UNIVERSITY OF CALGARY

Department of Electrical and Computer Engineering

ENEL 619.76

Biometric Technologies and Systems Design

FACIAL FEATURE EXTRACTION USING ACTIVE APPEARANCE MODEL

Cristobal Martinez

December 15, 2016

Instructor: Dr. S. Yanushkevich

INTRODUCTION

The task for landmark extraction or detection it's an important step in almost all facial software applications like face detection, face matching, facial expression recognition between others. In this project we explore the use of an Active Appearance Model (AAM) for finding this landmarks in the human face.

As a way to know how accurate this approach is, the main objective is to find the error between the landmarks or features found by a human user and the ones found with the use of the AAM.

On a secondary objective, I explore the use and reliability of other software frameworks other than Matlab. Matlab is known as one of the best solutions for scientists and engineers to analyze and design systems and products. One of the problems of this ecosystem, even when it's tremendously powerful, you are somehow tied to the use of a desktop computer. If I want to move my project from the computer to a different hardware solution is not a straight forward job. The approach explored in this project will let me do all the design on a more portable hardware like a raspberry pi or even a cell phone.

The framework found for this project is called Menpo (http://www.menpo.org/) and is Python based. I will describe the basics of this framework later on.

FEATURE EXTRACTION - LANDMARKS

The features or landmarks are the main properties or information extracted from a facial picture or image and is used in tasks like facial expression recognition, face detection or face matching. It's important to note that the number of landmarks that an algorithm could find in a face is not fixed and it depends on the application or on the designer itself.

The most common numbers are 10, 35, 47, 68 and 122. For this project I only use 68 points: 19 for the face contour, 6 per eye, 9 for the nose, 5 for each eyebrow, and 19 for the mouth.

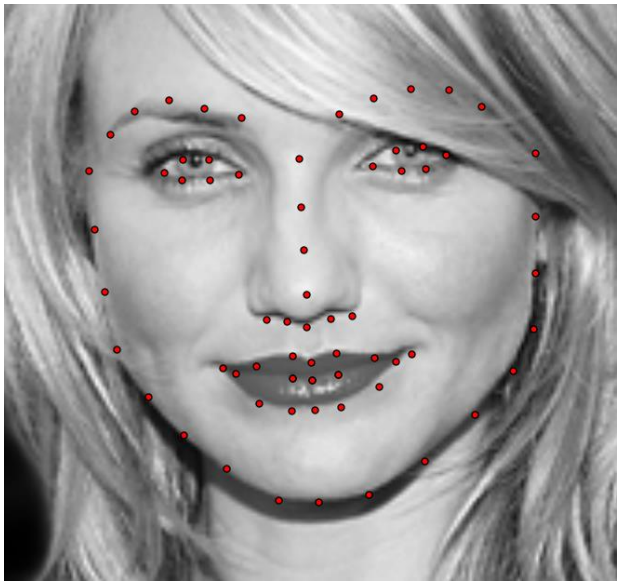

FIGURE 1. EXAMPLE OF A 68 POINT ANNOTATED IMAGE.

MENPO PROJECT

FIGURE 2. MENPO PROJECT LOGO

Before I get into the Active Appearance Model I want to talk about the Menpo Project. Students and professors at the Department of Computing on the Imperial College London, UK are constantly updating this tool. This project is a Python framework for deformable modeling. Since it is built on Python, is highly portable and easy to learn. It includes training and fitting code for various state of the art methods such as Active Appearance Model and Lucas-Kanade. At its core, the menpo package makes importing, manipulating and visualizing images and mesh data as simple as possible and particularly well suited for working with annotated or landmarked data.

ACTIVE APPEARANCE MODEL

The first time AAM was used was in 1998 by G.J. Edwards, C.J. Taylor and T.F. Cootes with the task of interpreting face images. They described the AAM as a statistical model of shape and grey-level appearance which can generalize almost any face. Basically is a computer vision algorithm used to match a model with another images.

This model, like many others, is built around a training phase in which it takes a set of pre-annotated images that are used to train the model. These pre-annotated images are nothing more that pictures that have the landmarks already found, usually by hand, by another human. You can find several free databases of pre-annotated images; in this project I use the Labeled Face Parts in the Wild (LFPW) dataset by the Intelligent Behavior Understanding Group (http://ibug.doc.ic.ac.uk).

FIGURE 3. DEFORMABLE OBJECTS

I would also like to note that an AAM can be created out of any deformable object like could be the human body or a cat face.

In the lines below I will briefly describe the basic mathematical definition of an AAM.

A deformable object is represented as $s = [x1, y1, \dots, xL, yL]^T$ a 2L x 1 vector that has L landmark coordinates(x_i, y_i), ∀=1 , … , L. An AAM is trained with a set of N images $\{I_1, I_2, \dots, I_N\}$ that have L landmarks that consist of the following parts:

Shape Model

The training shapes $\{s_1, s_2, \dots, s_N\}$ are aligned using Generalized Procrustes Analysis and then an orthonormal basis is created using Principal Component Analysis (PCA) augmented with four similarity transform eigenvectors. This results in $\{\bar{s}, U_s\}$ where $U_s \in R^{2L\,x\,n}$is the orthonormal basis of n eigenvectors and $\bar{s} \in R^{2L\,x\,1}$ is the mean shape vector. A new shape instance can be generated as $S_p = \bar{s} + U_s p$ where $p = [p1, p2, \dots, pn]^T$ is the vector of shape parameters.

Motion Model

Consists of the warp function $W(p)$ which is essential for warping the texture related to a shape instance generated with parameters **p** into a common reference. The reference shape is the mean shape $\bar{s}$.

Appearance Model

It's trained by:

- Extracting features from all the training images using the features function $F()$.
- Warping the feature-based images into the reference shape to get $F(I_i)\big(W(p_i)\big), \forall i = 1, \dots, N$.
- Build vectors out of the warped images as $F(I_i)\big(W(p_i)\big), \forall i = 1, \dots, N$ where $a_i \in R^{M\,x\,1}$.
- Applying PCA on the acquired vectors which results in $\{\bar{a}, U_a\}$. Where $U_a \in R^{M\,x\,m}$ is the orthogonal basis of m eigenvectors and $\bar{a} \in R^{M\,x\,1}$ is the mean appearance vector.

A new appearance instance can be generated by $a_c = \bar{a} + U_a c$ where $c = [c1, \dots c_m]^T$ is the vector of appearance parameters.

Warp Function

We can define the warp function as $t(W(p)) \equiv F(I)(W(p))$ as the feature based warped $M\ x\ 1$ vector of an image **I** given its shape instance generated with parameters p. Menpo project offers five different AAM version which only differ on the way that the appearance warping is performed which are:

- Holistic
- Masked
- Linear
- Linear Masked
- Patched

It is good to know that you can choose from five different options, but since for this project I only explored the Holistic appearance warping, that's the only method I will discuss.

Holistic: Uses a holistic appearance representation, obtained by warping the texture into the reference frame with a non-linear warp function $W(p)$. Menpo also provides two supported warp functions for this: Piecewise Affine Warp and Thin Plate Spline.

Cost Function and Optimization

The fitting process, which is the actual step where the algorithm will find the landmarks on a test image, involves the optimization cost function $\arg min \left\| t\big(W(p)\big) - \bar{a} - U_a\, c \right\|^2$ with respect to the shape and appearance parameters. An AAM aims to align the test image with a linear appearance model.

This optimization can be solved by two approaches: Lucas-Kanade Optimization and Supervised Descent Optimization. For this project only the Lucas-Kanade was explores but its important to note that we have another option available.

Lucas-Kanade Optimization

This method belongs to the family of gradient-descent algorithms. The existing gradient-descent techniques are categorized as: **Forward or Inverse** depending on the direction of the motion parameters estimation and **Additive or compositional** depending on the way the motion parameters are updated. Menpo provides Forward-Compositional and Inverse-Compositional of five algorithms. All this algorithms are iterative and the shape parameters are updated at

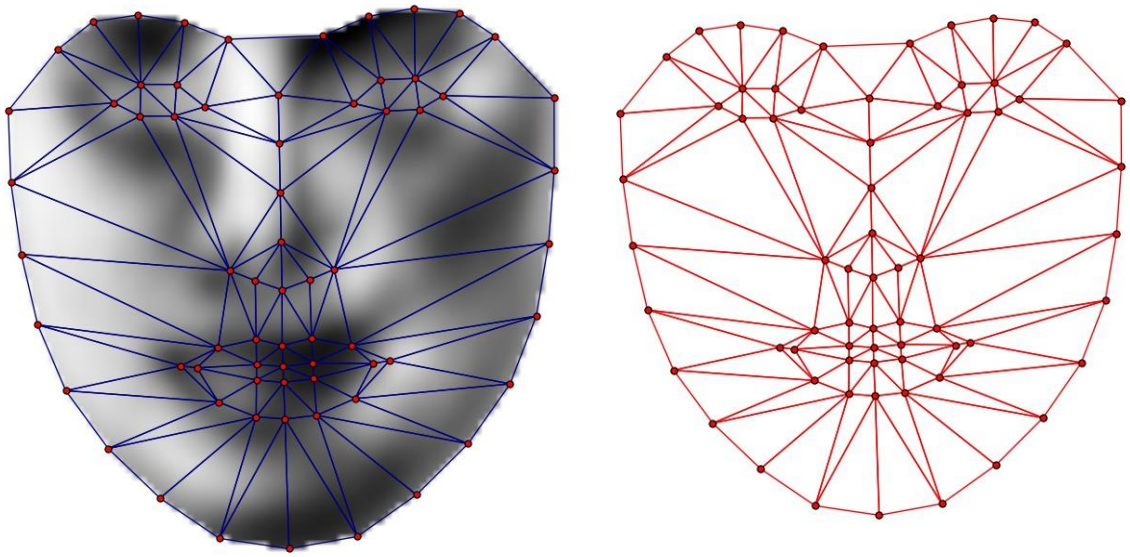

FIGURE 4. EXAMPLE OF APPEARANCE AND SHAPE MODEL. WE CAN VISUALLY SEE THE MODEL USING MENPO PROJECT WIDGETS.

each iteration in a compositional manner as $W(p) \leftarrow W(p) \cdot W(\Delta p)^{-1}$. The five different algorithms that can be used are:

- Project-Out
- Simultaneous
- Alternating
- Modified Alternating
- Wiberg

For this project only the Wiberg algorithm was explored, but is important to note that we have 4 more options to implement the optimization.

Wiberg algorithm

This algorithm is a very efficient version of alternating optimization. It involves solving two different problems in an alternating manner, one for the shape and one for the appearance parameters increments

$$\begin{cases} \boldsymbol{arg}\ \min_{\Delta p} \left\| \boldsymbol{t}\big(\boldsymbol{W}(\boldsymbol{p})\big) - \overline{\boldsymbol{a}}\big(\boldsymbol{W}(\Delta \boldsymbol{p})\big) - \sum_{i=1}^{m} \boldsymbol{c_i u_i}(\boldsymbol{W}(\Delta \boldsymbol{p})) \right\|^2_{\widehat{U}_a} \\ \boldsymbol{arg}\ \min_{\Delta p} \left\| \boldsymbol{t}\big(\boldsymbol{W}(\boldsymbol{p})\big) - \overline{\boldsymbol{a}}\big(\boldsymbol{W}(\Delta \boldsymbol{p})\big) - \sum_{i=1}^{m} (\boldsymbol{c_i} + \Delta \boldsymbol{c_i}) \boldsymbol{u_i}(\boldsymbol{W}(\Delta \boldsymbol{p})) \right\|^2 \end{cases}$$

where $\widehat{\boldsymbol{U}}_{\boldsymbol{a}} = \boldsymbol{I}_{\boldsymbol{eye}} - \boldsymbol{U}_{\boldsymbol{a}} \boldsymbol{U}_{\boldsymbol{a}}^{\boldsymbol{T}}$ is the project out operator. Given the current estimate of $\Delta \boldsymbol{c}$ the shape parameters increment estimated by solving the first optimization problem as $\Delta p = \widehat{H}^{-1} \hat{J}_a^T \big[t\big(W(p)\big) - \overline{a} \big]$ where $\hat{J}_a = \widehat{U}_a J_a$ is the projected out Jacobian with $J_a = \nabla \overline{a} \frac{\partial W}{\partial p}\Big|_{p=0} + \sum_{i=1}^{m} c_i \nabla u_i \frac{\partial W}{\partial p}\Big|_{p=0}$ and $\widehat{H} = \hat{J}_a^T \hat{J}_a$ is the Gaussian-newton approximation of the Hessian. Given the current estimate of $(\Delta \boldsymbol{p})$, the appearance parameters increment is computed by solving the second optimization problem as

$$\Delta c = U_a^T\left[t\left(W(p)\right) - \bar{a} - U_a c - J_a \Delta p\right]$$

MANUALLY ANNOTATING

For me to compute the error between the landmarks found by the model and what a human could find, I had to manually annotate my test set. The test set was a small database of 32 images that I annotate with another menpo tool called landmarker.io. Landmarker.io is an online tool where you can easily pick the landmarks by eye. Even when this processes its straight forward, it was very time consuming and it took several hours.

FIGURE 5. TOOL USED FOR MANUAL ANNOTATION.

RESULTS

The setup for using the menpo project was very straightforward. They already have a lot of sample code on github and step-by-step guides on how to install all the tools of the framework in their website. They also have a forum where some of the members answer questions to the users. I even contacted one of them by email and he helped me very politely.

At the end of this project I got very miscellaneous results. Some of the pictures got a very good fitting and the landmarks were placed almost perfectly. In other cases the landmarks were very off of where they should be. And also I had items that some landmarks (like the eyes) were very off but others (like the mouth) perfectly placed.

In the end I encountered a great problem; the error between my human-annotated pictures and the one the algorithm found was always large. Even when the algorithm found almost all the landmarks perfectly, with each iteration the error became bigger and bigger. I found that the manual annotation that I had done was not done correctly. I placed the landmarks manually in a way that felt natural to me. I had an image to guide me, but it wasn't by far a step-by-step guide. As a result, the error found for all of the images in the test set never decreased and only got bigger on every iteration.

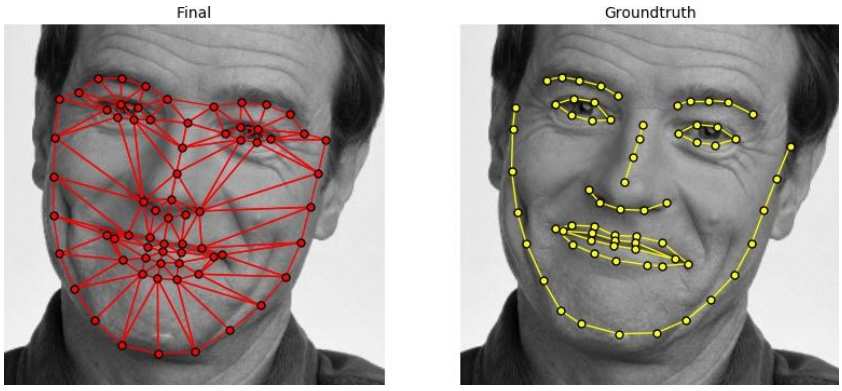

FIGURE 6. EXAMPLE OF A GOOD FITTING RESULT FOUND BY THE ALGORITHM

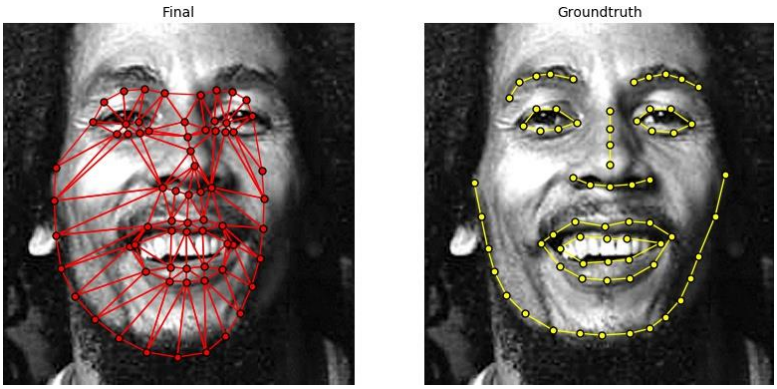

FIGURE 7. EXAMPLE OF A BAD FITTING FOUND BY THE ALGORITHM

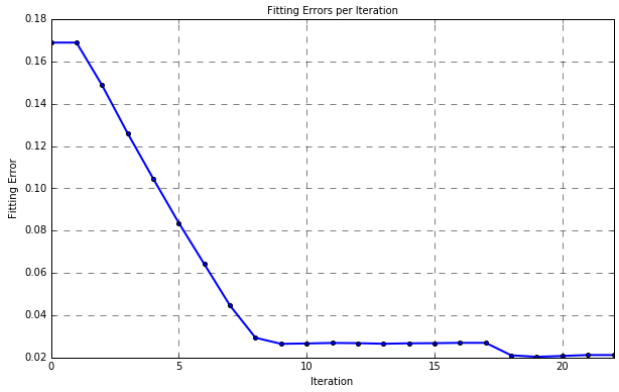

FIGURE 8. EXAMPLE OF A SUCCESFUL FIT WHERE THE ERROR DECREASES.

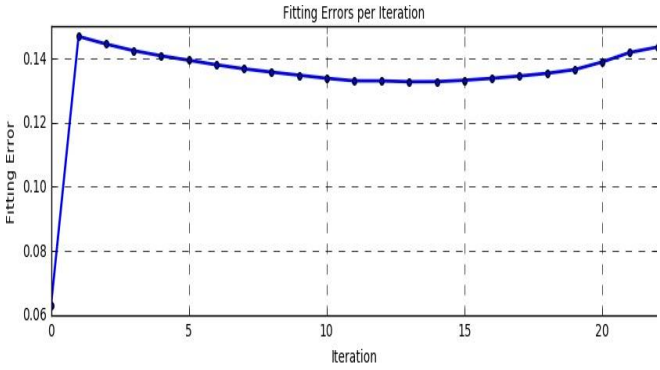

FIGURE 9. THE ERRORS FOR ALL MY CASES LOOKED LIKE THIS. THIS GRAPH BELONGS TO ONE OF THE BEST FITTINGS.

CONCLUSIONS

On one hand I think the objective of using another approach other than Matlab was a success. I used an open source platform based on python and successfully trained an Active Appearance Model which I used to fit my sample data set, which I manually annotated, and used a fitting algorithm to find the face landmarks. On the other hand, the error found between my human-annotated images and the landmarks found by the algorithm never decreased even when the landmarks found were fitted correctly. I found that the cause of this was that I did annotate incorrectly my dataset. I couldn't find anywhere how would be the best way to manually annotate the 68 points on the face.

One thing that could be calculated is the total error. This error would be how many images were correctly annotated by the algorithm versus how many failed. The user could easily find this just by looking at the final image annotation. One downside for this error is that it carries very little information.

Finally, there are still some options that have to be explored. Menpo project offers another four types of warp functions and another four algorithms for solving the optimization problem. Exploring this other options, with better manual annotation, could end up on finding better error results. Also, the framework has another four options that could be used instead of Active Appearance model that still could be explored. Of course, the student could also use the menpo project just as a basic tool and develop his/her own solutions for problems other than landmark localization.

REFERENCES

- Interpreting Face Images using Active Appearance Models - G.J. Edwards, C.J. Taylor and T.F. Cootes. 1998.
- Face Tracking for Model-based Coding and Face Animation - J. Ahlberg, R. Forchheimer. 2003.
- The Menpo Project - http://www.menpo.org/
- Fundamentals of Biometric System Design – S.N. Yanushkevich
- Intelligent Behavior Understanding Group - http://ibug.doc.ic.ac.uk/
- https://en.wikipedia.org/wiki/Active_appearance_model
- J. Alabort-i-Medina*, E. Antonakos*, J. Booth*, P. Snape*, and S. Zafeiriou. (* Joint first authorship) Menpo: A Comprehensive Platform for Parametric Image Alignment and Visual Deformable Models, In Proceedings of the ACM International Conference on Multimedia, MM '14, New York, NY, USA, pp. 679-682, 2014. ACM.
- G. Papandreou, and P. Maragos. "Adaptive and constrained algorithms for inverse compositional active appearance model fitting", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.
- G. Tzimiropoulos, and M. Pantic. "Gauss-Newton Deformable Part Models for Face Alignment In-the-Wild", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
- G. Tzimiropoulos, M. Pantic. "Optimization problems for fast AAM fitting in-the-wild", IEEE International Conference on Computer Vision (ICCV), 2013.