# Computer Aided Diagnosis
# Lab 1: Image Segmentation (Active Shapes Models)

**Submitted By**
Basel Alayfi
Md. Kamrul Hasan
Fakrul Islam Tushar

**Submitted To**
Arnau Oliver, PhD
Xavier Llado, PhD
Kaisar Kushibar
{arnau.oliver,xavier.llado,kaisar.kushibar}@udg.edu

May 3, 2019

## Contents

# 1 Introduction and Problem Definition

Image Segmentation is one of the prior steps to most of the image analysis tasks, specially if it relates to medical imaging. Among many other approaches, statistical shape model for image segmentation is a common practice in computer vision society. In this course work, we get introduced to one of these statistical shape models called Active Shape Model (ASM). ASM was implemented from scratch for hand shapes using the provided data of hands having 40 examples with 56 landmarks of each. The crucial objectives of this project are as follows:

- To understand the concept of Active Shape Model (ASM) and related theories to implement ASM from scratch using the provided data.
- To understand and implement Procrustes Analysis for aligning, transforming; scaling and rotating, the different shapes to the reference shape.
- To apply Principle Component Analysis (PCA) for reducing the dimensions of the feature space.

# 2 Algorithm Analysis

The mathematics underlying for ASM implementation are fairly straightforward. Procrustes analysis and Principal components analysis (PCA) are used to align the shapes and to find the major axes of a cloud of points in a high dimensional space, respectively [1]. Algorithms used for this project implementation are described briefly as follow.

## 2.1 Shape Representation

Any shape can be represented by a judiciously-chosen set of points, aka features, each of which is a k-dimensional vector. Those N feature points are stacked into a long vector having length kxN. Mathematically, it can be written as follow by considering dimension k = 2 in Eq. (1).

$$x = \begin{bmatrix} p_1, & p_2, & p_3, & \dots & p_N \end{bmatrix}^T \tag{1}$$

$Where, p_i = (x_i, y_i) \quad for \quad i = 1,2,3,\dots,N$. If we have **M** training sample, we generate M such vectors as like Eq. (2).

$$X = \begin{bmatrix} x_1, & x_2, & x_3, & \dots & x_M \end{bmatrix} \tag{2}$$

Before staring with the active shape analysis, statistical analysis has to be performed on those vectors to bring them to same co-ordinate frame. The shape of an object is normally considered to be independent of the position, orientation and scale of that object [1].

## 2.2 Procrustes Analysis

Procrustes analysis is the series of statistical methods that is used for to analyze the distribution of a set of shapes. The algorithm that have been followed to implement Procrustes analysis is shown in Fig. 1 which is described briefly as follows:

1. **Translation and Normalization:** Translation brings the shapes to its centre of gravity that is at the origin (i.e. its centroid). Translation can be done by the following Eq. (3) and Eq. (4).

$$\bar{x} = \frac{x_1 + x_2 + x_3 + ..... + x_N}{N} \quad and \quad \bar{y} = \frac{y_1 + y_2 + y_3 + ..... + y_N}{N} \tag{3}$$

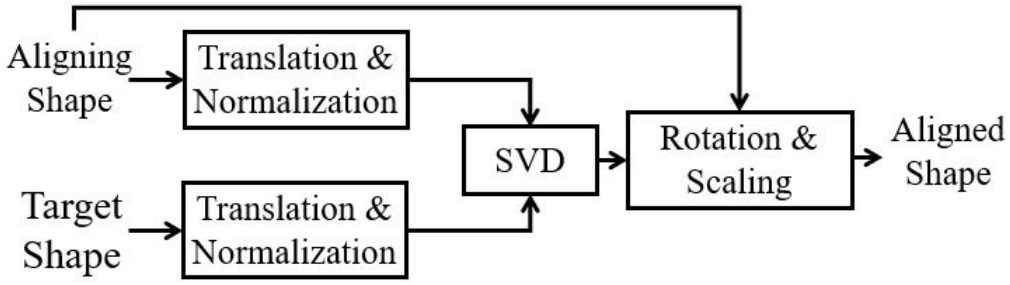



Figure 1: Block diagram for Procrustes analysis

$$(x_i, y_i) \rightarrow (x_i - \bar{x}, y_i - \bar{y}) \quad for \quad i = 1, 2, 3, ....., N \tag{4}$$

Normalize each shape so that it satisfies the Eq. (5).

$$|\bar{x}| = \sqrt{\bar{x_1}^2 + \bar{y_1}^2 + \bar{x_2}^2 + \bar{y_2}^2 + ..... + \bar{x_N}^2 + \bar{y_N}^2} = 1 \tag{5}$$

2. **Singular Value Decomposition:** After bringing centre of gravity to the origin and performing normalization, vector x can be expressed as follow.

$$x = U\Sigma V^T = \underbrace{\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_r\end{bmatrix}}_{\text{Col}, x} \underbrace{\begin{bmatrix}\mathbf{u}_{r+1} & \dots & \mathbf{u}_m\end{bmatrix}}_{\text{Nul}, x^T} \begin{bmatrix}\sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \dots & & & & & & \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & & & & & & \\ 0 & 0 & \dots & 0 & 0 & \dots & 0\end{bmatrix} \begin{bmatrix}\mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \dots \\ \mathbf{v}_r^T \\ \mathbf{v}_{r+1}^T \\ \dots \\ \mathbf{v}_n^T\end{bmatrix} \begin{matrix}\} \text{Row}, x \\ \\ \\ \\ \} \text{Nul}, x\end{matrix}$$

So, orthogonal or rotational matrix, R can be written as follows Eq. (6) from the Singular Value Decomposition (SVD) [2] of the vector x. As well as scaling will be as in Eq. (7).

$$R = UV^T \, (which\ means\ dot\ product\ of\ U\ and\ V^T) \tag{6}$$

$$Scale factor = \sum_{i=1}^{k} \Sigma(i);\ k\ is\ the\ dimension\ of\ each\ landmark. \tag{7}$$

3. **Shape Alignment:** After getting rotation and scaling parameters. The following process is applied on every shape in the data set, considering the mean of the shapes as the reference. Following mathematical operation Eq. (8) can be performed to get the aligned shape.

$$AlignedShape = (Shape \cdot R^T) * Scale factor \tag{8}$$

## 2.3 Principle Component Analysis (PCA)

The foremost reason for using PCA is to approximate the shape of any landmark using a small number of parameters. The steps to implement PCA are given in Algorithm 1.

**Algorithm 1** Steps for PCA implementation, part 1

***Step-1:*** Compute the mean of the feature vector using Eqs. (9, 10, 11).

$$\bar{x_j} = \frac{1}{M}\sum_{i=1}^{M} x_{ij}\ ;\ j = 1,2,3,\ldots N \tag{9}$$

$$\bar{y_j} = \frac{1}{M}\sum_{i=1}^{M} y_{ij}\ ;\ j = 1,2,3,\ldots N \tag{10}$$

$$\bar{p} = [\bar{x},\bar{y}]^T \in \Re^{2N} \tag{11}$$

***Step-2:*** Co-variance matrix can be calculated using Eq. (12).

$$\Sigma_{2Nx2N} = \frac{1}{M-1}\sum_{i=1}^{M}(p_i - \bar{p})(p_i - \bar{p})^T \quad ;\ i = 1,2,3,\ldots M. \tag{12}$$

***Step-3:*** Compute the eigen vectors $u_i$ and eigen values $\lambda_i$ of the covariance matrix. Sort the eigen values descendingly, then, reorder the eigen vectors correspondingly.

***Step-4:*** Choose the $t$ eigen vectors (columns of P) that correspond to the t largest eigenvalues that satisfy Eq. (13).

$$\frac{\sum_{i=1}^{t}\lambda_i}{\sum_{i=1}^{M}\lambda_i} \geq 0.98 \tag{13}$$

## 2.4 Active Shapes Model

The design was mainly inspired from the description found in [3]. The main algorithm followed was algorithm 2. Mathematically, $b$ can be written as $b = [b_1, b_2, \ldots, b_t]^T$. Varying the weights $b_i$ enables us to explore the allowable variations in the shape. To clarify the values used for basic parameters, table 1 is given.

| Parameter | Value |
|---|---|
| max_iterations | 140 |
| norm_range | 13 |
| N (number of landmarks per shape) | 56 |
| M (number of shapes) | 40 |

Table 1: Parameters definition

Where max iterations was enough for almost all the shapes to converge to a good solution. norm range was tested for different values, and 13 was a decent trade off between detecting nearest edges and preventing overlapping with neighbouring points.

**Algorithm 2** Core protocol

1: apply Procrustes on all shapes
2: apply PCA on the aligned shapes, algorithm 1
3: initialize using algorithm 3 to get initial $s, \theta, t_x, t_y$
4: **for** max_iterations **do**
5: deform the model to match image edges using algorithm 4
6: **end for**

**Algorithm 3** Initialization algorithm

draw the model with initial scale, theta, and translation parameters
use the mouse wheel to scale up/down the model
use the left mouse click and drag to translate the model
use mouse right drag to control theta.

**Algorithm 4** Matching model points to target points

1: initialize the shape parameters to zeros, i.e., $b = [0 \dots 0]^T$. Initial shape will be only the mean one ($\bar{x}$).
2:

$$x = \bar{x} + P\,b \tag{14}$$

3: transform x into X in the image space (using initial transformation parameters for the first iteration)
4: find image points, Y, using algorithm 5
5: find the pose parameters $(t_x, t_y, s, \theta)$ which best align the model points x to the current points Y using algorithm 6.
6: project Y onto the model co-ordinate frame by inverting the transformation T using Eq. (15).

$$y = T^{-1}_{t_x,t_y,s,\theta}(Y) = T_{-t_x,-t_y,1/s,-\theta}(Y) \tag{15}$$

7: ***Step-5:*** Update the model parameters to match $y$ using Eq. (16), with constraints on $b$.

$$b = P^T(y - \bar{x}); \quad -3 \times \sqrt{\lambda_i} \leq b_i \leq 3 \times \sqrt{\lambda_i}, \quad 1 \leq i \leq t \tag{16}$$

**Algorithm 5** Finding maximum edges along the norms

1: compute sobel gradient magnitude of gray image
2: **for** all model points, $p_i$ **do**
3: compute the normalized tangent at $x_i$

$$\hat{p}_i = \frac{p_{i+1} - p_{i-1}}{|p_{i+1} - p_{i-1}|}; \hat{p}_i = [\hat{x}_i, \hat{y}_i] \in \Re^{2N}$$

4: compute points coordinates along the norm

$$nx_i = x_i - k\hat{y}_i$$

$$ny_i = y_i + k\hat{x}_i$$

$$\forall k \in \{-norm\_range, \dots, 0, \dots, norm\_range\}$$

$$n_i = [nx_i, ny_i] \in \Re^{2(2*norm\_range+1)}$$

5: **end for**
6: find image coordinates of maximum gradient along each norm

$$Y_i = \underset{x,y}{argmax}(grad[ny_i, nx_i]); i = 1, 2, \dots, N$$

**Algorithm 6** aligning two shapes

1: **for** both shapes, $x_i$ **do**
2: centralize the shape

$$\hat{x} = x - \bar{x};\ \bar{x} = [mean(x_1, x_2, \ldots, x_N), mean(y_1, y_2, \ldots, y_N)] \in \Re^2$$

3: **end for**
4: $a = (\hat{x}_{mov} \cdot \hat{x}_{fix})/|\hat{x}_{mov}|$
5:

$$b = \sum_{i=1}^{n} (\mathrm{x}_{x_{mov}} * \mathrm{y}_{x_{fixed}} - \mathrm{y}_{x_{mov}} * \mathrm{x}_{x_{fixed}})$$

6: $scale = \sqrt{a^2 + b^2}$
7: $\theta = \arctan(b/a)$

# 3 Results Analysis

In this section, five results are shown with the parameters used on the initialization pictures themselves to make them reproducible. Use tips are available as well on the figure to make it easy to use them. All and additional images are avaialable in images folder attached. This algorithm is highly sensitive to initialization, fairly-far initial positions may result in completely unexpected results.

## 3.1 Little deformations, case image 0001

In this case, pretty good initial positions were possible to achieve. Figure 2 shows the initial pose. For reproducing the results, the used parameters are shown at the top of the figure. Use instructions are just under the image. Figure 3b and Figure 3a show relatively good results with fingers being matched. Only a few iterations (less than 40) were enough to converge.

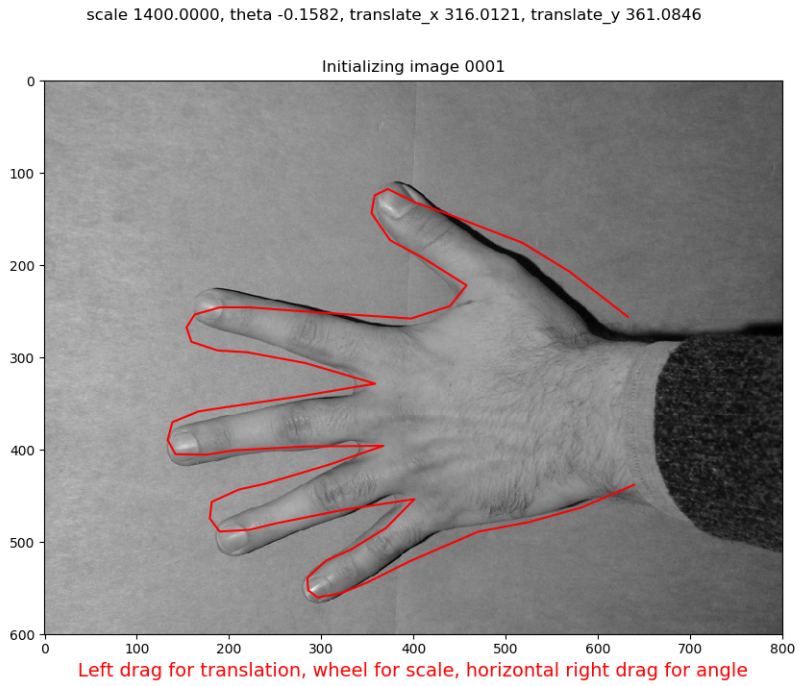
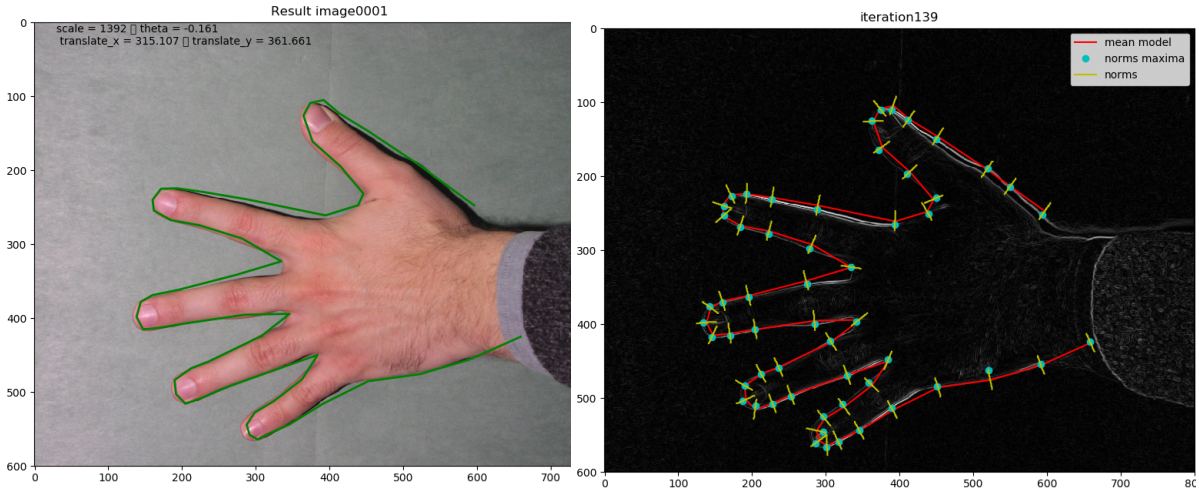



Figure 2: Initial positions for image 0001




(a) deformed model (green line) imposed over the image 0001

(b) Last iteration for image 0001

Figure 3: last iteration (right) and result (left) for image 0001

## 3.2 Medium deformation, case image 0000

In this case, the hand shape was fairly similar to the model (after scaling, translating and rotating), see Figure 4 below. To detect edges, sobel filter was used as in Figure 5b. It is obvious that edges are not perfect due to noise and illumination differences. Additionally, in Figure 5a, the norms could detect the edges of the nails, and no farther due to the range norm used. The resulted parameters are shown on top of the figure.

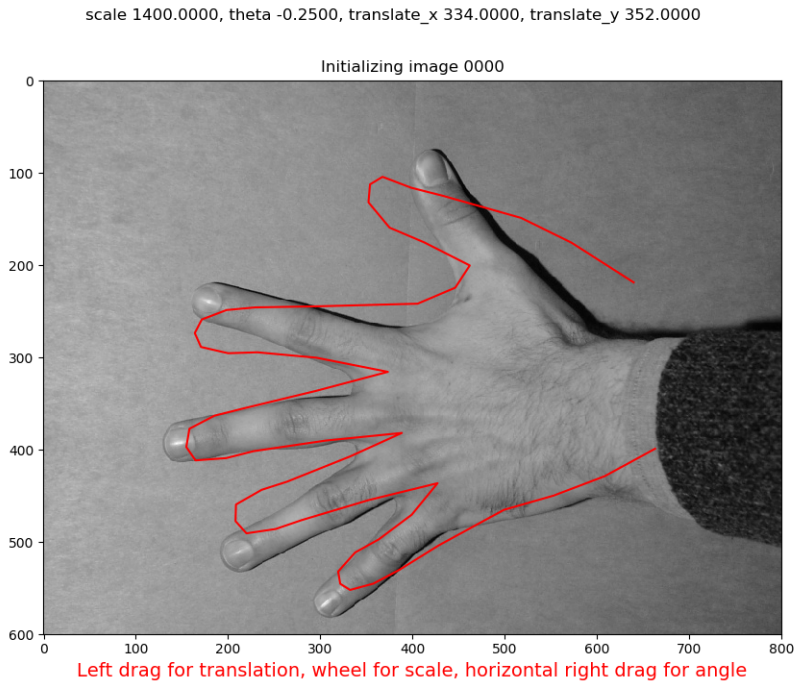
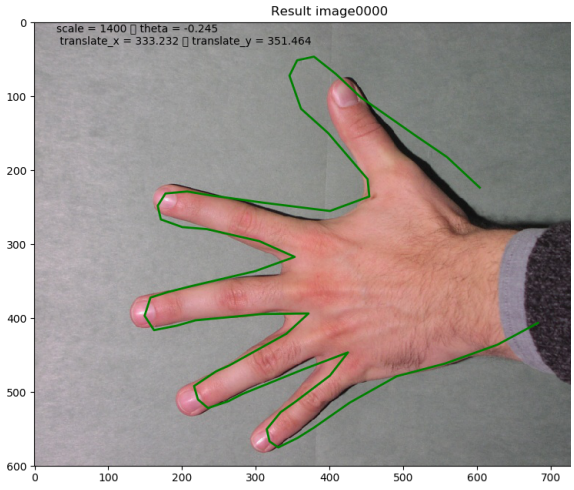
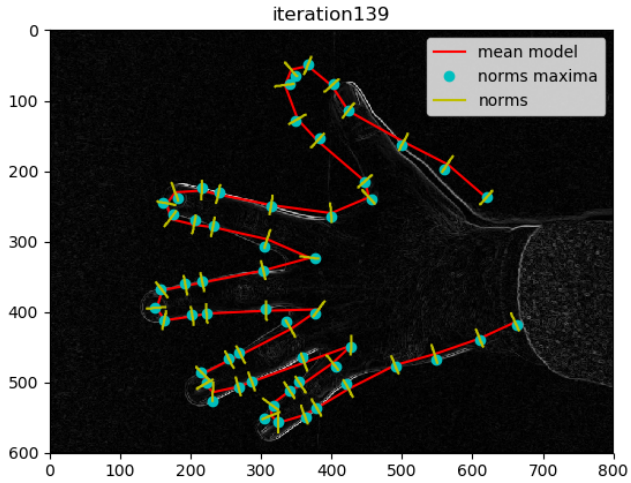



Figure 4: Initialization for image 0000




(a) deformed model (green line) imposed over the image 0000



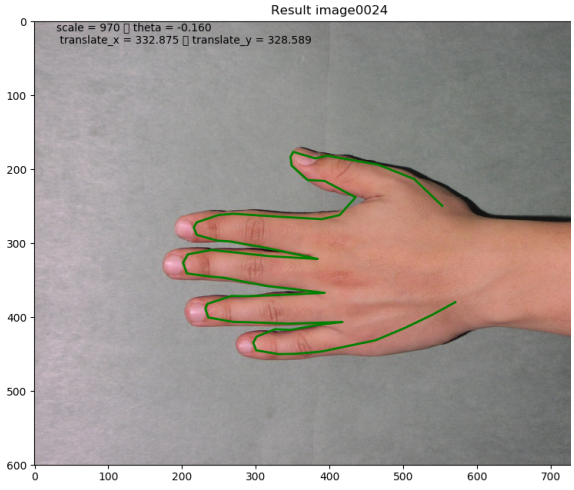

(b) Last iteration for image 0000

Figure 5: last iteration (right) and result (left) for image 0000

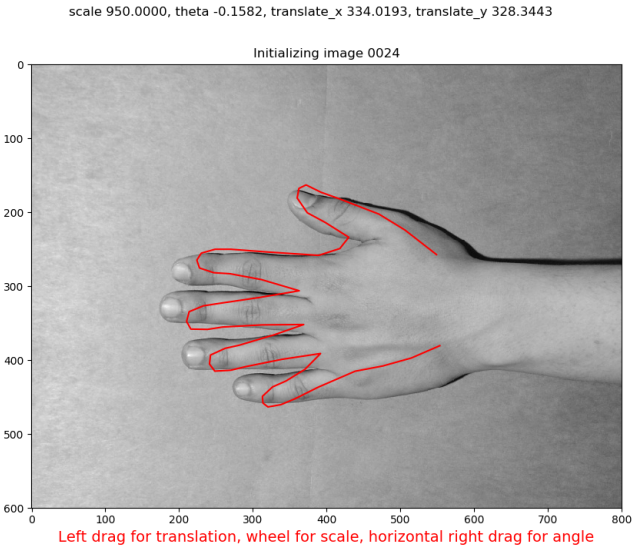
The thumb was quite far in the initial pose which caused the mismatch in the result for that finger.

## 3.3 Medium deformations, case 0024

In this case, fingers were almost closed with quite little distances among them. The initialization was a careful process to achieve acceptable results. Figure 6b and Figure 6a show the initial pose and the final result respectively.




(a) deformed model (green line) imposed over the image 0024

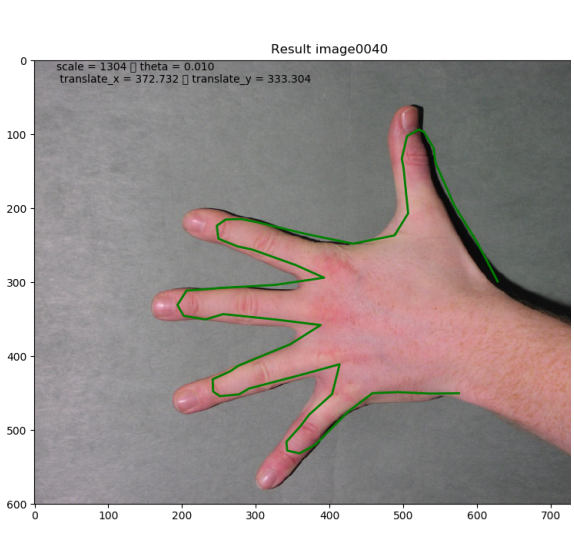



(b) Last iteration for image 0024

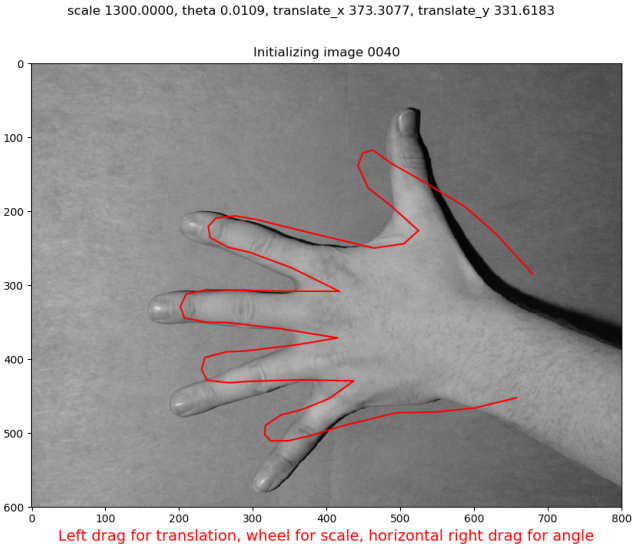
Figure 6: last iteration (right) and result (left) for image 0024

## 3.4 Large deformations, case 0040

In this case, the hand was opened almost completely, fingers were far from each other. The initialization process had to be really precise to get acceptable results as the case of image 0000. Figure 7b shows the initial pose, while Figure 7a shows the final result. The algorithm was able to catch fingers edges but not the tips due to nails and skin edges.




(a) deformed model (green line) imposed over the image 0040

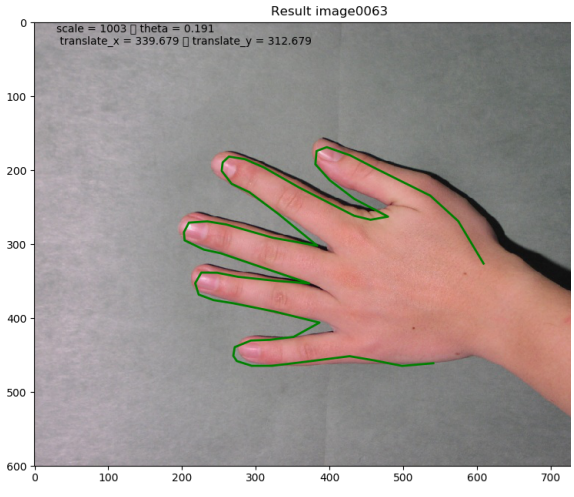



(b) Last iteration for image 0040

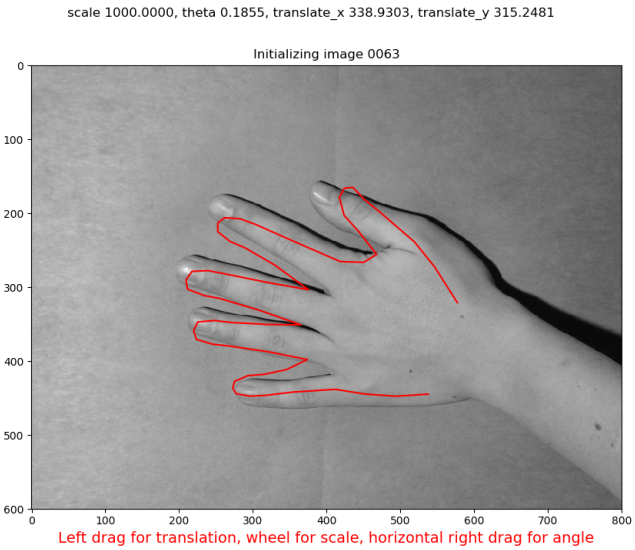
Figure 7: last iteration (right) and result (left) for image 0040

## 3.5 Larg deformations, case 0063

In this case, the hand was fairly different from the model. Figure 8b and Figure 8a show the initial pose and the results respectively.




(a) deformed model (green line) imposed over the image 0063




(b) Last iteration for image 0063

Figure 8: last iteration (right) and result (left) for image 0063

# 4 Conclusion and Future work

So far in this report, different results for different cases were shown. Fingers were detected in most cases and the technique worked properly. The main difficulty was in adjusting good initialization positions which was hardly ever possible in some cases. What can be done to improve the performance and ease the process is to utilize multi-resolution ASM which starts with a small version of the image and propagates the parameters from one layer to the next. In that case the quality will be enhanced significantly.

# A How To Use

To test the code, it is highly recommended to use PyCharm for that, however, to test Procrustes, it is suggested to use Spyder. All that due to issues related to `MAtplotlib` library.

## A.1 Testing active_shapes script

To test active shapes algorithm with different images and initializations, a few things need to be modified, see Figure 9

```
27  # the initial state of the initialization
28  norm_range  = 13
29  max_iter    = 140
30  image_name  = "hand/images/0001.jpg"
31  scale       = 1400;
32  theta       = -0.1582
33  translate_x = 316
34  translate_y = 361
```

Figure 9: code lines may need to be modified in order to change default settings

In order to run a different image, the name of the image has to be modified. Scale, theta, translate_x and translate_y can be controlled through the figure that comes up at the beginning of the run, see Figure 2 for example. Figure 10 tells the story of controlling the scale, theta, translation parameters in a convenient way.

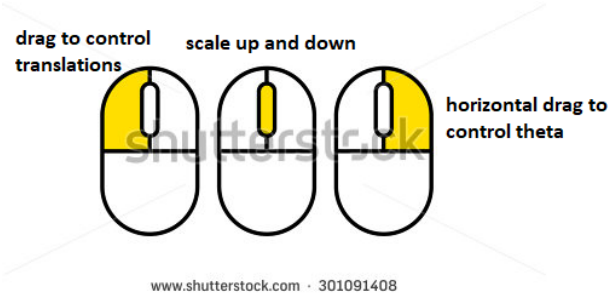



Figure 10: Initialization manual

# References

[1] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active Shape Models-Their Training and Application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38-59, 1995.

[2] Lecture on *Singular Value Decomposition* by Justin Solomon. CS 205A: Mathematical Methods for Robotics, Vision, and Graphics. Avaiable at: `graphics.stanford.edu/courses/cs205a-13-fall/assets/lecture_slides/svd.pdf`

[3] Tim Cootes, "Introduction to Active Shape Models ".