsymbol{\Sigma}_{\kappa}$ ($\kappa = 1, \ldots, R$ is the index of the new cluster). F. Chung proposes to do it like that:

$$ {}^{\mu}\boldsymbol{\mu}_{\kappa} = \frac{\sum\limits_{p=1}^{P} \sum\limits_{\forall k \in [1;K^p] | \eta(p,k)=\kappa} \sum\limits_{i=1}^{M} c_{ik}^{p} \boldsymbol{\mu}_{k}^{p}}{\sum\limits_{p=1}^{P} \sum\limits_{\forall k \in [1;K^p] | \eta(p,k)=\kappa} \sum\limits_{i=1}^{M} c_{ik}^{p}} \tag{6.2} $$

$$ {}^{\mu}\boldsymbol{\Sigma}_{\kappa} = \frac{\sum\limits_{p=1}^{P} \sum\limits_{\forall k \in [1;K^p] | \eta(p,k)=\kappa} \left(\sum\limits_{i=1}^{M} c_{ik}^{p} (\boldsymbol{x}_{i}^{p} - \boldsymbol{\mu}_{k}^{p})(\boldsymbol{x}_{i}^{p} - \boldsymbol{\mu}_{k}^{p})^{T} \right)}{\sum\limits_{p=1}^{P} \sum\limits_{\forall k \in [1;K^p] | \eta(p,k)=\kappa} \sum\limits_{i=1}^{M} c_{ik}^{p}} \tag{6.3} $$

Along with these new cluster parameters comes an equivalence table $\eta(p, k)$ that indicates to which new cluster κ the original cluster k of data-set p belongs.

6.5 Projection of the posterior probabilities onto the reference mesh

The last step consists in projecting the cluster memberships onto a reference mesh $\mathcal{M}^*$ of M^* vertices. To achieve this, first a reference mesh is defined and each original mesh $\mathcal{M}^p$ is registered to it and then the cluster memberships of each mesh are projected and re-sampled on the reference mesh using a closest point approach. To ensure that every clustering information of the separate data-sets are correctly transferred to $\mathcal{M}^*$ it is necessary that $M^* >= \max_p(M^p)$ (M^p is the number of vertices of mesh $\mathcal{M}^p$).

Finally, the posterior probabilities $\boldsymbol{\gamma}_i$ are normalised for each vertex $\boldsymbol{x}_i$ of the reference mesh:

$$ \gamma_{i\kappa} = \frac{\sum\limits_{p=1}^{P} \sum\limits_{\forall k \in [1;K^p] | \eta(p,k)=\kappa} c_{ik}}{\sum\limits_{k=1}^{R} \gamma_{ik}} \tag{6.4} $$

for each $i = 1, ..., M^*$.

Part III

Results

Chapter 7

NEM Algorithm Initialisation

Three methods were investigated to initialise the NEM algorithm:

1. Fuzzy C-Means
2. Spatially Aware Fuzzy C-Means
3. Hierarchical Clustering with Ward's Criterion

The first two were tested by exporting the data from Amira® to the R environment (`http://www.r-project.org/`) to avoid implementing the methods and then importing the results back into Amira®.

To enable the use of the Fuzzy C-Means (FCM) algorithm, the library `e1071` available on was installed `http://cran.r-project.org/web/packages/e1071/index.html` using the simple command `install.packages('e1071')` in the R-console.

The package provides a method called `cmeans` which is really simple to use as it possible to start with only two parameters : the data to cluster and the desired number of clusters (the default distance is the euclidean distance and the fuzzification parameter is set to 2). It returns an object which contains among others: the cluster centres, a membership matrix and a vector yielding a hard partitioning of the data (the cluster with the highest membership coefficient is chosen for each point). The cluster centres and membership matrix can then be used to initialise the EM algorithm. The initialisation of the NEM algorithm with these informations was however not implemented because of lack of time. The hard partitioning returned by the R algorithm was only useful in that it was thus possible to compare visually this result with the Hierarchical Clustering which was implemented in Amira®.

The two first methods only differ in that each row of the data matrix of the latter contains an extra three values which are the spatial coordinates of the associated point. In order to wheighten the contribution of these and the intensity values of the profile, a parameter α is used (as explained in Section 5.3).

If α yields 0, the second method is the same as the first one and, as it increases, the distance between two data samples comes down to a simple distance between the two points. An appropriate value as thus to be chosen in order to spatially smooth the clustering without denaturating it.

The algorithm was carried out several times with different values of α. The code can be found in Appendix A.

A hierarchical clustering algorithm was implemented in Amira® and used the Ward's criterion and the euclidean distance. This is the algorithm used for the rest of the present work.

The various solutions were compared visually as shown on Figures 7.1, 7.2, 7.3, 7.4 and 7.5.

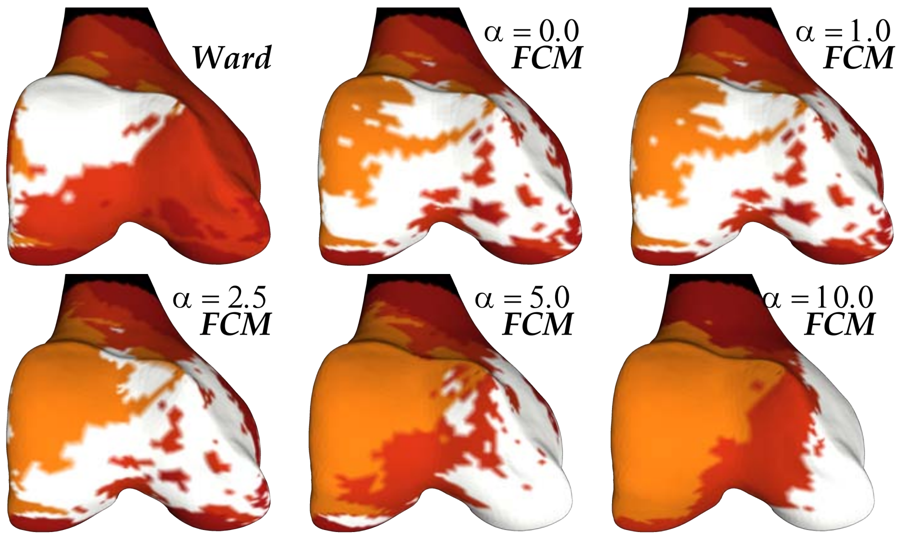


Figure 7.1: 6 different initialisations displayed on an anterior view of the distal head of the femur. The colours represent the 4 clusters (black indicates that the point was not used in the clustering) and the colour of the cluster which has the highest cluster membership was assigned to each vertex respectively.

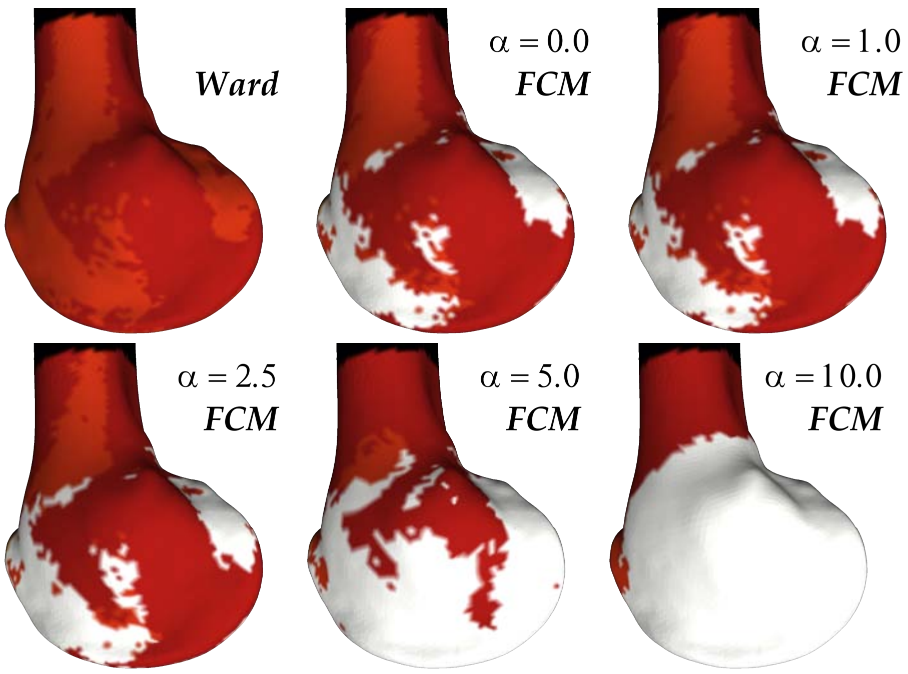


Figure 7.2: 6 different initialisations displayed on a lateral view of the distal head of the femur. The colours represent the 4 clusters (black indicates that the point was not used in the clustering) and the colour of the cluster which has the highest cluster membership was assigned to each vertex respectively.

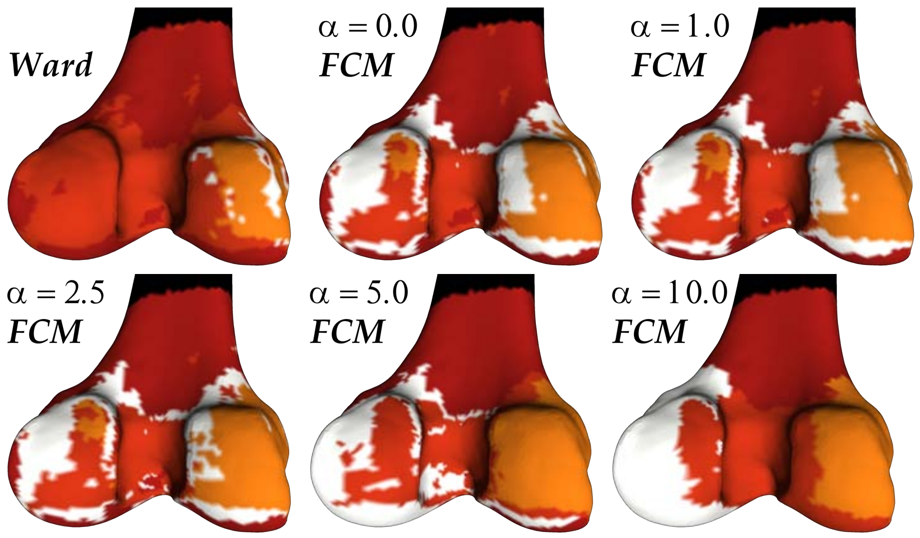


Figure 7.3: 6 different initialisations displayed on a posterior view of the distal head of the femur. The colours represent the 4 clusters (black indicates that the point was not used in the clustering) and the colour of the cluster which has the highest cluster membership was assigned to each vertex respectively.

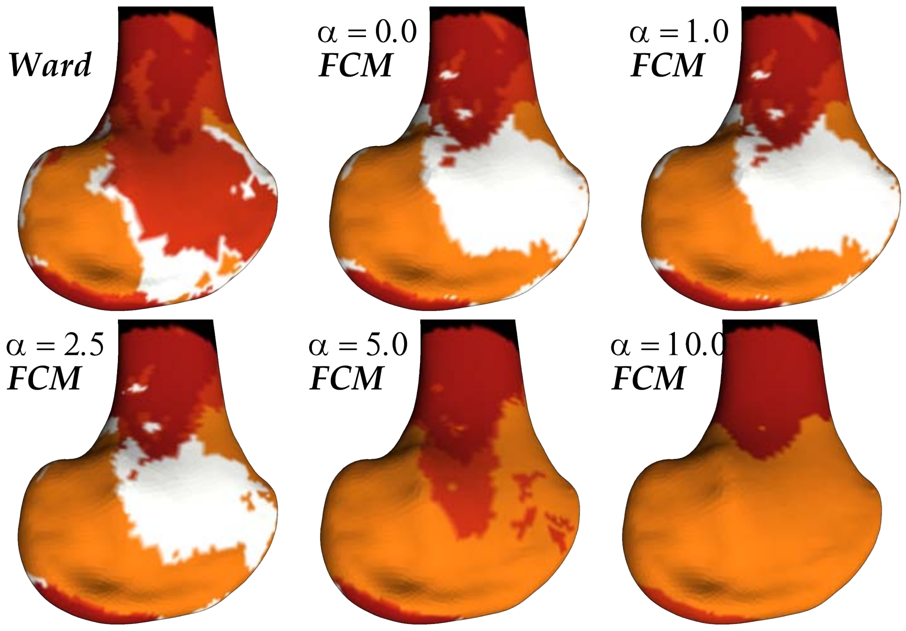


Figure 7.4: 6 different initialisations displayed on a medial view of the distal head of the femur. The colours represent the 4 clusters (black indicates that the point was not used in the clustering) and the colour of the cluster which has the highest cluster membership was assigned to each vertex respectively.

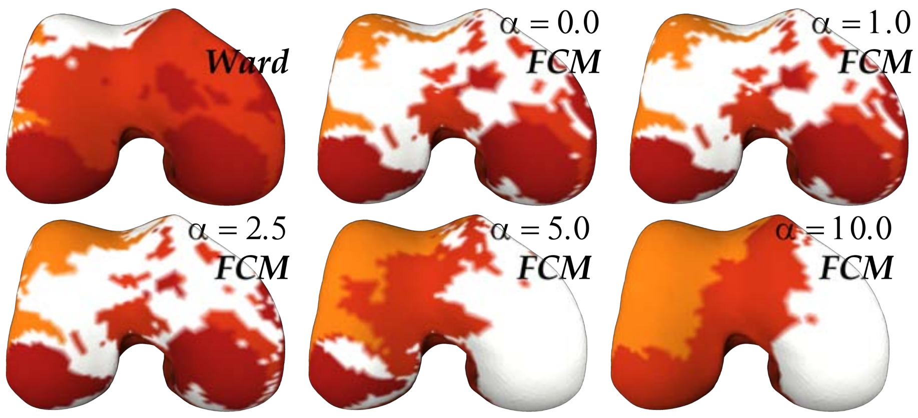


Figure 7.5: 6 different initialisations displayed on an inferior view of the distal head of the femur. The colours represent the 4 clusters (black indicates that the point was not used in the clustering) and the colour of the cluster which has the highest cluster membership was assigned to each vertex respectively.

As it can be observed, the biggest difference between the various initialisations with the Fuzzy C-Means algorithm is between the instances with $\alpha = 2.5$ and $\alpha = 5.0$ suggesting that the optimal value of α that balances spatial smoothing and clustering denaturation can be found between those two values.

The instances of the algorithm with $\alpha = 5.0$ and $\alpha = 10$ are clearly uninteresting since it can be seen that the weight given to the spatial data is too important which results in a clustering of the position of the vertices only.

It can be noticed that in such cases (especially on Figures 7.3, 7.1 and 7.4), the spatial smoothing has created clusters that are almost contiguous. This is due to the metric which was used to compute the distance between data samples (intensity values **and** position coordinates). The result could probably have been improved if this metric was different. A short list of possible improvements can be found below:

- Use another distance measure, for example the geodesic distance which takes the shape of the surface into account to measure a distance between two points,
- Use a relative distance instead of an absolute distance. This way, profiles that are at the same distance from a reference point are considered as close to each other even though they might not be close to each other considering a regular distance,
- Normalise intensity values and position coordinates (and preferably separately) to allow an easier weighting,
- Use different distance measures for the intensity values and the position coordinates.

Since these algorithms were not implemented in Amira$^{®}$, it was not possible to evaluate the benefits of using the initialisation based on the Fuzzy C-Means algorithm (with and without spatial regularisation) instead of the Hierarchical Clustering using Ward's criterion.

However, an obvious advantage of the FCM is that it returns a fuzzy clustering of the data and that the (N)EM algorithm is based on such a clustering. Moreover, the spatial regularisation seems to be a good approach since it fuzzifies the clustering thus spatially expanding the clusters wiping out some noise (see Fig. 7.6). Sadly it doesn't always work as desired and sometimes the clusters are only fuzzified without being spatially expanded (see Fig. 7.7).

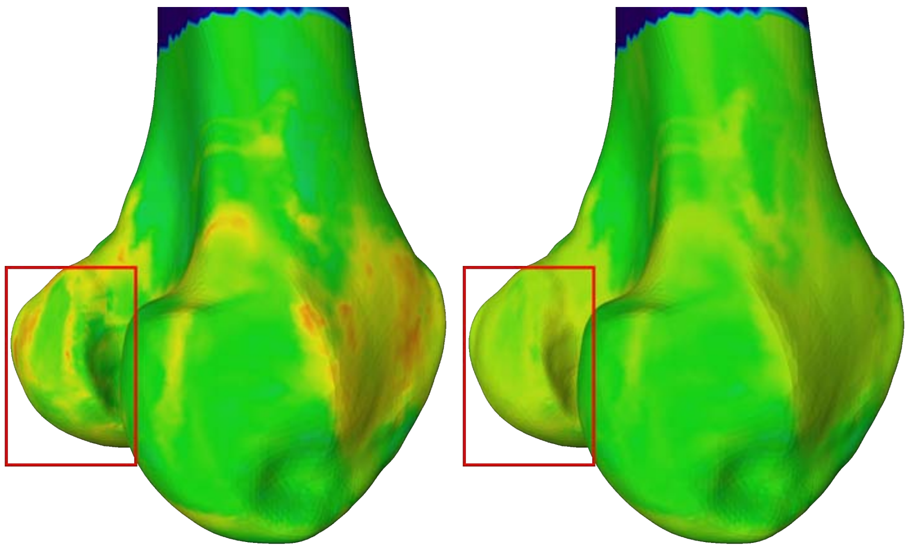


Figure 7.6: Fuzzy representation of the second cluster (red) of the same clustering as in Figures 7.1, 7.2, 7.3, 7.4 and 7.5. Blue is assigned to "no cluster", green to a membership value of 0.0, yellow to ≈ 0.5 and red to 1.0. The left figure shows the membership values of the first cluster with $\alpha = 0.0$ and the right one with $\alpha = 2.5$. It can be noticed in the red frame, that the green zone on the condyle in the left figure becomes yellow in the right thanks to the fuzzification.

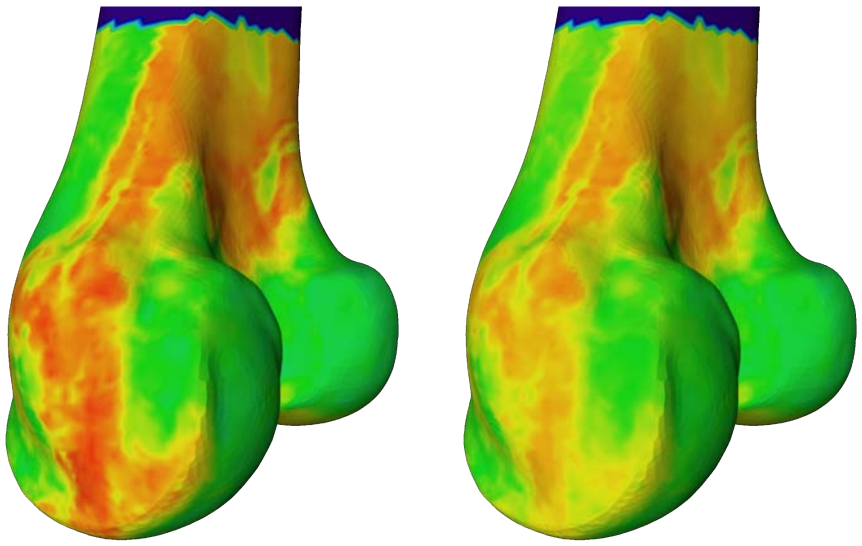


Figure 7.7: Fuzzy representation of the third cluster (white) of the same clustering as in Figures 7.1, 7.2, 7.3, 7.4 and 7.5. Blue is assigned to "no cluster", green to a membership value of 0.0, yellow to ≈ 0.5 and red to 1.0. The left figure shows the membership values of the first cluster with $\alpha = 0.0$ and the right one with $\alpha = 2.5$. It can be noticed that this time, there is apparently no green gap filled with yellow thus no wiping-out of noise but only a fuzzification of the cluster which doesn't bring anything.

Chapter 8

Spatial Regularisation Parameter Evaluation

Apart from the initialisation of the NEM algorithm, another parameter that has to be determined is the spatial regularisation coefficient β. In [5], F. Chung states that it can be observed that, as β increases, the likelihood of the clustering significantly drops when the weight given to spatial smoothing becomes to high.

It was in fact possible to see this drop and a heuristic algorithm was developed. As each point in the two-dimensional space "β-likelihood" represents the result of one run of the NEM algorithm which takes about 45 seconds to perform, it is not possible to finely sample the beta-space. This disables to simply compute the derivative of the curve.

Also, as there is no knowledge of how big the sought gap in the likelihood between two β, it is not possible to start from the the left ($\beta = 0$) and scan the likelihood values till the gap is found.

The proposed solution is to define an initial range where a coarse sampling of β will be performed. If the range is big enough, it will be possible to define a representative range for the likelihood values.

Then the angle formed by three successive points in the β-likelihood space is computed for the whole range. Therefore, the values of β and likelihood are first normalised with respect to their, now known, boundaries. Then two vectors are defined: v_1 describes joins the first point and the second one and the other, v_2, joins the second and third points. The angle between v_1 and v_2 is computed relatively to v_1 as shown on Fig. 8.1.

This enables to make two interesting measurements:

1. The likelihood gap between two successive samples,
2. The angle between three successive points.

If the gap is big enough (i.e. greater than a certain threshold for example $\tau = 0.2 * likelihoodRange$) and the angular value too (for example $> 60°$), the sought gap can be considered as found.

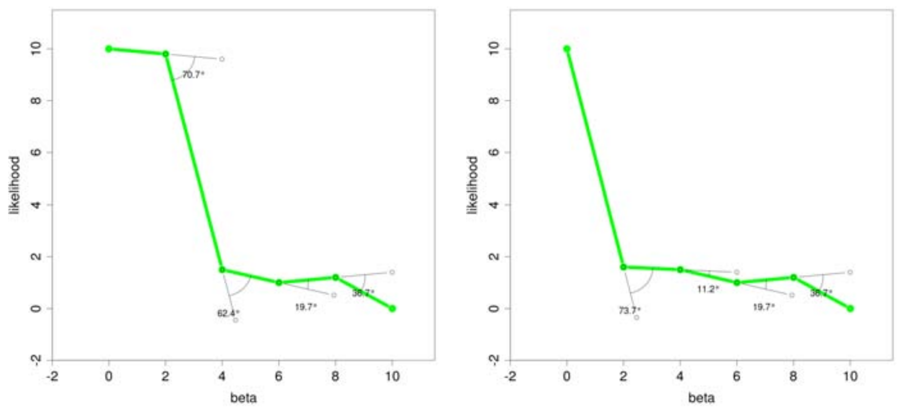


Figure 8.1: The figure shows two different sets of 6 samples of the likelihood for $\beta = 0, \ldots, 10$. Starting at the third point, the angle between this point and its 2 predecessors is computed. The angle between the two vectors defined by these 3 points is shown in dark.

However the simple case illustrated on the figure above is not the most common. More often, the first samples show values of likelihood that seem aligned and it is thus not possible to find the gap so easily. This is shown on Fig. 8.2

That is the reason why an iterative process takes place where the beta range will be re-sampled more finely. In order to avoid to re-sample the whole range (which is time-consuming), the algorithm searches two values of beta between which the gap might be located.

The first β value can be found when there is whether a rather big gap (i.e. $|\text{secondPointLikelihood} - \text{firstPointLikelihood}| \geq 0.1 * \text{likelihoodRange}$) or a big angular value ($\theta > 40°$) between the three current points.

- if the gap was big enough when the first beta was found but the next gap is too small, then the second β value is the one of the next sample since the assumption is made that the likelihood curve stabilises after a gap,
- if the angle was big enough but not the gap and a big enough gap is found in the next step then the second beta value is found since the assumption is made that the gap started just after the former big angle.

Once the new boundaries for β have been found, the beta increment is refined and the NEM algorithm is run for these new betas as shown on Fig. 8.3.

After a few iterations, the likelihood-gap between successive values of β will be too small to ever fulfil the conditions and the optimal β would only be refined due to the conditions on the angle. This is why a limited number of iterations as been set.

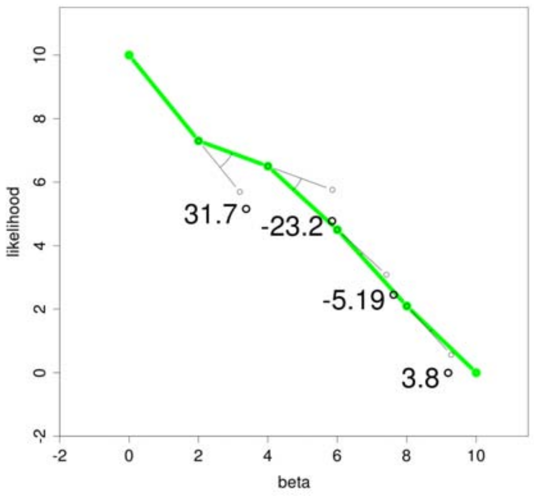


Figure 8.2: The figure shows a set of 6 samples of the likelihood for $\beta = 0, \ldots, 10$ where the points are almost aligned and no clear gap can be seen.

Moreover, the loose conditions constrain the search for the optimal β in a range of low values which is positive. In fact, even if a gap that satisfies all conditions exists for a higher value of β, the quality of the clustering would probably be poor since it would mainly depend on whether points are neighbours or not.

Plots of the results of this algorithm applied on some data-sets from the training-set can be found below. The commented C++ source of the algorithm can be found in Appendix B.
As it can be observed, most of the time, the algorithm succeeds in finding the appropriate value of β. When it fails, the optimal β is always greater than the chosen one. This desirable to avoid to denature the clustering.

The likelihood curve is sometimes severely non-monotonic and I wasn't yet able to find the reasons of such a behaviour (for example in Fig. 8.4 for cluster count = 3 and 5).

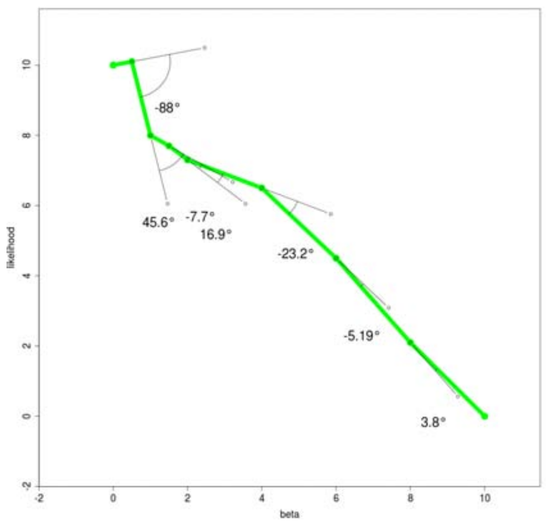


Figure 8.3: The figure shows the same plot as above but with a range of β values that was re-sampled more finely. The gap is thus visible.

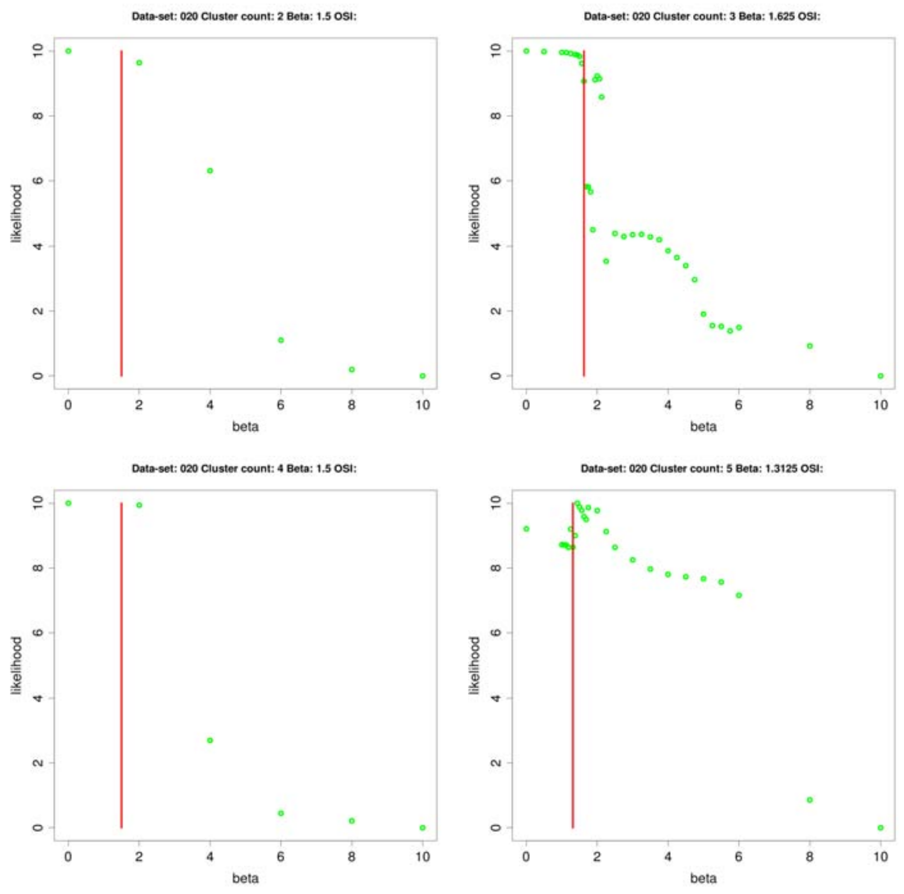
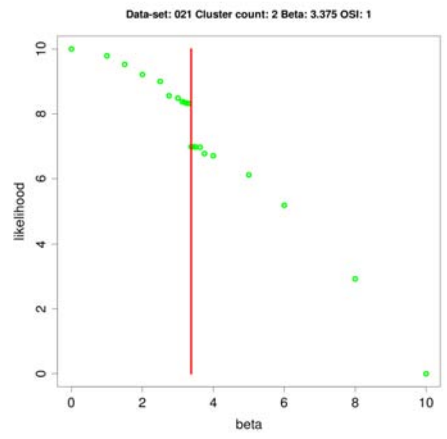
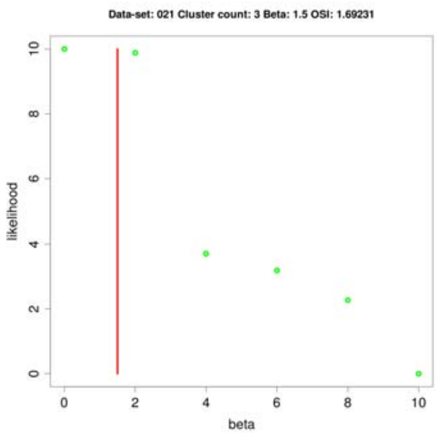
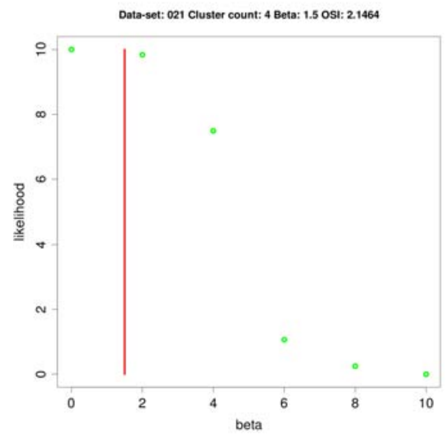
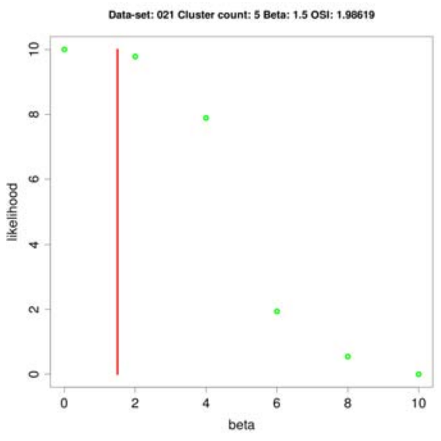


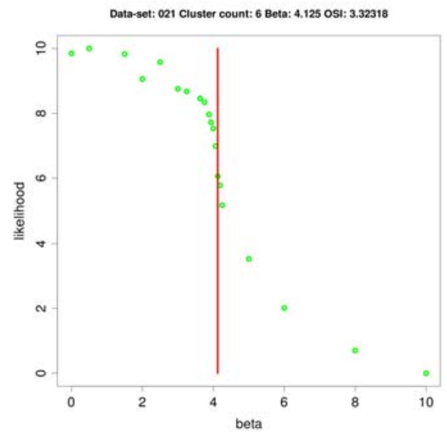
Figure 8.4: The figure shows the β samples and their likelihood values necessary to determine the optimal β (red line) for different number of clusters (2-5) in the 20th data-set of the training set. The range of the likelihood values was rescaled to $[0; 10]$.

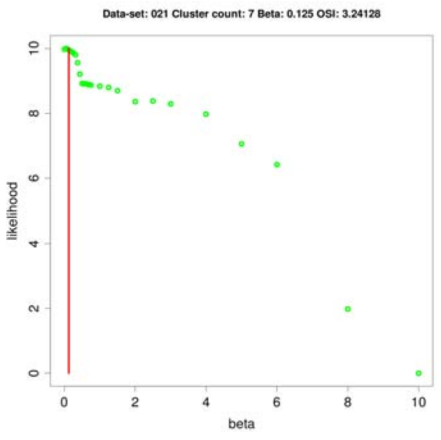

Data-set: 021 Cluster count: 2 Beta: 3.375 OSI: 1
likelihood
beta

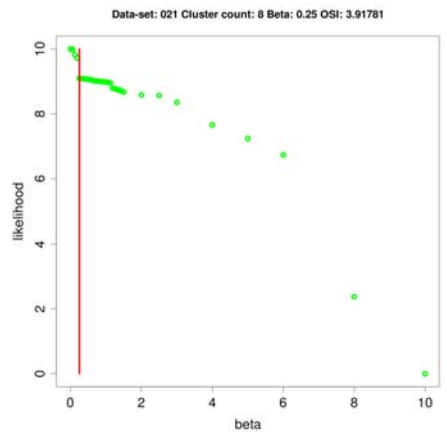

Data-set: 021 Cluster count: 3 Beta: 1.5 OSI: 1.69231
likelihood
beta

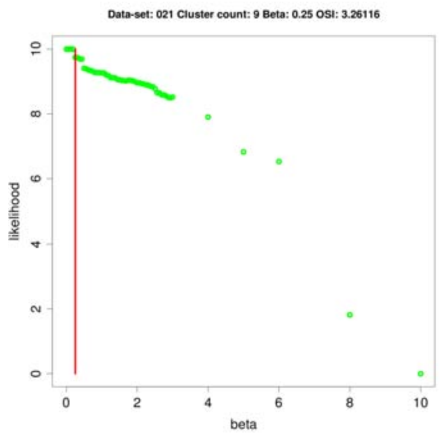

Data-set: 021 Cluster count: 4 Beta: 1.5 OSI: 2.1464
likelihood
beta

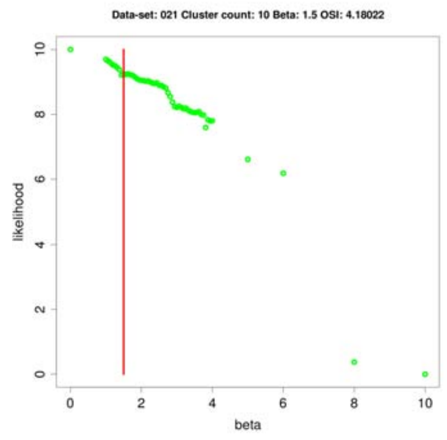

Data-set: 021 Cluster count: 5 Beta: 1.5 OSI: 1.98619
likelihood
beta

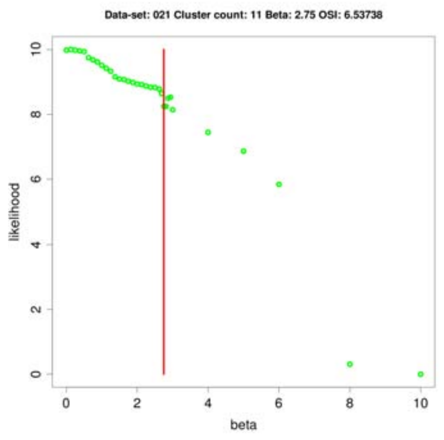

Data-set: 021 Cluster count: 6 Beta: 4.125 OSI: 3.32318
likelihood
beta

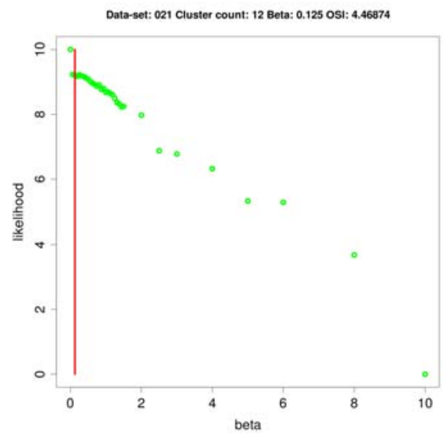

Data-set: 021 Cluster count: 7 Beta: 0.125 OSI: 3.24128
likelihood
beta

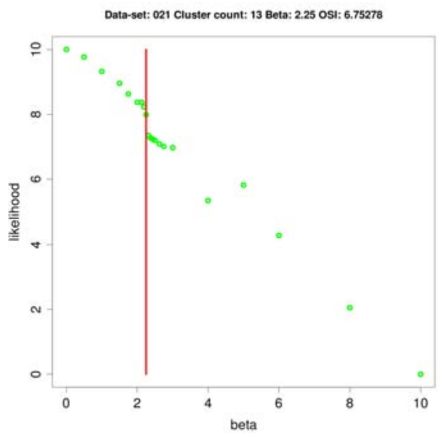

Data-set: 021 Cluster count: 8 Beta: 0.25 OSI: 3.91781
likelihood
beta


Data-set: 021 Cluster count: 9 Beta: 0.25 OSI: 3.26116
likelihood
beta


Data-set: 021 Cluster count: 10 Beta: 1.5 OSI: 4.18022
likelihood
beta


Data-set: 021 Cluster count: 11 Beta: 2.75 OSI: 6.53738
likelihood
beta


Data-set: 021 Cluster count: 12 Beta: 0.125 OSI: 4.46874
likelihood
beta


Data-set: 021 Cluster count: 13 Beta: 2.25 OSI: 6.75278
likelihood
beta

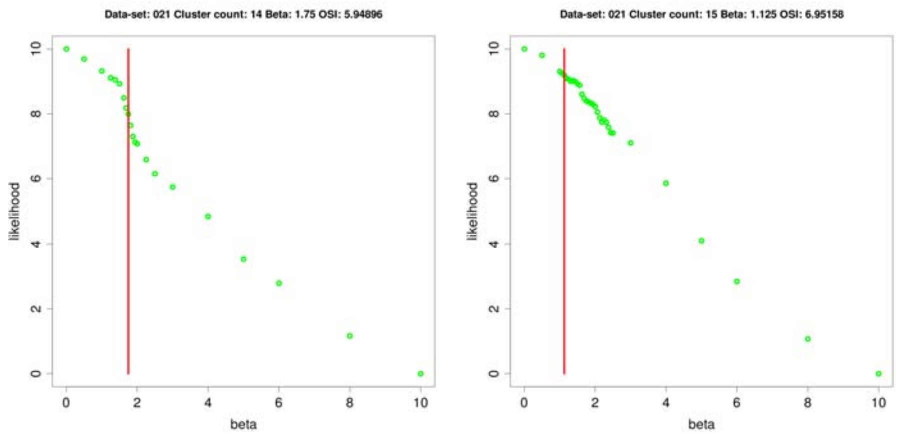


Figure 8.5: The figure shows the β samples and their likelihood values necessary to determine the optimal β (red line) for different number of clusters (2-15) in the 21th data-set of the training set. The range of the likelihood values was rescaled to $[0; 10]$.

Chapter 9

Selection of the Number of Clusters

The Overlap Separation Index proposed by F. Chung in [5] and described in Section 5.4 was implemented and tested. In the paper it is stated that the algorithm has to be performed with an increasing number of clusters and then the solution with the highest OSI has to be chosen.

Thanks to its nature, the criterion should ensure that the optimal number of clusters would not always be the highest that was tested. However, looking at the curve described by the OSI when the cluster count rises, it can be seen that the situation is not that simple as shown on Fig. 9.1.

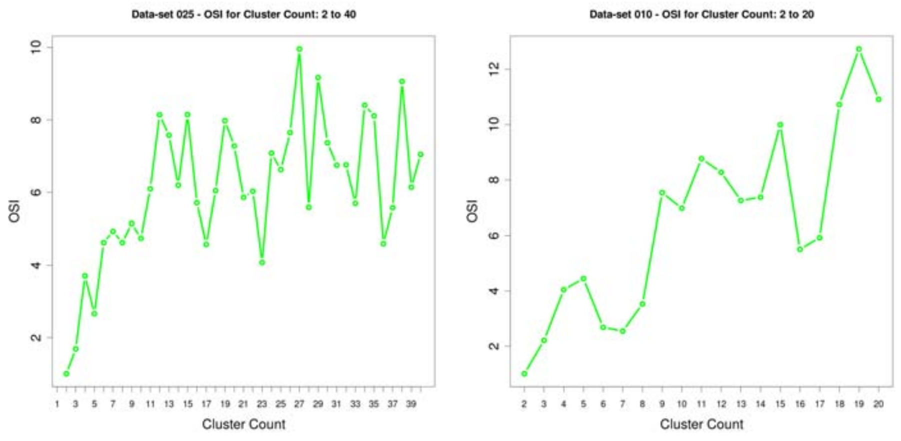


Figure 9.1: The figure shows the OSI plotted against the cluster count for the 25th data-set (left) and for the 10th data-set (right). The curves are strongly non-monotone and there is no sign of an asymptote.

One possible explanation for this behaviour is that the OSI is not suited for use with the posterior probabilities that the NEM algorithm outputs.

In fact, as it can be observed on Fig. 9.2, the posterior probabilities of all clusters are almost crisp, i.e. each intensity profile has a probability to belong to one of the clusters ≈ 1.0. Only a small surrounding area (1 or 2 vertices) of each cluster has a cluster membership that is not equal to 1.0.

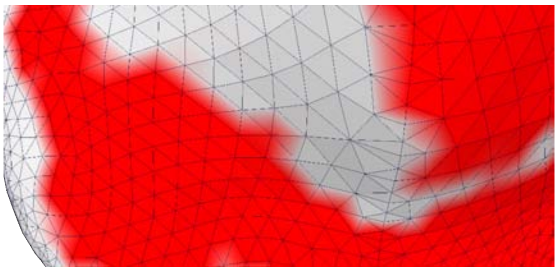


Figure 9.2: The figure shows a zoom on the surface of a mesh where the cluster membership values of a particular cluster were drawn for each vertex (white to red = 0.0 to 1.0). Only a small area around the cluster is plotted in pink where the cluster membership value amounts 0.5.

It is then obvious that the OSI index is not suited at all. In fact, it is computed as the ratio of the amount of overlap between two clusters C_1 over the minimum separation between all clusters C_2.

In Eq. 5.14, most of the vertices have a $c_{i\beta_i} \approx 0.0$ (β_i is the cluster to which vertex i has the second highest probability to belong to) and thus $C_1 \to 0$.

In the first sum of S_{kl} in Eq. 5.15, i only refers to vertices whose **highest** cluster membership is for cluster k, so $c_{ik} \approx 1.0$ and thus $c_{il} \approx 0.0$ and the first sum is thus ≈ 0.0.

In the second sum of the same equation, j refers to vertices whose **second highest** cluster membership corresponds to cluster l. Therefore, some terms in the sum could be not null if for those vertices $c_{ik} \approx 1.0$ but as C_2 (Eq. 5.16) is the **minimum** of all S_{kl}, it will probably also tend to 0.0.

The value of the OSI thus probably depends on the number of vertices whose intensity profile has been assigned in each cluster.

However, this doesn't mean that the OSI is not interesting at all. It shouldn't be used on such crisp clustering like the results of the EM algorithm. A possible solution would be to use the cluster means and variances to recompute distances from each profile to the clusters (for example the Mahalanobis distance, see Fig. 9.3) thus restoring a fuzzy classification on which the OSI could be used.

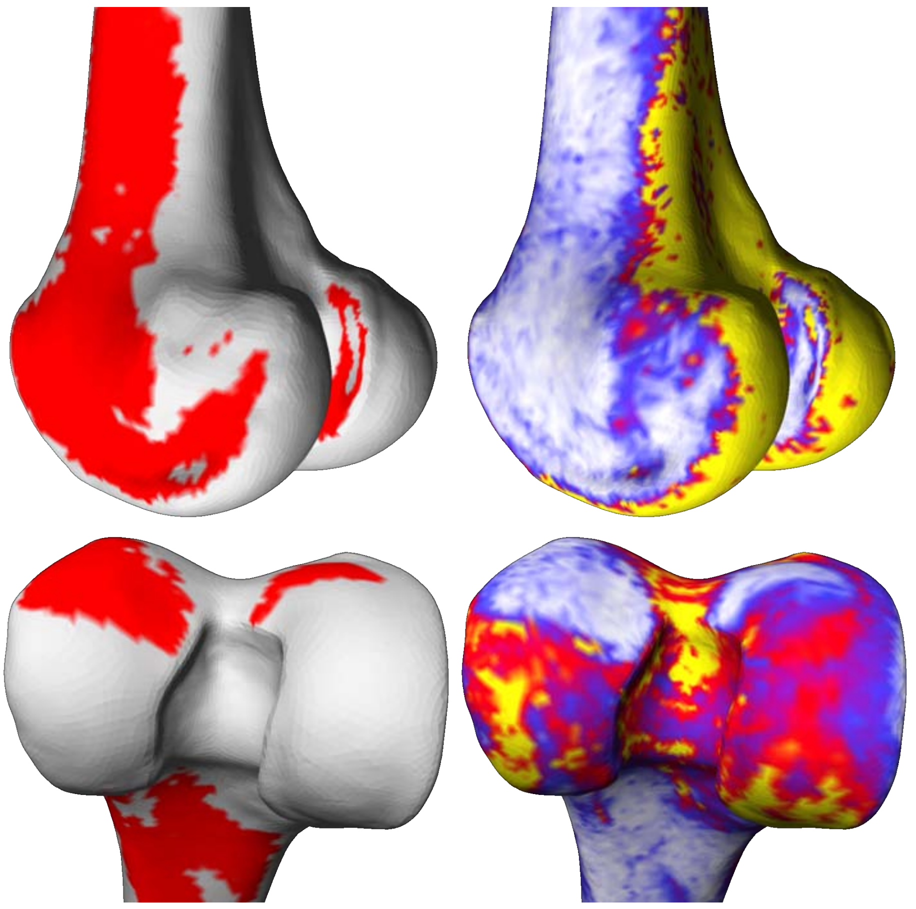


Figure 9.3: The figure shows two different views of the distal femur. Top-left: the cluster membership values of one cluster are shown on each vertex of the mesh (white to red = 0.0 to 1.0). Top-right: each vertex is attributed a colour that reflects its Mahalanobis distance to the cluster (white is closest, red middle and yellow far. The values are not given because they were not normalised). Bottom: Same as above but with another cluster.

Chapter 10

Comparison of two clusters

To compare two clusters and decide whether they should be merged or not, F. Chung proposes to use the Jaccard index as described in Section 6.2.

Three clustering solution were compared with 4, 5 and 7 clusters respectively and the results of the Jaccard index are visible in Table 10.1.

A few representative plots (among the 120 possible plots) of the clusters' mean profiles with their respective standard deviations are shown below in Fig. 10.1 with the associated Jaccard Index value.

Although this distance measure is not meaningless, some clusters who could be considered as equivalent are not. For example, in Fig. 10.1, the bottom-left plot shows two clusters who could be considered as equivalent.

The opposite situation has also been seen. Due to the use of the standard deviation to delimit the surface around the mean, the comparison of two clusters doesn't take differences between the means themselves very much into account. For example, in the next chapter on Fig. 11.1, although the surfaces are similar, the means are similar and the question arises: are these clusters really equivalent and should they be merged?

It would maybe be possible to compare the derivative of the profiles instead of the profiles themselves or even, after adaptation, use other techniques to compare the profiles like the ones that are used to align protein sequences [13].

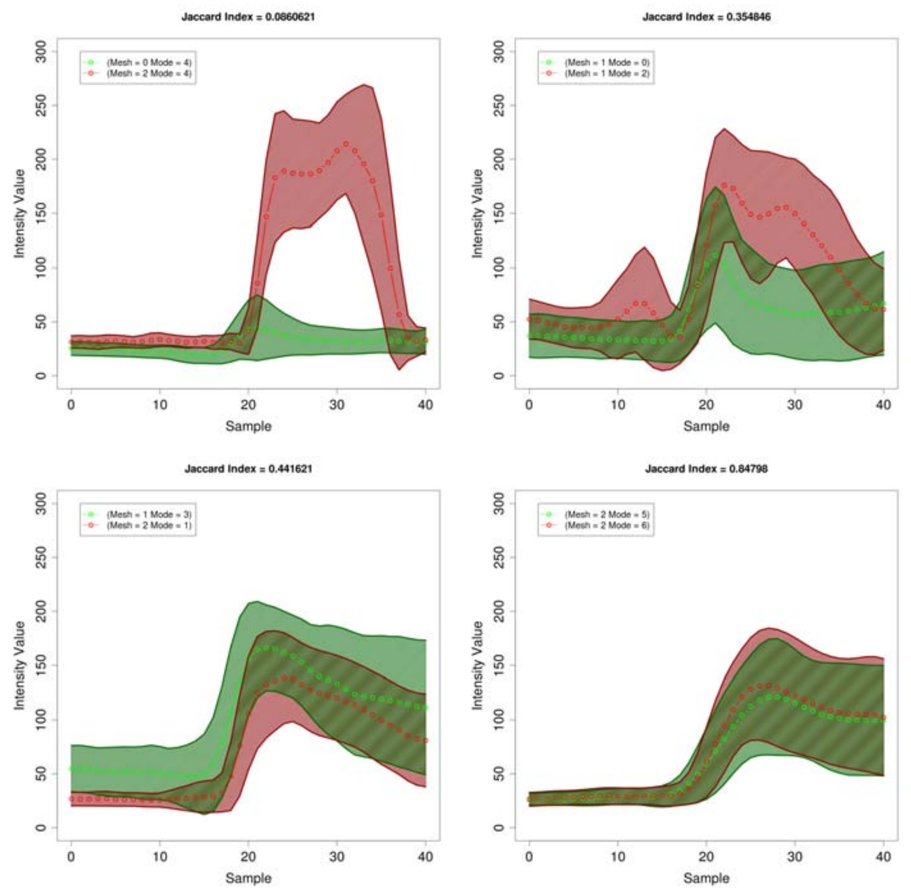


Figure 10.1: The figure shows 4 different comparisons of respectively 2 clusters. The plots are ordered by increasing Jaccard Index value from top-left to bottom-right. The two plots on the right show opposite situations for clusters in the same clustering solution. In the upper-one, the two profiles are really different whereas in the lower-one they are almost identical.

		0					1				2						
		0	1	2	3	4	0	1	2	3	0	1	2	3	4	5	6
0	0	1	0.35	0.59	0.60	0.05	0.26	0.53	0.53	0.52	0.66	0.53	0.16	0.35	0.47	0.47	0.54
	1	0.35	1	0.55	0.46	0.17	0.58	0.40	0.39	0.27	0.47	0.58	0.50	0.22	0.20	0.49	0.46
	2	0.59	0.55	1	0.69	0.08	0.43	0.59	0.54	0.47	0.64	0.69	0.32	0.41	0.32	0.72	0.75
	3	0.60	0.46	0.69	1	0.08	0.39	0.56	0.54	0.58	0.50	0.66	0.21	0.44	0.28	0.58	0.66
	4	0.05	0.17	0.08	0.08	1	0.27	0.05	0.07	0.01	0.09	0.09	0.34	0.14	0.08	0.11	0.10
1	0	0.26	0.58	0.43	0.39	0.27	1	0.37	0.35	0.25	0.32	0.38	0.60	0.21	0.16	0.38	0.33
	1	0.53	0.40	0.59	0.56	0.05	0.37	1	0.67	0.70	0.61	0.50	0.19	0.27	0.28	0.43	0.46
	2	0.53	0.39	0.54	0.54	0.07	0.35	0.67	1	0.63	0.53	0.50	0.17	0.24	0.33	0.38	0.42
	3	0.52	0.27	0.47	0.58	0.01	0.25	0.70	0.63	1	0.44	0.44	0.09	0.30	0.25	0.36	0.42
2	0	0.66	0.47	0.64	0.50	0.09	0.32	0.61	0.53	0.44	1	0.58	0.27	0.31	0.38	0.54	0.58
	1	0.53	0.58	0.69	0.66	0.09	0.38	0.50	0.50	0.44	0.58	1	0.27	0.33	0.24	0.63	0.65
	2	0.16	0.50	0.32	0.21	0.34	0.60	0.19	0.17	0.09	0.27	0.27	1	0.23	0.11	0.38	0.31
	3	0.35	0.22	0.41	0.44	0.14	0.21	0.27	0.24	0.30	0.31	0.33	0.23	1	0.17	0.53	0.52
	4	0.47	0.20	0.32	0.28	0.08	0.16	0.28	0.33	0.25	0.38	0.24	0.11	0.17	1	0.23	0.27
	5	0.47	0.49	0.72	0.58	0.11	0.38	0.43	0.38	0.36	0.54	0.63	0.38	0.53	0.23	1	0.84
	6	0.54	0.46	0.75	0.66	0.10	0.33	0.46	0.42	0.42	0.58	0.65	0.31	0.52	0.27	0.84	1

Table 10.1: The table shows the Jaccard index for each pair of clusters from the 3 clustering solutions (with respectively 5, 4 and 7 clusters).

Chapter 11

Cluster Merging

The clustering solutions to be merged were chosen randomly between all datasets in the training-set and as well as the number of clusters because, as Chapter 9 explains, the OSI could not be used to determine the optimal number of clusters. The following discussion thus only focuses on studying the Cluster Merging as described by F. Chung in [5].

As described in Section 6.3, it was possible to use two strategies. Only the first one, described by F. Chung, was tested. The results would of course have been different. For example, in Table 11.1, $(0, 2)$ and $(0, 3)$ are almost identical except that $(0, 2)$ is also merged with $(2, 5)$. This would have resulted in 2 different cluster with the second strategy presented in Section 6.3 but all the three of them would have been merged into a single cluster if the first strategy was used.

There was not enough time to implement both strategy and to compare the results hence only the first one was used.

The algorithm was tested as described in Section 6.4. The same 3 clustering solutions as in the previous chapter were used and the Table 10.1 was thresholded with a value of 0.65 resulting in Table 11.1.

The result of the fusion step with this threshold was 9 clusters. They were created as indicated below:

New cluster 0: merging of clusters $(0, 0)$ and $(2, 0)$. See Fig. 11.1.

New cluster 1: original cluster $(0, 1)$.

New cluster 2: merging of clusters $(0, 2)$, $(0, 3)$, $(2, 1)$, $(2, 5)$ and $(2, 6)$. See Fig. 11.2.

New cluster 3: original cluster $(0, 4)$.

New cluster 4: original cluster $(1, 0)$.

New cluster 5: merging of clusters $(1, 1)$, $(1, 2)$ and $(1, 3)$. See Fig. 11.3.

		0					1				2						
		0	1	2	3	4	0	1	2	3	0	1	2	3	4	5	6
	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1
	3	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1
	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1
	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
	6	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1

Table 11.1: The table shows the previous table thresholded: $\mathcal{J} > 0.65$. There are thus 9 clusters remaining.

New cluster 6: original cluster $(2, 2)$.

New cluster 7: original cluster $(2, 3)$.

New cluster 8: original cluster $(2, 4)$.

Each new cluster is shown below with the original clusters that were merged to create it (Fig 11.1, Fig 11.2 and Fig 11.3.

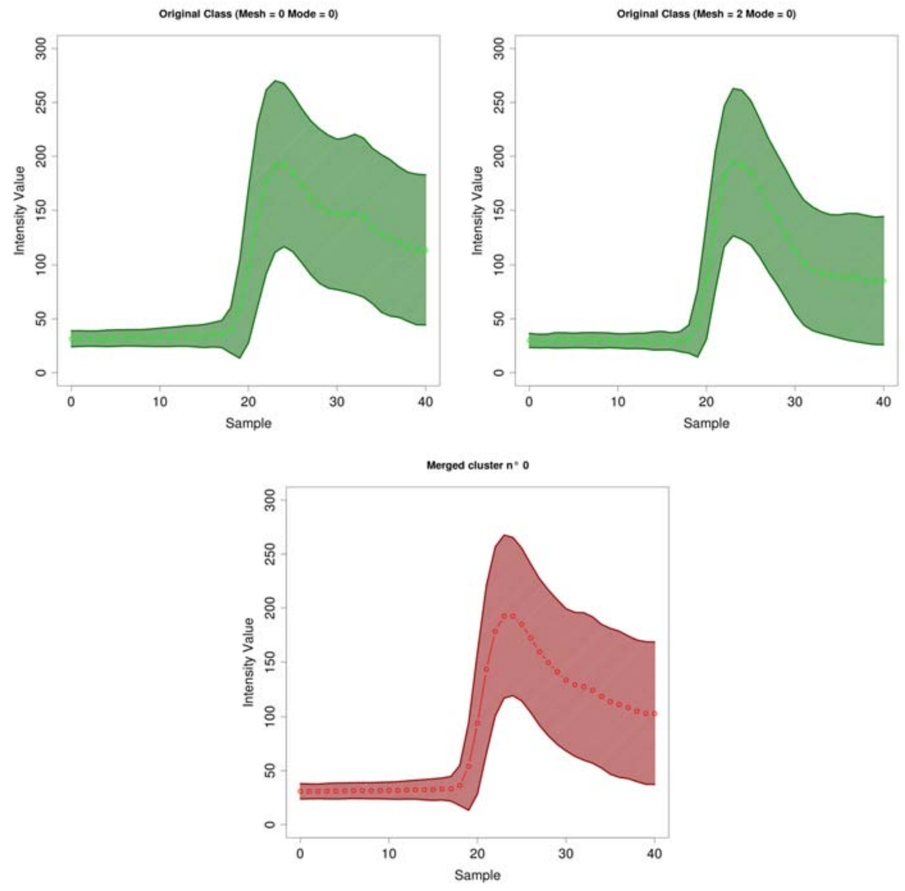


Figure 11.1: The figure shows the 2 original clusters' mean profile and standard deviation in green and the resulting cluster 0 in red.

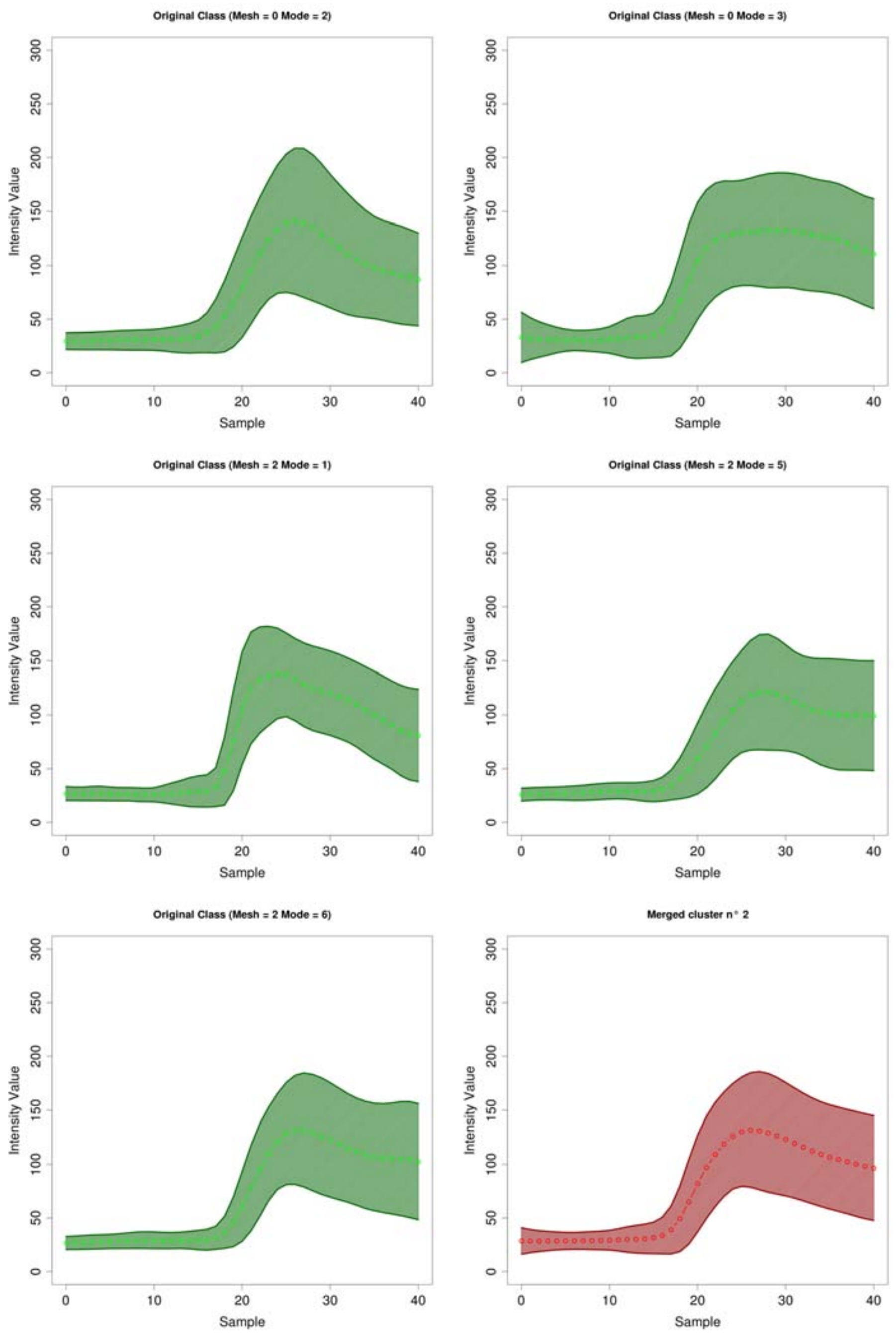


Figure 11.2: The figure shows the 5 original clusters' mean profile and standard deviation in green and the resulting cluster 2 in red.

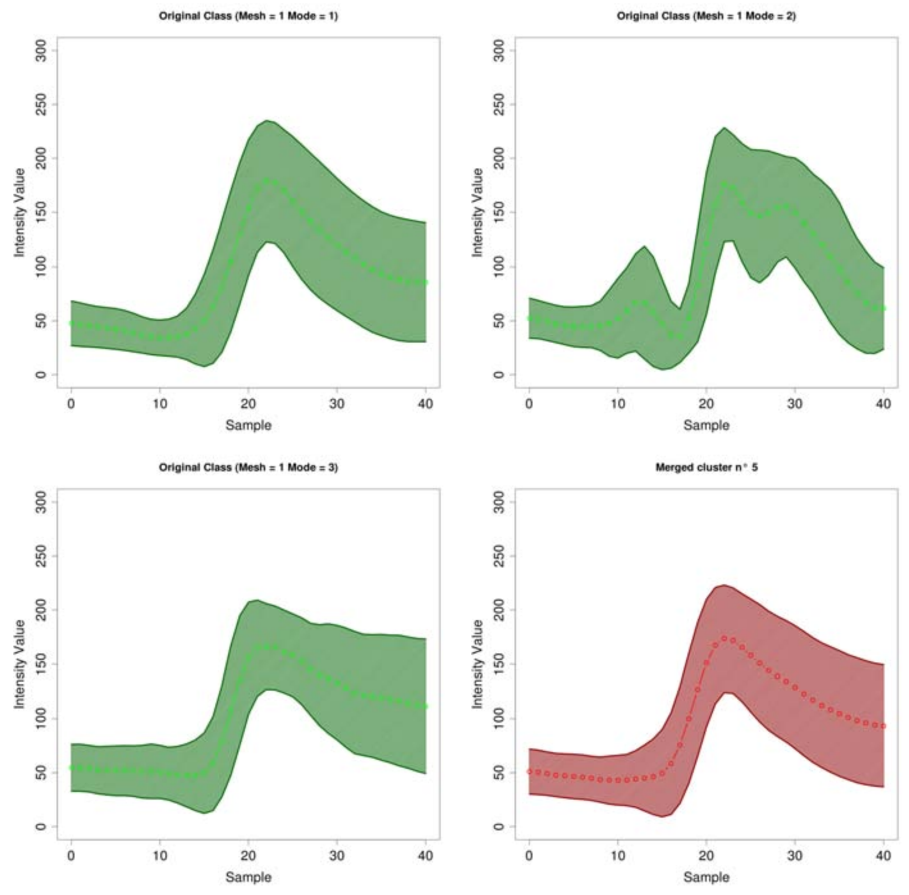


Figure 11.3: The figure shows the 3 original clusters' mean profile and standard deviation in green and the resulting cluster 5 in red.

The resulting cluster membership of each cluster and their associated profile are shown below in Fig. 11.4, Fig. 11.5, Fig. 11.6, Fig. 11.7, Fig. 11.8, Fig. 11.9, Fig. 11.10, Fig. 11.11 and Fig. 11.12.

As, in this case, there was a point-to-point correspondence between all meshes in the training-set, it was not necessary to perform the projection step and the posterior probabilities were simply normalised per vertex.

It is interesting to see that there is a clear cluster associated with the condyle (see Fig. 11.4). It is of course not completely smooth because the number of different clustering solutions that was used is too low.

For example, in one of the clustering, the tibia bone almost touches the femur which causes the profiles to be very different at this place. It is an interesting information that is not well taken into account due to this few number of used clustering.

The ideal result would be a spatially smooth cluster for the whole condyle when the tibia doesn't touch the femur and another cluster for when it is the case. It probably depends on how the patient lays when the exam is performed and with a sufficient number of data-sets in the training-set, this information could be taken into account.

This also the case for the lateral sides of the femur which are almost always clustered together in the individual data-sets and also merged into one cluster (see Fig. 11.6. Probably the corresponding cluster of two of the three original clustering solutions are merged together and one is slightly different and can was merged into cluster 5 in Fig. 11.9).

A general overview of the results shows that the clusters tend to grow spatially and smoothen with respect to their memberships when merged together. As the goal is to find zones on the surface where there is an high probability to find one particular type of clusters, this is not good.

Possible improvements include another way of merging cluster together. In fact, instead of just focusing on the intensity data, it should be possible to fragment the clusters in order to find zones in which there is a strong correlation between the intensity profiles associated to their points but also to the ones associated to corresponding points throughout the data-sets in the training-set.

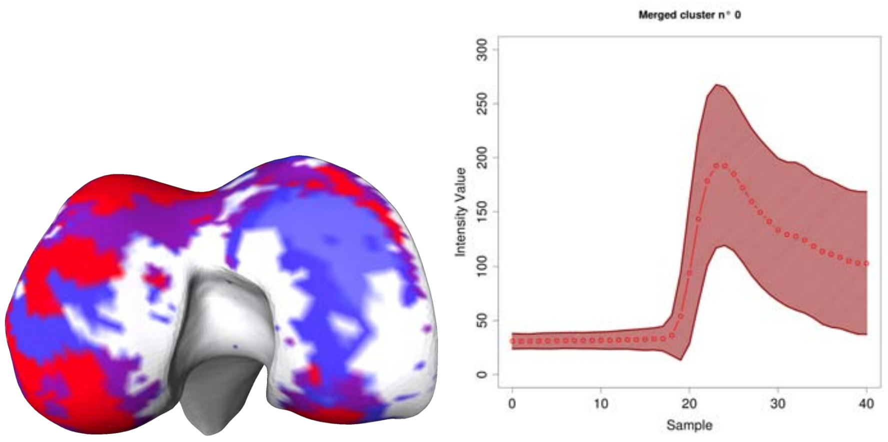


Figure 11.4: The figure shows the resulting cluster 0 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5 and red = 0.75.

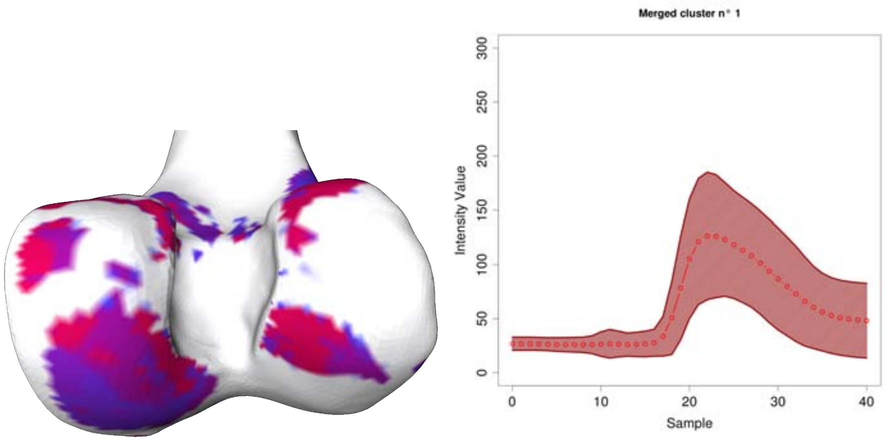


Figure 11.5: The figure shows the resulting cluster 1 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

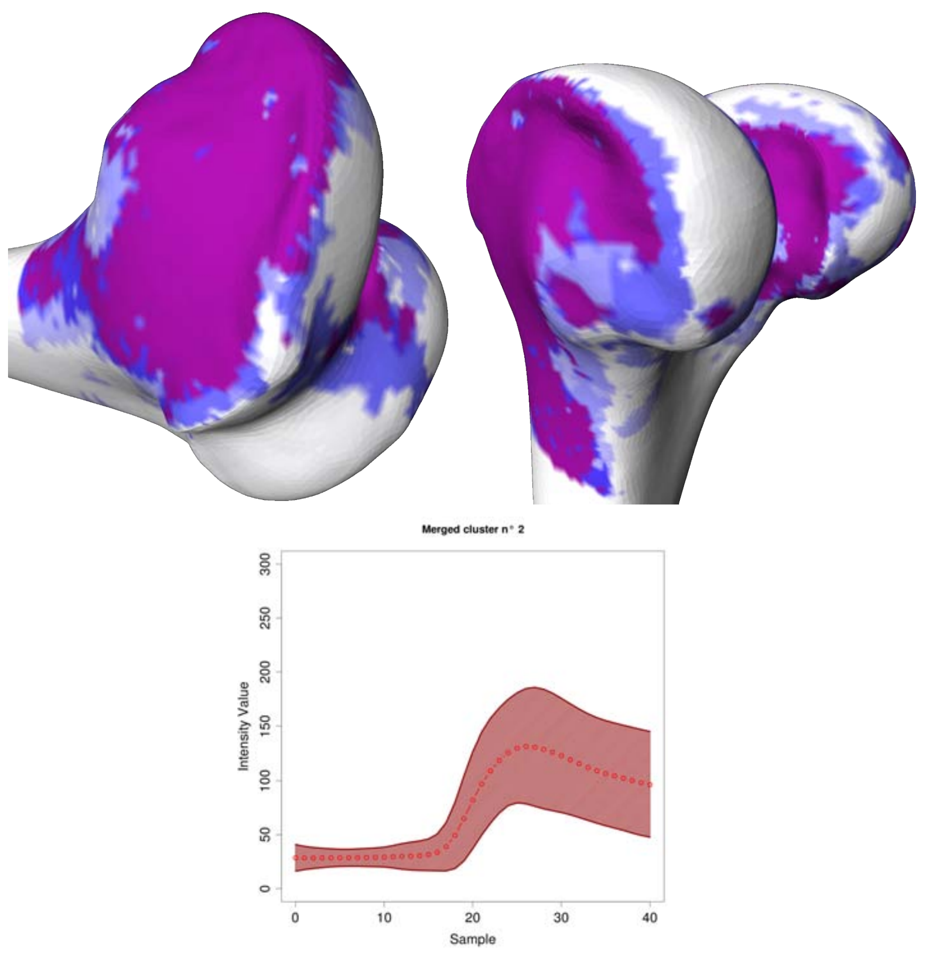


Figure 11.6: The figure shows the resulting cluster 2 with its membership values for each vertex of the surface (2 different views). The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

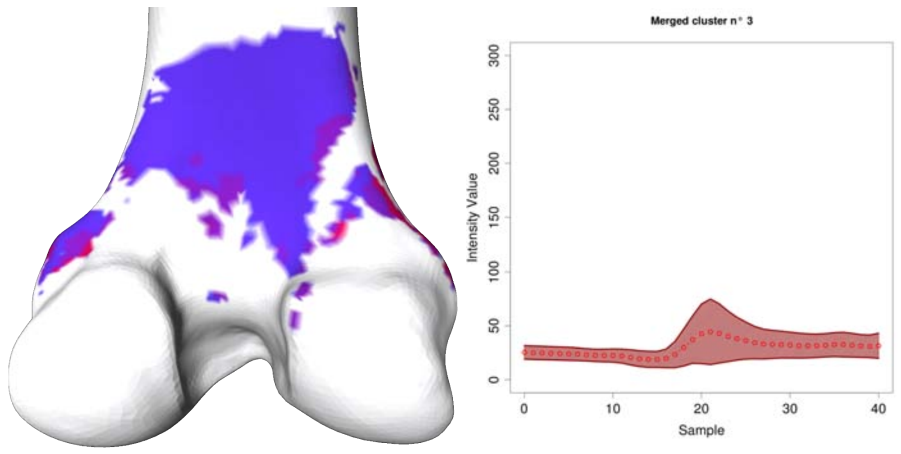


Figure 11.7: The figure shows the resulting cluster 3 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

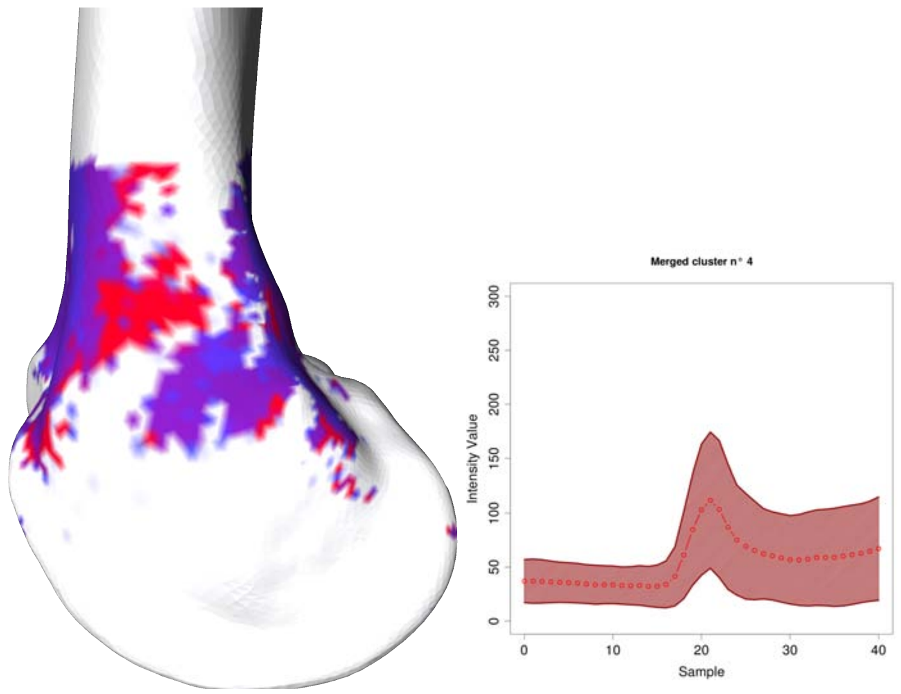


Figure 11.8: The figure shows the resulting cluster 4 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

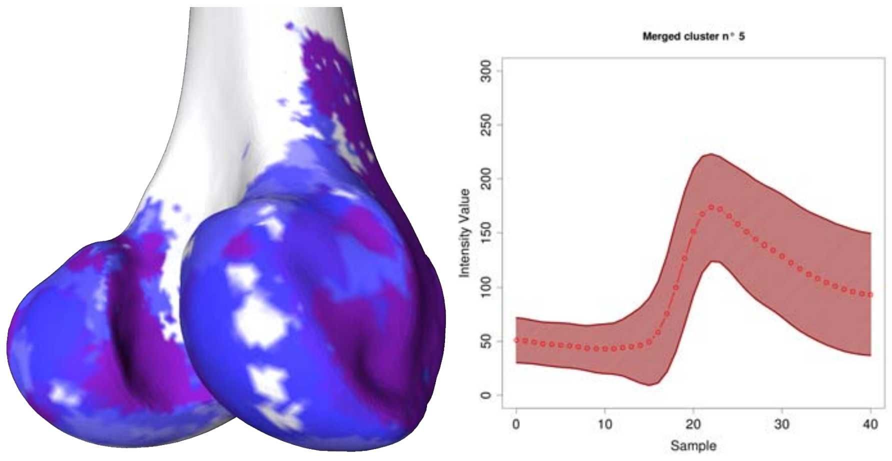


Figure 11.9: The figure shows the resulting cluster 5 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

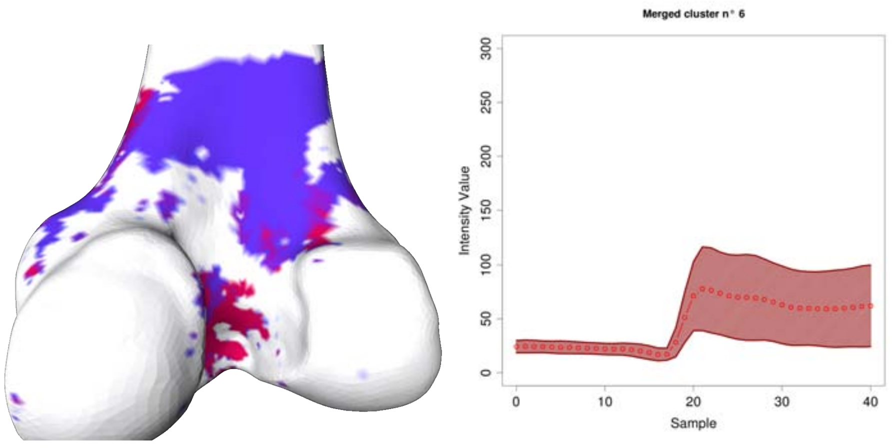


Figure 11.10: The figure shows the resulting cluster 6 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

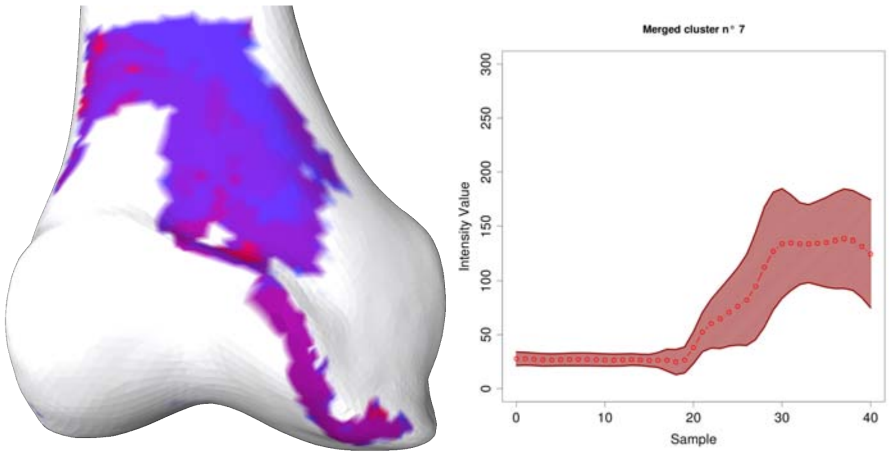


Figure 11.11: The figure shows the resulting cluster 7 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

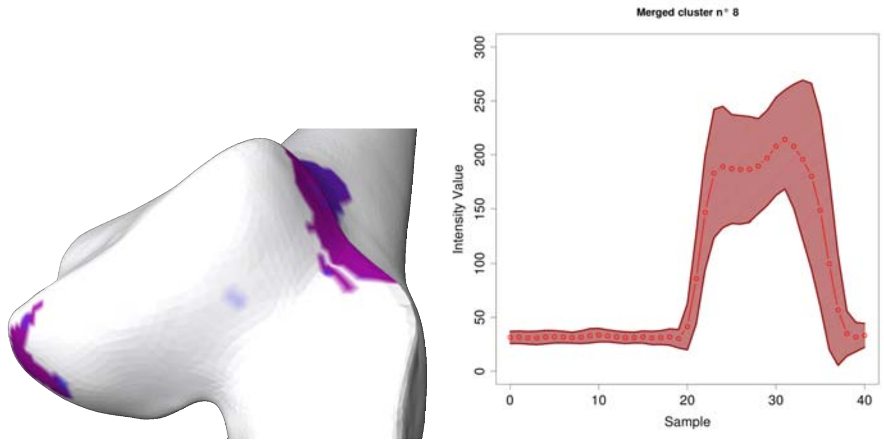


Figure 11.12: The figure shows the resulting cluster 8 with its membership values for each vertex of the surface. The colour map coding is the following: white = 0.0, blue = 0.25, magenta = 0.5and red = 0.75.

Conclusion and Future Work

The goal of this thesis was to propose and study a method for generating a profile intensity model which could be used to create cost functions in an automatic fashion for model-based segmentation.

The results have shown that it is in fact possible, in individual clustering solutions, to find zones where the intensity profiles are similar and spatially homogeneously distributed. For example, clusters associated with the condyles of the femur and with its lateral sides were found in the clustering of many data-sets.

The heuristic developed to determine the optimal parameter for the spatial regularisation in the NEM algorithm has proven to be quite good as it found the expected sudden drop in the likelihood value when the parameter was too high and suggested an acceptable value when it was not possible to find it.

There are however a lot of possible improvements at different levels of the process and ideas to explore which could be used to exploit the information given by these intensity profiles. Hereunder is a list of these ideas which popped up during the research and analysis of the results.

When sampling the data, not only intensity information could be used but also, for example, curvatures or any other type of morphological data that could enrich the model.

The different initialisation methods should be tested to compare their influence on the final result of the NEM algorithm. For the method that was specifically developed for the NEM algorithm, a proper normalisation of the intensity and spatial data should be found in order to enable the determination of the spatial regularisation parameter.

Although it works quite well, the heuristic for determining the spatial regularisation parameter β could be improved or replaced by a more efficient or precise method.

In order to determine the optimal number of clusters to use during the NEM clustering, the behaviour of the OSI should be further studied and probably applied to a real fuzzy clustering and not to the hard clusters produced by the EM algorithm.

As it was mentioned prior to the fusion step the clusters have to be compared and this step should be improved as well. In fact, the use of the surface delimited by the standard deviation around the mean hides some essential differences between clusters.

Most of the improvements should probably be made on the merging step. It was observed that the clusters tend to grow spatially and posterior probabilities diminish. Another method for merging the clusters should be developed which would allow to spatially fraction the clusters to create zones that are as large as possible whilst having high posterior probabilities for the points that are in it.

The intensity models could in fact provide information that could be used to improve the segmentation process with the Statistical Shape Models by guiding the deformation of their surface. This could be done because the intensity model would provide the typical profiles that can be encountered for specific zones of the SSM and the deformation could tend to move the points in these zones in order for them to match this profile.

To guide the deformation and match the profiles of the model with those found in the image that is being segmented, a suited distance should be found. Although a simple distance like the Mahalanobis distance could be used, more sophisticated algorithms could be used. For example if instead of matching the profile itself, it was transformed in a sequence of tokens that describes it, an algorithm could be used for matching multiple sequences as used in the field of genetic.

Appendix A

R Code for Fuzzy C-Means Initialisation of the NEM Algorithm

```
library("e1071")

load("~/EnhancedProfileScalarField.Rdata")

data <- array(EnhancedProfileScalarField.am, dim=c(60, 15172, 44))

trainingSetID <- 2

tData <- data[trainingSetID,,]
nRows <- dim(tData)[1]
nCols <- dim(tData)[2]

alpha <- 2.5

tData[,42:44] <- alpha * tData[,42:44]

# Remove rows which contain NaN values
removedRows <- array(dim=c(0,1))
newData <- array(dim=c(0,nCols))

for (i in 1:nRows) {
  NANFound <- FALSE
  for (j in 1:nCols) {
    if(is.na(tData[i,j])) {
      NANFound <- TRUE
      break
    }
  }

  if(NANFound) {
    removedRows <- rbind(removedRows, c(i))
  } else {
    newData <- rbind(newData, tData[i,])
  }
}

newRows <- dim(newData)[1]
remRows <- dim(removedRows)[1]

# Perform clustering using c-means algorithm
# with euclidean distance and 4 clusters
clust <- cmeans(newData, 4)

# Save hard partitioning to file by redirecting output
sink("~/R-output-clust-enh.txt")

nI = 1
rI = 1
for(i in 1:nRows) {
  if(rI <= remRows && removedRows[rI] == i) {
    print(-1)
    rI <- rI + 1
  } else {
    print(clust$cluster[nI]-1)
    nI <- nI + 1
  }
}

sink()
```

Appendix B

C++ Code for the Evaluation of the Spatial Regularisation Parameter

```
/* Number of samples for beta */
unsigned int nbrSamples = 6;
/* Value range for beta. If nbrSamples == 6 and
 * the range is [0, 10], then the 6 samples for
 * beta are: 0.0, 2.0, 4.0, 6.0, 8.0, 10.0
 * The range is chosen very wide in order to have
 * an scale for the values of the likelihood
 */
double betaStart = 0.0;
double betaStop = 10.0;
double betaRange = betaStop - betaStart;
/* In each loop the beta-space will be sampled
 * more precisely. After 6 iterations, the
 * increment between successive values of beta
 * will be 2^(1-i) with i = 6 thus 0.03125
 * This is the finest step that is allowed.
 */
unsigned int loopID = 0, maxLoops = 6;

/* Current increment for the values of beta */
double betaInc = (betaRange) /
                 (double)(nbrSamples - 1);

/* Array of results. The NEMData object contains:
 * - the parameters for the execution of the NEM algorithm (data matrix,
 * neighbourhood matrix, number of clusters, beta and the initial
 * values of the cluster membership matrix)
 * - the results of the NEM algorithm (mean and variance matrices of the
 * clusters, likelihood,...)
 */
Array<NEMData*> results;

/* First runs of the NEM algorithm for the initial
 * beta range.
 */
for(unsigned int i = 0; i < nbrSamples; i++) {
  /* Current value of beta */
  double beta = betaStart + i * betaInc;

  NEMData * nemdata = new NEMData(dataMatrix,
      neighbourhoodMatrix, clusterStart, beta,
      clusterMembershipMatrix);

  NEMClusterAlgorithm algo(nemdata);

  /* Performs the NEM algorithm and store
   * the result in the array of results.
   */
  results[i] = algo.compute();
}

/* This value indicates whether the optimal was
 * found or not.
 */
bool found = false;

/* The likelihood values extend this range. */
double range = abs(results[nbrSamples - 1]->likelihood - results[0]->likelihood);

/* Optimal value of beta */
double optimalBeta = 0.0;

/* Main loop: finishes if the number of iterations is too high or if
 * the optimal beta value was found (see corresponding break statement
 * below).
 */
while(loopID < maxLoops) {
  /* Indicates whether the new boundary values
   * for were found or not. The range is refined
   * at each iteration.
   */
  bool tmpFoundFirst = false, tmpFoundLast = false;
  /* IDs of the NEMData objects corresponding
   * to the betas of the new boundaries.
   */
  unsigned int firstID = 0, lastID = 0;
  /* Indicates wether the difference there was a rather
   * big gap between two successive points.
   */
```

```
bool hasBeenHigher = false;

/* Browse the results of the current beta range. */
for(unsigned int i = 2; !tmpFoundLast && i < nbrSamples - 1; i++) {
  NEMData* lastNEMData = results[i];
  NEMData* middleNEMData = results[i - 1];

  /* First search the new left boundary value of beta. */
  if(!tmpFoundFirst) {
    NEMData* firstNEMData = results[i - 2];
    NEMData* afterlastNEMData = results[i + 1];

    /* The goal is to compute the angle between two vectors
     * defined by successive points in the beta-likelihood plane.
     * So first compute firstVector which has two coordinates. Its
     * coordinates are normalised with respect to the global range
     * if both variable (beta and likelihood) and then the vector
     * itself is normalised.
     */
    Vector2D<double> firstVector;
    firstVector.x = (middleNEMData->beta - firstNEMData->beta) /
             betaRange;
    firstVector.y =
      (middleNEMData->likelihood - firstNEMData->likelihood) /
      range;
    firstVector.normalize();
    /* Same thing, except that the likelihood value has
     * been smoothed by taking the mean of the two points
     * right of the middleNEMData.
     */
    Vector2D<double> secondVector(2);
    secondVector.x = (
              lastNEMData->beta
              - middleNEMData->beta
              ) / betaRange;
    secondVector.y = (
              lastNEMData->likelihood / 2.0
              + afterlastNEMData->likelihood / 2.0
              - middleNEMData->likelihood
              ) / range;
    secondVector.normalize();

    /* The angle between the two vectors will be computed
     * relative two the first vector. This will enable to
     * see if the second vector has an higher slope than
     * the first.
     * Therefore, the
     */
    double projX = firstVector.dot(secondVector);
    Vector2D<double> tmp = secondVector - 1.0 * projX * firstVector;
```

```
      double projY = tmp.norm();
      double theta = atan2(projX, projY) * 180 / PI;

      /* Gap between the "last" and "first" value of the
       * likelihood.
       */
      double gap = abs(lastNEMData->likelihood - firstNEMData->likelihood);

      /* If there is a big gap and a hard angle, the
       * optimal value of beta is considered as found.
       */
      if(theta < -60 && gap > 0.2 * range) {
        found = true;
        optimalBeta = middleNEMData->beta;

        break;
      }

      /* If either the gap or the angle is big,
       * the left bound is found.
       */
      if(theta < -40 || gap > 0.1 * range) {
        if(gap > 0.1 * range)
          hasBeenHigher = true;

        tmpFoundFirst = true;
        firstID = i - 2;
      }
    }

    /* Then search the new right boundary.
     * If during the search, there was already a rather
     * big gap but the next one is to small, this is the
     * second boundary.
     */
    if(tmpFoundFirst && hasBeenHigher &&
      lastNEMData->likelihood - middleNEMData->likelihood < 0.1 * range
    ) {
      tmpFoundLast = true;
      lastID = i + 1;
      optimalBeta = middleNEMData->beta;

      break;
    }
    /* If during the search, there wasn't a rather big gap
     * (if the angle was big enough) but now, one has been
     * found, this is the second boundary.
     */
    else if(tmpFoundFirst && !hasBeenHigher &&
      lastNEMData->likelihood - middleNEMData->likelihood > 0.1 * range
```

```
  ) {
    tmpFoundLast = true;
    lastID = i + 1;
    optimalBeta = middleNEMData->beta;

    break;
  }
}

/* If the gap was found, stop searching ! */
if(found) {
  break;
}

/* If either the first boundary or the second one or
 * both weren't found, simply refine almost the entire
 * initial beta range.
 */
if(!tmpFoundFirst) {
  lastID = nbrSamples - 2;
}

if(!tmpFoundLast) {
  lastID = nbrSamples - 1;
}

/* If the maximum number of iterations wasn't reach,
 * it's time to refine the new beta samples range.
 */
if(loopID < maxLoops - 1) {
  /* Array where the results that are in the new
   * beta range are kept to avoid recomputing them.
   */
  Array<NEMData*> tmpResults;

  /* Search for such results, delete the others. */
  for(unsigned int i = 0; i < nbrSamples; i++) {
    if(i >= firstID && i <= lastID)
      tmpResults.append(results[i]);
    else
      delete results[i];
  }

  /* Clear the result array */
  results.clear();

  /* Resize it, there are always two times more new
   * samples.
   */
  nbrSamples = 2 * (tmpResults.size() - 1) + 1;
```

```
    results.resize(nbrSamples);

    /* Replace the elements of tmpResults in the current
     * result array.
     */
    for(unsigned int i = 0; i < tmpResults.size(); i++) {
      results[2 * i] = tmpResults[i];
    }

    /* New beta range and increment. */
    betaStart = results[0]->beta;
    betaStop = results[nbrSamples - 1]->beta;
    betaRange = betaStop - betaStart;
    betaInc = (betaRange) / (double)(nbrSamples - 1);

    /* If the beta range is zero, something went wrong
     * or the value was found.
     */
    if(betaRange == 0) {
      beta = betaStart;
      break;
    }

    #pragma omp parallel for
    for(unsigned int i = 1; i < nbrSamples; i += 2) {
      double beta = betaStart + i * betaInc;

      NEMData * nemdata = new NEMData(dataMatrix,
          neighbourhoodMatrix, clusterStart, beta,
          clusterMembershipMatrix);

      NEMClusterAlgorithm algo(nemdata);

      /* Performs the NEM algorithm and store
      * the result in the array of results.
      */
      results[i] = algo.compute();
    }
  }
  /* Maximum number of iterations was reached, select
   * the third beta in the beta range.
   */
  else {
    beta = results[firstID + 2]->beta;
  }

  loopID++;
}
```

List of Figures

List of Tables

Bibliography

[1] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision, 2004.

[2] Dagmar Kainmueller, Thomas Lange, and Hans Lamecker. Shape constrained automatic segmentation of the liver based on a heuristic intensity model, 2007.

[3] Heiko Seim, Dagmar Kainmueller, Hans Lamecker, Matthias Bindenagel, Jana Malinowski, and Stefan Zachow. Model-based auto-segmentation of knee bones and cartilage in mri data. In B. v. Ginneken et al., editor, *Proc. MICCAI Workshop Medical Image Analysis for the Clinic: A Grand Challenge*, pages 215 – 223, 2010.

[4] Hans Lamecker, Thomas Lange, and Martin Seebaß. A statistical shape model for the liver. In T. Dohi and P. Kikinis, editors, *MICCAI 2002*, Lecture Notes in Computer Science 2489, pages 422 – 427, 2002.

[5] François Chung and Hervé Delingette. Multimodal prior appearance models based on regional clustering of intensity profiles. In Guang-Zhong Yang, David Hawkes, Daniel Rueckert, Alison Noble, and Chris Taylor, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009*, volume 5762 of *Lecture Notes in Computer Science*, pages 1051–1058. Springer Berlin / Heidelberg, 2009.

[6] C. Ambroise, M. Dang, and G. Govaert. Clustering of spatial data by the em algorithm. 2007.

[7] Heiko Seim, Dagmar Kainmueller, Markus Heller, Hans Lamecker, Stefan Zachow, and Hans-Christian Hege. Automatic segmentation of the pelvic bones from ct data based on a statistical shape model, 2008.

[8] Hans Lamecker, Stefan Zachow, Hannes Haberl, and Michael Stiller. Medical applications for statistical shape models, 2005.

[9] Jeff Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, 1998.

[10] Richard J. Hathaway. Another interpretation of the em algorithm for mixture distributions. *Statistics & Probability Letters*, 4(2):53 – 56, 1986.

[11] Tianming Hu, Ji Ouyang, Chao Qu, and Chuanren Liu. Initialization of the neighborhood em algorithm for spatial clustering. In *ADMA'09*, pages 487–495, 2009.

[12] Niloofar Gheissari and Alireza Bab-Hadiashar. A comparative study of model selection criteria for computer vision applications. *Image and Vision Computing*, 26(12):1636 – 1649, 2008.

[13] Wikipedia. Multiple sequence alignment — wikipedia, the free encyclopedia, 2011. [Online; accessed 8-May-2011].