# Assignment 4 - Medical Image Analysis

2020/2021

Francesco Corti & Julie Mazzella
12002944 12040967

**Graz, June 20, 2021**

Contact: Francesco Corti & Julie Mazzella, corti@student.tugraz.at julie.mazzella@student.tugraz.at

**Abstract**

The following report aims to illustrate the work done for the fourth assignment of the Medical Image Analysis course. A segmentation method called Active Shape Model was implemented through a Python code and multiple experiments were applied varying several parameters, such as the number of set of points and the test images considered. The variation of the parameters lead to the analysis of the variation of the error in each experiment. What was observed is that the choice of the parameters is fundamental to avoid the local minima problem in the achievement of the solution.

# 1 Introduction

There are three main segmentation categories: basic low-level methods, high-level deformable methods, and shape prior-based deformable models.
The last ones deal with shape and appearance priors explicitly. Compared to high-level deformable models, strong model knowledge and similarities between the anatomy and patient data are used. The goal is to explicitly train on prior instances and incorporate this strong model knowledge by using generative learning, yielding sub-space models of appearance and shape. The prior instances are sample objects where anatomical variations are known. Once the parameterized model is obtained, by training on sample objects, it is applied on new instances. In Figure 1, $g$ is the typical shape, $P$ encodes the variations and $b$ are the parameters used to tune the model. These parameters are modified until the generated synthetic model fits.

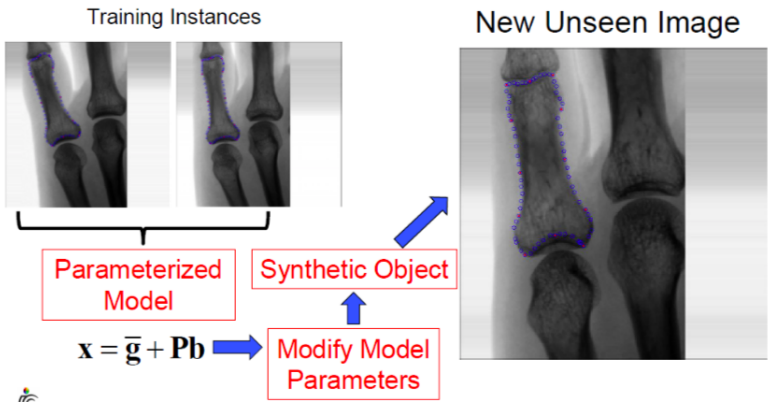



Figure 1: Shape prior-based deformable models algorithm

There exist three different approaches for model-based segmentation and subspace models:
- Patch-based models ("EigenFace")
- Statistical Shape models and Active Shape models
- Statistical Appearance models and Active Appearance models

## 1.1 Eigenface (Eigenpatch) Models

Following this approach, first a region is marked on images from a training set, sampled, normalized, and statistically analyzed. It is fundamental that bounding boxes are registered initially, to be able to synthesize an appearance by putting in parameters for $b$.
Each region can be represented by a feature vector, normalized to avoid lighting changes. The feature vector contains intensity values at corresponding pixel locations. All of these feature vectors are then put into a matrix, the training set $D$. At this point, dimensionality reduction

is required. To do so, the distributions of the samples have to be learned.
The simplest approximation for the model is a Gaussian model, the mean $g$ and covariance matrix $C$ of the $s$ feature vectors define the Gaussian model in an $n$-dimensional space. With this model, it is possible to achieve parametric density estimation.

$$\bar{\boldsymbol{g}} = \frac{1}{s}\boldsymbol{D_1} \tag{1}$$

$$\hat{\boldsymbol{D}} = \{\boldsymbol{g^1} - \bar{\boldsymbol{g}}\boldsymbol{g^2} - \bar{\boldsymbol{g}}...\boldsymbol{g^2} - \bar{\boldsymbol{g}}\} \tag{2}$$

$$\boldsymbol{C} = \frac{1}{s-1}\hat{\boldsymbol{D}}\hat{\boldsymbol{D}}^{\boldsymbol{T}} \tag{3}$$

After computing the mean and the covariance matrix, the projection lines that maximize the variance of the projected data have to be calculated. This Least-Squares problem can be solved by computing the eigenvectors $\boldsymbol{P}$ of the covariance matrix $C$. The eigenvectors are obtained through the Principal Component Analysis and they represent the direction of variations, while the eigenvalues quantify the variations in those specific directions.
In Figure 2, $\boldsymbol{b}$ are the parameters that define the deformation in a specific direction.

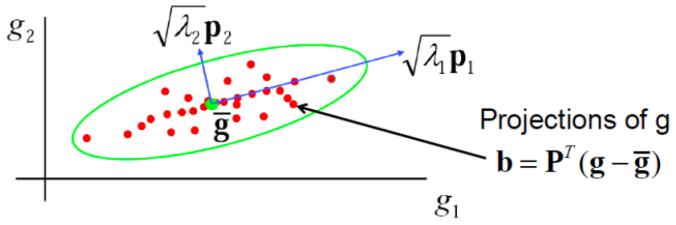



Figure 2: Principal Component Analysis projected data

Now it is possible to perform dimensionality reduction to get to a subspace representation. In Figure 3, all features are projected to the eigenvector with the largest variance. The first eigenvalue is the most significant, it is the one with the highest variance, so the most impact in changing the shape.

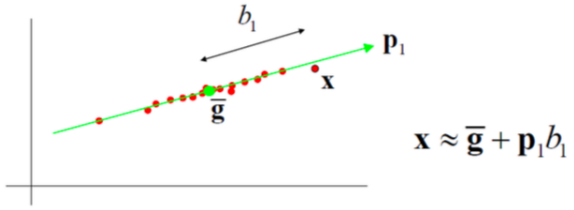



Figure 3: Projection of features in the most significant direction

In general, more than one eigenvector gives information. However, after a certain point, the contribution gets very small, therefore it is possible to define a threshold $t$ and perform dimensionality reduction.
Given a set of training examples $\{g_i\}$, after computing the mean and eigenvectors of the covariance matrix, the model is:

$$\boldsymbol{x} \approx \bar{\boldsymbol{g}} + \boldsymbol{P_t}\boldsymbol{b_t} \tag{4}$$

with $\boldsymbol{P_t}$ the $t$ first eigenvectors, and $\boldsymbol{b_t}$ the projections of the training examples into the subspace.

Now a model for appearances is obtained and it can be applied on new instances. However, registration and face deformation due to expressions remain problematic. For example, when a

face is smiling, the correlation between pixels changes and the model fails.
This variance in shape and the registration problem lead to the introduction of Statistical Shape models.

### 1.2 Statistical Shape Models

Differently from the Eigenface approach, statistical shape models are based on landmarks instead of patches: a shape can be represented using a set of points. However, this requires labeled training images, and it is very important that corresponding points are correctly associated with each other by a consistent labeling.

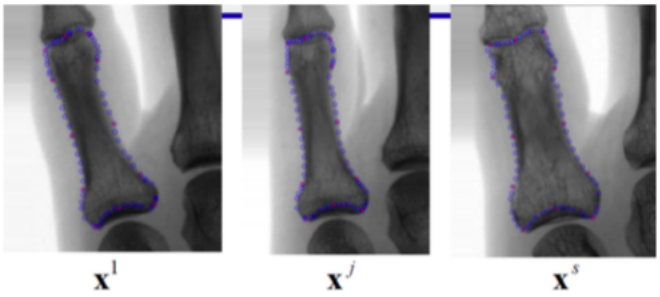



Figure 4: Correspondence between points

The location of each point in the image is known and there is correspondence between points. This means that for every point in the image it is possible to create a vector with the $x$ and $y$ coordinates, from which to build the model.
The shape vector of the corresponding points is defined as:

$$\boldsymbol{x}^j = (x_1, y_1, x_2, y_2, ..., x_N, y_N) \tag{5}$$

Given a set of training shapes (meaning $s$ point sets with $N$ corresponding points), shapes can be aligned in a common coordinate frame to remove the similarity transform, using generalized Procrustes analysis. After this registration step, only differences due to shape deformations remain. Then, it is possible to estimate a shape distribution, and for that a single Gaussian is sufficient (again using the PCA model).

Using generalized Procrustes analysis, it is possible to find the $N$ transformations $T_i$ which minimize:

$$\sum |\boldsymbol{m} - T_i(x_i)|^2 \tag{6}$$

where $\boldsymbol{m} = \frac{1}{N} \sum T_i(x_i)$ is the mean shape estimate.

An instance of $\boldsymbol{x^j}$ of the training set is a $2n$ column vector with $x$ and $y$ of all (aligned) points stacked together: this provides a high-dimensional feature space.
The goal is to model the $s$ aligned shapes as these form a distribution in $2N - dimensional$ shape space. Aligned shapes can be represented by their probability density function $p(x)$, which follows a multi-variate Gaussian distribution. Given this knowledge it is sufficient to compare a given shape to the model.

As a similarity metric the Euclidean distance $d_E$ or the Mahalanobis distance $d_M$ are used. The latter is modified by the covariance matrix as well, therefore the same Euclidean distance is interpreted differently depending on where the sample is located in relation to the covariance. For shape synthesis and model fitting a parameterized model is obtained through PCA: $\boldsymbol{x} =$

$f_{shape}(\boldsymbol{b})$.
Given aligned shapes $\{\boldsymbol{x_i}\}$, the mean is computed, then the PCA is applied to remove the eigenvectors corresponding to small eigenvalues. The final model is:

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + \boldsymbol{P_t b_t} \tag{7}$$

where $\boldsymbol{P_t}$ are the first $t$ eigenvectors of the covariance matrix and $\boldsymbol{b_t}$ are the shape model parameters defining a set of deformable model.

### 1.3 Active Shape Models

In this approach, the training is done explicitly on prior instances. Given some training instances x, a parameterized model can be derived:

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{P b} \tag{8}$$

where $\bar{x}$ and $P$ describe the model, while $b$ the model parameters. The parameterized model should be used to segment a new unseen image. In order to do so, the model parameters are modified to create a synthetic object which has to be placed into the new image to generate a segmentation.

It is possible to define a local model of intensity, along normals to the shape, derived from the training set. What is obtained is an iterative matching procedure, which is regularized using the parameterized model:
1: Look along the normals through each model point to find the best local match for the model at that point.
2: Update the pose and model shape parameters to best fit the model instance to the found points.
3: Repeat until convergence.
This problem is, however, non-convex and non-linear, which means that only local optima are obtained. A good initialization is thus fundamental.

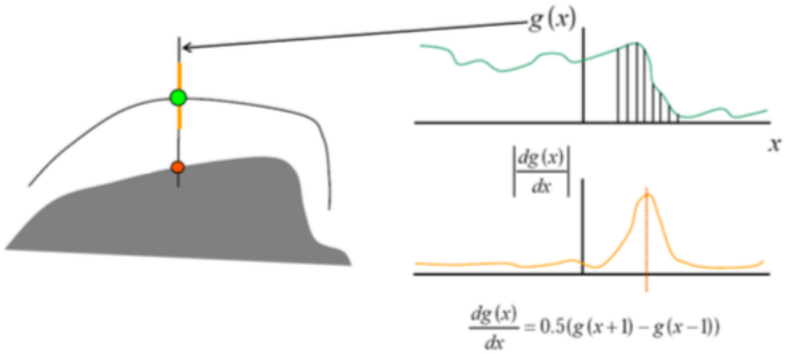



Figure 5: Active Shape Model

The goal is to go from the green to the orange point. Therefore, a local match is needed for each point along the profile. By simply searching for strong edges, the goal is to move towards the strongest gradient. However, the true point might not actually lay on the strongest edge.

To correctly detect points which are not on the strongest edge, it is possible to use available data during training. All points are annotated, the normal is known, as well as intensities along the profile. For each point it is possible to build up a 1D profile line model that gives grey values

along the profile in the training data.
Sample profiles are collected from the training set. Afterwards, the intensity profiles are normalized to allow for global lighting variations. From the training set, a generative PCA model of 1D lines is learnt.
An improvement to this method would be to search 2D intensity patches in a neighborhood N around the point, instead of only searching in the normal direction.

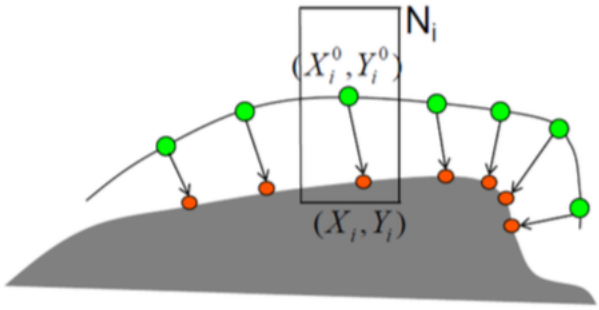
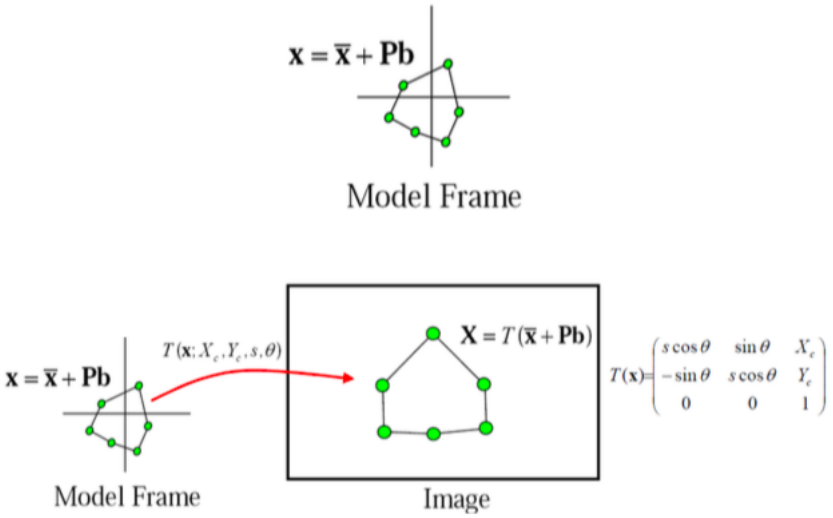



Figure 6: Template Matching model

In this case, the algorithm would change:
1: Search along the profile for a candidate X
2: Update similarity transformation $T$ and parameters $b$ to minimize:

$$|\boldsymbol{X} - T(\bar{\boldsymbol{x}} + \boldsymbol{Pb})|^2 \tag{9}$$

The similarity transformation $T$ and the parameters $b$ are unknown and their update is considered a regularization step. Since a generative model is being considered and the goal is to best fit a synthetic object into the image for segmentation, the placement of the model into the image is also an important issue. The model points are defined in a model coordinate frame. In order to place the model into an image, a global similarity transformation $T$ must be applied.




In order to update the parameters, a shape model ($b$) and a pose ($\Theta$) parameters have to be found to minimize the cost $S$:

$$S(\boldsymbol{b}; \Theta) = |\boldsymbol{X} - T(\bar{\boldsymbol{x}} + \boldsymbol{Pb}; \boldsymbol{\Theta})|^2 \tag{10}$$

with a pose $\Theta$ given as:

$$\Theta = (X_c, Y_c, s, \theta) \tag{11}$$

The $b$ parameters are described by variance, and can be computed by eigen decomposition.

In order to update the pose and shape model parameters alternately, the following approach can be used:
1: Fix $b$ giving model instance $X_b$ and find $\Theta$ minimizing $S(\Theta) = |\boldsymbol{X} - T(\boldsymbol{X_b}; \boldsymbol{\Theta})|^2$
2: Analytic solution (Procrustes) exists: $\Theta'$
3: Given fixed $\Theta'$, find $b$ which minimizes $S(b) = |\boldsymbol{X} - T(\Theta'; \bar{\boldsymbol{x}} + \boldsymbol{Pb})|^2$
4: Repeat until convergence

# 2 Methods

This assignment aimed at exploring the segmentation methods already discussed. A set of 20 images and a set of 18 points of reference for each image were given.

The five tasks that were asked are the following:
• Task 1 - Generalized Procrustes Alignment: compute the mean shape of all training point sets (shapes) and align all shapes to the mean shape.
• Task 2 - Principle Component Analysis: apply a PCA to the aligned training shapes and reduce the number of features to the most important ones (keep 97 % of the variance).
• Task 3 - Gradient Magnitude and Mean Gradient Patches: compute all gradient magnitude images and get the mean patches, which represent a local mean image for each feature and its surrounding pixels.
• Task 4 - Iterative Active Shape Model: iteratively fit the test image to the Active Shape Model and update the estimated test shape until convergence.
• Task 5 - Evaluation: conduct experiments and report the outcomes.

To do that a Python code was implemented, using Numpy [1] and Scipy [3] as scientific computing libraries.

## 2.1 Task 1: Generalized Procrustes Alignment

The first step was to initialize the mean shape $\boldsymbol{m}$: given a set of training point sets $X = \boldsymbol{x_1}, ..., \boldsymbol{x_N}$, the mean shape was initialized choosing a random point set $x_i$ and was then centered at the origin. Afterwards, the mean shape was iteratively optimized. All point sets $x_i$ were alligned to the current mean shape $\boldsymbol{m}$. The mean shape $\boldsymbol{m}$ was then updated by calculating the mean of all point sets in $X$. The sum of squared differences between the previous and current $\boldsymbol{m}$ was computed, terminating the loop early if the resulting error was below a threshold.

## 2.2 Task 2: Principle Component Analysis

The aim of this second task was to implement an algorithm for the Principle Component Analysis. First, *np.reshape*() was used to flatten the x and y coordinates of the aligned points and converted from a [19, 18, 2] matrix to a [19, 36] matrix. Then, the command *princomp*() was used to apply the PCA and retrieve the sorted eigenvalues and eigenvectors $\boldsymbol{P}$. Once obtained those variables, only the most important features $t$ were kept, corresponding to the 97 % explained variance in training data.

### 2.3 Task 3: Gradient Magnitude and Mean Gradient Patches

For the third task the gradient magnitude $g$ was computed as:

$$g = \sqrt{g_x^2 + g_y^2} \tag{12}$$

Then, the mean shape $\boldsymbol{m}$ and the training gradient magnitudes $g$ were used to compute the mean gradient patches $\bar{g}$, setting a patch size of 15. For this purpose, the function *getpatches* was used.

### 2.4 Task 4: Iterative Active Shape Model

For this task, the shape has to be adapted inside of the image. First, a matching area of dimension 45 was created (3 times the patch size). Then the best matches of the ASM for the test gradient magnitude were extracted: in order to do so, for each point of the current shape the minimal and maximal index in x and y inside of the matching area were found. After that, the index of the best matching point of each mean gradient patch in the respective matching area of the test gradient magnitude was obtained. To do that, the function index of best match ncc() was used.

The second part of this task is the update of the shape parameters: first to update the pose transformation the best matches were aligned to the mean shape, obtaining aligned matches.
Then, the difference dx = aligned matches – mean shape was computed and the shape model $\boldsymbol{b}$ was obtained as $\boldsymbol{b} = \boldsymbol{P^T dx}$. Each $b_i$ was then constrained with eigenvalues $\pm a\sqrt{\lambda_i}$.

The third part of this task is to compute the new shape as $\boldsymbol{x_{new}} = \boldsymbol{m} + \boldsymbol{Pb}$ and align it to the best matches through the Procrustes algorithm.
Finally, the SSE between the current shape and the new shape was computed and if the error was below a defined threshold, the matching area was reduced of 2 values.

### 2.5 Task 5: Evaluation

For this last task, 5 different evaluations were conducted:
1) A new shape was computed from the mean shape, considering a vector $\boldsymbol{b}$ composed first by one eigenva]lue in first position and then one eigenvalue in the second position of values equal to 50 and 100.
2) The change of the mean segmentation error (1-Jaccard Index) for each iteration was investigated using the first image of the test set.
3) A leave-one-out cross-validation was performed over all samples.
4) The changes in the final segmentation error were investigated using a different number of eigenmodes.
5) The changes in the final segmentation error were investigated when less set of points are considered for the model.

## 3 Results

The results for the 5 tasks are shown in the following subsections.

### 3.1 Task 1

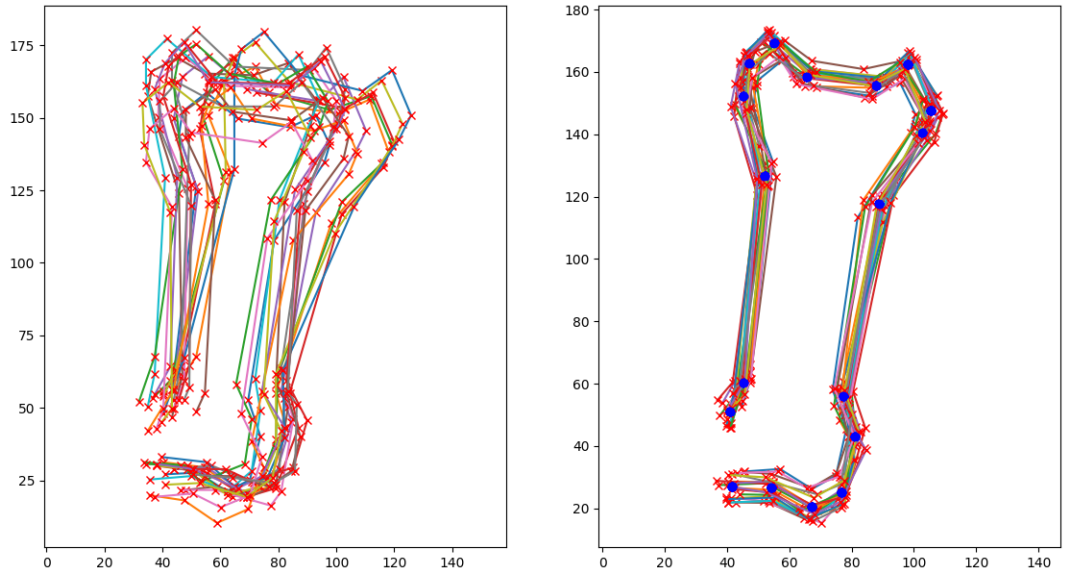


Figure 7: Alignment of points after Procrustes algorithm

### 3.2 Task 3

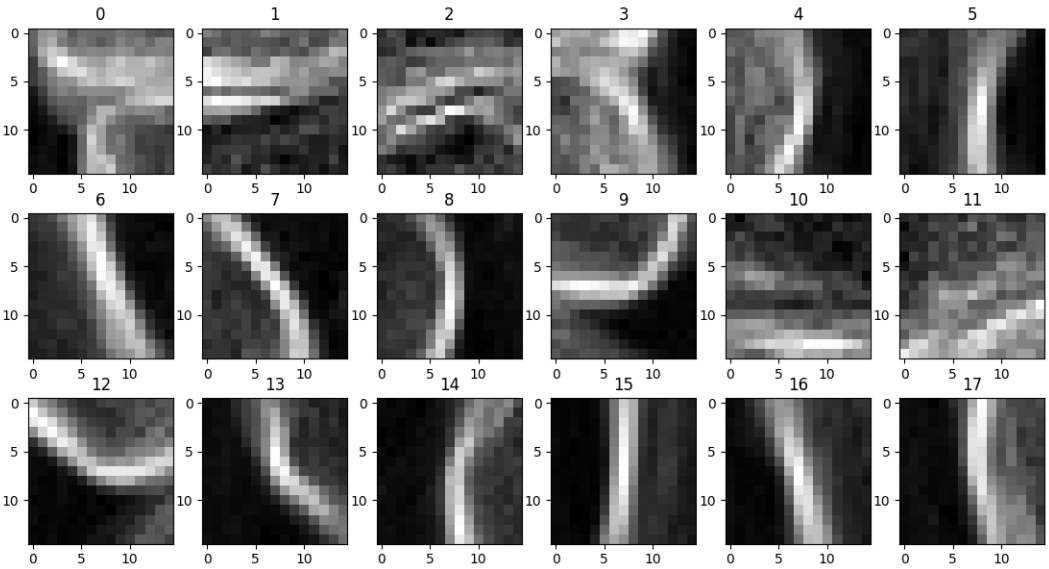


Figure 8: Mean patches obtained

### 3.3 Task 4

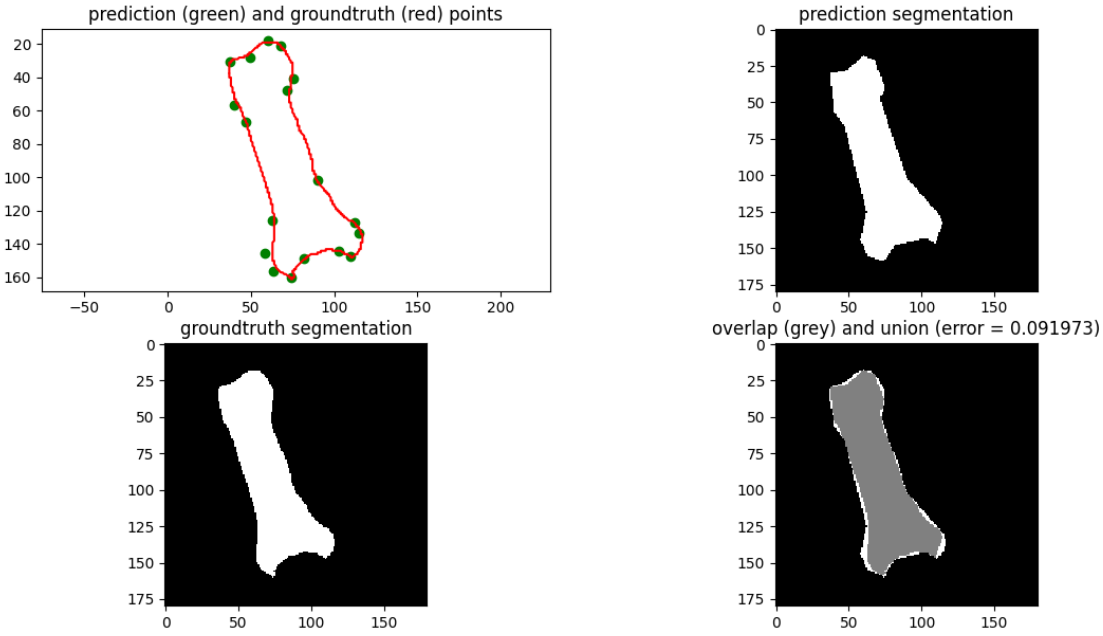



Figure 9: Result of ASM

### 3.4 Task 5

#### 3.4.1 First Evaluation

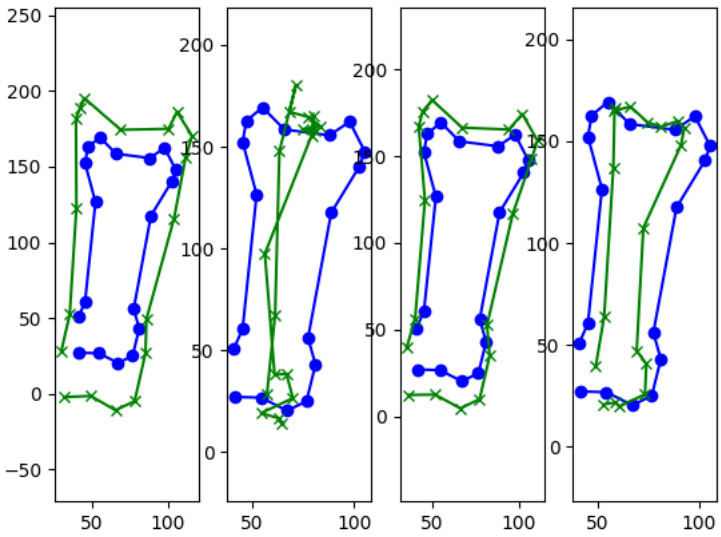


Figure 10: Result choosing 100 as value for first and second eigenvalues and 50 for first and second eigenvalues

### 3.4.2 Second Evaluation

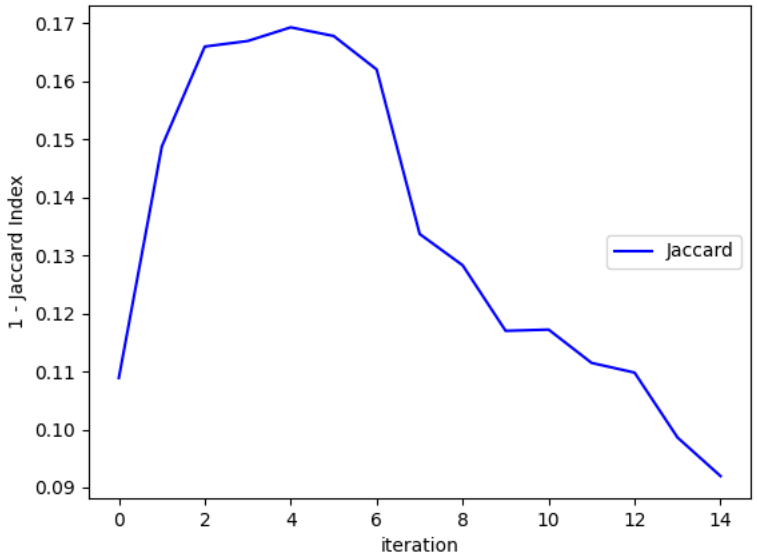



Figure 11: 1-Jaccard index

### 3.4.3 Third Evaluation

The following table shows the value of the error after a leave-one-out cross-validation performed on each image considered as test set.

| trial 1 | trial 2 | trial 3 | trial 4 | trial 5 | trial 6 | trial 7 | trial 8 | trial 9 | trial 10 |
|---|---|---|---|---|---|---|---|---|---|
| 0.0919 | 0.0838 | 0.1037 | 0.1081 | 0.0774 | 0.0731 | 0.1331 | 0.0920 | 0.0756 | 0.0895 |

| trial 11 | trial 12 | trial 13 | trial 14 | trial 15 | trial 16 | trial 17 | trial 18 | trial 19 | trial 20 |
|---|---|---|---|---|---|---|---|---|---|
| 0.0792 | 0.0868 | 0.1855 | 0.1337 | 0.1143 | 0.1239 | 0.0948 | 0.0840 | 0.0801 | 0.0877 |

Table 1: Final error for leave-one-out cross-validation, for each image as test set

### 3.4.4 Fourth Evaluation

The following table shows the values of error considering an increasing number of eigenmodes.

| num = 1 | num = 2 | num = 3 | num = 4 | num = 5 | num = 6 | num = 7 | num = 8 | num = 9 |
|---|---|---|---|---|---|---|---|---|
| 0.1448 | 0.1120 | 0.0917 | 0.0918 | 0.0959 | 0.0992 | 0.1002 | 0.1003 | 0.1003 |

| num = 10 | num = 11 | num = 12 | num = 13 | num =14 | num = 15 | num = 16 | num = 17 |
|---|---|---|---|---|---|---|---|
| 0.1004 | 0.1016 | 0.0994 | 0.0996 | 0.1019 | 0.1002 | 0.1016 | 0.1025 |

| num = 18 | num = 19 | num = 20 | num = 21 | num = 22 | num = 23 | num = 24 | num = 25 |
|---|---|---|---|---|---|---|---|
| 0.1001 | 0.1000 | 0.1000 | 0.1000 | 0.1000 | 0.1000 | 0.1000 | 0.1000 |

| num = 26 | num = 27 | num = 28 | num = 29 | num = 30 | num = 31 | num = 32 | num = 33 |
|---|---|---|---|---|---|---|---|
| 0.1000 | 0.0997 | 0.0996 | 0.0997 | 0.1029 | 0.1077 | 0.1046 | 0.1025 |

| num = 34 | num = 35 | num = 36 |
|---|---|---|
| 0.1056 | 0.1072 | 0.1070 |

Table 2: Error obtained by varying the number of considered eigenmodes

#### 3.4.5 Fifth Evaluation

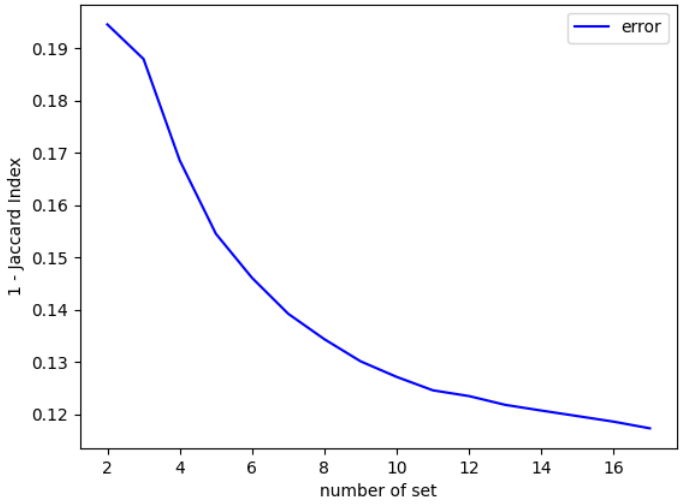



Figure 12: Jaccard error obtained by varying the number of considered sets of points

## 4 Discussion

### 4.1 Task 1

As it can be seen in Figure 7, the Generalized Procrustes algorithm is able to correctly align the set of points into a unique coordinate framework, creating a correspondence between the different sets, by choosing as initial points a random set, passed as input.

### 4.2 Task 2

From the Principal Component Analysis, it is possible to observe that of the 18 set of points trained on the first test image, the first 10 eigenmodes are able to explain the 97 % of the total variance. This means that these eigenvalues represent most of the variation in their specific directions.

### 4.3 Task 3

As it can be seen in Figure 8, from the Gradient Magnitude Analysis, it is possible to obtain mean patches for every point of the considered set. The patches represent a neighborhood of the image of dimension 15x15 around the specific point, from which we can update the point position towards the gradient direction. This is useful to search for a local match of each point along the profile.

### 4.4 Task 4

The implementation of ASM algorithm leads to the result shown in Figure 9. The Figure shows the original segmentation and the predicted one in the grey area. It is also possible to observe the position of the predicted points on the red line, which corresponds to the correct segmentation. From this, it is possible to compute an error index that considers the distance between the correct and the predicted points. In this case, after 15 iterations, the error converged to 0.0919, and this is reflected in the good overlap of the segmentation.

### 4.5 Task 5

#### 4.5.1 First Evaluation

In Figure 10, it is possible to observe the overlap of the correct and predicted points when the value of the first and second eigenvalues considered are alternatively equal to 50.

#### 4.5.2 Second Evaluation

In Figure 11, it is shown the value of the 1-Jaccard error considering only the first test image and how does this error varies increasing the number of iterations. What it can be seen is that the higher the number of iterations, the lower the error goes, arriving to a minimum of 0.0919.

#### 4.5.3 Third Evaluation

The Table 1 shows the values of the error for the leave-one-out cross-validation. The method considers as test set one image at a time, and as training set the rest of the images present. This means that at least once, every image is going to be the test set. The error for each trial is not significantly different in the different images, going from a minimal value of 0.0731, up to a maximum of 0.1855.

#### 4.5.4 Fourth Evaluation

In Table 2 are shown the values of the error when considering an increasing number of eigenmodes. What it can be seen is that after the 10th eigenmode, the error does not decrease significantly anymore, since most of the variance is explained in the first ones.

#### 4.5.5 Fifth Evaluation

In Figure 12 is shown the value of the error varying the considered number of set of points from 2 to 18. The more the considered set of points are, the less the error is, since the training is done on a bigger number of samples, thus resulting in a more efficient segmentation.

## 5 Conclusion

From the many experiments conducted, it is possible to conclude that segmentation methods highly depend on the initial conditions that are set. In fact, varying the image considered as test set and varying the number of set of points, the error in the segmentation decreases or increases. In fact, this is a non-convex problem, and this leads to a non-unique optimal solution. For this reason, segmentation problems must be faced with trial and error approach, varying the parameters considered depending on the issue that must be faced.
Also, the chosen segmentation model to be conducted depends on the problem that has to be solved, since simpler models such as the Eigenpatch could be equally effective than more computationally complex models such as the Active Shape one [2].

## References

[1] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[2] Jerry L.Prince and Jonathan M.Lins. *Medical Imaging Signal and Systems*. 2006.

[3] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.