2013

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

AN OPEN SOURCE EXTENSIBLE AUTOMATIC LANDMARKING SYSTEM USING DYNAMIC ACTIVE SHAPE MODELS (DASM)

David Daniel Macurak

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems & Operations Management

University of North Carolina Wilmington

2013

Approved by

Advisory Committee

Devon Simmonds

Jeffrey Cummings

Karl Ricanek, Jr.

Chair

Accepted by

Dean, Graduate School

TABLE OF CONTENTS

Abstract

An Open Source Extensible Automatic Landmarking System Using Dynamic Active Shape Models (DASM). David Daniel Macurak, 2013. Thesis Paper, University of North Carolina Wilmington.

This thesis proposes a modern implementation of an application designed to perform automatic landmarking based on Active Shape Models. The software is designed to locate and precisely identify the features of an object, based on its known structure. In this particular study, the process of extracting features from frontal facial images is explored. The goal of which is to achieve accurate landmarks which can be applied to further processing techniques such as facial, expression, or gender recognition, etc. All of these techniques are reliant upon accurate annotations of features in order to produce meaningful results. Manually locating these landmarks can be a tedious and slow process, which is why automatic techniques such as the one discussed in this paper have been developed. Based on a comparison of similar automatic techniques, the open source implementation published in this work shows promising results, with many facets for future improvements.

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ASM. Active Shape Model

STASM. Stacked Active Shape Model

PDM. Point Distribution Model

AAM. Active Appearance Model

CLM. Constrained Local Model

FEM. Finite Element Method

GPA. Generalized Procrustes Analysis

Chapter 1: Introduction

This thesis introduces an open source project involving the implementation of an automatic landmarking tool. The tool, entitled Dynamic Active Shape Models, or DASM, is a C++ implementation of Cootes et al. [14] Active Shape Model (ASM) framework. More specifically, the proposed project is based on an extension to ASM by Stephen Milborrow [35] entitled Stacked Active Shape Models (STASM). The problem that the tool seeks to solve is a means of accurately identifying the positions of a set of landmarks which identify key features of a human face. However, the modularity of the software design enables extensions to other types of modalities. Such extensions have been described extensively in literature. This work investigates the effectiveness of automatic landmarking on 2D images of frontal facing neutral expression faces, more generally termed "mugshots". Example images are provided in Figure 1.1. A similar open source project proposed by Mikkel Stegmann uses Active Appearance Models (AAM) to interpret various types of medical imagery, and can be found here [51].

Figure 1.1: Sample images from the MORPH dataset

1.1 Overview

The following sections provide a survey on shapes and landmarks, automatic landmarking, and the difficulties associated with automated techniques. Following that is an

introduction to statistical shape modeling and an outline of the goals of this work.

1.2 Shapes and Landmarks

We must first begin by giving a good definition of the term shape, as it is the basis on which this work is grounded. In itself, shape is a very general term for describing what an object looks like, based soley on its geometric features. In the terms of this project, a more suitable definition can be adapted from D.G. Kendall [44]: "*Shape* is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object." Essentially, the author is stating that the shape of an object remains constant regardless of any linear transformation that it may undergo. This is illustrated in Figure 1.2.

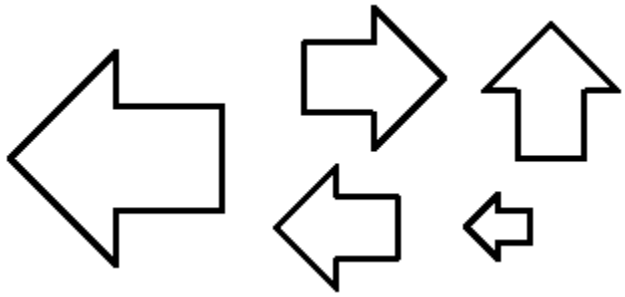


Figure 1.2: The arrows retain the same shape regardless of rotation, translation, or size

The capacity to describe and interpret the shape of an object provides us with a means of recognizing the identity of an object. There is an immediate appeal to seek to mirror this instinctive ability of humans to computer vision. To accomplish this, researchers have developed a means of algorithmically interpreting shape through a collection of *landmarks*. From [44], "a landmark is a point of correspondence on each object that matches between and within populations." The authors [44] further define three basic types of landmarks: anatomical, mathematical, and pseudo. Anatomical landmarks identify physical traits or characteristics that exist between every sample in a population. In the context of human faces, anatomical landmarks define features such as the lips, eyes, nose, etc. In greater detail, they can be anthropometric features such as the alare of the nose or endocanthion and exocanthion of the eye. Mathematical landmarks define specific areas of a

sample which have some geometrical significance. These are typically features which can be visually identified with ease by an abrupt shift in pixel intensity, also known as a gradient change. The skin creases in the crows feet or along the forehead as well as the outline of the jaw are good examples of regions of the face that cause significant changes in gradient. Finally, pseudo landmarks represent constructed points that help further define the structure of the object. From Figure 1.3 the following points/landmarks illustrate the three landmark types:

1. Anatomical: nose tip and eye centers, points 112, 180, 153
2. Mathematical: corners of the eyes and lips, points 170, 175, 206, 216
3. Pseudo: points between anatomical landmarks 200, 199, 198, 197

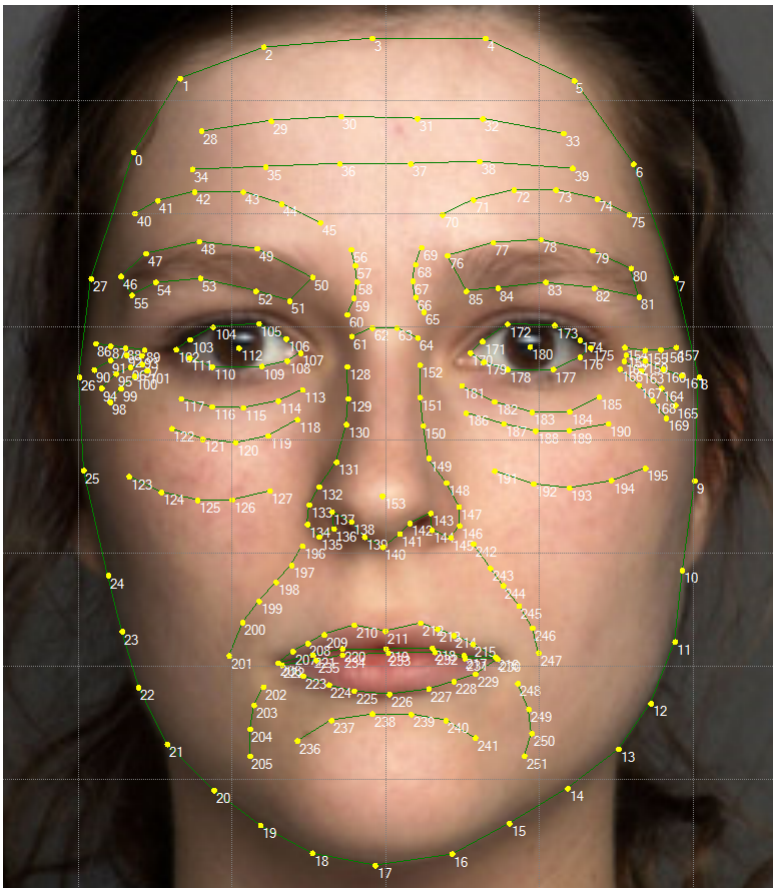


Figure 1.3: Face with landmarks

In some instances, landmark types may change as a result of age. For example, nasal labial creases exist on many faces, but not all, i.e. younger faces. These creases represent mathematical landmarks on faces where they exist, and pseudo on faces where they don't exist.

We now have a universal means of describing the shape of a two-dimensional object through a collection of landmarks. As the nature of this work primarily relates to applications based on facial analytics, the objects being described will reflect human faces. Landmarks are visually represented by annotation points on an object, as shown in Figure 1.3. It is not uncommon for landmarks to also be referred to as anchor points, fiducial markers, or key points.

The edges between annotations are used to help identify connecting feature *parts*, but have no further statistical significance. Parts provide a convenient means of grouping a set of landmarks. A facial feature, such as the nose, mouth, or eye can be defined by a single part or collection of parts, as illustrated in Figure 1.4. The component scheme adopted in this work, for instance, contains thirty-seven unique parts. This information is used to boost the performance of the landmark search process, as will be discussed later. Ultimately, landmarks provide an efficient means of characterizing a face in a reduced dimensionality space based upon the interrelations of the landmarks, such as the distances between them. Effectively, we are using a collection of landmarks to model the unique shape of individuals' faces. This enables applications to make recognition decisions such as verification or identification based solely on landmark comparisons [48],[49]. For these types of applications, it is critical to have precise locations of the landmarks to ensure correct classifications.

1.3 Automatic Landmarking

Accurate registration of feature points on faces in digital images is a critical practice which precedes many face-based automatic systems, e.g. face recognition, facial analytics, or anthropometric analysis. Typically, this is a manual process performed by either a trained individual or team with expert domain knowledge. Landmarks which have been

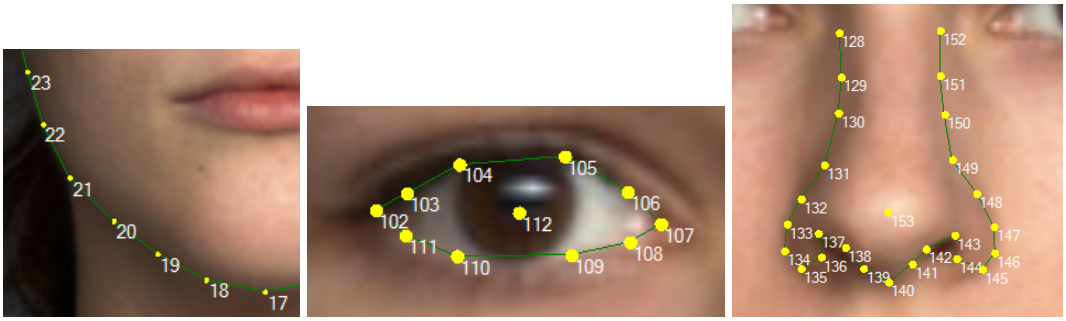


Figure 1.4: Various parts of the face

hand labeled by experts are considered ground truth data. It is not uncommon for ground truth landmarks to be included with facial datasets for the purposes of research and testing. Unfortunately, this data is often limited to a very few number of key point landmarks such as the eye centers, tip of the nose, or corners of the mouth. This is often sufficient information for applications such as face tracking, which only require the presence of easily identifiable facial features within a video frame capture. However, this work focuses on automatic registration of extremely dense landmark schemes, for instance over two hundred landmarks.

Dense landmark schemes are particularly appealing for applications involving face-based biometrics, such as age estimation, race and gender classification, expression and micro gesture recognition, etc. These tasks often require meticulous knowledge of the facial features in order to make accurate classifications. Intuitively, the greater the number of landmarks, the more detailed knowledge we have of the subject. Applications based on statistical modeling techniques often require hundreds or thousands of images to be manually annotated before they can utilize the feature information. This is often a very tedious and time-consuming process, especially when considering a dense landmark scheme. When compiling ground truth data, it is imperative that landmarks are labeled exactly to the specifications of the scheme across the entire training set. Failure to maintain consistency of landmarks results in inaccuracies and mis-classifications when applied to further processing as shown in [46]. Due to the immense time consumption of manually labeling very dense landmark schemes, it is obvious to seek out a method of automating the process.

1.4 Difficulties

Automatic landmarking of dense schemes is not a trivial task however, as even the best performing algorithms cannot compete with the accuracy of hand labeled landmarks [46]. Performance of landmarking algorithms degrades as a result of unconstrained "in the wild" images where pose, illumination, and expression are uncontrolled. Further degradation can come about as a result of image artifacts from low-quality or scanned images, or occlusions such as facial hair or eye-glasses. Each of these issues inhibits the algorithms ability to extract meaningful information from the image. Examples of poor quality image samples are shown in Figure 1.5. As many applications use unconstrained images gathered from Internet sources such as Facebook or Google's Picassa, it is critical to develop an automatic landmarking tool which not only produces the most accurate landmarks, but is robust to these complications.

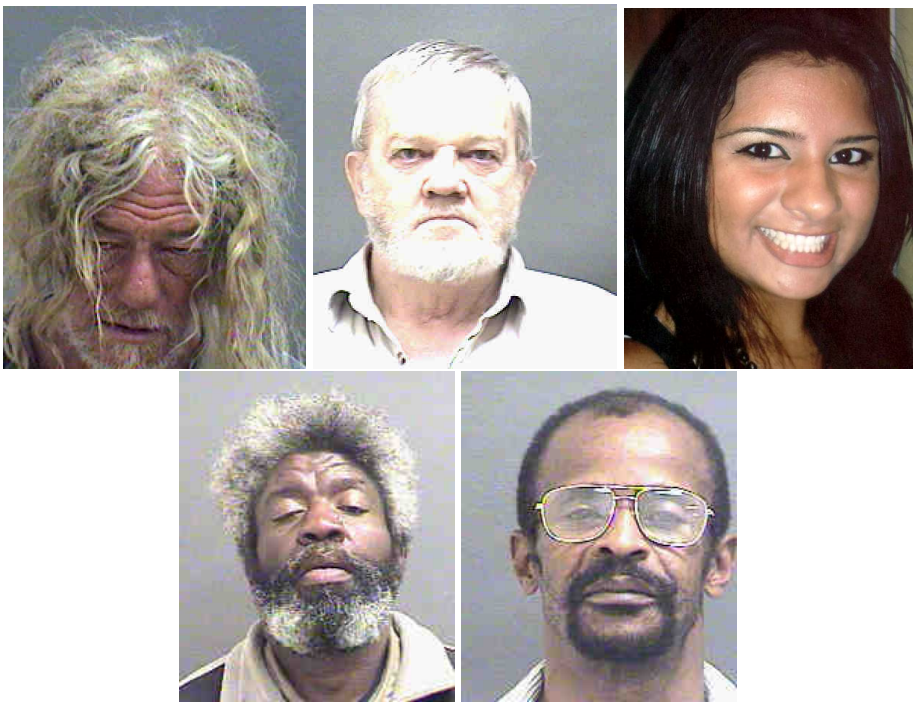


Figure 1.5: Poor image conditions due to off-angle pose, poor illumination, expression, and occlusions

1.5 Measuring performance

Sethuram et al. [47] define a number of protocols for measuring the performance

of landmarking algorithms. The study demonstrates the performance of three well-known landmarking techniques: CLM, STASM, and AAM, evaluated over a testing set with a dense landmarking scheme. I make use of the same baseline performance metrics to illustrate the performance of the tool developed in this work to the aforementioned algorithms. This work also considers computational complexity as a factor of performance. This information will be discussed in the Results chapter.

1.6 Statistical Shape Modeling

Statistical shape modeling techniques have played a key role in many computer vision and image processing applications. We can make use of such algorithms for interpreting geometric shape information of anatomical structures. Such techniques have commonly been applied to biomedical imagery, where internal organs can be automatically registered based solely on their shape. A study by Eviatar [17] introduced an approach to model features of the brain from magnetic resonance images using a type of statistical shape model: Active Contours. In the terms of the problem being addressed in this work, statistical modeling is utilized to interpret the geometric features of the face. We can define the shape of each feature of the face by a set of landmarks, which can then be automatically located and labeled by a model search. A shape model is typically constructed by analyzing and learning from a set of ground truth data. The more training data learned by a model, the greater the variation in shape of the object the model is able to interpret. As faces tend to vary according to age, gender, race, etc., increasing the amount of samples and incorporating a variety of demographic groups to training data provides a means of accurately capturing all of the typical geometric shape deformations related to the *general* anatomy of the face.

We identify shape models which are capable of locating features regardless of geometric shape variations as deformable or flexible models. Another well-known type of statistical shape model is a rigid model. Contrary to deformable models, rigid models seek to identify how well an object's features match to the model, whereas deformable models actively seek to conform to the shape of the object. A rigid model could, for example, be used to analyze objects passing along a factory production line. In this scenario, each

widget must retain the exact shape characteristics of the model, else it will be discarded. Further examples of rigid modeling techniques are in [9, 21]. For the reasons stated above, rigid models would not serve to solve the problem of automatically identifying facial feature landmarks.

The inspiration for this project was derived from the work proposed by Milborrow [35] called *Stacked Active Shape Models*. STASM is a well known and highly regarded software application specifically focused on automatically locating landmarks on human faces. The purpose of the tool proposed in this work, DASM, is to improve upon the software design of STASM through the inclusion of modern software engineering paradigms. Simultaneously, DASM seeks to maintain and in some cases improve upon the level of computational performance and accuracy of automatic landmarking. Both tools implement a well known and proven statistical shape modeling algorithm called *Active Shape Models*, originally proposed by Cootes et al. in [14]. The ASM algorithm defines a means of automatically 'fitting' landmarks to specific object features by performing an image search from a learned model. STASM and ASMs will be described in greater detail in the following chapter.

1.7 Locating Facial Features

DASM, like its predecessor, is designed with the primary notion of modeling and automatically landmarking features on human faces. Facial analytics is a rapidly growing domain of study in computer science as a result of its integration into the security and intelligence communities. Many biometric and soft biometric research studies explore the use of faces to ascertain useful information based on an individual; such as identity, age, gender, etc. Face recognition is an example of a well studied area of biometrics which has been applied to applications ranging from terrorist watch lists to commercial advertising. Tools such as STASM and DASM assist these applications by automatically locating landmarks of facial features. Once the facial features have been located, they can then be extracted and processed by other algorithms. Final conclusions can be drawn from the resulting classifications on the extracted features.

I have chosen DASM to be an open source project for the reason that the potential uses for a tool such as this reach far beyond the designs proposed in this work. As an open source tool, researchers in the community have the ability to collaborate and adapt the project to meet their own needs. For this reason, I have attempted to develop the tool to be as extensible and user-friendly to the best of my knowledge and skills. Furthermore, I feel that the tool only has the potential to benefit from continued support of the open source and research communities.

Ultimately, the goal of this work is to:

1. Create open source application which implements Active Shape Models and extends the work of STASM
2. Provide an object-oriented structure for managing modular extensions to the code
3. Incorporate parallelization to better suit the availability of multi-core processing
4. Utilize well supported open-source libraries to support computing tasks and platform independence
5. Evaluate the performance of the tool against competitive algorithms
6. Draw conclusions based on my findings

In the remainder of the paper, I will give an overview of the modeling techniques which lead up to Active Shape Models, describe some of the enhancements to ASM contributed by other researchers, step through the ASM methodology, define metrics for evaluation of similar algorithms, review the contributions of this work, display the results of my experiments, and state my conclusions.

Chapter 2: Background

This section provides some basic background information on image registration methods which gave rise to Active Shape Models. A more robust survey is provided in [5]. Next, I introduce ASMs and describe the original implementation in detail. Following that, I give a brief discussion of *Active Appearance Models* (AAM) and how they are related to ASMs. Finally, I discuss the extensions to ASMs provided by Stephen Milborrow's work in [35].

2.1 History

Computer vision tasks such as edge or corner detection, template matching, object recognition, and segmentation have a rich history of research and application. Methods of automatically locating features fall under the broad category of image segmentation. In particular, model-based vision algorithms, like ASMs, have shown to be excellent approaches to recognizing and locating known objects within an image.

There are a host of methods which have provided the foundation for Active Shape Models, the foremost being template matching algorithms. *Template matching* is a technique in digital image processing for finding small parts of an image which match a rigid exemplar [6]. For example, a very simple straightforward method of template matching is correlation filtering a mask or template that we wish to detect, across an entire image in the hopes of locating a specific feature. A major drawback of this type of template matching is that any changes in image scale or rotation, or variations to the original template features, decreases the chances of acquiring a match.

In an attempt to address the shortcomings of standard templates, researchers developed template models capable of deforming to the features of an object. *Active Contour Models* (ACM) was one of the earliest and most popular forms of deformable template modeling. Introduced by Kass et al. in [27], Active Contours or "snakes" represent energy-minimizing flexible splines which attach to the contours of an object. They are active in the sense that they continually deform and slither until they come to rest in a final position, hence the name snakes. The total energy of the system in conjunction with external con-

straint guides the snake to the nearest feature. The features are defined as the edges within boundaries surrounding an object. In this way, each snake attempts to minimize its energy function, with its final resting place representing a local minimum. They require some human intervention in terms of assigning the snake a starting location from which to begin its search. Furthermore, constraints must be placed on the internal energy of the spline to ensure that the snake does not take on an unnatural shape. An example would be putting a penalty on the snake taking on too much curvature. For instance, imagine the contour smoothness parameters have been set to search for rectangular objects and the snake has latched onto a picture of a box in an image. The smoothness parameters must enforce that as the snake conforms to the boundaries of the box, it latches to the contours in a rigid, non-curving fashion.

Advancement to ACMs came about as a result of some common problems which the original implementation was subjected. The method often displays slow convergence speeds and instability in noisy environments. Another pitfall lies in the potential for multiple local minima for which the snake can come to rest. Without some detailed knowledge of the organization of the structure, the system cannot solve the problem of accurately conforming to all feature contours without user interaction. Kass provides an example of a slice of wood in which multiple line-contours resolve to different local minima producing differing results within the same set of wood grains. Nevertheless, snakes have shown to be extremely useful in a number of applications, especially in medical imagery, feature tracking [24], object recognition [56], and stereo matching [32]. Snakes have also been used in earlier face recognition work by Ricanek [42].

Parametric B-splines, or B-snakes, and subsequent enhancements [4, 18], solved many of the original snake design flaws. Other researchers have described deformable modeling techniques using *Finite Element Method* [26]. Similar to snakes, FEM is based on an energy minimization problem. The algorithm overcomes many of the shortcomings of Euler-Lagrange Theory, on which the method is based. The method suffers from the same issues as snakes, in that there is no automatic way of determining the initial positions

of the nodes without direct user involvement. Furthermore, the internal and external forces which guide the splines are unique to the problem at hand, and must be hand-crafted for each application. Terzopoulos and Metaxas [54] propose a method of dynamic 3D modeling which incorporates the flexibility of spline models with the constraining effects of parameterized models in the form of a physical model. The local and global deformations of their "superquadrics" are based on the geometry, physical dynamics, and elasticity of a specific model. Unfortunately, their method is limited to modeling closed shapes and also requires a deep understanding of physical motion and mechanics. Furthermore, the user must provide initial estimates of the translation, rotation, and global and local constraints to the model.

It is evident from the techniques discussed so far that there is a common issue shared by each method. Each requires human interaction to determine the initial estimates of position and correct interpretation of the structure of the objects they seek to model. As best stated by Timothy Cootes [10], "The problem with existing methods is that they sacrifice model specificity in order to accommodate variability, thereby compromising robustness during image interpretation." Essentially, the models are not specific to any one object, meaning that they are capable of deforming in ways outside of the class of objects they are attempting to describe. In [13], Cootes and Taylor give a more thorough explanation for why they felt there was a need for a better modeling technique.

2.2 Active Shape Models

As a result, Cootes composed a body of research dedicated to Active Shape Models. Milborrow refers to this vast collection of literature as the *classical* ASM, spread across [14, 13, 15, 10, 28, 29]. ASMs intoduced a method of building object-specific models which have the power to capture the natural variability of the class of objects they define. The models are taught all of the possible deviations in shape through a collection of training samples. Once this is established, the model is capable of constraining the potential shape based on the learned material, rather than some user-defined global or local constrains as in ACMs. This is what separates ASMs from other methods and makes it such a powerful

modeling technique.

Active Shape Models are capable of interpreting any type of object given a few constraints. First, a model can define only one class of object. Secondly, the choice of object to be modeled cannot have an amorphous or completely unpredictable shape. The construction of the object can vary such that it remains the same general shape. However, the structure cannot vary so much that it takes on a different shape. Intuitively, we can derive that the effectiveness of the model search process is directly correlated to how well the examples in question match to the model.

As this work is concerned with both the ASM algorithm as well as extensions to the original implementation of Active Shape Models, an overview of the methodology is provided. A more comprehensive survey is provided by Milborrow in his master's thesis, available in [34]. The actual implementation of the techniques to be covered will be detailed in the Methodology chapter.

2.2.1 Modeling Faces

As mentioned earlier, a model requires a set of training samples from which to learn all of the typical shape variations of a specific class of objects (in our case, faces). Faces are capable of being modeled due to their simple structure. However, due to their highly variable construction, they exhibit a high degree of variation across each facial feature. Case in point, no two individuals' faces are exactly the same; even amongst identical twins [39, 40]. The shape of the face varies across individuals as a result of the gender, age, race, presence of facial hair, piercings, etc., unique to each person. Altogether, these personal characteristics incorporate an inimitable influence on the shape of the face. Yet, the general shape of a face is consistent across the population. Excepting some rare cases, all faces share the same anatomical properties; two eyes, two ears, a nose, a mouth, etc., which define the human face.

2.2.2 Overview

The first step involves acquiring a set of training data, from which each of the images must be manually labeled to obtain ground truth data. The models described require

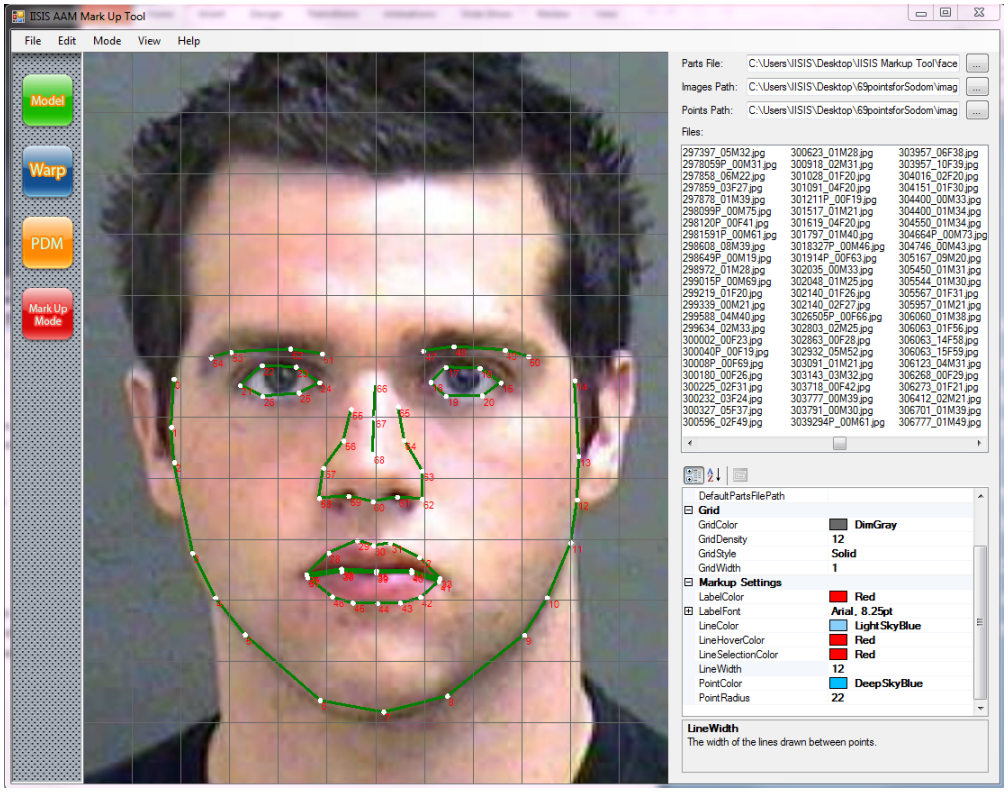


Figure 2.6: A screenshot of the landmark tool used in this work [1]

a user to be able to annotate the points on each of a set of training images in such a way that each point represents a distinguishable feature present on every example [10]. This statement puts forth two assertions: first, the features being defined must be present on every sample in the training set, and second, points must correspond across each shape. Basically, if point 1 defines the tip of the nose, that particular point must be consistent for every other sample.

Determining what features to label and how many landmarks to use is often dependent on the application. In [13], Cootes suggests labeling the features of the object which have the most discriminative information in regards to the application. He describes labeling points which define areas of high curvature, equally spaced along boundaries, and application-dependent points such as the eye centers. Milborrow [35] found that the inclusion of more landmarks to a system resulted in higher recognition performance. Logically, complex shapes with a significant degree of variation can be better defined by a greater amount of landmarks. Obtaining accurate ground truth data is a very important step, as the

model is only as accurate as the information on which it was trained. A more comprehensive description of labeling shapes is available through Stegmann [50].

2.2.3 Shapes and Alignment

Active Shape Models are characterized by the use of two sub-models: the shape model and profile model. The shape model, also referred to as the Point Distribution Model (PDM), is a composition of all of the shapes extracted from the ground truth data and aligned to some average shape. As mentioned before, a shape is quantitatively represented in terms of a set of landmarks which are defined by some points scheme. Each landmark is expressed by (x, y) coordinates in the two-dimensional space of an image. It is convenient to represent a shape $\boldsymbol{x}$ in terms of a vector of 2D points:

$$\boldsymbol{x} = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]^T \tag{2.1}$$

The process of aligning shapes is accomplished through a procedure called *Generalized Procrustes Analysis* (GPA) [52]. The algorithm involves the following iterative steps:

1. Translate each sample so that its center of gravity is at the image origin.
2. Assign an initial reference shape at random from the training set to be the mean shape.
3. Align all shape instances to the reference shape.
4. Compute a new mean shape from the aligned shapes.
5. Compute the Procrustes distance from the aligned shapes.
6. If the distance is less than some threshold, quit. Else, set the reference shape to the mean shape and return to step 3.

Appendix C from [16] provides the original ASM solutions to the problem of finding the optimal parameters to align shapes. Translation to the origin is accomplished by first finding the center of gravity of each of the samples:

$$(\bar{x}, \bar{y}) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i, \frac{1}{n} \sum_{i=1}^{n} y_i \right) \tag{2.2}$$

The Procrustes distance between two shapes is the Euclidean between corresponding points:

$$D = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^2} \tag{2.3}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are two shape vectors. With GPA, the approach is more concerned with the sum of the distances of all the training shapes to the reference shape. Mathematically, this can be shown as:

$$P_d^2 = \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})^2 \tag{2.4}$$

where P_d^2 is the squared distance, given n training samples. The square root of this value gives the total Procrustes distance of the system. The approach to the alignment process is to transform each shape by applying translation, rotation, and scaling parameters (a, b, t_x, t_y) to minimize the distance to the reference shape. Together, these three operations are characterized as a similarity transformation and their parameters can be computed by a simple linear equation[1]. The similarity transformation T proposed by Cootes from equation C.3, ensures that integrity of the actual shape has not been altered:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.5}$$

After each shape has been aligned to the reference shape, a new mean shape is calculated by extracting the mean point positions of the aligned shapes. The system is considered converged when some minimum distance threshold has been reached. The threshold value is often dependent on the scale of the shapes. If not converged, the new mean shape becomes the reference shape and the process is repeated. Figure 2.7 illustrates the results of aligning shapes during the training phase.

[1] Affine transformations such as shearing are not used because they corrupt the structure of the shape.

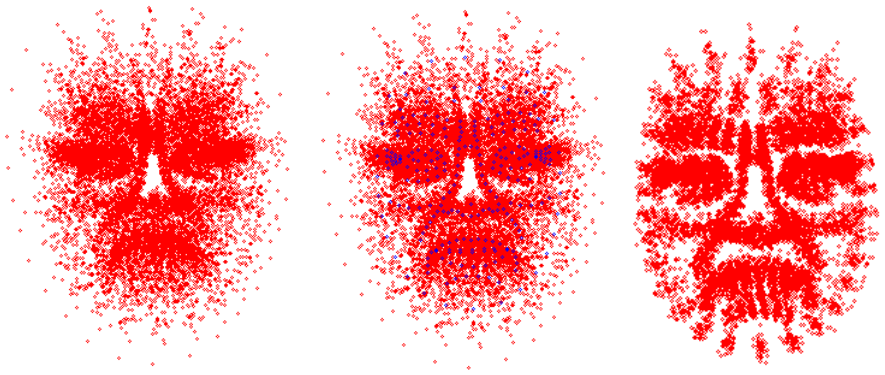


Figure 2.7: Left: the point cloud translated to the origin, Center: the computed mean shape in blue, Right: the post shape alignment from 1000 images

2.2.4 Modeling Shape Variation

Now that the shapes have been aligned to a common coordinate frame, the next step involves modeling the distributions of the shapes within the point cloud. Describing all of the variation present in the point distribution makes it possible to generate new examples, similar to those in the original training set. Chapter 4 of [16] describes the following procedures in greater detail. By enforcing constraints on certain model parameters, we can ensure that the newly generated samples are plausible shapes. Algorithmically, this is approached by seeking a parameterized model of the form $\boldsymbol{x} = M(b)$, where b is a vector of parameters of the model, and $\boldsymbol{x}$ is a generated instance from the model. Once the parameters of b have been found, it is possible to vary them so that the generated shapes are similar to model shapes.

An effective means of characterizing the distribution of shape variation is by performing a Principal Components Analysis (PCA) [37] on the data. The PCA is computed across the main axis of the point cloud, such that each shape of n points represents a single data point in the $2n$ dimensional space. The result is a set of a principal components which are ranked based on the amount of variance discovered in the data. The variance itself is represented by the eigenvectors $\mathbf{\Phi}$ of a covariance matrix $\mathbf{S}$. Each eigenvector ϕ_i is associated with an eigenvalue λ_i; sorted in such a way that the greatest eigenvalue reflects the first principal component, and so on. The value of the eigenvalue λ_i is equal to the variance

along the axis corresponding to the i^{th} eigenvalue. It is safe to assume that small-scale variation in the data, represented by the smallest eigenvalues, is noise which can be discarded. Hence, it is possible to model the main modes of variation with less than the $2n$ dimensions of the original data. The approach from [16] is as follows:

1. Compute the average shape from the aligned data,

$$\bar{\boldsymbol{x}} = \frac{1}{n_{shapes}} \sum_{i=1}^{n_{shapes}} \boldsymbol{x}_i \tag{2.6}$$

2. Compute the covariance of the data,

$$\mathbf{S} = \frac{1}{n_{shapes} - 1} \sum_{i=1}^{n_{shapes}} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \tag{2.7}$$

3. Finally, compute the eigenvectors and eigenvalues of $\mathbf{S}$.

It is now possible to approximate any shape, $\boldsymbol{x}$ from the training set by varying the parameters of b,

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + \boldsymbol{\Phi b} \tag{2.8}$$

In order to ensure a plausible shape, it is suggested to apply limits of $\pm 3\sqrt{\lambda_i}$ to the $\boldsymbol{b}$ vector. We can limit the number of eigenvectors to retain, such that the model represents some proportion of the total variance described by the data. Because the points have been aligned through multiple iterations of similarity transformations, the data is highly correlated. This means that a small percentage of the top principal components is able to describe a high percentage of the total variance in the model. The remaining components are assumed to be residual noise. Alternatively, if the data is highly uncorrelated, a greater percentage of principal components would be needed to satisfy the majority of variation in the system. It is possible to calculate the number of modes t which are characterized by some percentage p of the data by,

i	λ_i
1	1324.4387
2	1088.9042
3	531.57056
4	483.98535
5	459.80508
6	358.1214
7	299.00217
8	267.72964
9	238.89328
10	185.86554
11	167.22571
12	145.95769
13	126.96124
14	111.66627
15	105.29704
16	102.14168
...	

Table 2.1: Ranked eigenvalues

$$t = \frac{p}{100} \sum_{i=1}^{2n} \lambda_i \tag{2.9}$$

For example, performing a PCA on a training set of 1150 aligned shapes containing 252 points produced the following ordered eigenvalues (truncated to save space) in table 2.1. From the table, it is clear to see that the variance related to the eigenvalues drops off significantly after the first few entries. In this case the sum of the eigenvalues (including those not shown) is 8298.75. Using equation 2.9, if we want to keep 98% of the information, we would only need to retain 136 of the total 504 entries.

2.2.5 Conforming to the Model

Alternatively, it is possible to conform a new shape instance $\boldsymbol{x}$ to the shape model using the same calculations above. Given a model mean shape $\bar{\boldsymbol{x}}$ and a new shape instance $\boldsymbol{x}$, solve for the b parameters and the pose parameters T which minimize the sum of square error distances between corresponding model and image points.

$$|\boldsymbol{x} - T(\bar{\boldsymbol{x}} + \boldsymbol{\Phi} b)|^2 \tag{2.10}$$

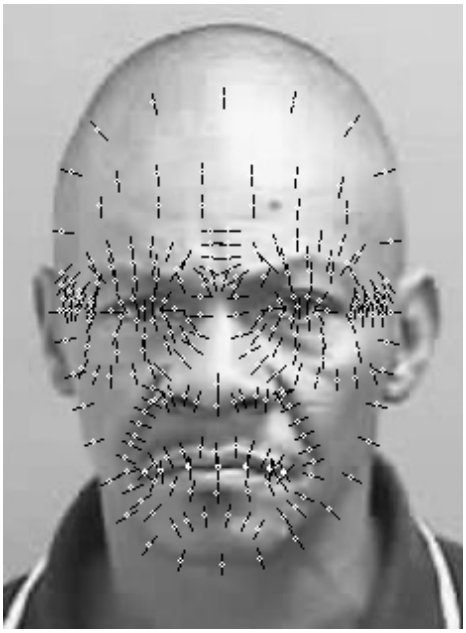


Figure 2.8: One-dimensional profiles of length 9 pixels surrounding each landmark

The iterative approach to this solution is given by Cootes and Taylor in section 4.8 of [16]. This method ensures that any suggested point movements computed by the profile model during the image search process, still produce a plausible shape by applying constraints to the values of the b parameters.

2.2.6 Modeling Gray-level Texture

At this point we have covered how to ensure generated shapes are similar to the model, but not how to search an image for the features which match to the model. This portion of the ASM is given by the profile model. As opposed to the shape model, the profile model is concerned with direct pixel information extracted from an image, local to each landmark. A profile model is formed by a collection of one-dimensional vectors, one for each landmark in the system. Milborrow [35] refers to each profile vector as a *whisker*. A whisker is created by passing a raster line through the landmark, perpendicular to the boundary formed by the landmark's two direct neighbors. The (x, y) coordinates of each landmark represent the center position of the profile, as illustrated in Figure 2.8.

As the section heading suggests, the profile model is only concerned with gray-level intensity values; color information is reduced to gray-level prior to processing. The

reason being that, gray-level texture is more resistant to changes in illumination, as well as being more robust to various skin textures related to an individual's race or ethnicity. The goal of the profile model is to gain an understanding of the texture surrounding each landmark. Earlier, I described that landmarks should be situated on feature edges or areas of high curvature. The reason for identifying these specific features is that they are areas which highlight a large change in pixel intensity, in so much that we can describe them by a gradient. During the image search process, the landmarks use this information in the form of a gradient descent search to identify where they should move. When the landmark identifies a search profile which provides the best match to what it learnt from the profile model, it is considered to have converged on the correct feature. Unfortunately, this is not always the case; it is not uncommon for landmarks to latch onto false features, thereby causing the system to converge to a false landmark position. During the training phase, the profile model is generated through the following steps:

1. Determine the length and angle of the profile as it relates to each landmark.

2. For each sample in the training set, populate each landmark's profile vector with the extracted pixel intensities along the whisker, ranging from 0-255.

3. Replace each element of the profile with its intensity gradient, computed by taking the difference of the current and prior element.

4. Normalize each vector by dividing each of its element by the sum of the absolute values of the vector.

Using the normalized derivative instead of the actual gray level profile provides uniform scaling and reduces the effects of changes in illumination across images. The result is a collection of n vectors (one for each landmark) containing normalized gradients. The profile model is finalized by computing the mean profiles $\bar{\boldsymbol{g}}$ and covariance matrices $\boldsymbol{S}_g$ for each landmark. Assuming that the landmarks correspond across the entire training

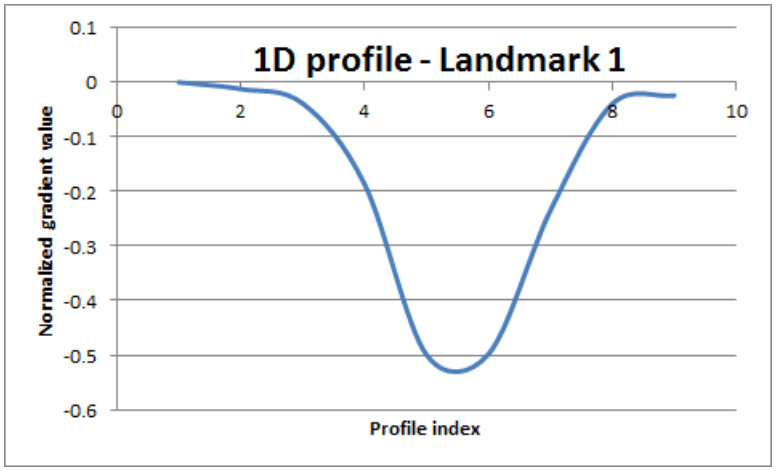


Figure 2.9: Gaussian distribution of the 1-dimensional mean profile for landmark 1 of the 252-point scheme

set, they should capture similar gradient information, such that the mean profiles form a multivariate normal distribution, as shown in Figure 2.9.

Therefore, it is theoretically possible to describe each landmark by its mean and covariance matrix. If there are n landmarks per shape with a profile length of 9, as in Figure 2.8, there will be n mean profiles $\bar{\boldsymbol{g}}$ having 9 elements. Associated with each $\bar{\boldsymbol{g}}$ is a covariance matrix, $\boldsymbol{S}_g$ of size 9×9. The following mathematical formulation for processing profiles is drawn from Hamarneh [22]:

The gray level profile of the landmark i is a vector n_g elements,

$$\boldsymbol{g}_i = [g_{i0}, g_{i1}, \ldots, g_{in_g-1}]^T \tag{2.11}$$

The derivative gradient profile is given by

$$\boldsymbol{dg}_i = [g_{i1} - g_{i0}, g_{i2} - g_{i1}, \ldots, g_{in_g-1} - g_{in_g-2}] \tag{2.12}$$

and its normalized form,

$$\boldsymbol{y}_i = \frac{d\boldsymbol{g}_i}{\sum_{j=0}^{n_g-2} |\boldsymbol{dg}_{ij}|} \tag{2.13}$$

The mean normalized gradient profile for landmark i is calculated by summing across every

sample j in the training set N,

$$\bar{\boldsymbol{g}}_i = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{y}_{ij} \tag{2.14}$$

Finally, the covariance matrix[2] of each landmark is given by

$$\boldsymbol{S}_g = \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{y}_{ij} - \bar{\boldsymbol{g}}_i)(\boldsymbol{y}_{ij} - \bar{\boldsymbol{g}}_i)^T \tag{2.15}$$

During the image search phase, each landmark forms multiple search profiles by sampling along each element of its whisker. Each search profile g_s undergoes the gradient and normalization calculations from above, and is compared to the mean profile for that particular landmark. The displacement along the whisker with the profile which is most *similar* to the mean profile becomes the new suggested location for the landmark. The measurement of similarity is calculated using the Mahalanobis distance of the sampled normalized gradient profile $\boldsymbol{g}_s$ to the corresponding landmark's mean normalized gradient profile $\bar{\boldsymbol{g}}$,

$$D_M = (\boldsymbol{g}_s - \bar{\boldsymbol{g}})^T \boldsymbol{S}_g^{-1} (\boldsymbol{g}_s - \bar{\boldsymbol{g}}) \tag{2.16}$$

After every profile has been searched and the suggested movements of each landmark recorded, a new shape $\boldsymbol{dx}$ is created based on the suggested points. It is often the case that suggested points will produce a shape that is not coherent to the model, so, we use equation 2.10 to conform $\boldsymbol{dx}$ to the model to produce an allowable shape. However, it is possible that there is little or no suggested movement. Such is the case when the current profile provides the best match to the mean profile. This point is said to have converged. This process is repeated until some percentage of the landmarks have met convergence, or for some maximum number of iterations. Hamarneh [22] suggests controlling the suggested movements of the landmarks by changing small suggested movement to no movement, and large suggested movements to only half of what is suggested. This would ensure

[2]The covariance matrix is necessary for the Mahalanobis distance calculation.

that points converge quicker when they are nearer to their desired feature. However, if the initial search shape is nowhere near the actual shape in the image, it would take a greater number of iterations before the landmark converges[3].

2.2.7 Multi-Resolution Image Search

To improve upon the image search process, ASMs use a multi-resolution search pyramid. On the model training side, each image is re-sampled to a number of lower resolutions. Typically, subsequent levels contain half the dimensions of the prior level as illustrated in Figure 2.10, until the final level of the pyramid is reached. The ground truth points associated with each image are scaled equivalently, allowing each level of the pyramid to have a unique profile model associated with each image resolution. Effectively, if each profile has a static length of 9 pixels, the ratio of the amount of information extracted by each profile in comparison to the image size, grows with each step of the pyramid. For example, a multi-resolution pyramid containing four levels could describe the following image resolutions at each level:

1. 320x400 (original resolution)
2. 160x200
3. 80x100
4. 40x50

At the first level of they pyramid, each profile would explain only .007% of the image. At the 4th level of the pyramid, the image resolution would be 40x50, meaning the same profile would account for .45% of the same image.

The reason for this approach is that the initial placement of the shape on a search image may not be close to its actual target. Performing a profile search at lower resolution levels allows the points to travel greater distances with each iteration. Meanwhile, searching at the highest resolutions allows for fine tuning of the landmarks once the target

[3]This was not tested in this work.

Figure 2.10: Image resolution pyramid: each level is downsampled by half of the previous

structure has been located. Therefore, the image search begins at the highest pyramid level (lowest resolution) and continues until convergence has been declared, before moving on to the next level. This process is repeated until all levels have been searched. Figure 2.11 illustrates the mean profiles across three pyramid levels of a single landmark extracted from the boundary of a face. The distributions are very similar, but are not entirely the same.

2.3 Active Appearance Models

Active Shape Models are limited to interpreting the shape of an object. They do not account for the texture (beyond that of the local profiles) of the object. Cootes and his colleagues expanded on ASMs by introducing Active Appearance Models [12], which provided a way of combining shape and texture to model the *appearance* of an object. AAMs require accurate locations of landmarks to define where and how the texture information should be extracted from the image for processing. Unlike ASMs, AAMs are able to generate new interpretations of model examples such that structure and texture can be reproduced visually. AAMs have been used in a number of applications related to facial biometrics and analysis. In [38], they were used to synthesize new instances of faces of individuals across a range of age. This has an interesting value to security applications attempting to identify

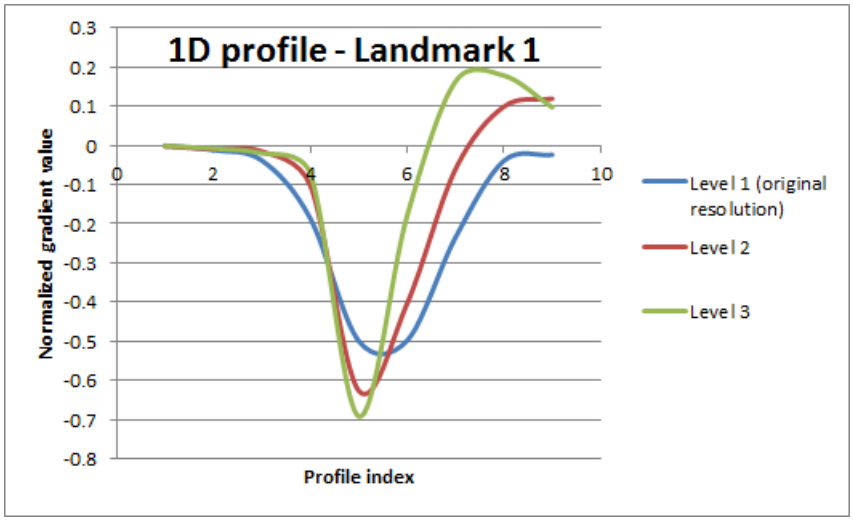


Figure 2.11: Normalized mean gradient profiles for landmark 1 across 3 pyramid levels

persons-of-interest with no current photographs. AAMs have also been used in applications related to face age estimation and gender and race classification [7, 57]. Sethuram shows in [46] that the accuracy of landmarks directly impacts the accuracy of classification techniques which utilize AAMs. A second study evaluated the performance of three landmarking techniques: STASM, CLM, and AAMs by the amount of error associated with the automatically generated landmarks of each method against the same testing set which had been manually labeled (ground truthed) [47]. The AAM approach produced notably more error per landmark than the latter methods, results recreated in Figure 2.12. This confirms the report by Cootes et al. [11], where they found that the landmarks generated by ASMs were better on a whole than AAMs. A potential use of this particular work could be to supplement AAMs with more reliable landmarks.

2.4 Stacked Active Shape Models

Over the past decade, many enhancements to the original ASM algorithm have been contributed by researchers within the community. As mentioned before, this work is based on the contributions of Stephen Milborrow [35], specifically his automatic landmarking tool STASM. STASM is a free-to-download software application that has been used extensively for detecting and locating landmarks on images containing faces. While not a true open source project, full source code is available on his website (`http://www.`

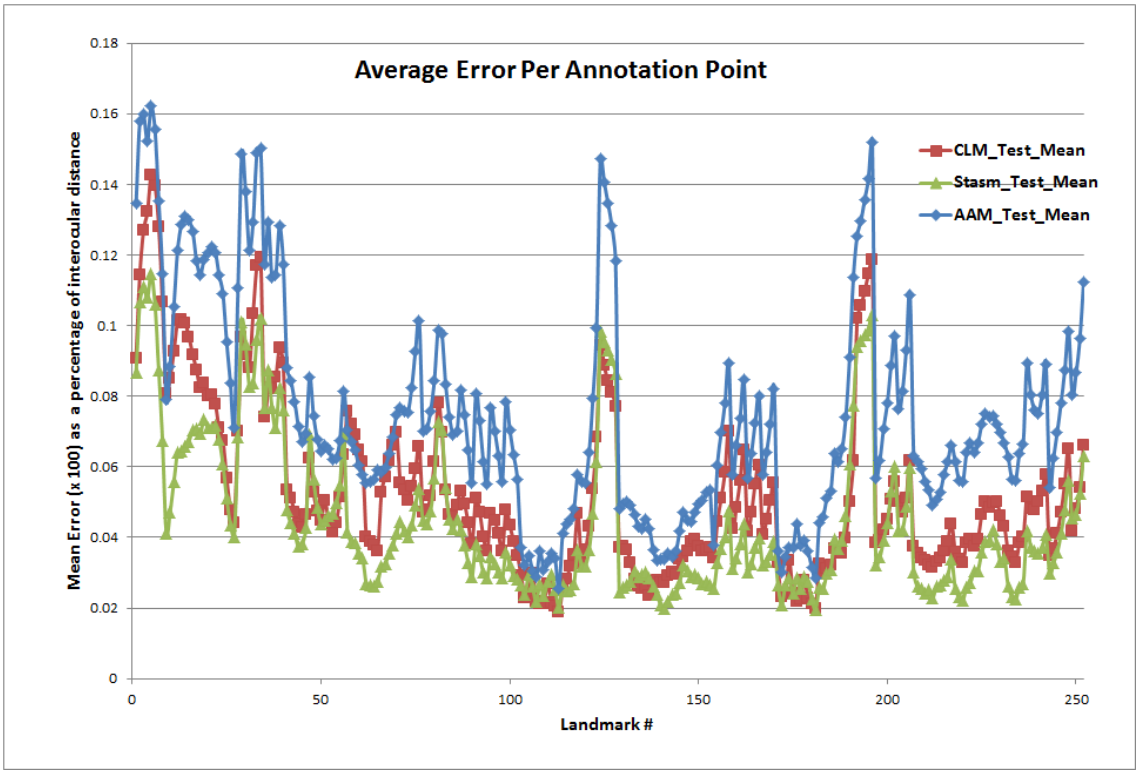


Figure 2.12: Comparison of the average error per landmark generated by three techniques: AAM, CLM, and STASM on the 252-point scheme

`milbo.users.sonic.net/stasm/)`, which can be reworked to meet any specific need. STASM provides many enhancements to the original ASM algorithm, which I will briefly describe here.

Foremost, STASM incorporates a means of *stacking* two models in succession during image search to improve upon the final accuracy of the points (hence the title of the project). The idea is that, the initial model search provides a good estimate of where the landmarks should be located, and the second model search refines the points to a more precise location. STASM's second major contribution was the development of the two-dimensional profile. Instead of the standard 1D vector used to describe the gradient changes of a line passing through each landmark, a 2D square profile has a greater ability to describe the region surrounding each landmark. The result is an increased likelihood that the suggested movements of landmarks actually move in the direction of the object features, which results in more accurate landmarks. Milborrow found that by extending the set of

landmarks, in other words incorporating additional pseudo landmarks to the scheme to better describe the features, the overall accuracy of the landmarks increased. The last major contribution was trimming the covariance matrices of the profile models. *Trimming* refers to creating a sparse matrix from the covariance matrix by setting values which had little or no effect on the outcome of the distance calculation from equation 2.16 to zero. As a result, he found a noticeable decrease in computation time without any negative impact on the landmark fittings. Of the contributions listed, all were implemented in this work apart from trimming the covariance matrix, which will hopefully be included soon!

Unfortunately, STASM has its own drawbacks. On one hand, it proved the concept that ASMs could be used to automatically identify landmarks on faces in images. However, it was not designed to be flexible or extensible to other classes of objects. The program itself was written primarily in C and had many dependencies on libraries which are mostly outdated at this time or are tied to the Linux platform. The majority of the data structures are modified extensions of the Gnu Scientific Library (GSL), which in itself is difficult to port to Microsoft Windows or build in 64-bit mode. The body of the code is immense in size and takes a great deal of time and patience to trace through in order to get a good grasp of what it is doing. Furthermore, STASM has no notion of parallelization, which can improve the execution time for training models as well as image search.

STASM is used extensively in academia and various research institutions as a quick and efficient source of attaining landmarks, which made it the logical choice as a model for this work. My goal is to build off of the success of STASM, while at the same time taking a more modern approach to the design of the application. The end result being a platform which gives researchers and developers the freedom to implement and test extensions to the ASM algorithm at a level of ease that STASM is not able to provide.

Chapter 3: Methodology

3.1 DASM

A fundamental function of this document is to explain the implementation of the algorithms mentioned prior. Therefore, it is relevant to discuss the design of the system and the course of decisions made throughout its development. In terms of the software engineering approach, I chose a very simple incremental development model that associates well with an open source project[4]: planning, software design, implementation and debugging, optimization, testing, and deployment. In short, my objective for the application was to reproduce the techniques of STASM with a fundamentally different software architecture, while creating an environment that lends itself to organic extensions for both researchers and end-users. Along the way, I introduced a few contributions to the system of my own design, which will be covered in subsequent sections.

During the planning phase, I aimed to satisfy my first requirement of limiting the amount of external dependencies of the application. STASM is tied to numerous external libraries, many of which are primarily supported only on the Linux platform, e.g. the Gnu Scientific Library (GSL). Furthermore, Milborrow modified the GSL libraries to suit his own requirements, making it increasingly difficult to update STASM with more recent releases of the library. In addition, many of the remaining standalone C libraries, such as libpng or libjpeg, have been collaborated into broader, well-supported C++ libraries. I attempted to find a balance between the number of external dependencies, and the amount of work that can be offloaded to pre-written functions from tried and tested open source libraries. I found that the use of three heavily supported and trusted C++ libraries helped to support my application while staying within my requirements. Each library was carefully chosen based on its reliability, level of support and use in the open source community, and the level to which it was able to contribute to the project. As an added benefit, all of the libraries are 64-bit friendly. The three libraries being: OpenCV, Boost, and OpenMP.

[4]Also, I was researching, designing, implementing, and validating all on my own, so a complex model was not necessary.

3.1.1 *OpenCV*

Originally developed as a research initiative from Intel, OpenCV [3], short for Open Source Computer Vision, was released publicly under a BSD license in 2000. The library itself was developed to provide universal access to over 2500 algorithms focused in the areas of computer vision, machine learning, and image processing. It was originally written in C/C++, but has since been extended to support Java and Python. Their website (`http://www.opencv.org`) states that OpenCV has more than 47 thousand active users in the community and an estimated number of downloads exceeding 5 million. OpenCV is a test driven platform; each release undergoes extensive testing prior to release by a host of developers. The most up to date version at the time of this thesis, OpenCV 2.4.5, was used in this project, but all versions from 2.0 upwards are compatible. In terms of DASM, I utilized a number of the libraries functions for processing images as well as performing various advanced statistical techniques, such as the PCA.

3.1.2 *Boost*

Boost[5] is actually a collection of peer-reviewed C++ libraries which support a wide variety of applications. Similar to OpenCV, Boost maintains an open source permissive software license that enables its use in public and commercial software. Version 1.53, used in this work, provides access to over 80 individual libraries ranging from multithreading and linear algebra to regular expressions. I utilized the Filesystem and Program Options libraries for directory and command line parsing, respectively.

3.1.3 *OpenMP*

Finally, OpenMP[6] is a C/C++ library which provides a set of compiler directives and routines that can be used to specify parallelism in shared memory applications. The library enables users to parallelize regions of code which are iterative by nature, such as loops, by surrounding the regions with compiler directives. Individual threads are spun up at run time and wait for work to be assigned to them. As almost every computer nowadays contains

[5] `http://www.boost.org/`
[6] `http://openmp.org/wp/`

multiple processing cores, it was an intuitive decision to incorporate mutlithreading into the application to effectively reduce the overall computation time of model training and image search. OpenMP was chosen over the Boost threading libraries because of its simple implementation and high-level approach to parallelism.

3.1.4 Software Design

The first design decision was whether or not to directly modify the STASM code base or construct the application from the ground up. My second requirement was to incorporate an object-oriented design which would allow for extensibility to the software. Based on this requirement, I concluded that the amount of effort required to convert the C-like structures and custom data structures to an object-oriented class structure, would be greater than developing from scratch. Instead, I used STASM as a model for designing the flow of the software, from function implementation to how the data should be organized. Ultimately, DASM follows STASM very closely as far as the methodology is concerned, but varies in its implementation.

My third requirement was to include all of the extensions to the original ASM algorithm that STASM introduced, as mentioned in the previous chapter. I will discuss the implementation of each contribution in the following sections, which describe the model building and image search processes.

3.2 Model Building

Prior to the image search process which involves fitting landmarks to features in images, it is required to construct a model which holds the information about the shape and profile models. Multiple models drawn from two classes of objects were constructed in this work: frontal faces and profile faces. Various point schemes and data demographics were adapted for each model to validate the performance of the software. Results from a 252-point general model of frontal faces[7] and 14-point model of profile faces are displayed in this work. Four databases were utilized for model training and testing.

1. The **MORPH** [41] database

[7] Considered the general model because it characterizes a wide variety of demographics.

MORPH is a longitudinal face database developed for researchers investigating any facet of facial analysis, but has primarily been associated with research dedicated to age estimation and face progression. The non-commercial (academic) version, hosted by the University of North Carolina Wilmington's ISIS Institute, contains over 56,000 images of frontal facing neutral expression faces (mugshot images). The demographics of the dataset span both genders, multiple races, and a range of ages from 16-77, with a median age of 32. As it is a longitudinal database, there are roughly 4 images per individual in the set with an average time between photos of 179 days. A subset of the MORPH database makes up a majority of the images in the 252-point general model. See Figure 3.13.

2. The Productive Aging Laboratory database (**PAL**) [36] database

   The PAL database is hosted by the Park Aging Mind Laboratory within the Center for Vital Longevity of the University of Texas at Dallas. It contains high quality color and gray scale images from a multitude of varying facial expressions. Demographic information for the database can be found in [36]. A small subset of PAL images contribute to the general model. See Figure 3.13.

3. Pinellas County Sheriffs Office (**PCSO**)

   The PCSO database is an enormous face dataset containing over 1.29 million digital images. Similar to the MORPH database, PCSO contains frontal facing mugshot photographs. Images from this database make up the final portion of the general model. See Figure 3.13.

4. Surveillance Cameras Face database (**SCface**) [20]

   SCface is a database of images taken in an uncontrolled indoor environment from five video surveillance cameras. Images were selected from a variety of angles to capture various facial poses from 130 subjects. A subset of a 90 degree pose images were used in this work to build the 14-point profile face model. See Figure 3.13.

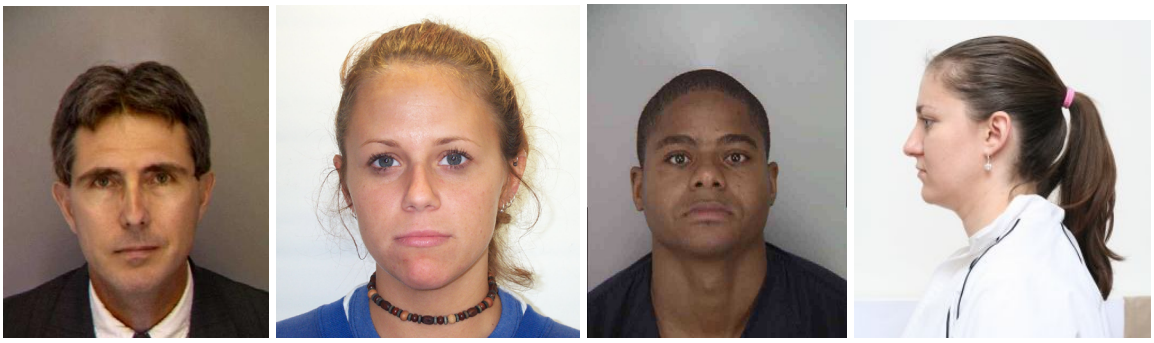


Figure 3.13: Sample images from MORPH, PAL, PCSO, and SCface

3.2.1 Ground truthing

Ground truth data for the above datasets was acquired over the course of a few years of dedicated labor by students (including the author) at the University of North Carolina Wilmington's Face Aging Group. Ground truth landmarks are necessary not only for training models, but also for validating the results of automated techniques such as DASM, STASM, etc. There exist a number of tools designed to help alleviate the difficulty associated with manually annotating landmarks. The tool used in this work was designed with the flexibility of labeling dense and sparse point schemes [1]. Figure 2.6 shows the interface of the Facemark tool. As mentioned earlier, obtaining ground truth landmarks is a tedious and very time consuming task. Experts must be trained on a specific landmarking scheme in order to reduce the effects of inconsistencies of landmark placements across the training set. In [47], Sethuram introduces a *component scheme*, in which a group of experts are trained to annotate specific regions or components of, in her case, the face. In effect, this ensures that there are minimal inconsistencies of correlated landmarks across the entire training set. This is an especially beneficial technique for dense landmark schemes, where it can often take a single individual ten minutes to precisely landmark an entire image.

3.3 Training Methodology

The size and quality of the training set will play an important role in the model's ability to generalize new examples. Over-training will result in the introduction of erroneous points to the system, while under-training will restrict the ability of the model to accurately describe all the variations of the object. A less complex structure such as the

shape of a hand would require a smaller training set than that of the face, based on the amount of variation present with faces. Both the general model built in this work, as well as the testing set, were adopted from the same data in the experiments of [47]. The general model training set was formulated with a distribution of images spanning age ranges of 18-71+, both male and female genders, and Caucasian and African American ethnicities. In total, 1155 ground truthed images were used in training in the hopes of capturing all of the variability across the various demographic groups. Two 14-point profile face models were constructed from the PAL and SCface databases, respectively. The purpose of this was to see how well each model generalizes against unseen images from the opposite dataset. The evaluation techniques and experiments will be discussed in the following chapters.

After ground truth data has been acquired, it is necessary to extract the image data to build each model; this is where the DASM software steps in. The application requires the user to define a configuration file which informs the program whether or not the user wishes to train a model, or use a pre-built model for image search. In regards to model training, the software attempts to load the training images, the associated points files, and the parts file which corresponds to the landmark scheme. Appendices 1 and 2 shows the formatting of a typical points and parts file. Furthermore, a number of additional parameter options can be specified, such as enabling OpenMP, i.e. how many threads the user wants spun up, or entering verbose mode. Additionally, model parameters such as the length and size of each one and two dimensional search profile, and the number of image pyramid levels can be defined. An example configuration file is illustrated in Appendix 3.

If one has a familiarity with STASM, it comes as no surprise that there are a number of fine tuning parameter adjustments that affect a model's ability to accurately generalize new examples. Milborrow [34] performed extensive testing to find the optimal parameters for training and image search on images from three databases: the AR face database [31], the BioId face database [25], and the XM2VTS database [33]. He defined the optimal parameters as those that produced the most accurate landmarks across each dataset, which did not result in a significant increase in computation time. I use his suggested model

parameter values as a starting point for my tests. It is recommended to read through his Master's thesis, where he explains the choice of each parameter in great detail.

3.3.1 Model parameters

The choice of model parameters is contingent upon a number of factors; the number of landmarks, the image resolution, the object being described, the quality of the image data, etc. Instead of recording the optimal parameters selections related to data acquired from a specific dataset, it makes more sense to define a range of parameter values which generalize across varying data, whether it be faces or any other class of objects. For example, the optimal number of pyramid levels for a 252-point landmark scheme of a face may not be the optimal number for a 10-point scheme describing the geometry of the hand. Unfortunately, this would require a great deal of testing which is outside the scope of this thesis. Instead, I explored a number of experiments which evaluated models trained under varying conditions, i.e. tweaking the length of the search profiles. The purpose of the experiments was to find an optimal model which, when compared against STASM and other landmarking techniques, would show the effectiveness of the DASM implementation. The following chapter shows the results of the experiments and the optimal model parameters.

3.4 Training Implementation

Integrating the model training methodology to the software was a fairly straightforward process. The implementation closely follows the methodology described in the ASM overview section of chapter 2. A simple class structure was designed to handle all of the fundamental tasks involved with model training and image search, illustrated in appendix 4. During training, a single instance of the Model class is instantiated, which houses all of the information that is necessary to perform image search. Every image and corresponding points file is associated with an instance of the Shape class. The Shape class stores the image data in terms of a two dimensional matrix of pixel intensities and landmark coordinates in the form of a one dimensional vector of points. Each Shape's image is either resized using bicubic interpolation over 4x4 pixel neighborhood, or padded with extra pixels to ensure a standard size for all of the training data. Maintaining a constant size prevents disparities

which arise when extracting gradient information from various size images. Essentially, the profiles would not match across images, meaning we won't see the nice normal distribution for each mean profile, as seen in Figure 2.9. Unfortunately, there is no way of knowing how large the object is within the image without some a priori knowledge, i.e. the location of the eyes, which would allow scaling the image by the interocular distance.

At the same time, the image resolution also affects the choice of 1D profile length. A profile length of 11 pixels on a very high resolution image, i.e. 3200x2400, does not make sense as the profile is hardly capable of describing an extremely small amount of the image. Also, through empirical testing I found that processing on high resolution images versus low resolution images does not increase the accuracy; in fact it only hinders the system by increasing the amount of processing and memory requirements. Factors such as image artifacts, poor illumination conditions, and unconstrained pose had a much greater impact. Ultimately, forcing images to a standard size, either by resampling or padding, for both training and testing, proved to be the most efficient means of defining a standard set of model parameters.

3.4.1 Training the Profile Model

The next step is to develop the profile model. However, it is necessary to first set up the image pyramid. Each Shape's image data and points are down-sampled by the pre-selected number of pyramid levels. At each level, the profiles for each landmark are trained across the entire dataset and the mean profiles are computed and stored by the Model class. It is pertinent to discuss the distinction between one and two dimensional profiles. Two dimensional profiles are constructed by defining a square patch around each landmark as opposed to a one dimensional vector. Processing of 2D profiles is extended slightly beyond the standard 1D profiles by performing matrix equalization, but the end goal remains the same. Each profile region reflects a square matrix, such that the landmark rests in the center of the profile. The matrix contains the extracted image pixel intensities, which are then processed by a correlation filter, shown in Figure 3.14. Figure 3.15 illustrates 9x9 2D profiles surrounding each landmark. The gradient matrix is then normalized and equalized

$$\begin{matrix} 0 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Figure 3.14: 3x3 Gaussian correlation mask

by applying a Fast Sigmoid Transform to each element i of the matrix $\boldsymbol{X}$,

$$\boldsymbol{X}' = \frac{\boldsymbol{X}_i}{abs(\boldsymbol{X}_i) + c}. \tag{3.17}$$

Milborrow suggests choosing a shape constant c of 10 or more, meaning that only a small amount of equalization is applied to the matrix. The resulting mean profiles after training, illustrated in Figure 3.16, show a similar inverted Gaussian distribution found in the 1D mean profiles. On a whole, 2D profiles have shown to produce more accurate landmarks than 1D profiles. However, they do increase the computation time due to the matrix processing as well as the memory requirements, particularly due to the covariance matrices. For instance, a 1D profile of length 9 produces an 81 element covariance matrix, whereas a 9x9 2D profile results in a 6561 element covariance matrix. Multiply that by 252 data points and 4 pyramid levels and you are talking about a substantial amount of memory. Thankfully, memory is less of an issue today, so this does not result in a limiting factor to the application.

3.4.2 Incorporating Parts Information

During training, 1D profiles also make use of the parts information. The direction of each search vector is determined by the perpendicular angle of the current landmark's two immediate neighbors, i.e. landmark 55's search profile would be perpendicular to a line connecting points 54 and 56. Unfortunately, not all landmarks have two neighbors related to the same *part*, and in some cases no neighbors at all, such as the nose tip and eye centers. In these situations, it is not logical to determine the search direction based on points from a different part, which could be located anywhere on the object. Instead, profiles for landmarks which fall on the end of an open boundary, i.e. points 236 and 241 from Figure 3.17, determine their search direction based only on the neighbor within the

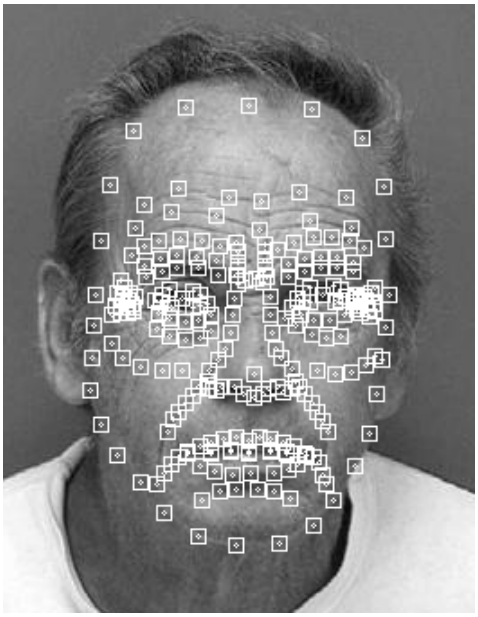


Figure 3.15: 9x9 pixel 2D profiles surrounding each landmark

same part. In rare cases where there are single point parts, the search direction is forced to be vertical. Typically, these are anatomical landmarks which are not necessarily defined by a strong gradient, so the search direction is less important. The difference between profiles with and without parts information can be seen in Figure 3.18. 2D profiles do not use parts information during training, because they are all vertically and horizontally aligned with the boundaries of the image, as seen in Figure 3.15. Parts files are not a novel concept; they are used in conjunction with Cootes' am_markup tool[8]. However, their integration with profile search in ASMs is a new design.

3.4.3 Finalizing the Model

Training the profile model consumes the majority of the training time, due to the immense number of calculations, albeit simple ones. The two remaining major tasks involve developing the PDM and determining the correct shape initialization from the object detector. The implementation of the PDM abides by the same methodology from the original ASM algorithm, described in section 2.2.4. The *mean* shape, which is stored by the model, is the final reference shape when the alignment process has converged. This shape

[8] `http://www.isbe.man.ac.uk/~bim/software/am_tools_doc/index.html`

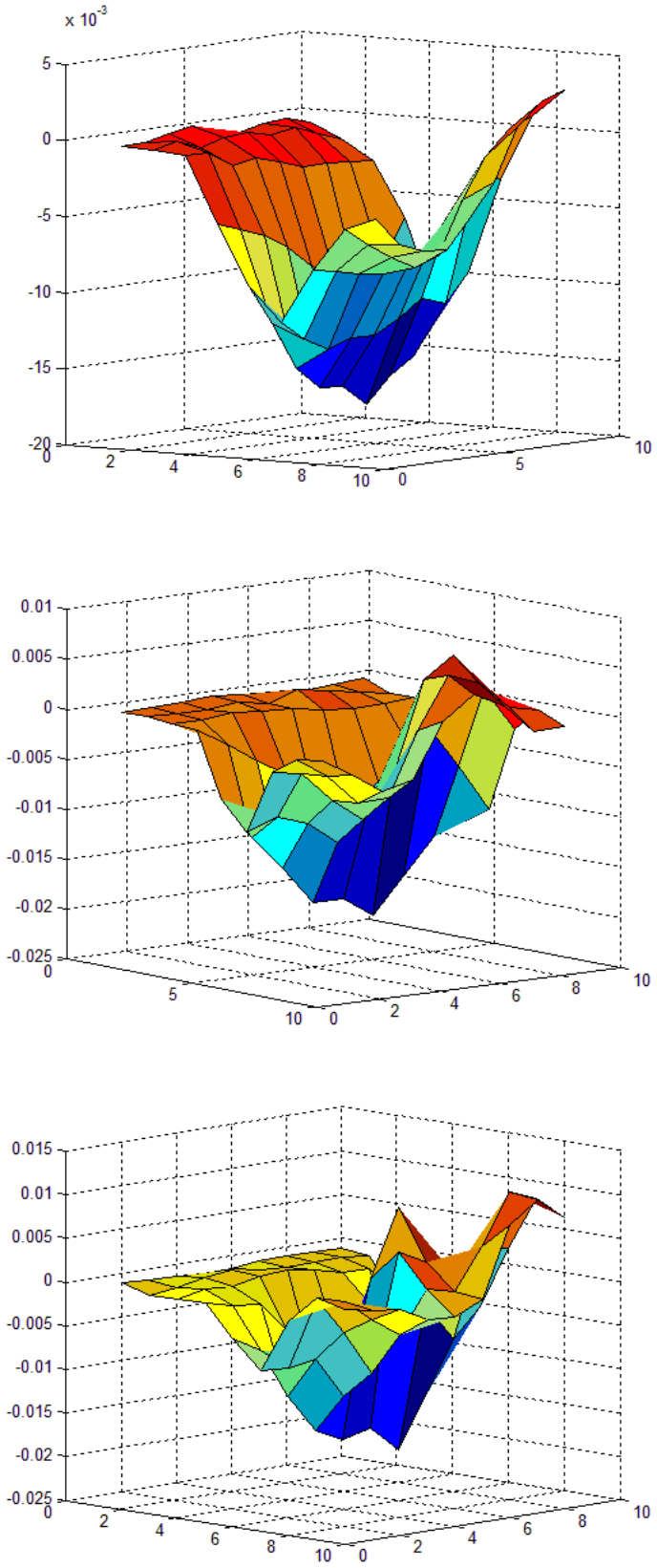


Figure 3.16: 2D Normalized mean gradient profiles for a landmark located on the boundary of a profile face image across 3 pyramid levels. (top) Level 1, (middle) Level 2, (bottom) Level 3.

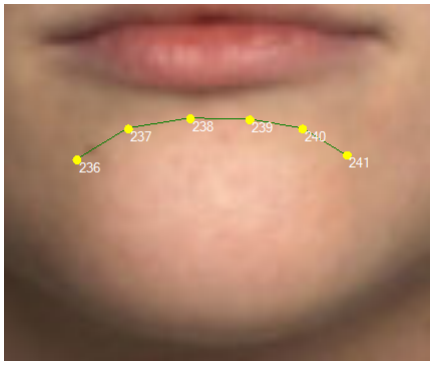


Figure 3.17: An open boundary part along the chin

Figure 3.18: Profile search directions of the chin without and with parts information

is used as the initial shape (starting point) when beginning the image search process.

3.4.4 Object Detection

Object detection is a key contributor to automatic landmarking systems. In controlled scenarios, we can manually define a starting location for the initial shape which is close to the structure to be searched. In real-world situations, it is not possible to have direct human interaction with the initialization of the shape with respect to the object within the image. Automatic detection alleviates this problem by giving a good estimate of the location of the structure within the image.

Object detection is a domain of study in computer vision concerned with *finding* a desired object within an image, rather than attempting to identify it. For example, locating an automobile within an image, but not necessarily recognizing its make and model. A significant amount of research has been invested into detecting or recognizing faces, as well as specific features of the face, such as the eyes or mouth [23]. Unfortunately, the technology does not exist yet to be able to universally detect miscellaneous structures based on semantic notions of content, i.e. ”find the boy holding a red balloon” [19].

Two object detectors were tested in this software. The first was provided by the OpenCV distribution. Their object detection functionality is based on the work by Viola

Figure 3.19: Frontal face detections from the OpenCV detector: (left) Successful detection, (center) Poor bounding rectangle, (right) Failed detection

and Jones[55], and extended by Rainer Lienhart[30], that enables users to construct custom object classifiers for detection. Once a classifier has been trained from a set of examples, it can be used to detect the known object in an image. Upon successful detection, a bounding rectangle is placed on the image surrounding the detected object. Various classifiers, including frontal face, profile face, eye, and full body classifiers come prepackaged with the distribution.

The two biggest flaws with the OpenCV implementation of face detection is that it, (1) often fails to find the object under poor image conditions and, (2) occasionally produces false positives. For example, the profile face classifier when applied to 124 images from the SCface database, successfully located the face only 102 times. The frontal face classifier is much stronger, but is often inconsistent as to the placement of the bounding box on the face, i.e. it is notorious for cutting off the chin. Figures 3.19 and 3.20 show examples of good and poor detections. Without an accurate detection, we cannot initialize the mean shape, meaning the model has no chance of accurately landmarking the features of the object. In fact, the software does not even attempt to search the image if the detection fails.

An alternative face detector was acquired for testing purposes from PittPatt [45]. The PittPatt 5.2.2 face detector is more robust to varying image conditions and has shown under test conditions to rarely produce false positives. Instead of coupling the use of either

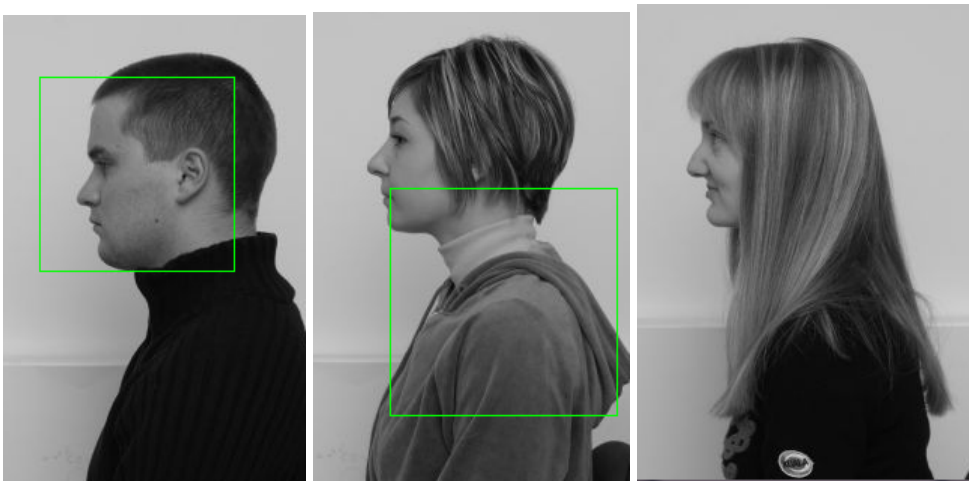


Figure 3.20: Profile face detections: (left) Successful detection, (center) Poor bounding rectangle, (right) Failed detection

detector to the software, the detector functionality was designed to be as abstract as possible. Ultimately, any user defined object detector can be implemented into the program as long as it is capable of supplying coordinates and dimensions for the bounding box. This is a distinct design advantage over STASM, which enforces the use of either the Viola-Jones detector or the Rowley detector [43].

Shape initialization is one of the most important steps during the search process. It is possible to use additional information to produce a better initialization; I will touch on this topic in my conclusions. During the training phase, the bounding rectangles dimensions are accumulated across every detected image and an average is computed. It is necessary to discover the relationship between the center of the average box and the center of gravity of the mean shape, such that when the mean shape is initialized, it is understood by how much to scale and translate the shape to the detected bounding rectangle. Basically, we want to know exactly where to put the start shape, according to how the landmarks from the training data relate to the detected box.

Once complete, the model is written out in a text and binary format. The text file is useful for debugging and discovering inconsistencies in the data. The binary format is preferred for use in the program, because it requires less space and loads exponentially faster than the text based format. Appendix 5 illustrates an abridged textual model file.

This concludes the model training implementation.

3.5 Training Summary

In summary, the training process involves developing a model which contains information about the typical variations of the object's shape, what each landmark's local neighborhood looks like, and how to initialize a shape based on where the object is located in an image. With all of this information, it is possible to search new images and automatically identify landmarks with a certain degree of confidence.

3.6 Search Methodology

Given a trained model, we want to be able to use it to produce accurate landmarks on new examples. The search begins by initializing the model's mean shape to the detected location of the object in an image. Based on the scale and location of the bounding rectangle, we can give a close approximation of the starting point for the search. It is clear from Figure 3.21 how great of an effect the initialization has on the search process. A poor start shape makes it difficult for the model to recover from, if it is able to at all.

3.6.1 Profile Search

The profile search begins at the lowest level of the image pyramid and works upwards to the original resolution. For 1D profile search, search profiles are offset at each pixel along the whisker. 2D search profiles are computed at every other location along the whisker, as well as an additional ± 1 along the y-displacements of the landmark. The search profile which is the most similar according to equation 2.16, becomes the suggested location for the landmark. After all of the landmarks have been searched, the suggested shape is realigned with the model's mean shape, to ensure that the new shape is a plausible shape. Figure 3.22 shows the result of a landmark search that was not conformed to the model. Without shape constraints, the points are free to latch onto any feature that they find is the best profile match, as is evident by the point that attached to the man's shirt collar.

3.6.2 Applying Shape Constraints

We control how much to conform the suggested shape by applying limits to the $\boldsymbol{b}$ vector from equation 2.10. Limits are applied based on a percentage of the information

Figure 3.21: Shape initializations based on the detected bounding box. (top left) Cutoff chin, (top right) Good initialization, (bottom) Below the chin

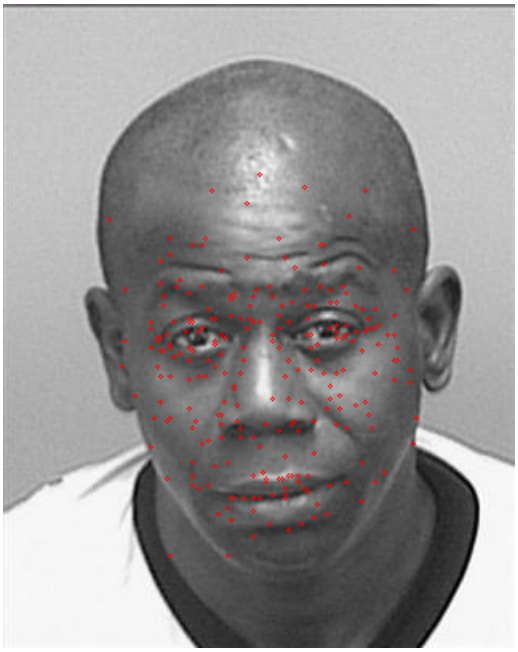


Figure 3.22: Produced landmarks without shape constraints

contained in the eigenvalues. For instance, we may only wish to retain the top 75% of the ranked eigenvalue information (not the number of eigenvalues), which describes the main modes of shape variation. The less eigenvectors used in the calculation, the more similar the conformed shape will be to the model's mean shape. Additional constraints of $\pm 1.8\sqrt{\lambda_i}$ control the maximum allowable value of each element in the $\boldsymbol{b}$ vector. The value of 1.8 was borrowed from Milborrow's tests[35]. In order to preserve the length of $\boldsymbol{b}$ after it has been constrained, the clipped values are replaced with zeros. It has shown to be useful to *loosen* the model by increasing the amount of eigenvectors by a small amount during the final iteration of search. This allows for additional wiggle room for the suggested landmarks at the original (full) image resolution.

3.6.3 Stacked Models

Once every level of the image pyramid has been searched and the landmarks converged, there is the option of repeating the search a second time using the newly acquired landmarks as the starting shape. Tests have shown that stacking the model search produces slightly better results overall. The unfortunate drawback is the increased computation time required to repeat the entire search, as will be seen in the experiments chapter. In the cur-

rent implementation, DASM is limited to using the same model for the second search. It would be beneficial to allow for a separate model to be specified for the stacked search, with different model parameters. For instance, given a decent initial shape, we can say with some confidence that the landmarks from the initial search are likely to be close to their destined features. Therefore, it isn't necessary to repeat every level of the pyramid search, especially the lower resolution levels. In fact, repeating the search at the lowest levels could potentially drive some landmarks further away from the object.

3.7 Image Search Summary

The search heuristic can be summarized by the following steps:

1. Initialize the model's mean shape to the detected bounding rectangle
2. Beginning at the highest level of the image pyramid, perform a profile search to generate the suggested landmarks
3. Conform the suggested shape $\boldsymbol{x}$ to the model's mean shape $\bar{\boldsymbol{x}}$
4. Repeat the search at each level until the landmarks converge
5. Optionally repeat the search with a stacked model
6. Output the new landmarks to a points file

3.8 Testing Methodology

Two experiments were performed to to illustrate the performance of DASM. The first measures how well the DASM implementation holds up against similar automatic landmarking techniques. Specifically, I want to explore the accuracy of rendered landmarks of the competing algorithms on a single large dataset. How well each algorithm performs is measured by the error and classification rates when applied to 1015 images of frontal faces using the 252-point scheme. The performance metrics and distribution of the testing set were already in place, thanks to the study by Sethuram et al. [46]. However, before I could perform the experiment, various tests had to be administered to find the optimal model parameters related to the test data.

The second experiment was designed to measure the speed associated with converging to the landmarks. I am most concerned with the search process, as this is where we see the most gain from parallel execution. By assigning each image a single thread, we are theoretically reducing the overall computation time by $\frac{1}{n}$ where n is the number of available threads. The results of both experiments are illustrated in greater detail in the following chapter.

Chapter 4: Experiment and Results

This chapter introduces a series of experiments and results related to:

1. finding the optimal model parameters for the 252-point frontal face model
2. comparing DASM against similar automatic landmarking techniques: STASM, AAM, CLM
3. measuring the computational efficiency of multi-threaded image search

4.1 The Training and Test Data

Both the training and testing data for the 252-point scheme were formulated by manually annotated images from the MORPH, PAL, and PCSO databases. Tables 4.2 and 4.3 describe the distribution of the training data according to the demographics of the individuals and databases used, respectively. The data closely resembles an even distribution across various age ranges and ethno-gender groups, i.e. African American Females (AAF), Caucasian Males (CAM), etc. A perfectly even distribution across the ethno-gender groups was not possible due to the lack of data in certain instances, such as African American Males in the age range 71+. In total, the 1155 images which formulate the *general* model provide the baseline for the frontal face experiments. The testing set of 1015 images is comprised of a similar distribution of data, described in tables 4.4 and 4.5. There is no crossover of images in the training and testing sets.

Table 4.2: General model training distribution based on ethno-gender groups and age ranges

Age Range	AAF	CAM	AAM	CAF
18-30	50	50	50	50
31-40	50	50	50	50
41-50	50	50	50	50
51-60	50	50	49	50
61-70	50	50	48	50
71+	38	21	49	50

4.2 Model Parameters

Table 4.3: General model training distribution across the databases

Database	AAF	CAM	AAM	CAF
PCSO	88	62	140	180
MORPH	200	199	149	67
PAL	0	10	43	53

Table 4.4: Testing set distribution based on ethno-gender groups and age ranges

Age Range	AAF	CAM	AAM	CAF
18-30	50	50	50	50
31-40	44	49	50	50
41-50	41	50	50	50
51-60	49	50	50	50
61-70	11	50	43	50
71+	0	50	2	26

Table 4.5: Testing set distribution across the databases

Database	AAF	CAM	AAM	CAF
PCSO	103	240	140	244
MORPH	80	44	101	6
PAL	12	15	4	26

This section shows the evaluations of finding the optimal model parameters for the general model. The four major tests involved were:

1. examining the profile types and lengths
2. the number of image pyramid levels
3. stacked versus non-stacked models
4. and the percentage of eigenvectors used by the PDM.

There are three additional factors that affect the performance of the model which were not extensively tested in this work. The convergence threshold value determines when the profile search at the current pyramid level can be stopped and the next level begun. For instance, a convergence factor used in these tests, 51%, meaning in order to declare convergence, over half of the suggested landmarks did not shift more than one third of the profile length away from the current location. Since it is possible that the system may never

converge, a cap is placed on the maximum number of search iterations at each level. The maximum number of searches per level was 6 during these tests. The final factor relates to how many eigenvectors are supplemented during the final level of search. This simulates "loosening" the model, which gives the suggested landmark movements more freedom at the original image resolution. A fixed value of 10 eigenvectors was the added factor for the following tests.

The experiments were run on a Windows 7 machine, containing a Intel Core I7 at 3.4GHz with 8 cores. The tests were all run in 32-bit mode.

The performance metrics that are used to evaluate the techniques put forward in this work are taken from [46]. Of note are measures of the cumulative error distribution (CED) and average error per annotation point. The CED measures the percentage of images that have less than a specific mean average error. That is, the cumulative error per image, averaged across the entire testing set. Alternatively, the average error per annotation point gives us in depth look at a model's ability to localize each individual landmark. The error measures are normalized by the interocular distance d_{io} of each subject.

$$e_i = \frac{\|T_i - \hat{T}_i\|}{d_{io}} \tag{4.18}$$

Interocular distance is defined as the distance between the centers of the left and right eyes. This type of normalization ensures that an equal measurement can be evaluated under variant image sizes. The following tests were all evaluated using the 252-point scheme on the 1015 image test set.

4.2.1 Profile Configuration

The first experiment compared the effectiveness of one- versus two-dimensional search profiles. I examined four different profile lengths (number of pixels) for each search criteria: 7, 9, 11, and 13 for 1D profiles, and 7x7, 9x9, 11x11, and 13x13 for 2D profiles. The results, displayed in Figures 4.23 and 4.24, show that for 1D profiles, a 7 pixel whisker is the best choice. However, if you look closely at the scale of the horizontal axis, you

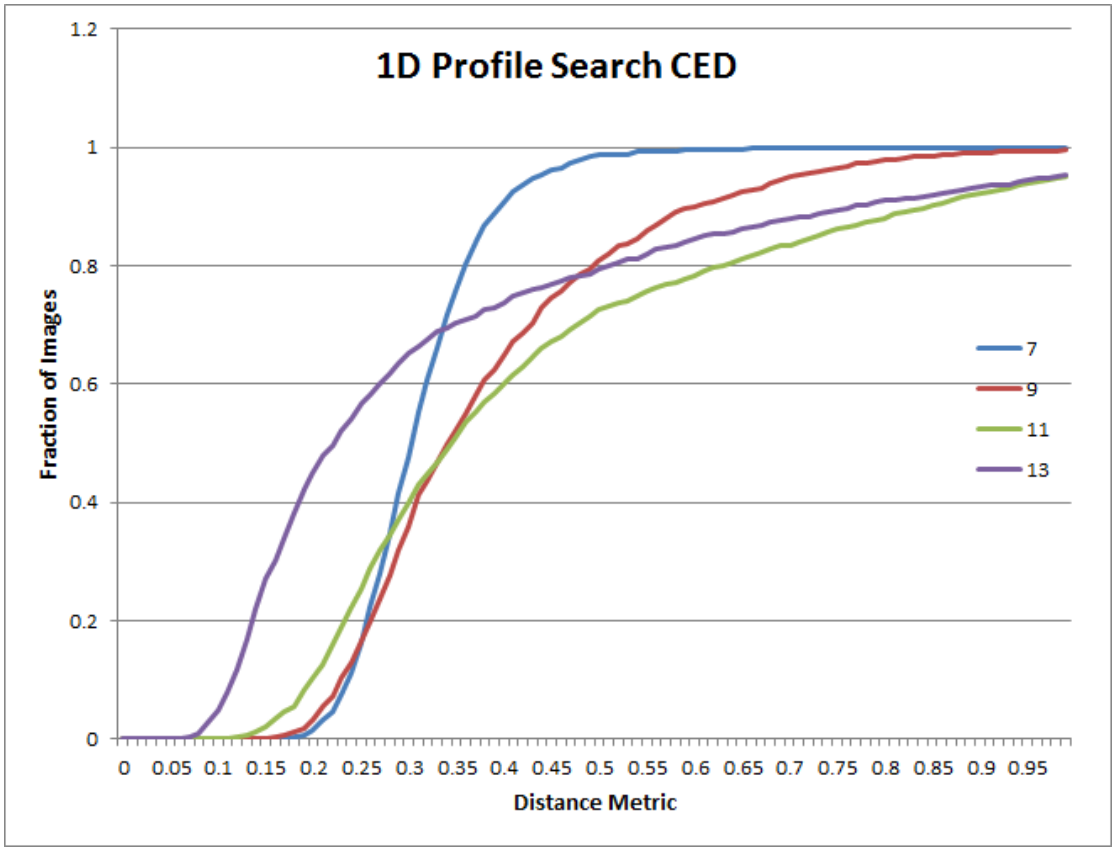


Figure 4.23: Results of varying the profile length for 1D profiles.

will see that the 2D profiles produce significantly less error than any of the 1D profiles. This is furthered exemplified in Figure 4.25, which shows the mean error per landmark between the optimal 1D profile model versus the optimal 2D profile model. The difference in profile size for the 2D profiles was minimal, however, the 11x11 parameter was chosen for the optimal model. As the 1D profiles showed to be less effective for search, the rest of the experiments focus only on 2D profile models.

4.2.2 Pyramid Levels

The second experiment evaluates the performance of the model search when the number of pyramid levels is adjusted. Figure 4.26 illustrates that increasing the number of pyramid levels correlates to a higher accuracy of landmarks. Essentially, the results reveal that given the model's ability to perform a greater amount of profile searches, the landmarks have a better chance of converging on their features. All of the images in the training and testing set had an image original resolution of 320x400.

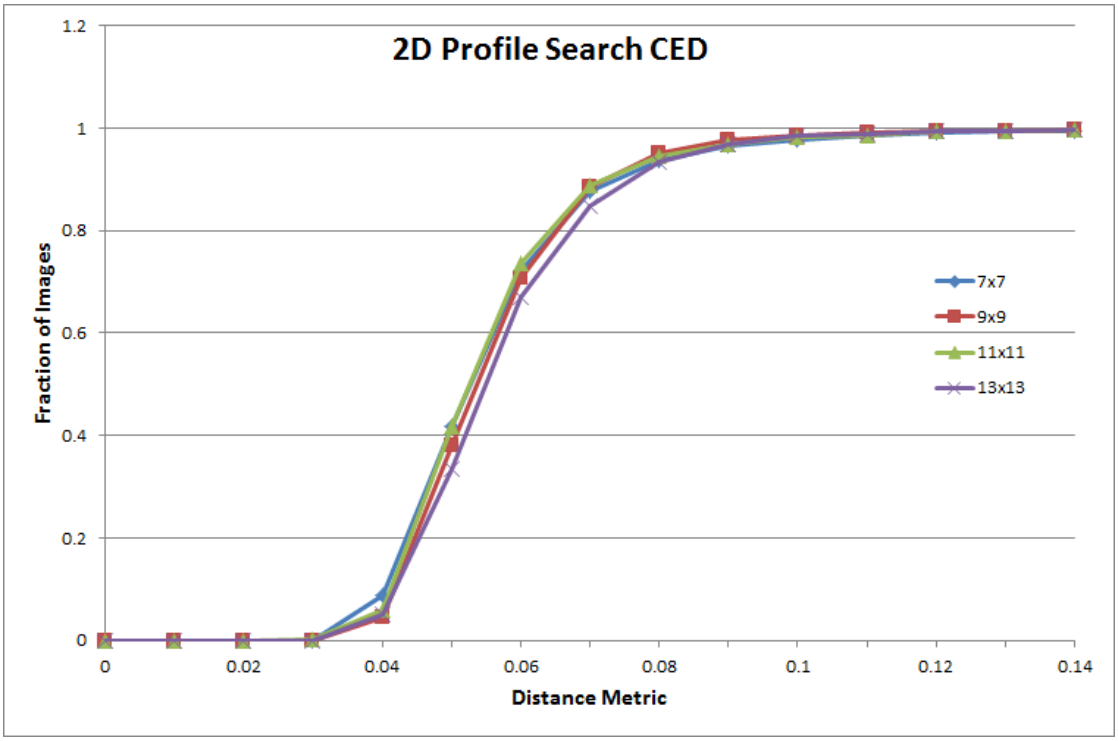


Figure 4.24: Results of varying the profile size for 2D profiles.

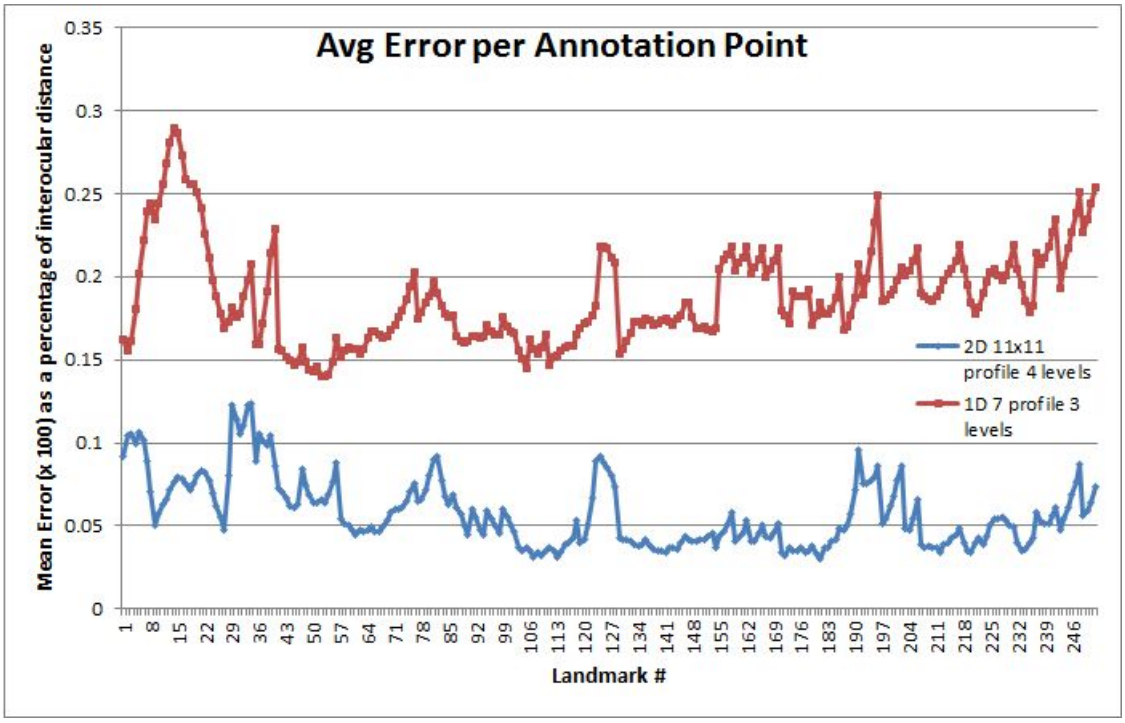


Figure 4.25: Disparity between the best 2D model and 1D model search

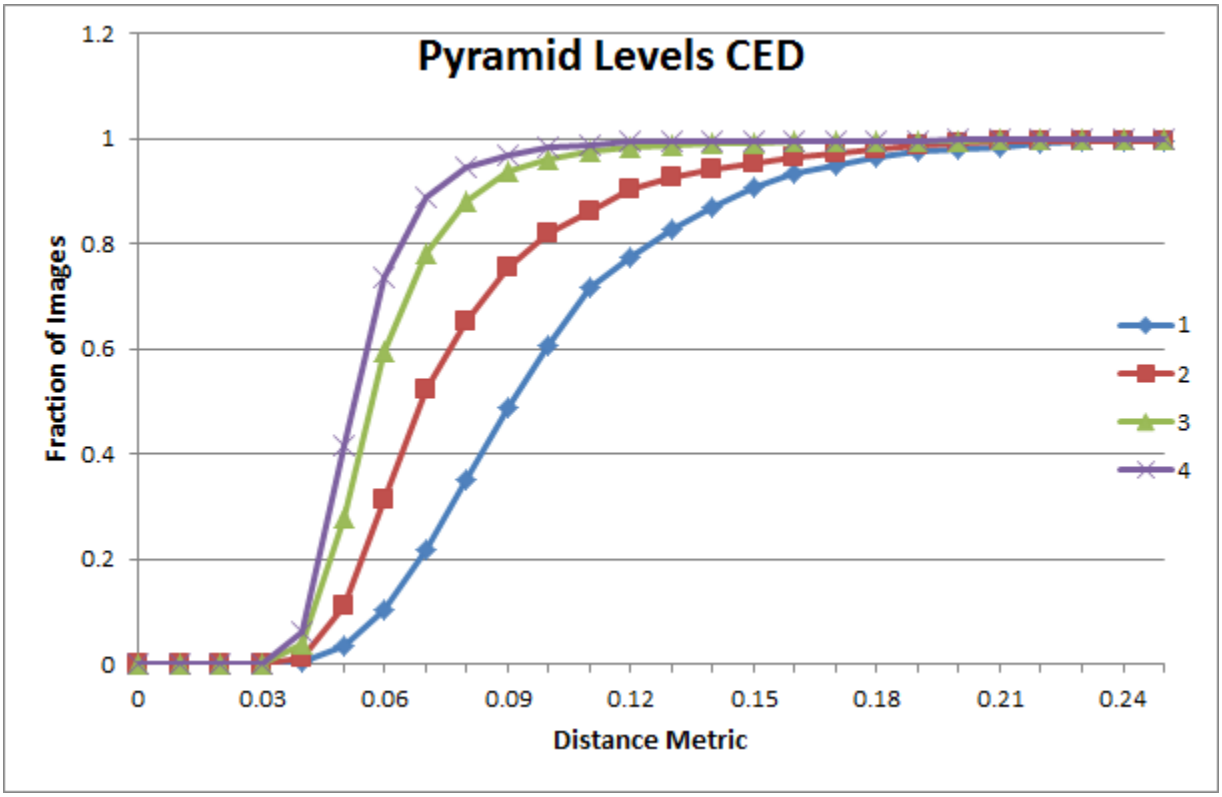


Figure 4.26: Difference between multiple levels of search.

4.2.3 Stacked Models

The next test took the optimal 2D profile model and evaluated it in a stacked versus non-stacked search environment. Figure 4.27 suggests a minor improvement from stacking the model in succession. As mentioned earlier, the current implementation only allows the same model to be stacked. It would be interesting to see if there is a more significant increase in performance if a different model is stacked. As we will see in section 4.4, there is a drawback with stacking models in reference to computation time.

4.2.4 Percentage of Eigenvectors

The final experiment related to discovering the optimal model parameters explores the percentage of eigenvectors to incorporate when conforming a new shape to the model's mean shape. The lower the percentage, the more similar the shape produced from the profile search will be conformed to the mean shape. Conversely, a higher percentage allows for more freedom for the landmarks to differ from the shape model. Four percentages were

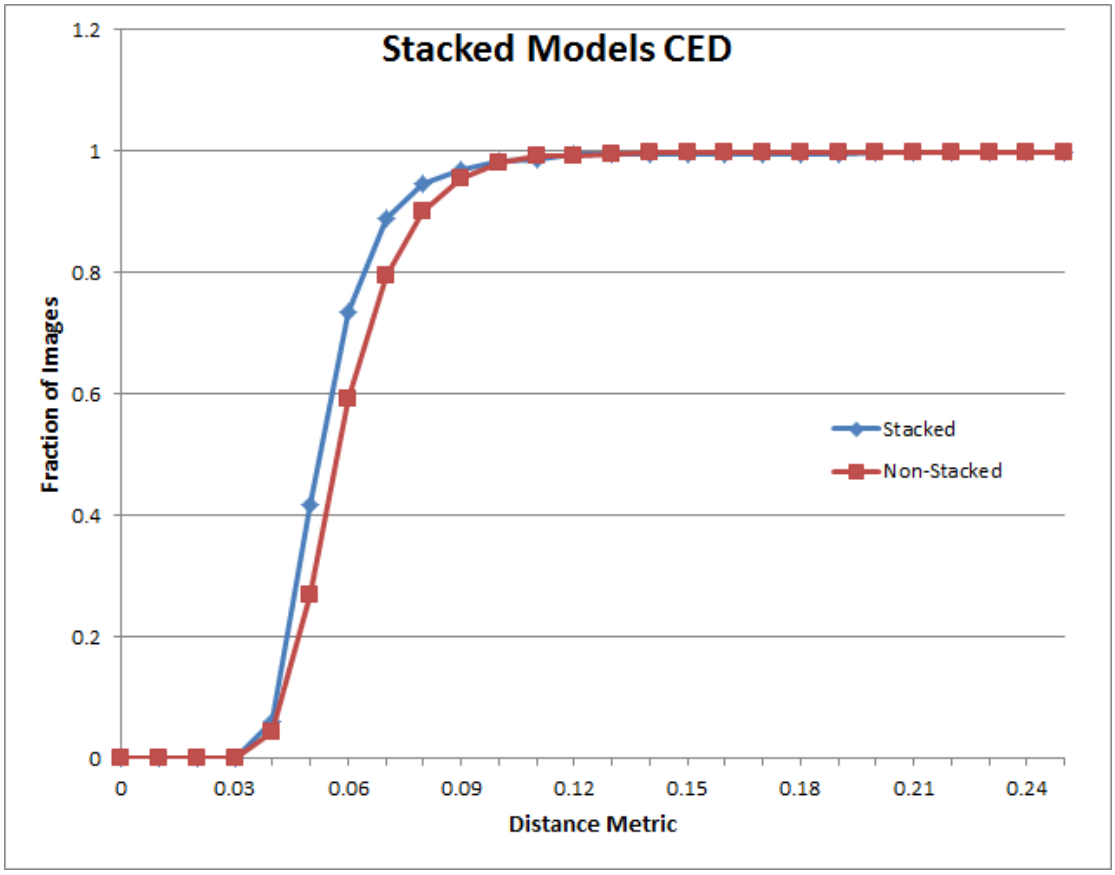


Figure 4.27: Performance of stacked versus non-stacked model search

tested: 65%, 75%, 85%, and 95%. Note that this is a percentage of the eigenvalues which have been trimmed during the training phase. When the PCA is computed, only the top 98% of the data is preserved. The purpose of this was to eliminate residual data which gets generated when computing over a large amount of data points. So the 65%, 75%, etc. are actually a percentage of the preserved 98% PCA space. Figure 4.28 shows that the less eigenvectors that are present, the greater the accuracy of the landmarks. The actual number of eigenvalues associated with the percentages were 11, 18, 31, and 67, respectively.

4.3 Comparison to Similar Methods

Once the optimal model parameters were found, I compared the results of landmarks produced by DASM against three competitive techniques: STASM, AAM, and CLM. Figure 4.29 shows that the current implementation of DASM produces slightly worse results than STASM, yet right in line with the CLM method. Figure 4.30 shows the aver-

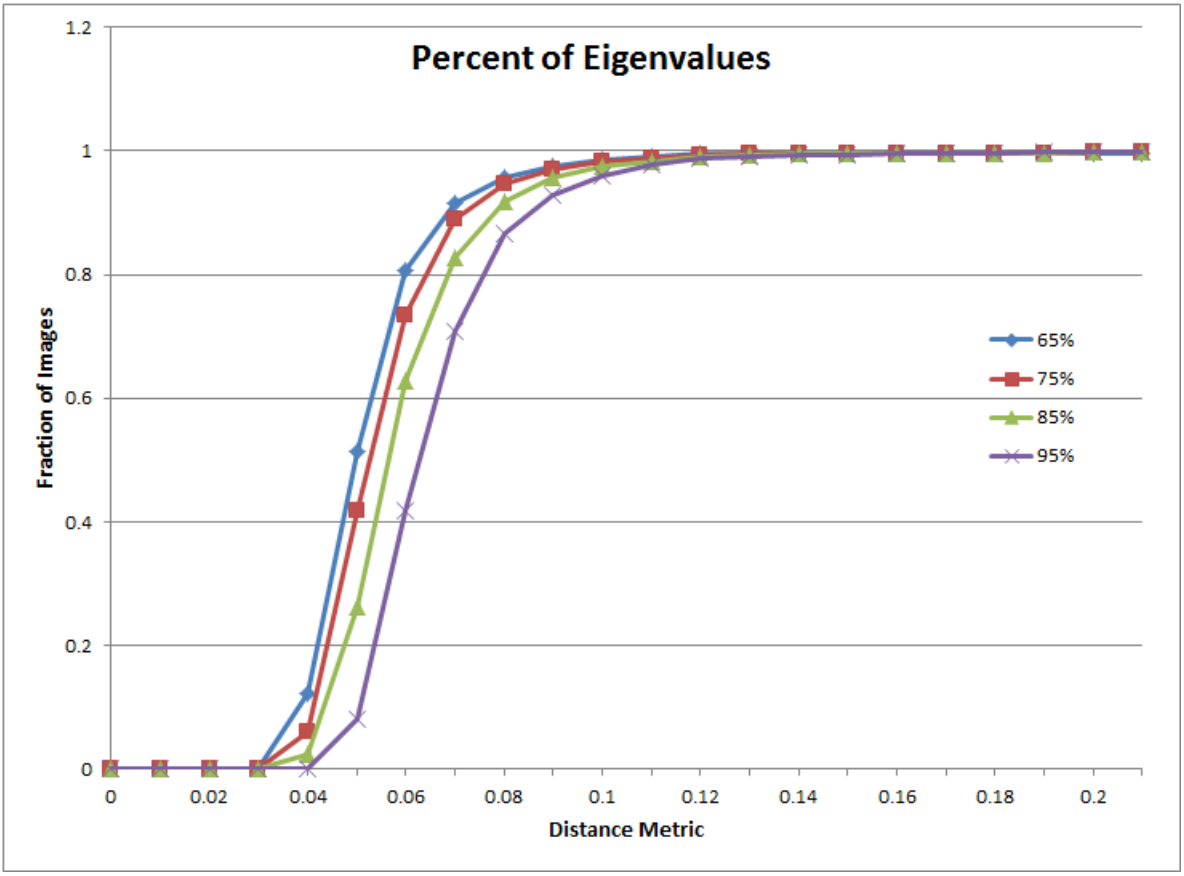


Figure 4.28: Percentage of eigenvalues applied to the system

age error per landmark from the top three algorithms. The spikes in the graph, i.e. points in the range 123-127 and 131-135 relate to anatomical landmarks representing the cheek bones of the face. Anatomical landmarks are typically more difficult to locate with the ASM methodology because their mean profiles are not defined by a strong mathematical gradient that the profile model can match to during image search. Instead, they rely heavily on a correct shape initialization and the PDM conforming them to the general region of the facial feature they define. I will touch on this dilemma again in my conclusions.

4.4 Parallel Speedup Test

The final experiment relates to the computational efficiency of the software. Figure 4.31 shows the search time speedup as a result of including additional OpenMP threads. There is not a linear gain from the addition of a new thread because of a bottleneck that is caused by a critical section surrounding the face detection function. From the way the

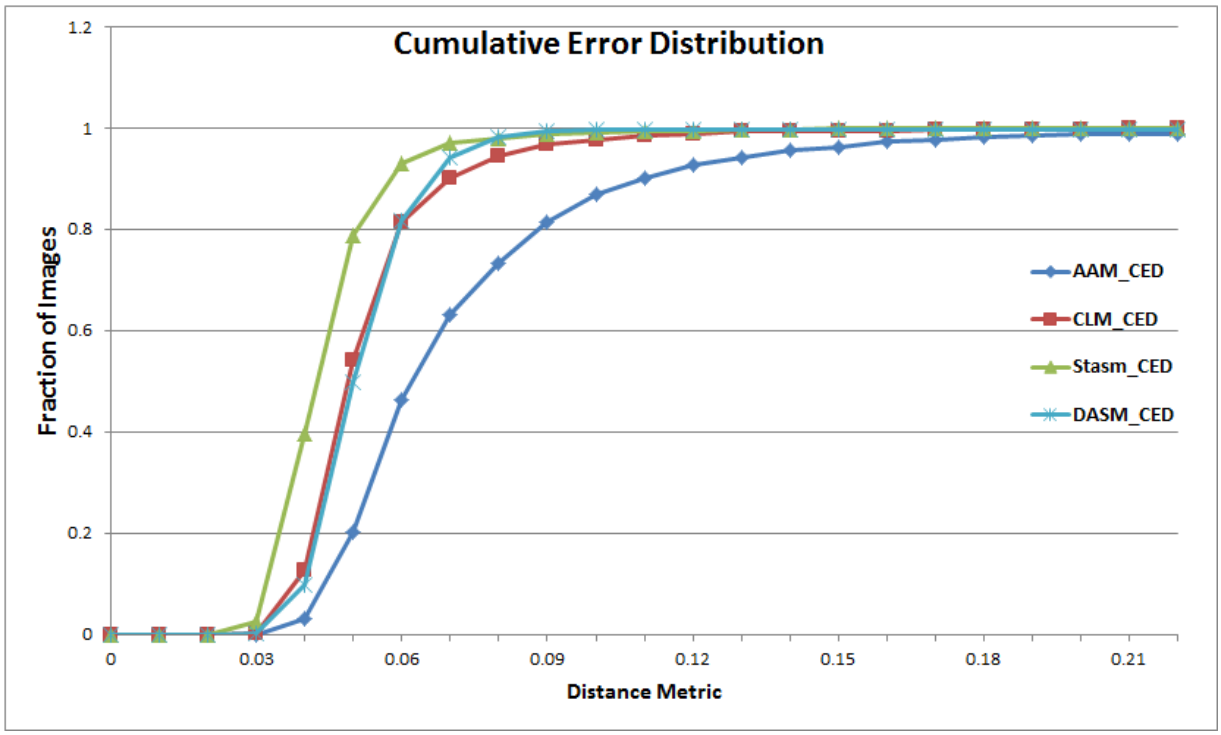


Figure 4.29: Cumulate Error Distribution of DASM against similar landmarking techniques

OpenCV detector is currently implemented within the software, it is not safe to use in a multi-threaded environment. To prevent irregular program behavior, I enforced that detection functionality be done within a critical section, meaning only one thread has access to the routine at a time. So, what we are seeing from the graph is that although there is a significant speedup with the inclusion of more threads, many of the threads are forced to wait on the other threads to complete the face detection before they can continue processing. Obviously, if a detector is implemented in such a way that it proves to be thread safe, the critical section can be removed, and a more linear trend should be evident with the addition of each thread. It is also clear from the chart that stacking the model effectively doubles the search time. This should be taken into account if speed of computation is more important to the user than higher accuracy of landmarks.

4.5 Profile Face Experiment

To prove that DASM is not limited to frontal faces and dense point schemes, I explored a secondary test on profile faces with a 14-point scheme. The point scheme was

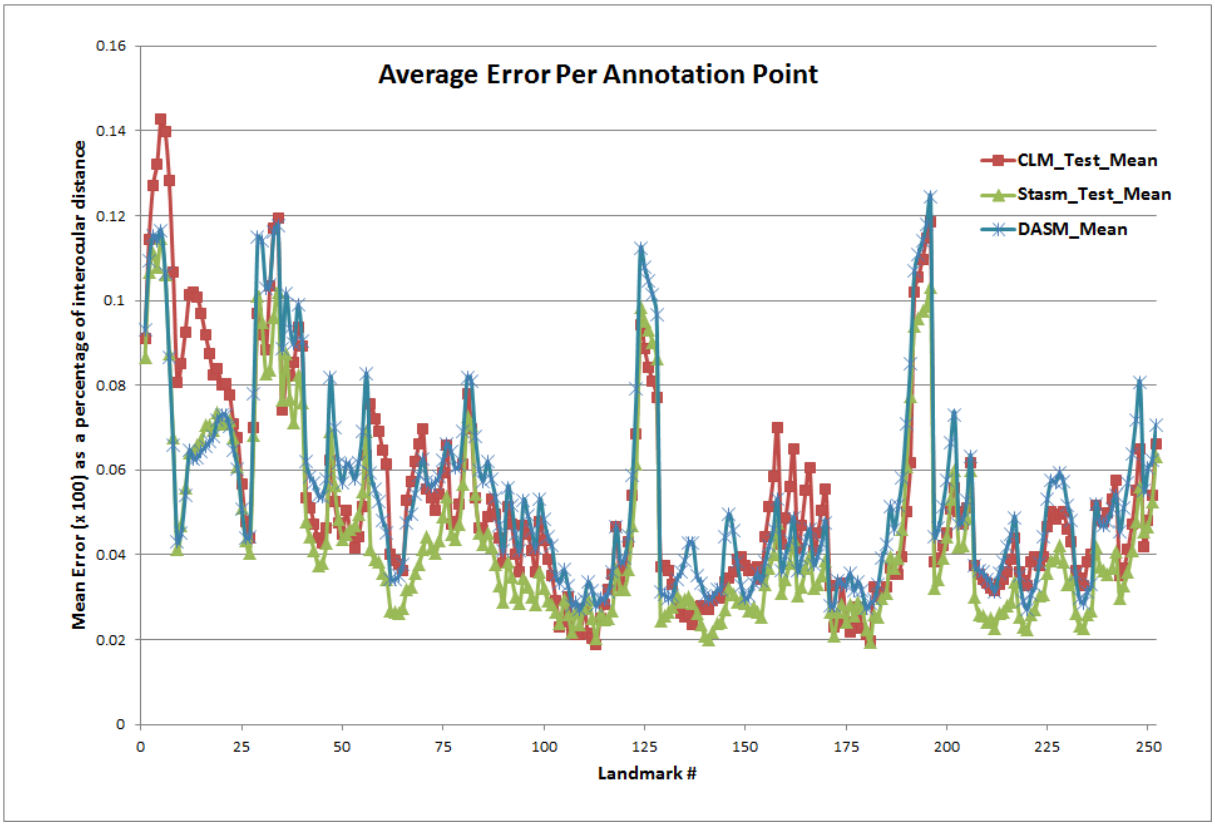


Figure 4.30: Comparison of the average error per landmark from CLM, STASM, and DASM on the general model

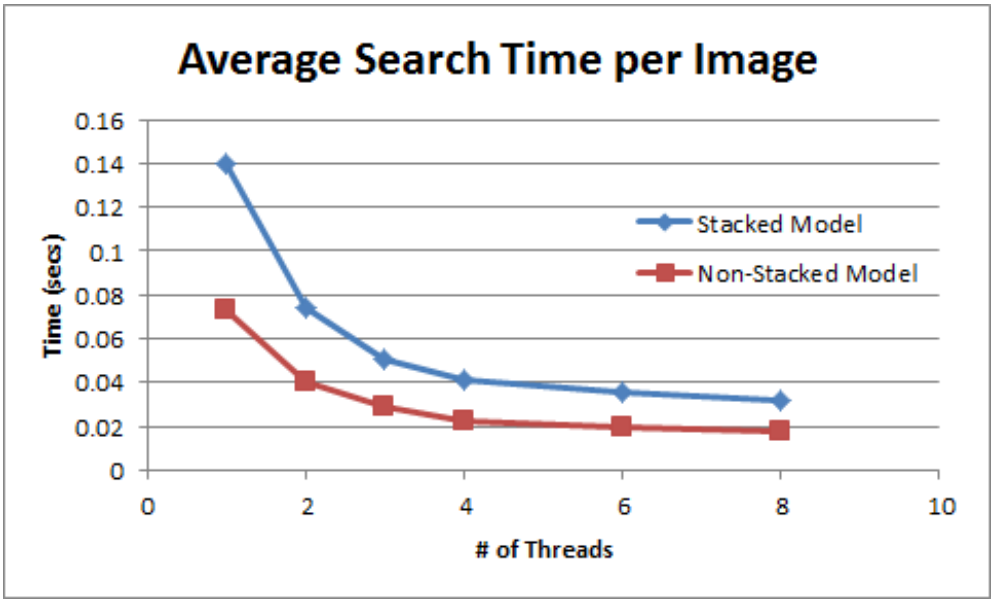


Figure 4.31: Tested on the general model with 4 pyramid levels and a profile length of 11.

abstracted from a study by Bottino et al.[2]. A few minor adjustments were made to the two-part scheme, specifically the addition of pseudo points on the nose and along the con-

tour of the chin. However, the majority of the anatomical facial landmarks, i.e. the exocantion, alare, and cheilion, were held consistent with the study. 124 images were extracted from the SCface database which exhibited 90 degree pose and minimal occlusions. A few samples, both good and poor results, are shown in Figure 5.32. It was evident from examining the results that the lack of training data, in comparison to the general model's 1150 images, had a significant impact on the accuracy of the produced landmarks. It would be interesting to see if the accuracy of the landmarks increases if additional landmarks are included to the scheme or the size of the training set increases.

Chapter 5: Conclusion

This thesis proposes a modern approach to solving the problem of automatic landmarking by introducing an open source application framework, designed to be efficient in terms of accuracy and speed, and easily extensible by end-users. This project reproduced many of the contributions of Stephen Milborrow's STASM [35], while maintaining key elements of Cootes et al. Active Shape Models. As is evident by the results of the experiments, the software is competitive with current published methods for locating facial landmarks. The major contributions of this work are articulated below:

1. Designing and implementing a software platform which enables researchers the ability to experiment and develop extensions to the ASM algorithm
2. Incorporating parts information to improve upon the landmark search directions
3. Making use of well-supported external libraries to support computation tasks
4. Creating a multi-threaded environment to increase the computational efficiency
5. Providing a robust solution to object detection by harnessing the power of abstraction

5.1 Limitations of ASMs

Active Shape Models were originally designed with the purpose of analyzing simple shape structures such as internal organs or bone structures. They have been extended to be an effective method of analyzing human faces, under certain conditions. For instance, it is possible to use ASMs to accurately locate facial features of individuals within images, as demonstrated in this work, such that each face is similar in terms of expression and pose. Visual inspections have shown that ASMs struggle when forced to interpret faces with expressions which vary from the training data, i.e. smiling, laughing, sad, surprise, etc. It would be interesting to train specialized models related to each subset of expression, as opposed to a general expression model. We can safely assume that individual models would perform better when analyzing faces related to their own specific type expression.

The difficulty would lie in knowing which model to use when applied to an unconstrained environment such as video analysis, where individual expressions are likely to change over short periods of time.

The second notable limitation of ASMs are their inability to handle minor pose variations. This is a natural drawback of attempting to analyze a 3-dimensional object in 2-dimensional space. An object's pose in three dimensions is defined by its yaw, pitch, and roll. Of the three, yaw and pitch have greatest impact on an object's 2D shape. For example, a face viewed at a 90 degree angle versus a frontal view, has nowhere near a similar shape. Roll, or rotation of the face, can also introduce complications during image search. During training, the model's mean profiles are accustomed to upright features which produce a certain gradient. If a face is rotated by 45 degrees, the profile gradients would not match to the model's profile features. Empirical testing showed that even minor facial pose variations caused drastic errors in landmark localization. However, if we have knowledge of rotational pose of the face, it would be possible to rotate the image itself to realign the actual features with the model's. The result would likely be a better overall landmark fitting. The challenge is interpreting the rotation of the face prior to processing.

There is a potential solution to this problem if we have known locations of the eye coordinates and nose tip, or some other well distinguished feature of the face. Advanced face detectors, such as the one provided by PittPatt, have this functionality built in. With three well defined points on the face, we can determine the initial scale, rotation, and translation of the mean shape within the image. Since the algorithm relies heavily on a close initial approximation of the location and size of the object in an image, employing this method could drastically improve automatic landmarking.

5.2 Final Thoughts and Suggestions

It may be in the best interest of the user to train multiple models with varying model parameters, and test to see which model produces the most accurate results with the data at hand. Adjusting certain parameters such as the profile lengths, even by minor amounts, can have drastic effects on the quality of the landmarks. Secondly, if you know you are

going to be analyzing high quality images, incorporate these same types of images into the training set. For instance, it doesn't make sense to have a training set of very high quality images if you are going to be searching on images extracted from a low resolution webcam or security camera. Remember to include an adequate amount of images to the training set, in order to describe all of the typical shape variations of the objects shape. Finally, for best results when working with faces, train on a single form of expression and try to keep a uniform pose.

5.3 Future Work

DASM is an open-ended project with great potential for growth and improvement. The software itself is by no means error-proof, nor has it been rigorously tested or debugged. There are a number of areas which would benefit from overhaul in design, which could potentially result in better memory management and more code reuse. There is also an immediate appeal to build the software in the form of a dynamic library as opposed to a standalone application. The current version of the program is designed to process a directory of images, regardless of the volume. A library could potentially supplement other applications, such as real-time video processing, by providing fast and accurate landmarks of images. It is a constant struggle when designing research applications to maximize the usability, maintainability, extensibility, and readability of the software. One of the foremost reasons for developing DASM, was that STASM, to a certain degree, suffered from poor readability and extensibility. With support from the research community, DASM has the potential to be a highly valuable application.

Given another year of research, there are a number of additional experiments that I wish I could have run. It would be interesting to evaluate the general model on a variety of landmark schemes beyond the dense 252-points. Other schemes, such as the 69- or 127- point scheme, are similar in nature to the 252-points, but disregard many of the facial features which are defined by anatomical landmarks. It is likely that these schemes would outperform the 252-point scheme based on how much greater the error associated with anatomically defined landmarks is compared with features defined by strong image

gradients. Furthermore, the optimal model used in this experiment was based on only a few tests related to the varying four major model parameters. Extensive testing of every possible adjustment to the model would almost positively result in more accurate rendered landmarks. It would also be intriguing to explore how well the algorithm performs on objects besides faces.

As far as extensions to the algorithm, it would be interesting to test different feature descriptors, such as an a Local Binary Pattern (LBP), in place of the 1D and 2D profiles. LBP's are isotropic in nature, which is a distinct advantage over the 2D search profiles that rely on the features having the same orientation as the model's profiles. The algorithm could also be extended to look at the global texture of the shape, as proposed by Sun and Xie [53]. A similarly titled project Boosted Dynamic ASM (BDASM), showed increased performance in noisy image environments when incorporating the Gentleboost regression algorithm [8].

Figure 5.32: Examples of landmarks (good and bad) produced by DASM on profile faces

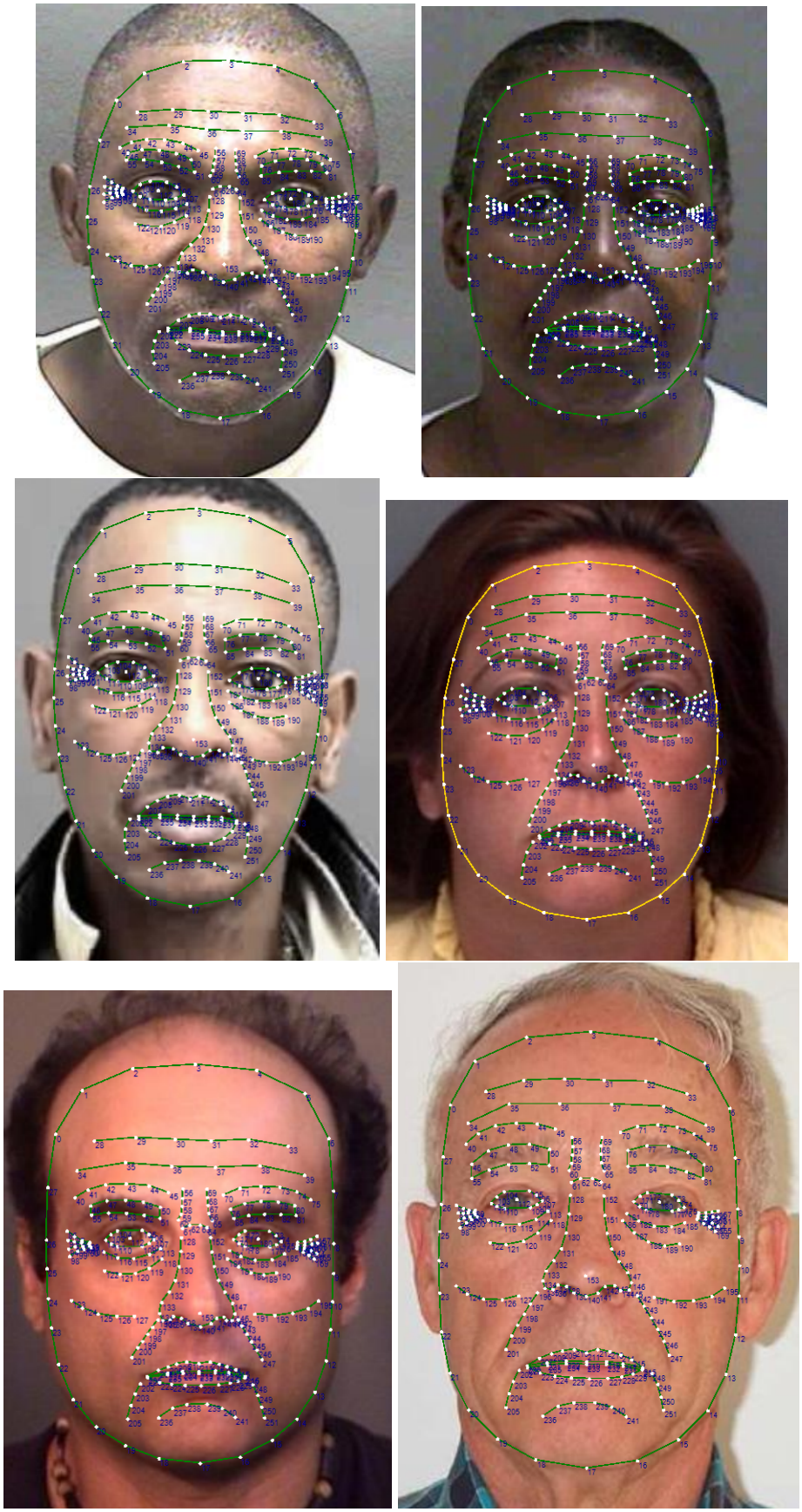


Figure 5.33: Examples of good landmarks produced by DASM on frontal faces

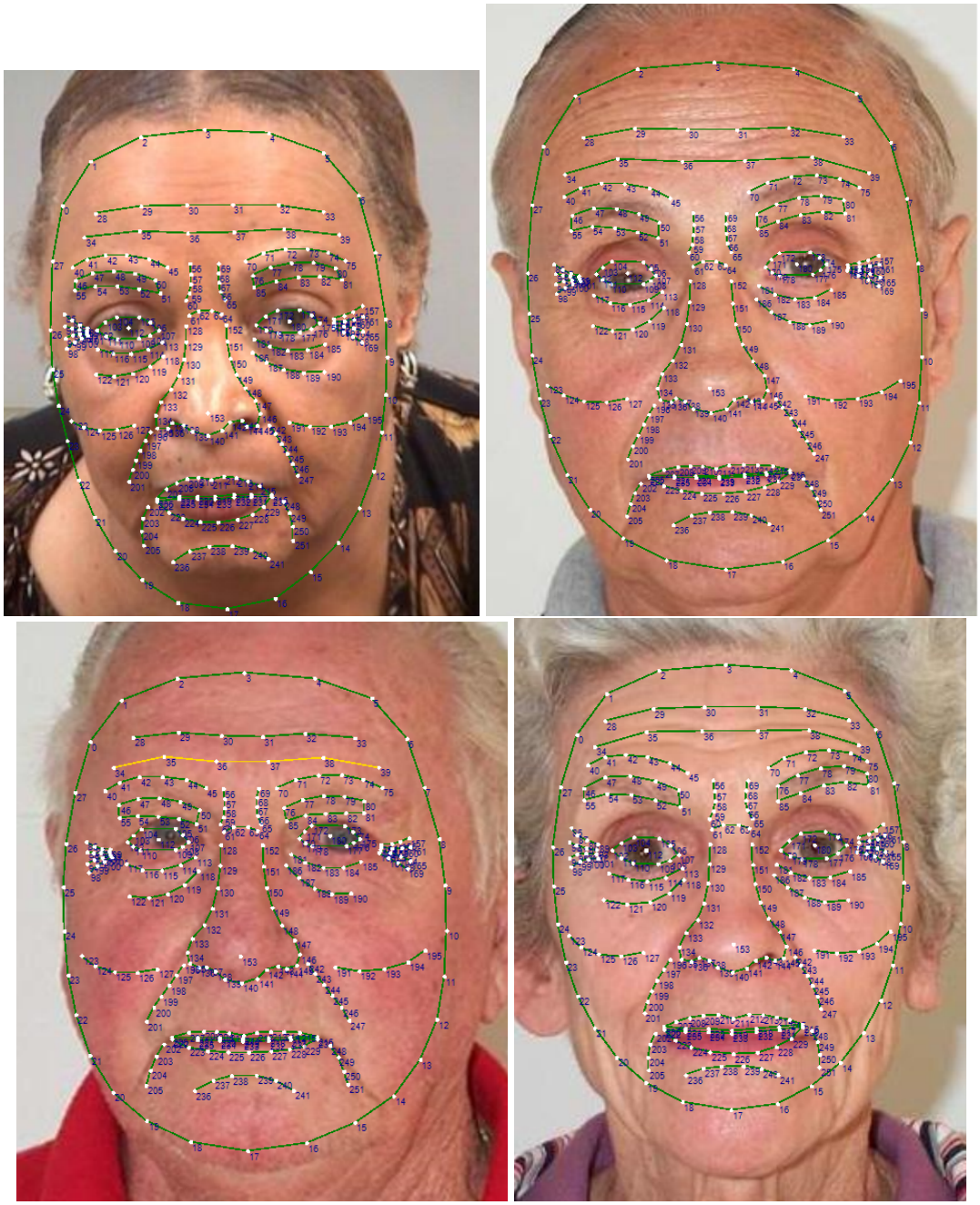


Figure 5.34: Examples of poor landmarks produced by DASM on frontal faces

Acknowledgments

I would like to acknowledge the following for their contributions and support throughout this work.

Amrutha Sethuram for her immense patience and guidance from the conception of this project to its completion, and everyday in between.

My friends Jeffrey Raynor and Michael Sodomsky for putting up with my antics in the lab.

Ben Barbour for his tutelage in the areas of OpenCV and C++, and for being an absolute debugging wizard.

Dr. Karl Ricanek for his everlasting support and enthusiasm over the past three years. Your guidance and mentorship has had an inimitable effect on the person that I am today, for which I will be forever grateful.

Finally, to my parents for their love and support over the past 24 years. Thank you for constantly pushing me to better myself and my education. This work is dedicated to you both.

References

[1] B. Barbour and K. Ricanek Jr. An interactive tool for extremely dense landmarking of faces. In Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications, page 13. ACM, 2012.

[2] A. Bottino and S. Cumani. Robust identification of face landmarks in profile images. In Proceedings of the 12th WSEAS international conference on Computers, ICCOMP'08, pages 107–114, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[3] G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

[4] P. Brigger, J. Hoeg, and M. Unser. B-spline snakes: a flexible tool for parametric contour detection. Image Processing, IEEE Transactions on, 9(9):1484–1496, 2000.

[5] L. G. Brown. A survey of image registration techniques. ACM Comput. Surv., 24(4):325–376, Dec. 1992.

[6] R. Brunelli. Template matching techniques in computer vision. Wiley Online Library, 2009.

[7] C. Chen, Y. Chang, K. Ricanek, and Y. Wang. Face age estimation using model selection. In Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, pages 93–99, 2010.

[8] Y. Chen, X. Cai, and A. Sowmya. Boosted dynamic active shape model. In Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference, pages 215–220, 2009.

[9] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. ACM Computing Surveys (CSUR), 18(1):67–108, 1986.

[10] T. Cootes. An introduction to active shape models. Image Processing and Analysis, pages 223–248, 2000.

[11] T. F. Cootes, G. Edwards, and C. J. Taylor. Comparing active shape models with active appearance models. In British Machine Vision Conference, volume 1, pages 173–183, 1999.

[12] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In Computer VisionECCV98, pages 484–498. Springer, 1998.

[13] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Training models of shape from sets of examples. In Proc. British Machine Vision Conference, volume 9, page 18. Citeseer, 1992.

[14] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models: their training and application. Comput. Vis. Image Underst., 61(1):38–59, jan 1995.

[15] T. F. Cootes, C. J. Taylor, et al. Statistical models of appearance for computer vision. World Wide Web Publication, February, 2001.

[16] T. F. Cootes, C. J. Taylor, et al. Statistical models of appearance for computer vision. Imaging Science and Biomedical Engineering, University of Manchester, Manchester M13 9PT, UK March, 8, 2004.

[17] H. Eviatar and R. Somorjai. A fast, simple active contour algorithm for biomedical images. Pattern Recognition Letters, 17(9):969 – 974, 1996.

[18] M. Flickner, H. Sawhney, D. Pryor, and J. Lotspiech. Intelligent interactive image outlining using spline snakes. In Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on, volume 1, pages 731–735. IEEE, 1994.

[19] D. A. Forsyth and M. M. Fleck. Automatic detection of human nudes. Int. J. Comput. Vision, 32(1):63–77, Aug. 1999.

[20] M. Grgic, K. Delac, and S. Grgic. Scface — surveillance cameras face database. Multimedia Tools Appl., 51(3):863–879, Feb. 2011.

[21] W. E. L. Grimson. Object recognition by computer: the role of geometric constraints. 1991.

[22] G. Hamarneh. Active shape models, modeling shape variations and gray level information and an application to image search and classification. 1998.

[23] E. Hjelms and B. K. Low. Face detection: A survey. Computer Vision and Image Understanding, 83(3):236 – 274, 2001.

[24] M. Hoch and P. C. Litwinowicz. A semi-automatic system for edge tracking with snakes. The Visual Computer, 12(2):75–83, 1996.

[25] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust face detection using the hausdorff distance. In Audio-and video-based biometric person authentication, pages 90–95. Springer, 2001.

[26] P. Karaolani, G. Sullivan, K. Baker, and M. Baines. A finite element method for deformable models. In Proceedings of the Fifth Alvey Vision Conference, Reading, pages 73–78, 1989.

[27] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. INTERNATIONAL JOURNAL OF COMPUTER VISION, 1(4):321–331, 1988.

[28] A. Lanitis, C. Taylor, and T. Cootes. Recognising human faces using shape and grey-level information. In Third International Conference on Automation, Robotics and Computer Vision. Citeseer, 1994.

[29] A. Lanitis, C. Taylor, and T. Cootes. Recognising human faces using shape and grey-level information. In Proceedings of the British Machine Vision Conference, pages 327–336, 1994.

[30] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–900–I–903 vol.1, 2002.

[31] A. Martinez and R. Benavente. The ar face database. CVC Technical Report #24, June 1998.

[32] S. Menet, P. Saint-Marc, and G. Medioni. B-snakes: Implementation and application to stereo. In Image Understanding Workshop, pages 720–726, 9 1990.

[33] K. Messer, J. Matas, J. Kittler, and K. Jonsson. XM2VTSDB: The Extended M2VTS Database. In Audio- and Video-Based Biometric Person Authentication, 1999.

[34] S. Milborrow. Locating facial features with active shape models. Master's thesis, University of Cape Town, November 2008.

[35] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. ECCV, 2008.

[36] M. Minear and D. Park. A lifespan database of adult facial stimuli. Behavior Research Methods, Instruments, & Computers, 36(4):630–633, 2004.

[37] A. A. Nielsen. Analysis of regularly and irregularly sampled spatial, multivariate, and multi-temporal data. Science, 21(4):555–567, 1994.

[38] E. Patterson, K. Ricanek, M. Albert, and E. Boone. Automatic representation of adult aging in facial images. In IASTED06: Proc. 6th International Conference on Visualization, Imaging, and Image Processing, pages 171–176, 2006.

[39] P. J. Phillips, P. J. Flynn, K. W. Bowyer, R. V. Bruegge, P. J. Grother, G. W. Quinn, and M. Pruitt. Distinguishing identical twins by face recognition. In Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on, pages 185–192. IEEE, 2011.

[40] M. T. Pruitt, J. M. Grant, J. R. Paone, P. J. Flynn, and R. W. V. Bruegge. Facial recognition of identical twins. In Biometrics (IJCB), 2011 International Joint Conference on, pages 1–8. IEEE, 2011.

[41] K. Ricanek and T. Tesafaye. Morph: A longitudinal image database of normal adult age-progression. In Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on, pages 341–345. IEEE, 2006.

[42] K. Ricanek Jr. Variable lateral pose face recognition using anthropometric analysis. 1999.

[43] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. IEEE Trans. Pattern Anal. Mach. Intell., 20(1):23–38, Jan. 1998.

[44] M. Rudemo. Statistical shape analysis. i. l. dryden and k. v. mardia, wiley, chichester 1998. no. of pages: xvii+347. price: 60.00.isbn 0-471-95816-6. Statistics in Medicine, 19(19):2716–2717, 2000.

[45] H. Schneiderman, N. Nechyba, and M. Sipe. Pittpatt. 2010.

[46] A. Sethuram, K. Ricanek, J. Saragih, and C. Boehnen. Facial landmarking: Comparing automatic landmarking methods with applications in soft biometrics. In A. Fusiello, V. Murino, and R. Cucchiara, editors, Computer Vision ECCV 2012. Workshops and Demonstrations, volume 7584 of Lecture Notes in Computer Science, pages 290–299. Springer Berlin Heidelberg, 2012.

[47] A. Sethuram, J. Saragih, K. Ricanek, and B. Barbour. Extremely dense face registration: Comparing automatic landmarking algorithms for general and ethno-gender models. In Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on, pages 135–142, 2012.

[48] J. Shi, A. Samal, and D. Marx. Face recognition using landmark-based bidimensional regression. In Data Mining, Fifth IEEE International Conference on, pages 4 pp.–, 2005.

[49] J. Shi, A. Samal, and D. Marx. How effective are landmarks and their geometry for face recognition? Computer Vision and Image Understanding, 102(2):117 – 133, 2006.

[50] M. B. Stegmann. Active appearance models: Theory, extensions and cases. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, aug 2000.

[51] M. B. Stegmann. The AAM-API: An open source active appearance model implementation. In Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003, 6th Int. Conference, Montréal, Canada, LNCS 2879, pages 951–952. Springer, nov 2003.

[52] D. Stoyan. Bookstein, f. l.: Morphometric tools for landmark data. geometry and biology. cambridge university press, cambridge/new york/port chester/melbourne/sydney 1991, xviii, 435 s., 50.00; $ 89.95. Biometrical Journal, 35(4):512–512, 1993.

[53] C. Sun and M. Xie. An enhanced active shape model for facial features extraction. In Communication Technology, 2008. ICCT 2008. 11th IEEE International Conference on, pages 661–664, 2008.

[54] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. In Computer Vision, 1990. Proceedings, Third International Conference on, pages 606–615. IEEE, 1990.

[55] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings

of the 2001 IEEE Computer Society Conference on, volume 1, pages I–511. IEEE, 2001.

[56] J.-Y. Wang and F. S. Cohen. Part ii: 3-d object recognition and shape estimation from image contours using b-splines, shape invariant matching, and neural network. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 16(1):13–23, 1994.

[57] Y. Wang, K. Ricanek, C. Chen, and Y. Chang. Gender classification from infants to seniors. In Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on, pages 1–6, 2010.

Appendices A

Points File

```
version: 1
n_points: 252
{
57.3801 140.853
76.0201 115.507
106.65 102.423
142.387 98.4289
178.544 101.625
206.663 115.089
221.273 140.901
224.773 170.318
226.501 202.818
227.065 220.081
226.963 237.053
225.992 255.143
224.316 275.267
219.151 295.704
209.727 313.668
195.073 329.326
174.758 340.274
148.51 344.995
120.06 343.183
95.0579 335.855
74.2285 322.277
59.1148 303.808
50.0112 281.3
45.348 257.617
44.0923 240.324
44.2862 223.647
45.2722 208.865
49.3251 173.252
80.1329 141.608
105.141 135.414
129.903 133.319
154.691 133.089
178.439 134.054
200.151 139.735
71.6082 157.934
100.697 149.701
129.131 148.081
⋮
```

```
version: 1
n_points: 14
{
100.661 135.554
102.357 144.978
94.2352 153.029
85.497 166.946
96.5261 173.836
94.5639 182.574
97.692 187.407
94.0065 192.593
99.6991 201.126
99.1263 215.499
128.469 228.257
116.147 149.041
102.601 171.741
104.035 188.644
}
```

Example points files from the 252- and 14-point schemes

Appendices B

Parts File

```
⋮
part "M.Forehead" {
    indices(28,29,30,31,32,33)
    form open_boundary
        jagged
    point_density 1
}
part "B.Forehead" {
    indices(34,35,36,37,38,39)
    form open_boundary
        jagged
    point_density 1
}
part "L.AboveEyebrow" {
    indices(40,41,42,43,44,45)
    form open_boundary
        jagged
    point_density 1
}
part "L.Eyebrow" {
    indices(46,47,48,49,50,51,52,53,54,55)
    form closed_boundary
        jagged
    point_density 1
}
part "L.MidBrow" {
    indices(56,57,58,59,60)
    form open_boundary
        jagged
    point_density 1
}
part "B.MidBrow" {
    indices(61,62,63,64)
    form open_boundary
        jagged
    point_density 1
}
part "R.MidBrow" {
    indices(65,66,67,68,69)
    form open_boundary
        jagged
    point_density 1
}
⋮
```

Example subsection of the 252-point parts file

Appendices C

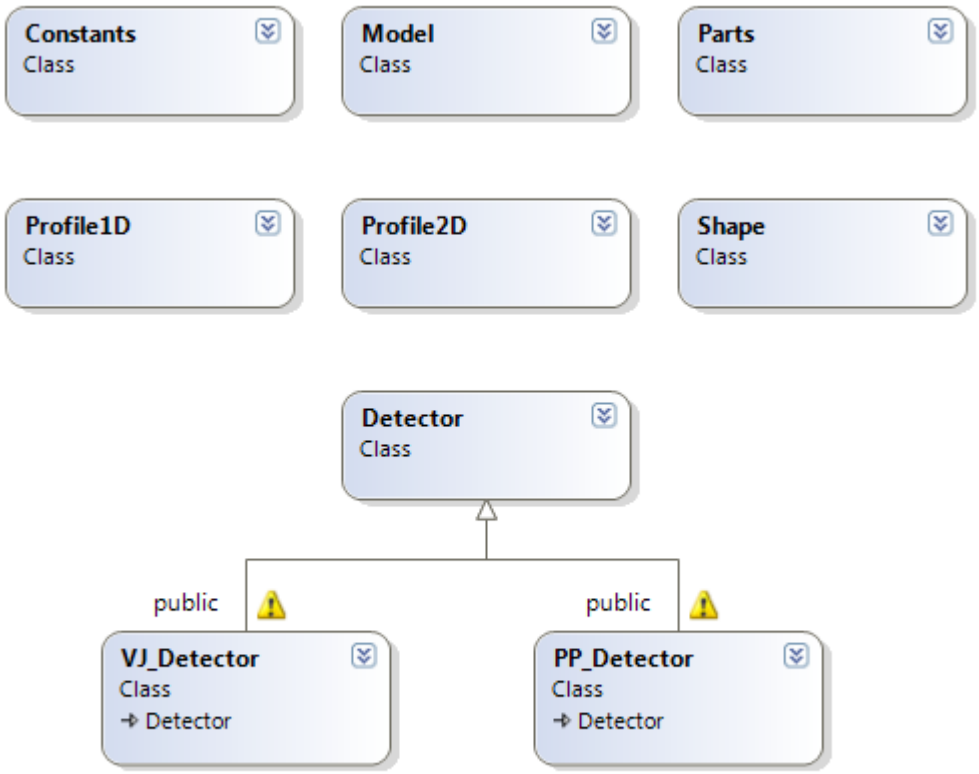
Configuration File

```
[DASM_Config]

# Configuration File for DASM Training

# search/train  = true (One of the two must be true but not both)
# input-dir     = Full path of the directory of training images/points
#                   or images to be searched
# model-path    = Full path to where the model will be output
# parts-path    = Full path to the parts file
# detector-**   = Full path to the necessary files for the object detector
# profLength1d  = Length of the 1D profile vector (must be odd value)
# profLength2d  = NxN size of the 2D profile matrix (must be odd value)
# num1DLevels   = Number of pyramid levels to be searched by 1D profiles
# num2DLevels   = Number of pyramid levels to be searched by 2D profiles

train = true
input-dir = C:\Users\IISIS\Documents\PAL
model-dir = C:\Users\IISIS\Documents\PAL
parts-path = C:\Users\IISIS\Documents\face252.parts
detector-vj = C:\Users\IISIS\Documents\haarcascade_frontalface_alt2.xml
profLength1d = 9
profLength2d = 11
num1DLevels = 1
num2DLevels = 3
```

```
[DASM_Config]

# Configuration File for DASM Image Search

# search/train  = true (One of the two must be true but not both)
# input-dir     = Full path of the directory of training images/points
#                   or images to be searched
# model-path    = Full path to the model file to load for search
# detector-**   = Full path to the required files for the object detector
# out-dir       = Full path to where the points will be output after
#                   search (Default current working directory)
# searchParam   = 1 for 1d profile search only, 2 for 1d/2d profile
#                   searches (Default 2)
# eigPercent    = % (1-100) of the eigenvectors to use for search
#                   (Default 75)
# stacked       = 1 to perform single model search, 2 to perform
#                   stacked model search (Default 1)

search = true
input-dir = C:\Users\IISIS\Documents\PAL
model-path = C:\Users\IISIS\Documents\asm.bin
detector-vj = C:\Users\IISIS\Documents\haarcascade_frontalface_alt2.xml
out-dir = C:\Users\IISIS\Documents\PAL_Searched
searchParam = 2
eigPercent = 75
stacked = 2
```

Appendices D

Class Diagram



A simplified class diagram of the DASM application. The Detector class is abstract. The two subclass examples reflect the OpenCV Viola-Jones based detector and the PittPatt detector.

Appendices E

Model File

```
DASM Model Produced: 2013-Jun-26 15:35:58

---------- Model Attributes ----------
{
nPoints 252
nLevels1dProfile 1
nLevels2dProfile 4
1dProfileLength 9
2dProfileLength 11
DetectorRefWidth 219
DetectorTranslation X -0.000256469
DetectorTranslation Y 0.0028305
nParts 37
}
---------- Parts Info ----------
{ Part 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Closed Boundary 1
}
{ Part 1
28 29 30 31 32 33
Closed Boundary 0
}
{ Part 2
34 35 36 37 38 39
Closed Boundary 0
}
{ Part 3
40 41 42 43 44 45
Closed Boundary 0
}
:
```

```
⋮

{ Part 36
248 249 250 251
Closed Boundary 0
}
---------- Avg Model Shape ----------
{
-78.0615 -81.5262
-62.1194 -101.849
-35.7962 -113.068
-3.03564 -116.421
29.3961 -114.131
56.7759 -104.451
74.1211 -84.8893
83.9172 -57.3463
89.5304 -25.5006
90.7075 -6.07438
90.1311 13.1487
88.014 33.1493
84.3788 53.4006
77.6871 73.4148
66.8889 92.0086
51.6491 107.665
31.4969 120.415
4.36985 125.593
-23.5889 121.825
-45.684 110.281
-62.2457 94.8817
-74.181 76.4431
-81.9379 56.142
-86.6297 35.3815
-89.5689 16.0464
-90.9897 -3.21816
-90.6786 -22.9744
-86.4531 -54.7781
⋮
```

⋮

```
46.9876 51.6791
49.5172 58.8267
41.9807 69.5944
44.9313 76.3823
46.1068 83.9786
45.5061 91.6213
}
---------- Eigen Values ----------
{ 136
[1324.4387; 1088.9042; 531.57056; 483.98535; 459.80508; 358.1214; 2
}
---------- Eigen Vectors ----------
{ 136 504
[-0.042822968, 0.064163625, -0.029381711, 0.071023509, -0.019832613
  -0.05724135, 0.014059714, -0.035623722, 0.012491478, -0.016504342
  0.099747077, 0.1252709, 0.089031659, 0.17184383, 0.057946514, 0.1
  0.034034431, -0.045806941, 0.018854421, -0.075534202, 0.01170971,
  0.015496667, 0.090695605, 0.0099543026, 0.10573915, -0.0015409377
  0.099607304, 0.022362174, 0.11997289, 0.11816192, 0.065860234, 0.
  -0.017545743, -0.032713693, 0.018047925, -0.016898682, 0.03405114
  -0.0023870878, 0.11318485, 0.00046907185, 0.073090784, -0.0007623
  0.036633413, 0.085420452, 0.0077895955, 0.084292494, -0.006312453
  0.031391058, 0.03824503, 0.022901358, 0.011069084, 0.023279706, -
  0.08422558, 0.034433376, 0.071905486, 0.076322965, 0.02325378, 0.
  -0.00049223105, 0.015572099, -0.014567844, 0.0026183862, -0.02616
  0.014956752, -0.018718719, 0.057773568, -0.01055958, 0.083890453,
  -0.038439851, 0.0024514478, -0.063342385, -0.012648759, -0.062399
  -0.017147377, -0.028762145, -0.010819895, -0.071263425, 0.0077553
  -0.059091169, 0.034273025, -0.083915375, -0.035226054, -0.0331480
  0.1135044, -0.098935723, 0.13274805, -0.040471371, 0.059341189, -
  -0.00082093122, 0.10122718, 0.0041326084, 0.11212566, 0.035439942
  -0.056116764, -0.0057830759, -0.069546029, 0.018699523, -0.101816
  0.018422205, -0.057783164, 0.015792625, -0.07551302, 0.026992049,
  0.09634582, -0.070613578, 0.098228626, -0.030121915, 0.038570553,
  0.068372667, -0.035527773, 0.13154349, -0.031807933, 0.14054053,
  -0.088838294, 0.047585059, -0.061263047, 0.051269028, -0.02477482
  0.0017474253, 0.048811544, -0.092826985, 0.02107862, -0.11673795,
  -0.023298547, 0.032096334, -0.064158998, 0.036423519, -0.08478688
  0.063116737, -0.046773378, 0.064940311, -0.048099719, 0.065138653
  0.063020736, -0.038391791, 0.04954331, 0.0024125285, 0.0060025086
  0.085451625, -0.13527574, 0.14225292, -0.078506291, 0.13161111, -
  -0.012033871, 0.083774984, -0.077638276, 0.00062554894, -0.096970⋮
```

```
⋮
}
---------- Mean Profiles 1D ----------
--! Level 0 !--
{ Pt 0
0.160371 0.140462 0.173727 0.165334 0.143675 0.14312 0.138811 0.124782 0.119938
}
{ Pt 1
0.202807 0.139897 0.186157 0.156801 0.100822 0.128979 0.0939322 0.079692 0.0566
}
{ Pt 2
0.187006 0.199081 0.193273 0.137158 0.114578 0.131719 0.0861956 0.101518 0.0826
}
{ Pt 3
0.114271 0.100662 0.072543 0.0497241 0.0449085 0.0418071 0.00933947 -0.00842069
}
{ Pt 4
-0.0431774 -0.0842271 -0.124245 -0.107689 -0.103093 -0.139928 -0.135226 -0.1838
}
{ Pt 5
-0.0774285 -0.0696789 -0.106803 -0.120788 -0.102838 -0.162441 -0.144481 -0.1895
}
{ Pt 6
-0.112599 -0.100598 -0.14473 -0.14125 -0.145362 -0.166742 -0.156585 -0.164028 -
}
{ Pt 7
-0.102292 -0.137168 -0.156936 -0.185141 -0.192934 -0.213332 -0.16849 -0.170707
}
{ Pt 8
-0.0575161 -0.0762276 -0.13175 -0.184727 -0.235906 -0.256925 -0.178869 -0.11598
}
{ Pt 9
-0.126783 -0.119498 -0.134241 -0.152223 -0.196516 -0.205891 -0.148905 -0.078815
}
{ Pt 10
-0.0966604 -0.0982339 -0.103845 -0.115281 -0.117065 -0.0764334 -0.0214729 0.015
}
{ Pt 11
-0.0722099 -0.0802575 -0.0827105 -0.106813 -0.0940321 -0.0678482 0.0222489 0.05
}
{ Pt 12
⋮
```