University of Colorado at Colorado Springs

Masters Thesis

Adapting GRAB Features for Unconstrained Face Recognition

Author:
Ethan Rudd
B.S., Trinity University, 2012

Supervisor:
Dr. Terrance Boult

A thesis submitted to the Graduate Faculty of the University of Colorado at Colorado Springs in partial fulfilment of the requirements for the degree of Master of Science

in the

Department of Computer Science
College of Engineering and Applied Sciences

November 2014

This thesis for the Master of Science degree by

Ethan Rudd

has been approved for the

Department of Computer Science

by

Dr. Terrance Boult (Advisor)

Dr. Jugal Kalita

Dr. Chuan Yue

Date

UNIVERSITY OF COLORADO AT COLORADO SPRINGS

Abstract

College of Engineering and Applied Sciences

Master of Science

Adapting GRAB Features for Unconstrained Face Recognition

by Ethan Rudd

In this thesis we research and develop approaches for incorporating intra-image scale information into grid-based feature vectors for facial recognition. We use GRAB (Generalized Regions Assigned to Binary) as a base feature type and estimate intra-image scale using a rigid 3D model and an active appearance model (AAM). We assess our results using the View 2 protocol for the Labeled Faces in the Wild (LFW) dataset. A survey of face shape and texture modeling in the context of recognition is included.

Acknowledgements

I would like to thank my graduate committee, Dr. Chuan Yue, Dr. Jugal Kalita, and especially my advisor Dr. Terrance Boult. I would like to thank Securics Inc. for providing resources and funding for this work. Finally, I would also like to thank all friends and family who gave me moral support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Although face recognition technologies have gained increased attention and accuracy in recent years, unconstrained face recognition still has limited application due to difficulties that have yet to be overcome. These difficulties stem from a wide range of factors, including but not limited to, scale variations between gallery and probe, blur, noise in illumination intensity, fiducial point detection and alignment errors due to variations in pose and expression, occlusions, and classification algorithm limitations. Put mathematically, unconstrained face recognition requires minimizing noisy variance while maximizing discriminative variance. To get an idea of the unconstrained problem, consider figure 1.1: On the left hand side are unconstrained images of Angelina Jolie taken from the Labeled Faces in the Wild (LFW) [10] dataset. On the right hand side are images of different individuals from a constrained gallery. A machine learning algorithm could do a reasonable job discriminating between a match and a non-match of a constrained probe image to an identity in the constrained gallery by simply examining the relative locations of the images in a feature space built from raw pixels or histograms of pixels. However, given any of the unconstrained images on the left hand side, any person of similar pose, albedo, expression, hairstyle, sunglasses, makeup, etc. could easily exhibit less per-pixel difference with a given image of Angelina Jolie than two unconstrained images of Angelina Jolie with each other.

Work to address accuracy limitations due to unconstrained factors must necessarily be conducted across multiple stages of the recognition process. Although what constitutes the recognition process per se is implementation dependent, all face recognition implementations have two components in common: feature extraction and classification. At a high level, face recognition is simply another machine learning problem, in which the feature vectors extracted from a set of gallery images, along with their corresponding class labels, are submitted to a learning algorithm which learns the classifier that, by some

FIGURE 1.1: A constrained gallery of faces and an unconstrained gallery of faces. Constrained gallery courtesy of [1]. The constrained faces are taken from the Labeled Faces in the Wild (LFW) dataset.

FIGURE 1.2: Which of the two images is more similar? Humans recognize the pair on the right as having the same identity. In a pixel-wise feature space, however, the pair on the left is closer by a Euclidean metric even though the facial identities are quite different.

mathematical criterion, best separates classes within the training data. During testing, feature vectors from probe images returned by the feature extractor are submitted to the learned classifier.

Improvement of learning algorithms is a relevant and important area of research both within and far beyond the domain of face recognition. Regardless of the choice of learning algorithm and classifier representation, however, classification performance is contingent both on the statistical representation of the training data and on the statistical relevance of features used for classification. Attempts to bolster unconstrained face recognition features are numerous, largely due to the number of free parameters.

A popular means of addressing scale variations, for example, is to incorporate multiple scales into the feature vector. This can be done with both grid-based 2D features, such as GRAB (Generalized Regions Assigned to Binary)[11, 12], in which the feature vector consists histograms of texture sampled regions over an image at multiple scales, or keypoint-based features, such as SIFT (Scale Invariant Feature Transform) [13] and SURF (Speeded-Up Robust Features)[14], in which keypoints are isolated via a difference of Gaussian convolutions in scale-space, noise and edges are removed via contrast

thresholding, and keypoints between images are matched and compared by local gradient behavior. Disadvantages of GRAB and similar grid-based methods include sensitivity to alignment and occlusions and the fact that additional scales increase the length of the feature vector and therefore classification time complexity. Disadvantages of keypoint methods include computation costs and the dearth of keypoints in blurry images. Grid based features can be made more robust to illumination and pose variations via affine normalizations based on detected fiducial coordinates and lighting normalizations such as self quotient image (SQI) [15]. However, pose normalization does little to counteract out of image plane rotations. Such rotations are also a problem for keypoint methods, since keypoints vanish due to the self-occluding geometry of the human face.

Many of the illumination-robust features in the literature rely on combining multiple gallery images of a single subject into a relatively illumination invariant subspace and projecting each probe image onto that subspace for comparison. These approaches have a major drawback in that they assume the availability of multiple gallery images for each subject, sometimes with specific lighting characteristics. Such an abundance of favorable images is improbable in many of the unconstrained face recognition applications in which illumination irregularities are especially problematic. Certain quotient image techniques, however, such as SQI (Self Quotient Image) and its spinoffs [15, 16] exploit the fact that illumination noise is of low spatial frequency with respect to identity information within the image[1], and can largely be overcome by dividing the image in question by a Gaussian convolution on that image, which has the effect of a high-pass filter, removing the low spatial frequency components.

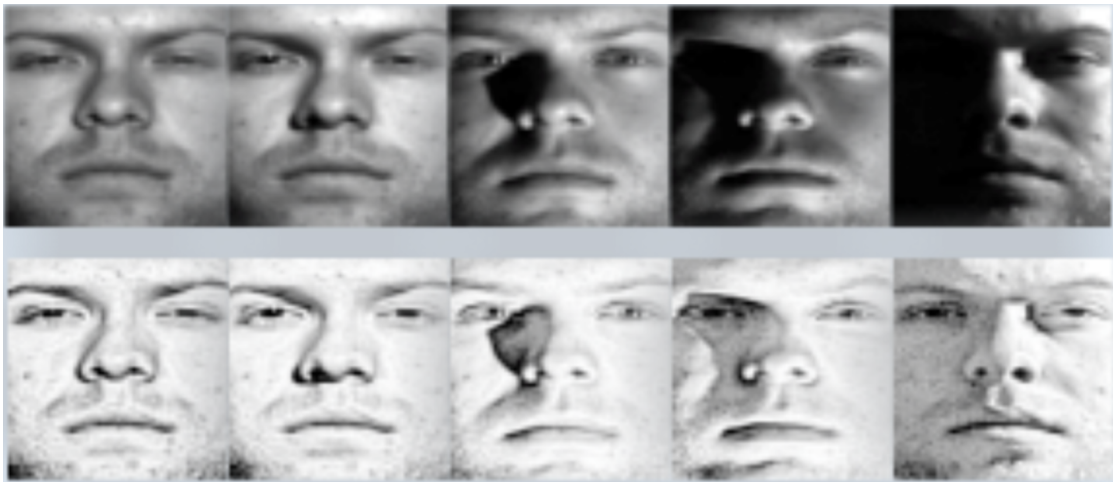

FIGURE 1.3: Examples of self-quotient image (SQI) illumination normalization from [2].

Associate-predict models, introduced in [17] by Yin et al., address both the problems of pose and illumination variations in a novel way: Based on the observation that variations

[1]This is provable by spherical harmonics analysis.

in pose and illumination explain a greater fraction of variance within images than do identity related variations[2], appearance prediction associate-predict models estimate the pose and illumination of both probe and gallery images as the closest statistical match to the average of all poses and illuminations within the MULTI-PIE dataset (assumed disjoint from both probe and gallery). If the illumination and pose of probe and gallery are sufficiently close (i.e., within a specified statistical threshold), probe and gallery are simply submitted to the classifier. Otherwise, associate-predict partitions both gallery and probe image into regions, and looks for the closest match subject for the same region for the estimated pose and illumination within the MULTI-PIE dataset. Once the subject is found, for gallery and for probe, the gallery region is replaced by the region of its closest MULTI-PIE match for the probe's pose and lighting. A dual procedure is carried out for the probe. The probe region modified to match the gallery in both pose and illumination is compared to the original gallery region. Likewise, the modified gallery region is compared to the original probe region. A weighted sum of the similarity between gallery and probe is submitted to the classifier for that region.[3] The same process is carried out for the entire probe image. For gallery and probe images of sufficiently different pose or lighting, associate-predict effectively obtains an approximation to the original probe (gallery) image normalized in illumination and pose to the gallery (probe) image.

2D and 3D model based methods take an entirely different approach, attempting to synthesize shape and texture representations of the face in question from one or more input images. Models can then be used for recognition by fitting them to the gallery/probe faces in question and generating new virtual images normalized for pose, lighting, and expression for 2D feature construction, using model parameters themselves as features, or using the model to augment 2D features. While 3D methods can model the full 3D face and environment parameters, including camera angle, focal length, location of the light source, and illumination model, 2D models commonly use machine learning in conjunction with interpolation and warping schemes to model in-plane rotations. In some respects, 2D and 3D models are the most capable methods of approaching unconstrained face recognition, but they have several disadvantages: First, the quality of the fit depends on the quality of the gallery and probe images themselves as well as how the model is constructed: a popular approach to model construction seen in ASMs (Active Shape Models), AAMs (Active Appearance Models), and 3DMMs (3D Morphable Models) is to represent models as a linear combination of eigenfaces decomposed from a ground-truth dataset of shape and/or texture. The dataset therefore must be representative of the

[2]This can be observed in principal components analysis, when discarding the component corresponding maximum magnitude eigenvalue yields superior recognition performance.

[3]This weighting corresponds to the statistical similarity between the approximated gallery (probe) region and the original gallery (probe) region.

modes of variations seen in the face images to be classified. If the model's basis shapes and textures are not representative, or the gallery and probe images are blurry, then model fitting will be poor. Second, although the face can be well well represented as superpositions of eigenfaces, other aspects of the human head, including hair, eyeglasses, and facial jewlery cannot, and can induce errors in the recognition algorithm when fitting the model to images. Finally, accurate fits of sophisticated model representations to images demand hefty serial computation costs. Although 2D ASMs and AAMs can be fit to images in real-time, accurate 3DMM representations cannot.

In this thesis, we discuss our work to address two correlated aspects of unconstrained face recognition: namely pose and scale variations. Unlike [11], which addresses scale differences between entire gallery/probe images, we address local scale variations within images which exist due to pose differences between the faces in the images. Our aim is to develop grid-based features which are more resilient to pose variations at low computational expense by leveraging 2D and 3D models. To this end, we implement two novel approaches: one which fits a generic 3D model to the image in question and uses the relative scale of the model points to imbue the feature vector with rough scale information and face location information. The second approach uses a 2D active appearance model to find mesh points on images and using these mesh points adjust the local scale at which GRAB features are extracted. The remainder of this thesis is structured as follows: Chapter 2 discusses related work, focusing on different types of 2D/3D models and their application to face recognition as well as their formulation of GRAB features. Chapter 3 discusses our techniques for using models to synthesize scale information into GRAB feature vectors. Chapter 4 discusses our experiments, dataset, and results. Chapter 5 concludes.

Chapter 2

Related Work

In this chapter, we review previous work related to leveraging appearance and shape models for pose robust face recognition, as well as the formulation of Generalized Regions Assigned to Binary (GRAB) grid based features, the 2D features which we augment with pose information in later chapters. The pose problem is epidemic to unconstrained face recognition and many pose-invariant and pose-adaptive approaches have been developed that do not involve 2D or 3D shape and appearance models. Our aim in this section, with respect to 2D and 3D models is to provide the requisite theory, framework, and survey of similar approaches to be able to communicate the motivation and theoretical basis of our approach. Likewise, GRAB is only one of several grid-based feature types. We focus on GRAB, simply because we chose to use this feature type in this thesis for its generalized nature which makes it suited to augmentation in several respects; The reader should not believe that GRAB is the end all, be all of grid-based features.

2.1 2D Models

2.1.1 Active Shape Models

Active shape models (ASMs) are models in a coordinate frame centered about an object type in question[18]. For pose robust face recognition, the object type is the human face, although ASMs can represent any topologically invariant object type, for example, ASMs were originally applied to organ detection in medical imaging [18].

For face recognition purposes, ASMs are built from training sets of labeled landmarks on the human face. The intuition behind the application of ASMs is straightforward: pick an initial rough approximation to the location of the face, given an initial (e.g., average) set of shape parameters, then iteratively optimize the shape parameters until

the model converges to the face in question. From there, the pose can be estimated and corrections can be performed.

FIGURE 2.1: An ASM fit to several images of celebrities.

The procedure for building an active shape model consists of first acquiring a training set of face images with N ground truth landmarks, labeled by their X and Y coordinates. The images are then normalized into a common coordinate frame.

A rank-f dimensionality reduction is then applied via principal components analysis (PCA) so that for any facial landmark configuration X, that landmark configuration can be approximated by $X \approx \bar{X} + Pb$, where P is a matrix containing the f eigenvectors of the data matrix explaining the greatest amount of variance (i.e., the eigenvectors with largest corresponding eigenvalue), and b is a vector containing weights on these eigenvectors – the modes of variation, as it were. The purpose of the dimensionality reduction is both to make optimization more manageable and to avoid overfitting on new images – i.e., images not in the training set. The modes of variation are completely dependent on the data, and hence a generalizeable model demands a well labeled training set, varying in demographic, expression, and pose. Modes of variation corresponding

to highest variance often explain macroscopic differences within an image such as pose, facial geometries, and major expression variations, while modes corresponding to smaller variance within the dataset explain discrepancies between images in minor expression variations.

Although the kth mode of variation may seem to correspond to a particular change, for example a change in head pitch, one must be careful not to impute any meaning to this observation beyond the fact that this mode corresponds to the kth orthogonal distortion of highest variance within the dataset in $2N - D$ dimensions. A few more or less training images could lead to a wildly different result. In other words, the modes of variation are directly learned from the training set via a linear model and do not correspond to an underlying physical model of head pose and expression.

Once PCA decomposition is performed and the model acquired, the coefficients on the modes of variation can be altered to fit the model to new face shapes by some optimization criterion, e.g., by minimizing the sum of the squares of residuals between the model landmark points and the image landmarks in question. The optimization procedure is carried out as follows: First, the model is initialized to the mean shape by zeroing all coefficients in the shape parameter vector. Candidate model points on the image are located and a similarity transformation (translation, rotation, and uniform scaling) is learned[1]. Note that the similarity transformation is with respect to the model points back projected into the original image space; not the dimensionally-reduced model space. Once the similarity transformation is learned, its inverse is used to project the candidate image points into the model space and projected into the plane tangent to the mean model. The model parameters are then updated so that the model and the projected image landmark estimates align. The procedure is iterated until convergence is reached; i.e., when further iteration produces negligible change on the shape parameters. Often, shape parameters are constrained to ensure plausible face shapes. This is done by bounding each mode of variation by a multiple of its corresponding eigenvalue.

When fitting ASMs to actual images, the choice of fit function becomes relevant, since the landmarks must be detected. A naïve choice is minimizing the sum of squares of orthogonal distances between model points and the nearest strong edges[2] However, facial landmarks do not always lie along strong edges – consider for example the tip of the nose. A better approach consists of statistically learning from the training set itself the pixel intensity behavior to search for, by building statistical models of gray level pixel structure normal to the boundary of the shape defined by the landmark points.

[1]Several optimization methods can be used here to learn the similarity transform, depending on the optimization criterion. A least squares loss function is often employed here for simplicity, lending the problem well to least squares regression or gradient descent.

[2]Edges can be extracted via a thresholded gradient or Laplacian image, or even better via a Canny edge detector.

By treating each channel separately and aggregating the result at each stage of the iteration, the approach can easily be extended to synthesize color information as well.

The gray-level modeling procedure outlined in [18, 19] consists of sampling $2k+1$ pixels for each landmark point. k adjacent pixels lie on each side of each landmark point and are orthogonal to the shape defined by the landmarks. To attenuate global intensity biases, each vector consists of the normalized derivative of the pixel intensity values rather than the raw intensities themselves. After sampling for all training images, N sets of normalized sample vectors are obtained, each set corresponding to a landmark point. A $2k+1$ dimensional Gaussian is then fit to each set and the mean vectors and covariance matrices are obtained. The quality of fit of an estimated landmark point is then assessed by the Mahalonobis distance parameterized on the mean and covariance matrix corresponding to that point. Under the multivariate Gaussian assumption, the most probable a posteriori estimate for a landmark point lies along the line normal to the a priori estimate at the pixel location at which the Mahalonobis distance is minimized. During the search, points on each side of the current point are sampled. The pixel location of maximum probability is chosen for each new estimated landmark point and shape and pose parameter updates are performed.

To better accommodate the multi-scale nature of real images, decrease convergence time, and mitigate false positive landmarks, a coarse-to-fine scale space search is generally performed. Once the algorithm has converged for a particular scale, the model parameters are saved and used to initialize a search at a finer scale.

2.1.2 Active Appearance Models

Active Appearance Models (AAMs) can be considered extensions to ASMs which incorporate image texture – i.e., the pixel intensity patterns over patches in the image. Instead of simply fitting shape, AAMs can be used to generate virtual images corresponding to parameter choices. Like ASMs, AAMs are attractive in that they offer a description of the image with a small number of parameters. There exist methods that incorporate texture into ASMs, but AAMs take a different approach, combining shape and texture parameters into one statistical model.

AAMs, are trained using a set of annotated images, with landmark points marked on the face defining the main features. First, PCA is performed on the landmark points. Each image is then warped so that its corresponding landmark points line up with the mean shape and texture is sampled and energy-normalized at specific locations for each image. PCA is then performed on the texture vectors. The eigenshapes and eigentextures are then concatenated and another round of PCA is performed. The end result is a synthesis

of all shape and texture information into *one statistical model*, so that an image's shape and texture can be reconstructed by

$$x = \bar{x} + \mathbf{Q}_s \mathbf{c} \tag{2.1}$$

$$g = \bar{g} + \mathbf{Q}_g \mathbf{c}, \tag{2.2}$$

$$\tag{2.3}$$

where $\mathbf{c}$ is a vector of model parameters, x is the shape, $\bar{x}$ is mean shape, g is texture, $\bar{g}$ is the mean texture, and $\mathbf{Q}_s, \mathbf{Q}_g$ are matrices containing the modes of variation in shape and texture.

Fitting an AAM to an image involves minimizing the difference between the real face image and a virtual synthesized face in texture projected space. Given parameter vector $\vec{p}$, and residual $\vec{r}(\vec{p})$, the residual at a small perturbation $\delta\vec{p}$ from the parameter vector is given by the first order Taylor series expansion[3]

$$\vec{r}(\vec{p} + \delta\vec{p}) = \vec{r}(\vec{p}) + \frac{d\vec{r}(\vec{p})}{d\vec{p}}\delta\vec{p} \tag{2.4}$$

Setting the derivative of squared error ($\vec{r}^T\vec{r}$) to zero allows us to pick $\delta\vec{p}$ so as to minimize the residual. The solution is

$$\delta\vec{p} = -\mathbf{R}\vec{r}(\vec{p}); \mathbf{R} = \left(\frac{\partial\vec{r}^T}{\partial\vec{p}}\frac{\partial\vec{r}}{\partial\vec{p}}\right)^{-1}\frac{\partial\vec{r}^T}{\partial\vec{p}}. \tag{2.5}$$

Due to a relatively large number of parameters needed to explain an acceptable portion of variance this is a computationally difficult optimization problem. However, Cootes et al. made the following observation: For each face fit, the optimization is similar and this similarity can be leveraged as a prior in the fitting algorithm by modeling relationships between error and parameter adjustments offline in a linear model. The offline approximation is feasible because the optimization occurs in normalized projected texture space – not the raw image space. This approximation amounts to solving the gradient $\frac{\partial\vec{r}}{\partial\vec{p}}$ and implicitly $\mathbf{R}$ offline, a task which can be accomplished by generating a training set of

[3] $\vec{r}$ is a vector, so $\frac{d\vec{r}}{d\vec{p}}$ is a matrix whose ijth element is $\frac{d\vec{r}_i}{d\vec{p}_j}$.

perturbations in model parameters from the known optimum and computing a weighted average of the residuals.

The final fitting algorithm becomes a procedure of projecting the sampled texture into texture space, determining the residual, solving for $\delta\vec{p}$ and setting $\vec{p} \leftarrow \vec{p} + k\delta\vec{p}$, where $k \leq 1$, adjusting the model by the new $\vec{p}$ value, sampling the texture at the new model points, projecting back into texture space and analyzing the residual. If the residual has been reduced, then the algorithm continues. If the residual has increased, $\vec{p}$ is returned to its previous value and k is reduced. When reduction in the residual below a certain threshold stops, the algorithm terminates. As with ASMs, a coarse-to-fine strategy is employed, via a scale-space pyramid.

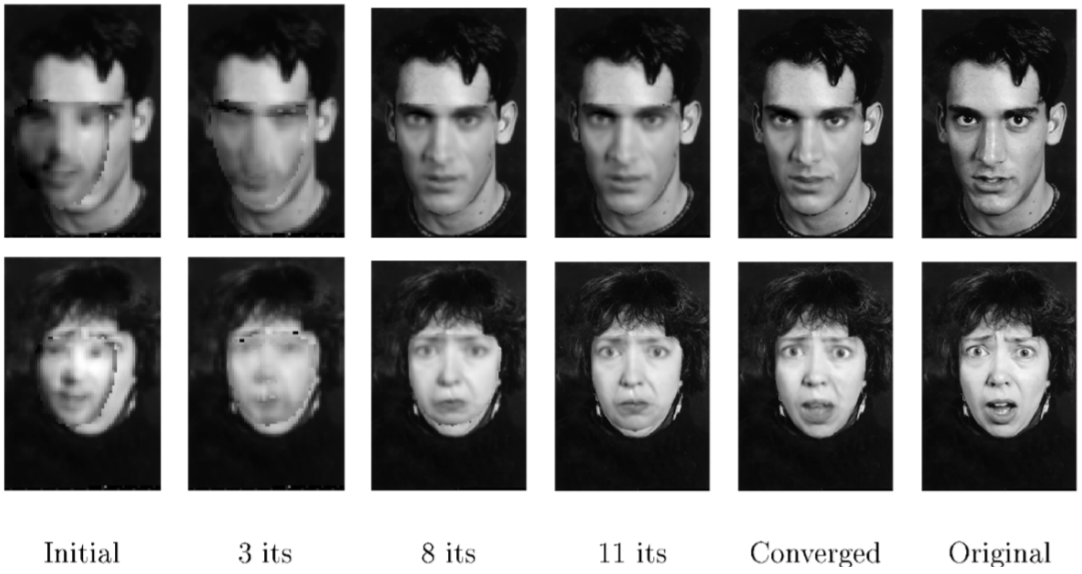

FIGURE 2.2: An AAM image search over several iterations of the algorithm. The accuracy of the final result is spectacular, although lack of background noise makes the fit easier. Image courtesy of [3].

2.1.3 Leveraging 2D Models

When leveraging 2D models for pose normalization/invariance, a key difficulty is out of plane image rotations – a degree of freedom which 2D methods inherently lack. However, 2D methods can still be used to synthesize 3D faces by statistically learning the changes in 2D shape in either image space or feature space at patches on the face and interpolating.

For example, in [4] Chai et al. justify that for pixel-wise aligned Lambertian faces of different poses, there exists an approximate linear mapping between images under different poses. After all, an image is simply a projection of a rotated 3D facial structure; Both rotation and projection are linear operators. Of course, projection reduces rank,

but missing points on the face due to occlusion constitute a relatively small number which can be inferred/estimated via linear interpolation/extrapolation of nearby points on the face. Globally linear regression (GLR) aims to learn the inverse rotation/projection operation to map a rotated image back to its frontal pose for an arbitrary face. For each pose, the regression is carried out on a training set, with pairs of frontal and non-frontal images of the specific pose as observations. A least squares estimate is obtained in 2D image space. Due to differences in face geometries/albedos for individual people/images, however, as well as the extreme difficulty of pixel-wise alignment – images are typically aligned via only a few fiducial points. Therefore, Chai et al. ultimately turn to locally linear regression (LLR), performing regression on overlapping local patches, using a cylindrical face model for patch selection on non-frontal faces. In reconstructing the frontal pose, patches are recombined, by fusing overlapping regions.

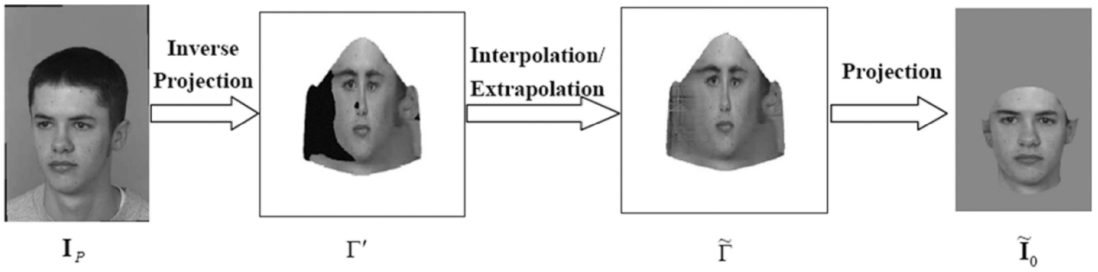

FIGURE 2.3: Mapping a non-frontal view image to frontal view using linear regression and interpolation. Image from [4].

In [20], on the other hand, Jiménez and Castro take a different approach, constructing statistical models of in-plane pose variations and synthesizing virtual views via thin plate splines interpolation. The pose parameters are isolated by observing changes to the model obtained from toggling weights on the eigenvectors. Naturally, the parameters relevant to pose depend on the dataset, from which the model is constructed. Hence, Jiménez and Castro used a dataset with great pose variation and little expression variation, causing the pose parameters to be associated for the most part with the top principal components. They further introduced virtual symmetric meshes to the dataset in order to decouple identity/expression variations from rigid pose variations, a technique for which they provide theoretical justification. Rather than attempt to warp the face image onto a mean face via a piecewise-affine transformation, thereby shearing out important identity information, Jiménez and Castro use localized Gabor jets – responses from convolutions with banks of Gabor wavelets – as features. Gabor jets are popular local features for pose-invariant matching without image normalization, and although their discussion is not specific to ASMs/AAMs, Jiménez and Castro provide several useful similarity measures for matching Gabor jets and graphs of fiducial points obtained either via an elastic bunch graph search or via ASMs/AAMs in [5].

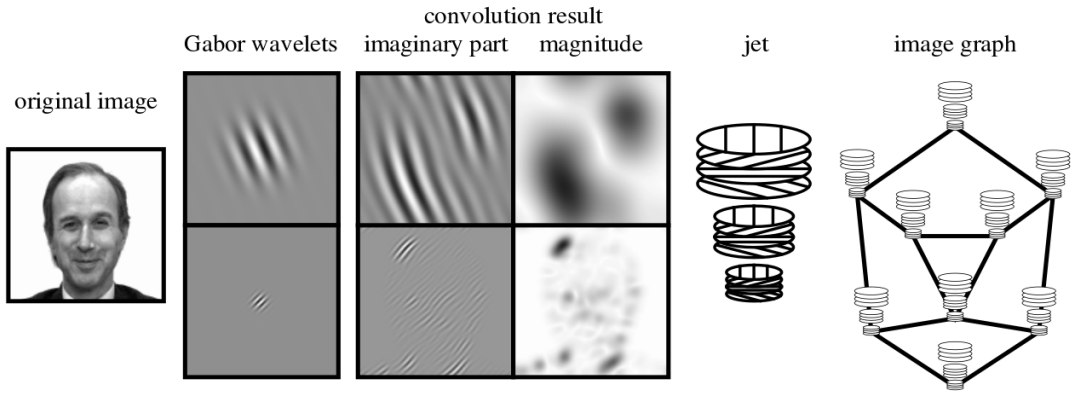

FIGURE 2.4: Gabor jets are localized features which can be obtained for specific model points on an image via convolution with banks of Gabor wavelets. Image from [5].

2.2 3D Models

2.2.1 3D Morphable Models

3D morphable models (3DMMs) are similar in concept to ASMs and AAMs. 3DMMs are parametric reconstructions, where each face is represented as a linear combination of exemplar faces from the gallery. Both shape and texture are sampled for each of the exemplar faces and each exemplar face is normalized in 3D in a common pose about a common origin, a 3D correspondence, in computer vision and computer graphics parlance. Rather than relying on sparse landmarks as ASMs and AAMs do, however, 3DMMs sample shape (the surface of the face) and texture (the color of the face) in 3D at n points, cylindrically uniform in azimuth and altitude. Hence, the shape of the ith face can be represented as the vector $S_i = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^T$, while the texture can be represented by the vector $T_i = (r_1, g_1, b_1, \ldots, r_n, g_n, b_n)^T$, where S_i and $T_i \in \mathbb{R}^{3n}$. New faces of shape $\hat{S}$ and texture $\hat{T}$ can be synthesized via the linear combinations $\hat{S} = \sum_i a_i S_i$ and $\hat{T} = \sum_i b_i T_i$, where a_i and b_i are scalar weights under the constraint $\sum_i a_i = \sum_i b_i = 1$.

For practical computation purposes as well as noise attenuation, dimensionality reduction by principal components analysis is performed on 3DMMs. For both shape and texture this involves subtracting all faces from the mean face, performing separate eigendecompositions of the associated covariance matrices for the data, ranking the eigenfaces by their associated eigenvalues, and discarding eigenfaces that explain little variance. The rank-F approximations for shape and texture are then given by

$$\hat{S} \approx \bar{S} + \sum_{i=1}^{F-1} \alpha_i s_i$$

and $$\hat{T} \approx \bar{T} + \sum_{i=1}^{F-1} \beta_i t_i,$$

where s_i are orthogonal eigenshapes, t_i are orthogonal eigentextures, and α_i and β_i are the corresponding coefficients in projected and rotated space. If F is equal to n, then the approximation becomes an equality. The $F-1$ superscript on the summations is a consequence of the fact that one degree of freedom is absorbed in the centering about the mean face.

As with ASMs and AAMs, statistical measures must be taken to ensure that 3DMMs produce shapes corresponding to plausible faces. Blanz and Vetter approach this by examining the statistics of their dataset. Under the assumption that $\vec{\alpha}$ and $\vec{\beta}$ values are distributed as $F-1$ dimensional Gaussians which are well summarized by the training set, the probabilities that a synthesized face's shape and texture coefficients correspond to real shapes and textures are given by the probability density functions

$$p(\alpha) = (2\pi)^{(-\frac{F-1}{2})} |\Sigma_S|^{(\frac{-1}{2})} e^{-\frac{1}{2}\sum_{i=1}^{F-1}\left(\frac{\alpha_i}{\sigma_i}\right)^2} \tag{2.6}$$

$$p(\beta) = (2\pi)^{(-\frac{F-1}{2})} |\Sigma_T|^{(\frac{-1}{2})} e^{-\frac{1}{2}\sum_{i=1}^{F-1}\left(\frac{\beta_i}{\sigma_i}\right)^2} \tag{2.7}$$

where the $|\sigma|$s are the respective determinants of the texture and shape covariance matrices. These PDFs can be used to bound shape and texture texture parameters by their minimum corresponding probabilities and thereby bias the 3DMM toward realism.

Although 3DMMs have many applications beyond the realm of face recognition, in face recognition, the objective is to fit a model to one or more images. Once a fit is acquired, several matching methods can be employed. Among these matching methods are pose normalization, where the model is used to aid in linear algebraic normalizations to frontal view of probe and gallery images in question, synthesis, in which virtual gallery or probe face images are generated for a corresponding match in pose and illumination to the gallery/probe image in question, and parameter matching, in which gallery and probe fitting parameters are compared for recognition[21]. In each case, the common step is the algorithm for fitting a 3DMM to one or more face images.

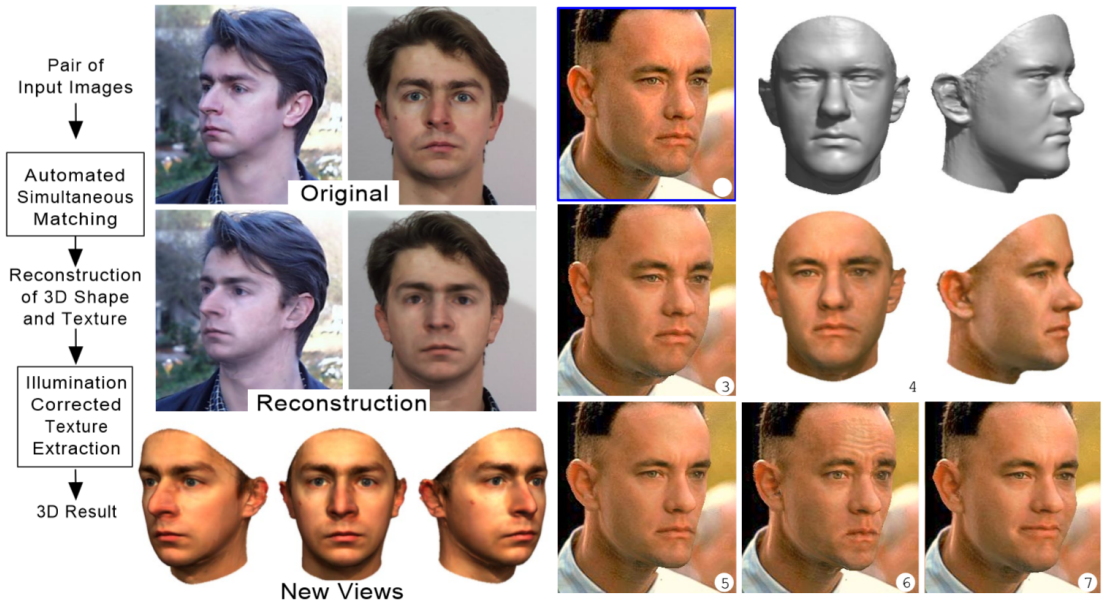

FIGURE 2.5: Images of 3D morphable models, courtesy of [6]. On the left, in the top two panes, we see original images along with 3DMM reconstructed images. The virtual images are difficult to discern from the original images. On the right, we see that by cleverly altering parameters on a 3D morphable model, one can make Forrest Gump appear to gain or lose weight and even perform the unthinkable act of smiling.

The algorithm employed to this end is discussed by Blanz and Vetter in [6], and is effectively and optimization algorithm over shape parameters contained in $\vec{\alpha}$, texture parameters contained in $\vec{\beta}$, and environmental parameters, contained in ρ. Environment parameters might include pose or camera angle, scale, camera distance, ambient and directed light intensities in all 3 channels, color contrast, light direction, light contrast, and surface shininess. Since this thesis focuses on the pose problem, for understanding purposes, the reader may think of $\vec{\rho}$ as containing only pose information.

Often, $\vec{\alpha}$ and $\vec{\beta}$ vectors are zeroed, i.e., the original shape and texture parameters are set to those of the mean face. The initialization of ρ may often be improved with a priori knowledge, given, e.g., by fiducial points detected by a cascade classifier. The 3D model is then rendered in the 2D plane, a procedure which can be as simple as orthographic or perspective projection of red, green, and blue channels for the given pose of the model, or can involve complicated illumination models depending on the parameters within the environment vector. Blanz and Vetter use the Phong illumination model [6]. Once a model is rendered, a rasterized representation of the image can be obtained, where

$$I_{model}(x, y) = [I_{modR}(x, y)|I_{modG}(x, y)|I_{modB}(x, y)]^T.$$

The residue between the image reconstructed from the model I_{model} and the original input image I can then measured via a least squares, Euclidean distance metric

$$E_I = \sum_{x,y} ||I_{model}(x,y) - I(x,y)||^2. \tag{2.8}$$

However, there are two problems with simply minimizing over equation 2.8. First, real images may contain background or foreground objects (e.g., glasses) with "face-like" geometries, a bad match which looks little like a real human face may do a better job of minimizing least squares error over an image than a match which a human would classify as correct. Therefore, it is necessary to balance the probability that the model parameters correspond to a face with the goodness of fit.

The second problem is that although the rendered image derived from the model can be *precisely* parameterized, the input image is noisy about any projected paramerization of the 3DMM. Assuming that this noise is Gaussian, with standard deviation σ_N, the probability that the correct image is parameterized with least squares error E_I observed error goes as a 1D Gaussian with standard deviation σ_N. Parameterized in terms of E_I, $p(E_I, \mu, \sigma) = \frac{1}{\sigma_N\sqrt{2\pi}} e^{-\frac{{E_I}^2}{2\sigma^2}}$ is the probability that the model matches the input image.

The optimization problem can then be described as, follows: Given multidimensional Gaussians PDFs corresponding to shape, texture, and environment parameters and the $1D$ PDF corresponding to "correct" observation of the input image in terms of model parameters, obtain a maximum a posteriori (MAP) estimate of parameters. Bayesian statistics teaches us that this corresponds to minimizing the squared sums of parameter values weighted by their respective variances. In other words, the objective becomes to minimize the cost function

$$E = \frac{E_I}{{\sigma_N}^2} + \sum_i \frac{{\alpha_i}^2}{{\sigma_{S,i}}^2} + \sum_i \frac{{\beta_i}^2}{{\sigma_{T,i}}^2} + \sum_i \frac{{\rho_i - \bar{\rho}_i}^2}{{\sigma_{\rho,i}}^2}.$$

The optimization problem is nonconvex and would be entail *substantial* complexity to perform using all mesh points and texture samples over the 3DMM. Instead, Blanz and Vetter randomly select K points to achieve both some resilience against shallow local minima and cut down on computational complexity. Intricacies of the optimization algorithm, specifically involving the Phong illumination model are discussed in [6]. At a high level, however, the optimization problem is simply solved by gradient descent or stochastic gradient descent on E.

As with ASMs and AAMs, Blanz and Vetter employ four coarse-to-fine techniques in their fit.

1. Using a scale space pyramid they optimize first over low scales with a subsampled model and use the acquired fits for initialization of higher scales.

2. They begin by optimizing over just the first few principal components and add more components with additional iterations.

3. They adjust σ_N throughout the coarse of the algorithm. Specifically, they imbue a larger variance to the "observed" input image value toward the beginning of the fit and shrink this variance toward the end of the fit. This has the effect of upweighting the importance of face shape and texture relative to the fit at the beginning of the optimization and up-weighting the importance of the fit relative to the face shape toward the end of the optimization.

4. Once a rough fit is obtained they optimize parts of the face separately so as to increase the number of degrees of freedom and better match fine-grained details of the face. After optimizing over parts, parts are stitched together via interpolation.

Although Blanz and Vetter produced their original paper in 1999, 3DMMs yield exceptional fitting performance, even to this date, and have been implemented in several commercial applications[22]. As with ASMs and AAMs, 3DMMs are suited to modeling non-rigid but topologically invariant objects. However, 3DMMs have their drawbacks. First, even on modern computers, 3DMMs require tens of seconds to perform fitting. Additionally, they sensitive to alignment, occlusion, and image quality. This means that in unconstrained scenarios, 3DMMs may fail miserably. Although multiple images from multiple angles may be used to generate a 3DMM [6], expression differences throughout the images may also produce nontrivial results.

3DMMs can be applied toward pose robust face recognition in several ways: First, they can be used to normalize off-pose faces to frontal, so that frontal images can be matched to frontal images. Second, they can be used to render and match virtual faces using traditional 2D algorithms. Third, recognition can be performed by comparing their shape and texture parameters ($\vec{\alpha}$ and $\vec{\beta}$). While all of these approaches have their merits, the inordinate amount of time required for the fit limits the extent to which 3DMMs have been applied in unconstrained and real time recognition algorithms.

2.2.2 Adapting 2D Features via 3DMMs

A hybrid approach to pose robust face recognition pursued by Yi et al. in [21] is the published approach most similar to ours in concept. Yi et al. use a 3DMM built *only* on shape and pose information in conjunction with a rougher but faster fitting algorithm

to overcome the heavy computational costs of fitting a 3DMM. The authors then use the information gleaned from the fit to change how 2D features are extracted. Yi et al. refer to this technique as *pose adaptive filtering*.

Once a shape-only 3DMM is obtained, Yi et al. define evenly-spaced feature points within the image plane on the 3D face shape. Pose and shape parameters for a face in question are then obtained via a 3DMM fit. The feature points are then projected into the image plane and a bank of Gabor filters is used to extract features for classification.

During the fit, Yi et al. use a 3-view active shape model (ASM) to localize face shape. To find each landmark within the image, their ASM employs a boosted cascade of LBP classifiers. This process returns a probability map for each landmark. Once the probability map is obtained, the mean-shift algorithm is used to move prior estimates of the landmarks to their most probable locations. Empirically, Yi et al. found that the 3-view ASM works well from -60 to 60 degrees.

The fit then consists of the following optimization: Given landmark x on face image and vertex index I on 3D model, the goal is to solve for pose ρ and shape $\vec{\alpha}$ by minimizing

$$E = ||x - \rho(m(I) + \sum_{i=1}^{n} \vec{\alpha}_i \vec{w}_i(I))||_2^2 + \lambda \vec{\alpha}^T \Sigma^{-1} \vec{\alpha} \tag{2.9}$$

In equation 2.9, $\vec{w}_i$s are the eigenfaces, λ is a regularization factor, Σ is the diagonal matrix of singular values from the singular value decomposition – values which are equivalent to the eigenvalues of the shape covariance matrix in this case. Equation 2.9 can be read as follows: for a given model point, the objective is to minimize the regularized Euclidean distance between the 2D projection of the model's landmark point and the corresponding point on the image. The regularization term consists of the product of a constant coefficient and an inner product, which amounts to the sum of the squares of $\vec{\alpha}$'s elements weighted by the inverse of their respective singular values. This means that the elements of $\vec{\alpha}$ with high corresponding singular values – i.e., the elements that explain more variance are allowed to vary more than those that explain less variance. Alternatively, this can be viewed as letting the hyper-sphered elements of $\vec{\alpha}$ vary equally, a process known as data "whitening". The error function is a trade-off between goodness of fit and deviation from the peak location of the n-dimensional Gaussian PDF.

In the design employed by Yi et al. ρ is a 2×4 affine matrix which is composed of translation, rotation, scaling, and projection operations. Note that in order for equation 2.9 to make sense, homogeneous coordinates are assumed. Homogeneous coordinates

are non-Euclidean geometric coordinates, which can be used to represent points at infinity. The 3D point (x, y, z) in Euclidean space is represented by the homogeneous point (X, Y, Z, K) in homogeneous space, where $x = X/K, y = Y/K, z = Z/K$. Note that there are infinitely many possible representations of the same Euclidean space point in homogeneous coordinates. Another advantage of homogeneous coordinates is that they allow translations to be expressed as matrix multiplications. Affine transformations can therefore be represented in one matrix multiplication. For example, the equivalent to a translation in 3D Euclidean coordinates can be expressed by multiplying the corresponding vector in homogeneous coordinates by the translation matrix

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where t_x, t_y, and t_z are the translation amounts in the x, y,and z directions.

Yi et al used a weak perspective projection model – a scaled orthographic approximation to a perspective projection, which assumes that the mean variation in depth of points is small compared to the magnitude of the mean depth. Since ρ and $\vec{\alpha}$ are coupled, gradient descent alone cannot solve the optimization problem as there exists no closed-form solution for E. Yi et al therefore employed an alternating least squares algorithm in which $\vec{\alpha}$ is first initialized to the mean shape, ρ is optimized to minimize error, then $\vec{\alpha}$ is varied with ρ held fixed. The procedure iterates until convergence is reached. The final model for a given fit consists of S with learned weights w_i, where $S = m + \sum_{i=1}^{n} w_i \alpha_i$, as well as ρ, the learned 2×4 affine projection matrix.

For the Jth feature point projected onto the at image at location, $\vec{x} = \rho S_J$, Yi et al extract Gabor features via a bank of Gabor Wavelets. Gabor wavelets are used to extract localized spatial frequency information and are 2D directed sinusoids modulated by Gaussians. The impulse responses of their convolution with the image at the feature point is used as an element in the feature vector. The Gabor wavelets that Yi et al. used are defined by

$$\Psi(\vec{x}, \vec{k}) = \frac{k^2}{\sigma^2} e^{\frac{\vec{k}^2}{-2\sigma^2} x^2} \left(e^{i\langle \vec{k}, \vec{x} \rangle} - e^{-\frac{\sigma^2}{2}} \right). \tag{2.10}$$

Although the convolved value at each feature point is complex, the sensitivity of phase to proper alignment led Yi et al. to simply use amplitudes as features [21]. Yi et al. used

a bank of Gabor wavelets with 8 wave orientations and 5 standard deviations, resulting in 40 elements per feature point. When feature points were occluded or blurred, Yi et al exploited facial symmetry and extracted features at the points on the visible side of the face. After extracting features at all, Yi et al. performed PCA and classification via a cosine metric.

In addition to the faster fitting algorithm and accuracy boosts, the real novelty of of [21] was using a 3D model to localize and improve the accuracy of 2D features.

2.3 GRAB: Generalized Regions Assigned to Binary

Generalized Regions Assigned to Binary, or GRAB features, are a grid-based feature type introduced in [11]. GRAB is a generalization of local binary pattern features, or LBP, designed to account for scale and resolution differences between images and to blend both local and global image texture information. Both LBP and GRAB assign to each pixel an 8-bit string which encodes the relative intensities of each neighboring pixel at a given offset from the center pixel. Since GRAB and LBP are represented in terms of bytes, GRAB/LBP images can be visualized. Histograms of both feature types (or raw pixels for small images) can serve as feature vectors for machine learning classifiers. The process of extracting GRAB features is shown in figure 2.6.

The key advantage of GRAB over LBP is that it accommodates multiple scales. LBP is simply GRAB with a neighborhood offset of one pixel, a threshold of zero and the LBP bit ordering in figure 2.7. Similar extensions to LBP have been developed with the same ends in mind and are discussed in some detail in [12] and [11]. A promising spinoff of LBP is MB-LBP [23] (Multi-scale Block Local Binary Patterns), which combines local and global gradient information in a manner quite similar to GRAB, comparing averages of eight ordinal square regions with the average of a center region. However, unlike GRAB, MB-LBP maintains the same bit ordering as LBP and does not allow for overlap among regions.

In [12], multi-scale experiments were conducted on artificially scaled versions of the LFW (Labeled Faces in the Wild) and FERET (Face Recognition Technology) datasets using GRAB and MB-LBP features. GRAB was found to outperform MB-LBP at small scales. By extracting a scale pyramid of GRAB features, better accuracies can be obtained. However, this comes at the computational cost of increased dimensionality, and machine learning classifiers all scale at least linearly with the size of the feature vector; More advanced classifiers scale polynomially, for example kernelized support vector machines scale between quadratically and cubically on feature vector length at train

time and between linearly and quadratically at classification time. Using a scale pyramid therefore demands considerable computational resources as well as dimensionality reduction techinques such as PCA or exploitation of uniform patterns.

FIGURE 2.6: An illustration of the GRAB process. The raw image is shown in (a). The image is converted to single channel, fiducial point locations are detected to obtain an affine transformation used to geometrically normalize the image (b). The image is averaged over neighborhoods of a given size (c), and the GRAB operator is applied (d). The feature vector submitted to a classifier consists of the concatenation of normalized histograms over fixed-size regions within the image (e).

In implementing GRAB, we first select a neighborhood size and shape. In traditional implementations of GRAB, this neighborhood has been a square of size $N \times N$ pixels where $N \in \mathbb{Z}^+_{odd}$. However, this neighborhood can be of any shape. We shall constrain ourselves to rectangular shapes of size $M \times N$ where $M, N \in \mathbb{Z}^+_{odd}$. Each pixel, p_c, within the image is then set to the average value of all pixels within the neighborhood centering on p_c. Each new pixel value can be set efficiently with only 4 arithmetic operations on an integral image, another reason for choosing rectangular neighborhoods. The purpose of this blurring operation is to aggregate per-pixel texture information with texture information contained in surrounding pixels and to achieve some tolerance to pixel noise.

After blurring, each pixel within the image is assigned an integer representing the local image gradient, measured with respect to pixel intensities of n surrounding neighbors. This binary string for center pixel c is defined as

$$GR(c) = \sum_{i=1}^{n} g_i(p_c, p_i) 2^{(i-1)}, \tag{2.11}$$

where $g_i(p_c, p_j) = 0$ if $sgn(p_c - p_j - T) < 0$ and 1 otherwise for threshold $T \in \mathbb{Z} \geq 0$. In practice T is statistically selected to account for pixel noise. Previous implementations of GRAB have used square neighborhoods, with eight neighbors selected, uniformly offset N, E, S, W, NE, SE, SW, and NW from the center pixel by a constant number of pixels. Specifically, conventional GRAB-1 is defined as a 1-pixel offset from the center pixel, in which no averages occur. Likewise, conventional GRAB-k (k odd) corresponds to a

$\lfloor k/2 \rfloor$ pixel offset from a center pixel p_c, where p_c is defined as the mean of the original image pixel values within the $k \times k$ region centered about p_c.

The ordering of the pixel values in equation 2.11 with respect to their relative geometry is also an important consideration. In canonical GRAB-k implementations, pixel values are ordered in a noise-tolerant manner, such that a permutation in the values of two adjacent pixels will have no more than a two order of magnitude impact on the center pixel's GRAB value.To see this, consider binary strings 00000001 and 10000000. In the conventional LBP ordering shown on the left-hand-side of figure 2.7, a single pixel rotation can cause these strings to be identical to one another, whereas under the GRAB bit ordering, 10000000 can be decreased at most to 00100000 and 00000001 can be increased at most to 00000100. This is one reason for GRAB-k's superior robustness to local pixel variations in comparison with other grid-based methods, such as conventional LBP.

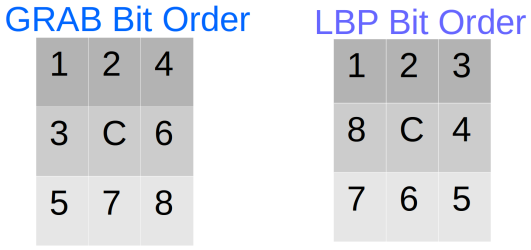

FIGURE 2.7: Canonical bit orderings for GRAB (left) and LBP (right). The GRAB bit ordering is up to five orders of magnitude more robust to a two-pixel permutation.

A typical application of, GRAB-based recognition proceeds as follows: First, fiducial points within an image are obtained via human ground-truth or automated detector. Each face image is geometrically normalized with respect to its fiducial points via some alignment scheme (e.g., similarity transform, affine transform, piecewise/nonlinear transform). Lighting normalization may also be applied. GRAB features are then extracted from the entire image. Finally, the image is partitioned into regions. Normalized histograms are obtained for each region with each bin corresponding to one of 256 possible pixel values. The concatenation of histograms serves as the feature vector for the image. Feature vectors from multiple GRAB scales can be concatenated to offer a wide range of local and global image gradient information. This concatenation can increase recognition accuracy, although it does so at the expense of time required for training and classification.

For pairwise verification experiments, in which we would like to determine if two or more images correspond to the same identity, feature vectors can be merged prior to submission to a classifier, for example, for two images, one can simply take the element-wise difference. For verification experiments, one classifier is learned for many images.

For identification experiments in which we would like to determine the identity within the gallery to which an image corresponds, if any, a separate classifier is typically generated for each identity and the outputs of multiple classifiers are synthesized into similarity scores.

Chapter 3

Approach

In this chapter we discuss our two approaches to incorporate scale information due to pose variations into the GRAB feature vector. In our first approach, we perform a rough fit of a 3D model of a generic human face to each image. We then incorporate scale information from this model into the GRAB feature vector by 1.) weighting portions of the feature vector by their corresponding scales and 2.) appending scale information to the feature vector. Although the fit is not precise, its utility or lack thereof hinges on the statistical and geometric similarity of many human face examples to a "generic face" – the assumption that the shapes of most faces are statistically constrained in structure to the extent that relevant pose information provides useful additional information to the classifier.

We use a simple fitting technique and a rigid 3D model in our approach. Although the approach could be extended to incorporate 3D Morphable Models (3DMMs), the dearth of readily available 3D face data and long fitting times required for 3DMMs place such an approach outside the scope of this thesis. Our approach is an exploratory investigation; not a quest for optimal fitting or state of the art accuracy.

Our second approach uses face graphs obtained via an active appearance model (AAM) to estimate the rough scale of each face image and pick the GRAB scale at which to extract local features on the face – a piecewise implementation of GRAB, as it were. The utility and lack thereof of this approach depends on how the granular changes made to the scale of the GRAB operator affect classification performance and on the AAM's ability to localize landmarks in the image.

Both approaches aim to introduce scale information to the feature vector but in a different ways. The first weights the feature vector after GRAB features have been extracted, while the second controls the scale at which gradient information is extracted.

3.1 Localized Scale Estimation Via Rigid 3D Model Fit

3.1.1 Estimating Pose

Fitting a rigid 3D model to a face requires an estimation of pose, i.e., pitch, roll, and yaw angle of the human head defined in figure 3.1. In this section, we turn to a simple and lightweight technique by Sankaran et al. [24]. Although there are a number of heavyweight methods to obtain this estimate, one of which we pursue section 3.2, we implement Sankaran et al.'s method method for variety and simplicity, with the additional observation that a lightweight method for pose estimation based on few fiducial points has its own advantages: First, the detection of a few prominent fiducial points on rigid parts of the face is less error prone than detecting *many* landmarks due to more consistent definedness of points. Second a lighter weight algorithm offers faster processing time and therefore easier incorporation into real time recognition pipelines. Finally, Sankaran et al.'s method relies only on a small number of fiducial points, the ground truth locations of which are more readily available for many databases, while the ground truth locations of all points needed for advanced techniques are more expensive to obtain and generally less available.

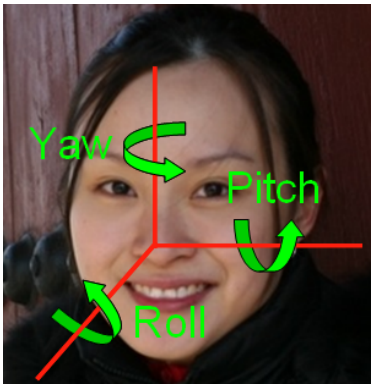

FIGURE 3.1: In the context of human face recognition, not all Euler angles are created equally. Since roll refers to rotations within the image plane, it can easily be corrected for by simply rotating the image. Out of plane pitch and yaw rotations are more difficult to correct for, since pitch and yaw rotations are perpendicular to the image plane. Due to human physiology, however, yaw variations account for greater variance within face images than do pitch rotations.

Sankaran et al.'s technique is effective for estimating yaw and roll, but not pitch. While pitch is important, for our dataset, significant pitch variations are uncommon, and yaw variations are prevalent. This is largely due to physiological limitations on human head orientation (turning sideways is easier than doing a back flip).

We note that the pose estimation technique in [24], or any pose estimation technique loses accuracy as poses approach ± 90 degrees from frontal, since fiducial points get

occluded. When this occurs, however most face detection algorithms also fail, so this is not a grave concern for the scope of this paper, since all images within our dataset can be obtained via a cascade classifier.

As illustrated in figure 3.1 yaw refers to the angular offset of a face about its vertical axis. Sankaran et al. estimate yaw by considering geometric relationships between 4 line segments: The line segment connecting the left eye to the right eye, the line segment connecting the left eye to the center of the nose, the line segment connecting the right eye to the center of the nose, and the line segment that goes through the center of the nose and is perpendicular to the line segment connecting both eyes. We can label the the coordinates of the left eye, right eye, and nose as $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) respectively, and the aforementioned line segments as L_1, L_2, L_3, and L_4 respectively. The slopes corresponding to the line segments are then $m_1 = \frac{y_2-y_1}{x_2-x_1}$, $m_2 = \frac{y_3-y_1}{x_3-x_1}$, $m_3 = \frac{y_3-y_2}{x_3-x_2}$, and $m_4 = \frac{-1}{m_1}$. All of these points, line segments, and slopes are shown in figure 3.2. The angle between slopes m_1 and m_2 is $\alpha_1 = tan^{-1}\left(\frac{m_1-m_2}{1+m_1m_2}\right)$, while the angle between slopes m_3 and m_4 is $\alpha_2 = tan^{-1}\left(\frac{m_3-m_4}{1+m_3m_4}\right)$. Assuming a cylindrical face model, yaw is approximately the difference between the two angles, $\alpha_1 - \alpha_2$. Converting from radians to degrees and shifting rotation so that such that yaw takes values between -90^o and 90^o, corresponding to the head facing left and right respectively[1], the equation for yaw becomes

$$yaw = (\alpha_2 - \alpha_1)\frac{180}{\pi} + 90. \tag{3.1}$$

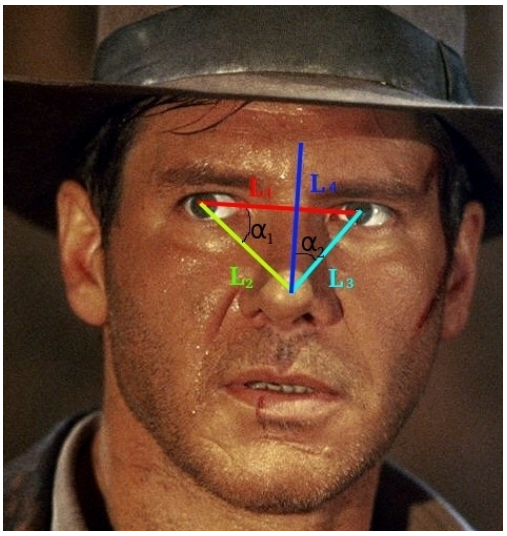

FIGURE 3.2: Geometric fiducial relations from which yaw can be estimated.

[1]This assumes the perspective of the image viewer.

When implementing this yaw determination technique, we must account for two considerations not mentioned by Sankaran et al.: First, as denominators $\rightarrow 0$, numerators $\rightarrow \infty$. We therefore check if divisors equal zero and set them to a small fraction (0.001) if they do. Second, due to inherent variations in facial geometries and fiducial point detection errors, estimated yaw angles can fail as the face approaches $\pm 90^o$ from frontal. We deal with these cases by checking the side of the eyes on which the tip of the nose is located. If $\alpha_1 - \alpha_2$ is positive and the nose is to the right of both eyes as seen by the viewer (i.e., $x_3 > x_2 \geq x_1$), we assign the yaw angle to 90^o. If the nose is to the left of both eyes (i.e., $x_3 < x_1 \leq x_2$), then we assign the yaw angle to -90^o. We tested the yaw estimation technique on several images from the FEI face dataset along with their ground truth fiducial coordinates. Empirical results for two of the subjects are shown in figure 3.3. The results appear qualitatively accurate.

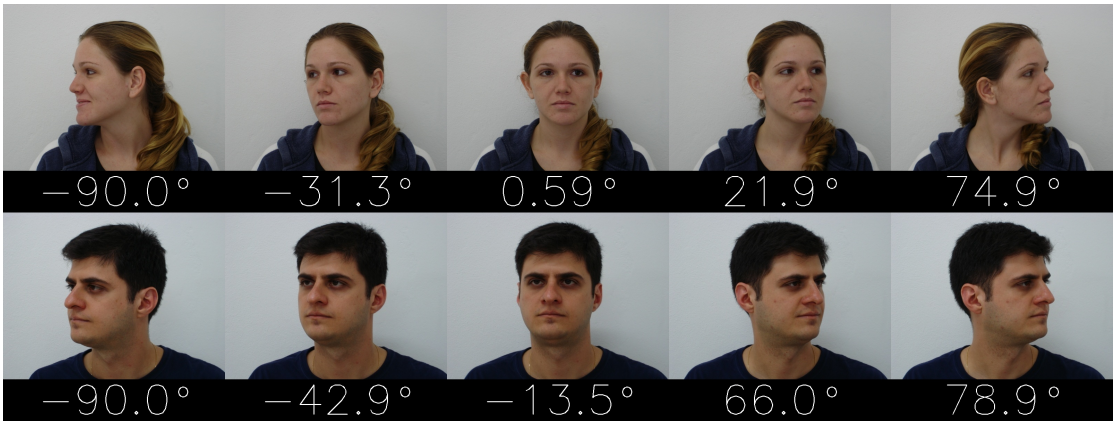

FIGURE 3.3: Qualitatively accurate yaw estimates of images of two subjects from the FEI face dataset [7] using the technique of Sankaran et al.

3.1.2 Estimating Roll

As illustrated in figure 3.1, roll describes the angle about the axis perpendicular to the image plane. Given eye points roll is quite easy to estimate and correct for: It is simply the deviation of left eye - right eye axis from horizontal, or $roll = tan^{-1}(m_1)$.

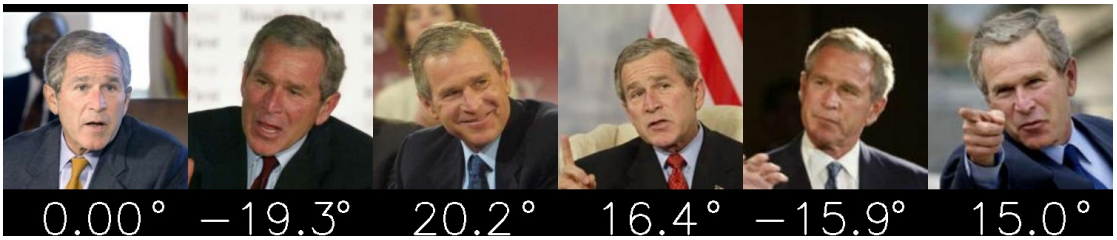

FIGURE 3.4: Detected roll angles for several LFW images.

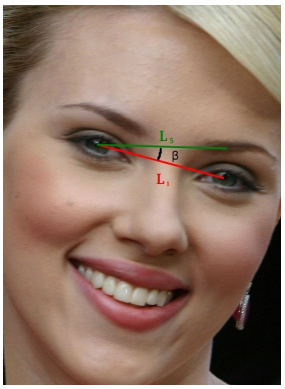

FIGURE 3.5: Let β constitute the roll angle, and define L_5 to be the horizontal line going through the left eye. β is simply the angle between lines L_1 and L_5.

3.1.3 Projecting Mesh Points onto A 3D Model

We would like a way of estimating scale from a model once it is fit to the approximate pose and scale of the human face within an image. For this, we follow the lead of Yi et al. in [21], using frontal pose as a frame of reference and orthographically projecting a mesh of points onto the face. To obtain our face model, we use a generic model of the human head obtained from 3D scan in .obj waveform format, a format in which vertices are defined by their X,Y,Z coordinates and triangles are defined as 3 numbers, each corresponding to an index of a mesh point within the model. Our model consists of 1,781 vertices and 3,410 triangles. However, not all points on the human head are relevant to the face and having to sort out which of two triangles (i.e., on the front of the face or on the back of the head) a projection corresponds to is computationally inefficient. We therefore choose a threshold along the Z-axis, i.e., the axis perpendicular to the model face, beyond which to ignore mesh points and triangles when intersecting a mesh with the model.

To intersect a mesh of points with the face model, we "shoot" a grid of 1600 (40×40) vectors at the face, with each vector parallel to the z-axis and evenly spaced between others in x and y coordinates. We determine x and y ranges of the grid by observing a plot of the model. An illustration of the "shooting" process is shown in figure 3.7. For each of the vectors, we then iterate over all triangles to determine which, if any, of the triangles that vector intersected and the point of intersection. We determine if a vector intersects a triangle by checking if its xy coordinates lie on the same side of all lines defining the triangle. Denote the triangle defined by 3 points (A, B, C) in $\mathbb{R}^3$ projected into the xy plane as (a, b, c) in $\mathbb{R}^2$. Let point $\vec{V}$ be a vector in $\mathbb{R}^3$ that intersects the xy plane at point p in $\mathbb{R}^2$, corresponding to the xy location of the mesh point of interest. Then the point is in the triangle if it is on the same side of the line defined by $\vec{ab}$ as c, on the same side of the line defined by $\vec{bc}$ as a, and on the same side of the line defined by

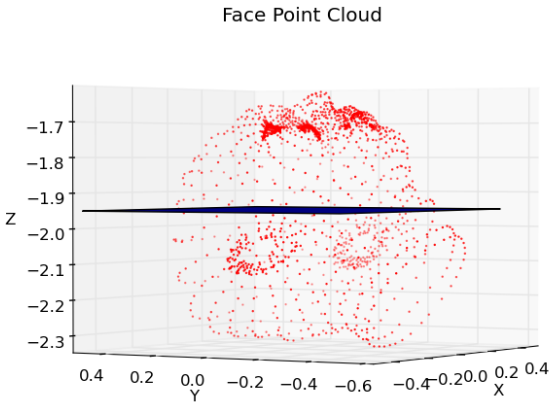

FIGURE 3.6: The threshold below which which we did not draw mesh points corresponds to the blue plane at $z = -1.95$ (in the .obj file's original model coordinates).

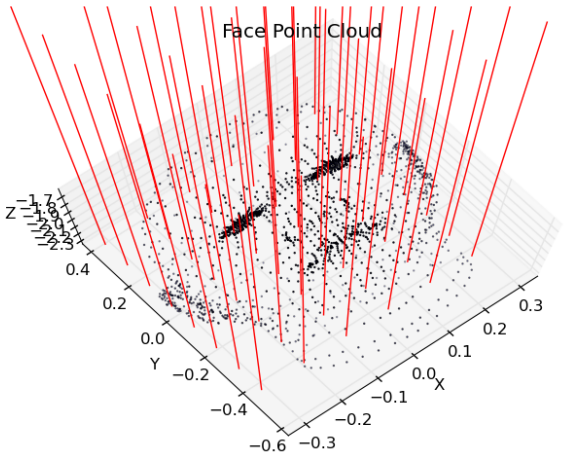

FIGURE 3.7: Mesh points were obtained by "shooting" vectors at the frontal face and finding the points of intersection.

$\vec{ac}$ as b. We can check if the point p is on the same side of the line parallel to $\vec{bc}$ as point a by checking the value of s in 3.2. If $s > 0$ then the point is on the same side of the line as a. An analogous check can be performed for the two other lines. An even more efficient technique can be performed using Barycentric coordinates (cf. section 3.2.2).

$$s = (\overrightarrow{(c-b)} \times \overrightarrow{(p-b)}) \cdot (\overrightarrow{(c-b)} \times \overrightarrow{(a-b)}) \tag{3.2}$$

Once we have determined that a mesh point viewed in $\mathbb{R}^2$ lies inside triangle (a, b, c) we can determine $\vec{V}$'s intersection Q with the plane defined by (A, B, C) via

$$Q = p + \left(\frac{\overrightarrow{(a-p)} \cdot \vec{n}}{\hat{z} \cdot \vec{n}} \right) \hat{z}, \tag{3.3}$$

where p is the point of intersection of $\vec{V}$ with triangle (A, B, C) in the xy plane (i.e., z coordinate zero), $\vec{n} = \overrightarrow{(b-a)} \times \overrightarrow{(c-a)}$ is the normal to the plane, and $\hat{z}$ is the unit vector $(0, 0, 1)$.

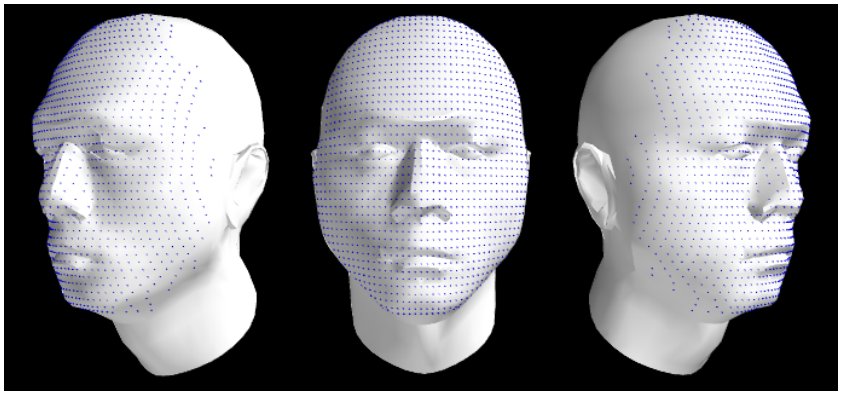

FIGURE 3.8: The mesh points on the model post projection. In the center, we see the frontal view of our face model. On the left and right we see the model rotated by a yaw angle of $\pm 45°$ respectively. Notice that mesh point spacing within the image plane changes with pose due to out of plane rotations, conveying both information about scale and information about the boundaries of the face. Note that although hidden surface and point removal is performed by the visualization software used to synthesize this image, we did not implement occlusion culling in our algorithm. We chose to ignore this detail because all images in our dataset are detectable via a frontal cascade classifier and mesh points are spaced far enough apart that hidden surface effects are negligible in most cases.

3.1.4 Fitting Mesh Points to Face Images

Using the technique discussed in section 3.1.3, we can project mesh points onto our face model and thereby obtain their 3D locations. However, we still require a means of fitting the points to a face image post projection. The fit entails first rotating the model to the pose of the face via rigid body rotation, projecting to 2D, and translating and scaling the warped 2D mesh of points so that the eye points align with those of the image. Given estimation of pose parameters, the following rotation matrices can be multiplied with any vector in $\mathbb{R}^3$ to perform perform pitch, roll, and yaw perturbations about the origin, which we define as the barycenter of all mesh points.

$$R_{roll}(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} R_{yaw}(\theta) = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix} R_{pitch}(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix}$$

Once the mesh points have been multiplied by their appropriate rotation matrices, we scale them to the corresponding face image by multiplying by the ratio of the Euclidean distance between eyes in the image and the Euclidean distance between projected eye points of the model on the xy plane, presuming the barycenter of the mesh points as the origin.

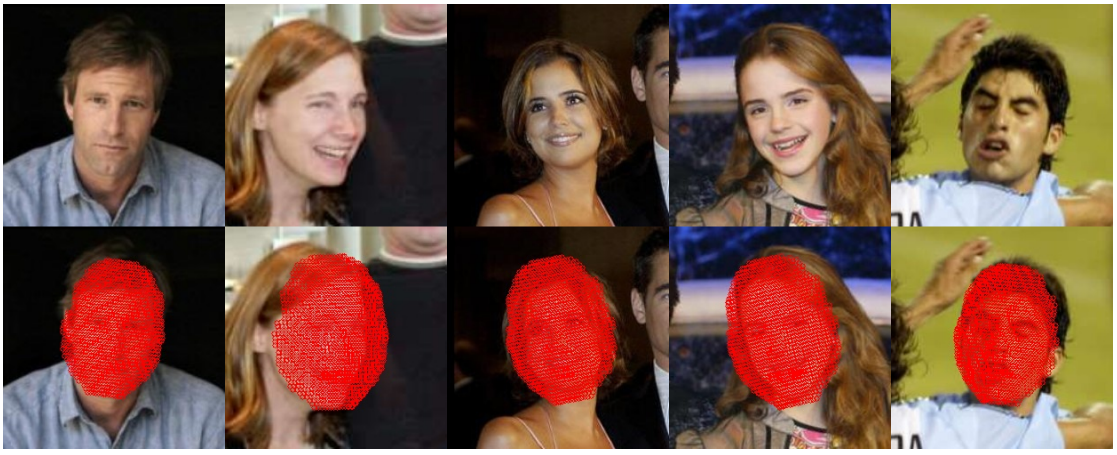

FIGURE 3.9: A fit of the mesh points, transformed and projected from the face model to five images from the *LFW* face dataset. Points are plotted as overlapping circles to offer insight into how scale due to in-plane rotation varies with pose. Note that although eye coordinates between models and images align, model $\leftrightarrow$ face correspondence is better for some images than for others due to variations in human face shape.

After scaling the mesh points, we convert them to image coordinates, where the origin is at top left corner of the image, x increases from left to right, and y increases from top to bottom. Given (x_{lem}, y_{lem}) corresponding to the left eye point on the model post projection, (x_{lei}, y_{lei}) corresponding to the left eye point on the image, and h_i corresponding to the height of the image, this conversion amounts to the following assignments:

1. $x \leftarrow x - x_{lei} - x_{lem} \forall x \in$ model
2. $y_{lei} \leftarrow (h_i - y_{lei})$
3. $y \leftarrow y + y_{lei} - y_{lem} \forall y \in$ model
4. $y \leftarrow h_i - y \forall y \in$ model.

Several 2D meshes are shown in figure 3.9 fit to several of the LFW images using the techniques in this section as well as sections 3.1.1 and 3.1.3.

3.1.5 Using Mesh Points to Incorporate Scale information into Feature Vectors

In section 3.1.4 we discussed how to fit a face mesh to an image. To incorporate scale and pose information into the feature vector, however, the face along with the mesh points must still be transformed to line up approximately. For this alignment, we use a similarity transformation discussed in greater detail in section 4. Note that the same transformation must be applied to the mesh points as to the image so that information consistent with initial alignment of the model with the face is imparted from the model to the feature vector.

Upon realignment of the mesh points and cropping, a small number of the mesh points may lie outside of the image boundary. We simply filter out these mesh points, ignoring them since they do not lie in a region corresponding to a GRAB histogram. Of the points that remain, we bin them over the GRAB histogram regions and divide by the total number of remaining points. In this way, we have a rough estimate of the relative scale within each region which we can use to either weight GRAB histogram or append to the GRAB histogram. A visualization of how the classifier "sees" this added information is provided in figure 3.10.

The standard GRAB feature vector consists of a concatenation of normalized histograms, each corresponding to a square, non-overlapping region within the image. Specifically, the ith histogram may be defined as

$$H_i = \left[\frac{\sum_{p \in R_i} P(p,0)}{|R_i|} | \ldots | \frac{\sum_{p \in R_i} P(p,255)}{|R_i|} \right], \tag{3.4}$$

where $|$ designates concatenation, $|R_i|$ is the number of pixels in region R_i, $P(p,k)$ is 1 if pixel $p = k$ and zero otherwise. The baseline feature vector can then be written as

$$F = [H_1 | H_2 | \ldots | H_N]. \tag{3.5}$$

Now, let scale vector S, be an N-element vector derived from mesh points, in other words,

$$S = \left[\frac{\sum_{k=1}^{|M|} M_k \in R_0}{|M|} | \ldots | \frac{\sum_{k=1}^{|M|} M_k \in R_N}{|M|} \right], \tag{3.6}$$

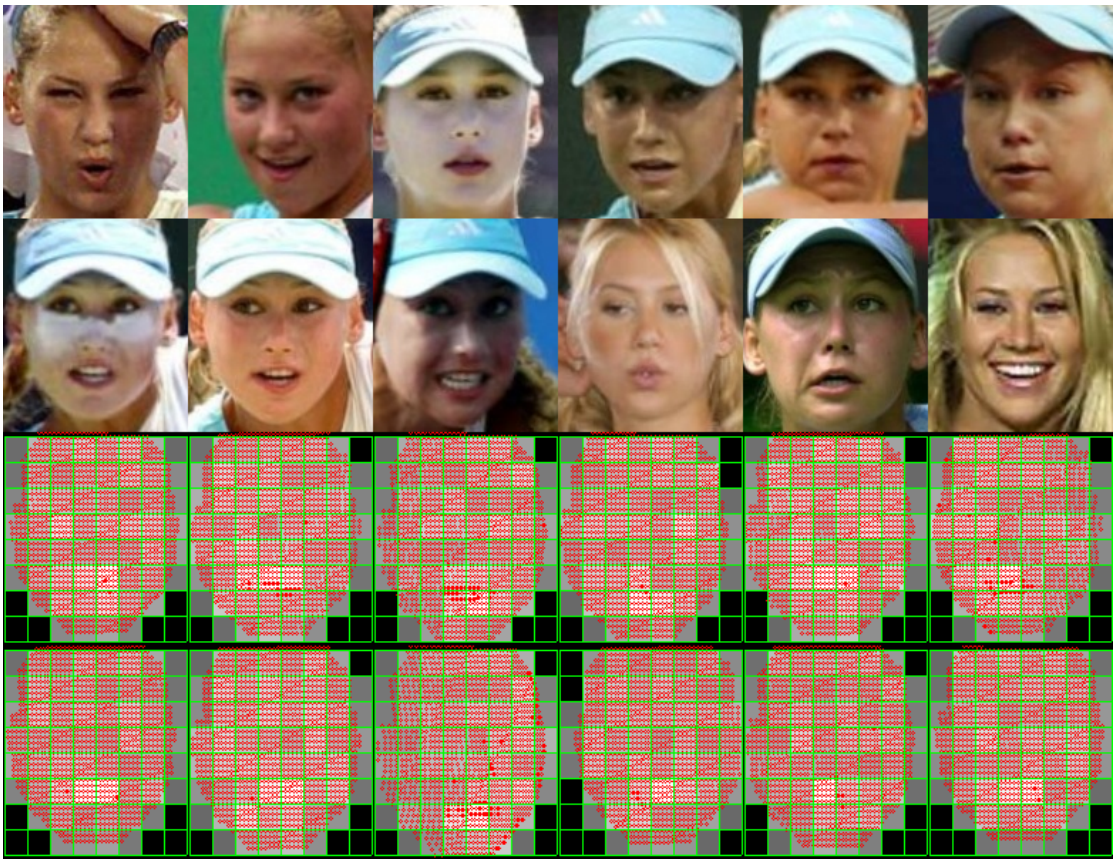

FIGURE 3.10: Geometrically normalized images along with heat map representations of scale density obtained by binning mesh points. Mesh points are shown in red, green boxes denote GRAB histogram bins and grayscale intensity indicates relative number of points per bin.

where M is the set of mesh points, M_k is the kth mesh point, and $|M|$ is the number of mesh points. Then, we have two choices for incorporating scale into the feature vector: We can either append so that our new feature vector becomes,

$$F' = [H_1|H_2|\dots|H_N|S]\,, \tag{3.7}$$

or we can weight our feature vector, so that F' maintains the same length as F and

$$F' = [w_1H_1|w_2H_2|\dots|w_NH_N]\,, \tag{3.8}$$

where w_i is a derived from some function on S_i and w_iH_i denotes the multiplication of w_i with each element of H_i. The question then becomes choice of function $f : S_i \to w_i$ to use. Naturally there are a lot of options, the simplest being $f(x) = x$. The problem with using $f(x) = x$ is that some useful information outside of the mesh points (e.g.,

parts of the head or even parts of the face not covered by mesh points) may be lost entirely, since S_i for that histogram region is 0. A manner which is more forgiving of alignment errors is to choose $f(x) = 1 + cx$, with c as a constant. This has the effect of upweighting regions by their relative number of mesh points, the weighting determined by c, while down-weighting, but not ignoring the rest of the image.

3.2 Localized Scale Estimation with an Active Appearance Model

In the previous section, we introduced a technique which involved fitting a roughly pose-aligned rigid 3D model of a generic face to each image and using the relative scale variations induced by changes in pose on the model to alter the feature vector obtained from the image. The effectiveness of this technique is dependent on the extent to which scale information from a generic 3D face can be generalized to specific images. In this section, we use a more rigorous fitting algorithm in the form of an Active Appearance Model (AAM), pre-trained on the MUCT dataset [9]. The MUCT dataset consists of 3,775 faces with 76 hand-labeled landmarks per face. The MUCT dataset is an ideal dataset for training a 2D model due to diversities in age, ethnicity, and gender throughout the dataset (cf. figure 3.12). Additionally, MUCT contains five different camera angles and three different illuminations for each subject (cf. figures 3.13 and 3.14). Rather than code our own implementation we used the source code provided in the CSIRO face analysis SDK [8], the state of the art in open source AAM implementations, with the AAM trained on 66 of the 76 landmark points per image in the MUCT dataset. The landmark points used by the AAM are shown in figure 3.11.

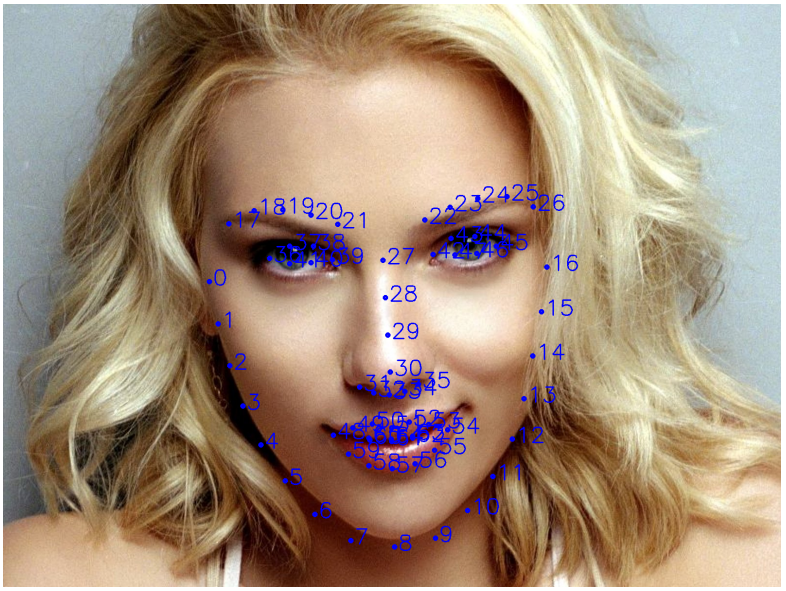

FIGURE 3.11: Landmarks for this image were obtained by fitting an AAM trained on the MUCT dataset. 66 landmarks were used to train the AAM. These landmarks are numbered 0-65. The source code for the AAM implementation was provided by [8].

FIGURE 3.12: Sample images of subjects from the MUCT dataset used to train the AAM. Subjects are diverse in terms of age, gender, and ethnicity. Images from [9].

FIGURE 3.13: The different poses each subject assumes within the MUCT dataset used to train the AAM. Different poses in the training set yield better fit across pose. Images from [9].

FIGURE 3.14: The three different lightings present in the MUCT dataset used to train the AAM. Different lightings in the training set yield better robustness to different illumination conditions. Images from [9].

3.2.1 Piecewise Scale Estimation

To estimate the scale of the face, we define triangles from the mesh points in figure 3.11. For horizontal scale estimation, we define many triangles ranging the fullest vertical extent of the face possible. For simplicity, these triangles are non-overlapping and are similar in horizontal extent under frontal pose. For vertical scale estimation, we define triangles, each of which range half the horizontal extent of the face – from left/right side to center or visa versa. These triangles are shown in figures 3.15 and 3.16 respectively.

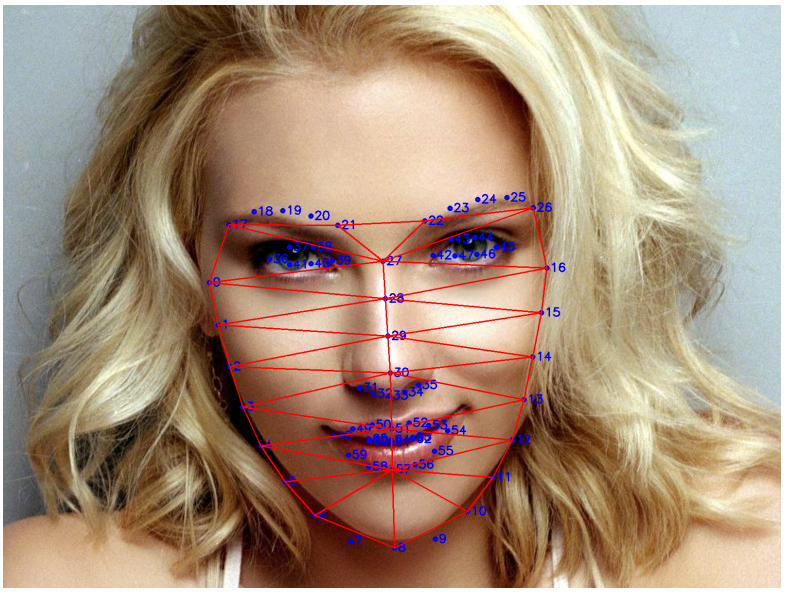

FIGURE 3.15: Latitudinal triangles defined using the AAM landmarks. We use these triangles to estimate scale changes due to rotations in pitch in the image plane.

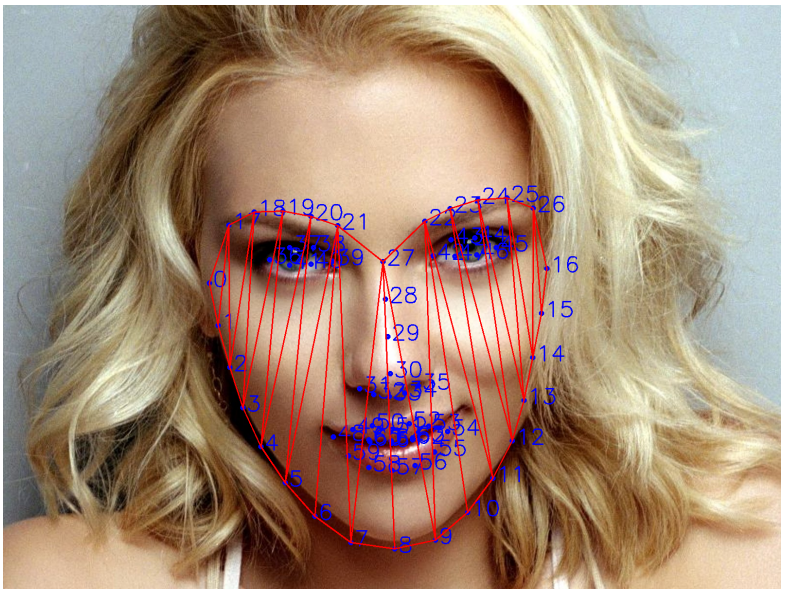

FIGURE 3.16: Longitudinal triangles defined using the AAM landmarks. We use these triangles to estimate scale changes due to rotations in yaw in the image plane.

Given our triangle definitions and an AAM fit to a face image, the question then becomes how to estimate localized scale of the face. For this, we exploit the topological invariance of AAMs: Triangles defined by the same vertices for two different face images correspond to approximately the same locations on the face. The accuracy of this correspondence depends on the quality of fit of the model, which is inherently dependent on the quality of labeling of the training set. For a given triangle T_i in one fit, and a corresponding triangle T_i', from another fit, we can compare the scale of two triangles by inspecting their relative widths $\bar{w_i}/\bar{w_{i'}}$ and heights $\bar{h_i}/\bar{h_{i'}}$. For triangle $T_i = ((x_{i1}, y_{i1}), (x_{i2}, y_{i2}), (x_{i3}, y_{i3}))$

$$\bar{w_i} = \frac{1}{3}\left(\sum_{j=1}^{3} x_{ij} - argmin_k(x_{ik})\right) \text{ and} \tag{3.9}$$

$$\bar{h_i} = \frac{1}{3}\left(\sum_{j=1}^{3} y_{ij} - argmin_k(y_{ik})\right). \tag{3.10}$$

Relative average width and height measurements can be obtained either via comparison between corresponding triangles obtained from an AAM fit to two different images or via comparison of the AAM fit to one image with the aggregate shape of several images. For each pixel within the triangle, we then define its vertical scale as the relative average height between the triangle and a baseline triangle. We define the horizontal scale analogously. This give us a piecewise definition of scale for the image.

Were multiple overlapping triangles used for the scale definition, pixels subsumed by those triangles could simply be assigned to the respective means of the relative widths and relative heights of all triangles which subsume them. However, we do not superpose triangles in this work.

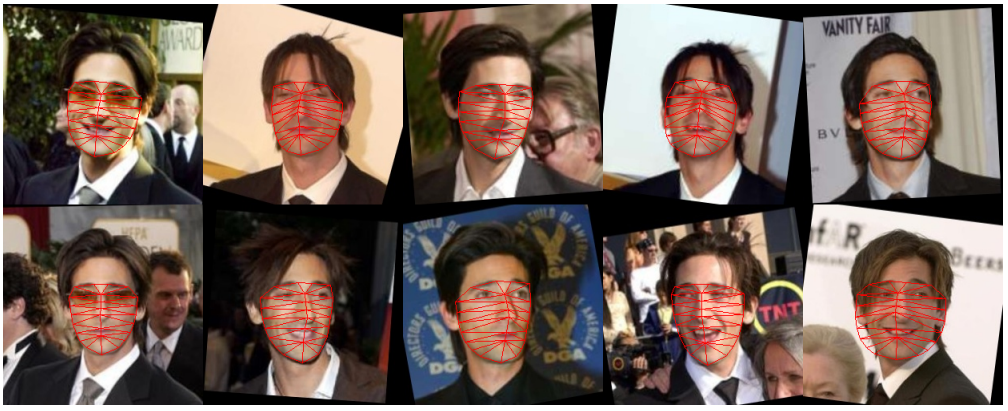

FIGURE 3.17: Note how the latitudinal triangles capture scale changes due to pitch rotations for several images of a subject from LFW. Yaw changes are also captured. However, when the AAM fit fails the associated scale information is spurious.

FIGURE 3.18: Note how the longitudinal triangles capture scale changes due to yaw rotations for several images of a subject from LFW. Pitch changes are also captured. However, when the AAM fit fails the associated scale information is spurious.

3.2.2 Practical Consideration: Efficient Triangular Closure in Barycentric Coordinates

In order to apply our technique, we need a way of determining which pixels within each image are in a given triangle. This is the same problem that we addressed in the section 3.1.3 when determining which mesh points intersect a particular triangle, but on a much larger scale, especially when working with data sets containing thousands of images. We therefore turn to a more efficient technique which leverages barycentric coordinates.

Given a triangle defined by points A, B, C in Euclidean space, along with arbitrary point P, the barycentric parameterization of P is $P = A + u(\overrightarrow{C - A}) + v(\overrightarrow{B - A})$. A point is in the triangle if

1. $u + v < 1$,
2. $u > 0$, and
3. $v > 0$.

We can turn our parameterization into two equations by subtracting A from the initial parameterization,

$$P = A + u(\overrightarrow{C - A}) + v(\overrightarrow{B - A}) \text{and} \tag{3.11}$$

$$(\overrightarrow{P - A}) = u(\overrightarrow{C - A}) + v(\overrightarrow{B - A}). \tag{3.12}$$

The equations in 3.11 constitute a system of two equations in two unknowns. Solving the system allows us to solve for u and v as follows:

$$u = \frac{(\overrightarrow{(B-A)} \cdot \overrightarrow{(B-A)})(\overrightarrow{(P-A)} \cdot \overrightarrow{(C-A)}) - (\overrightarrow{(B-A)} \cdot \overrightarrow{(C-A)})(\overrightarrow{(P-A)} \cdot \overrightarrow{(B-A)})}{(\overrightarrow{(C-A)} \cdot \overrightarrow{(C-A)})(\overrightarrow{(B-A)} \cdot \overrightarrow{(B-A)}) - (\overrightarrow{(C-A)} \cdot \overrightarrow{(B-A)})(\overrightarrow{(B-A)} \cdot \overrightarrow{(C-A)})} \quad (3.13)$$

$$v = \frac{(\overrightarrow{(C-A)} \cdot \overrightarrow{(C-A)})(\overrightarrow{(P-A)} \cdot \overrightarrow{(B-A)}) - (\overrightarrow{(C-A)} \cdot \overrightarrow{(B-A)})(\overrightarrow{(P-A)} \cdot \overrightarrow{(C-A)})}{(\overrightarrow{(C-A)} \cdot \overrightarrow{(C-A)})(\overrightarrow{(B-A)} \cdot \overrightarrow{(B-A)}) - (\overrightarrow{(C-A)} \cdot \overrightarrow{(B-A)})(\overrightarrow{(B-A)} \cdot \overrightarrow{(C-A)})}. \quad (3.14)$$

A check for membership in any triangle then reduces to inspecting the values of the above dot products, an operation which lends itself well to implementation on a GPU. The ability to mask out triangles derived from an AAM at the pixel level allows us to perform powerful image manipulations, working with individual triangles or multiple triangles, for example, in figure 3.19, we use the technique to remove extraneous background and non-face information from the image in 3.11, while in figure 3.20 we manipulate individual triangles, spreading each horizontally from its nearest neighbor by 10 pixels.

FIGURE 3.19: By checking per-pixel triangular closure in barycentric coordinates, we can perform background subtraction, isolating only the parts of the image relevant to facial identity.

3.3 Using Localized Scale Estimation for GRAB Scale Selection

Unlike our technique in 3.1.5, when estimating pixel-wise GRAB scale, we do not alter the GRAB histograms themselves. Rather, we adjust scales on the GRAB images from which histograms are obtained. Specifically, we select vertical and horizontal scales for each triangle within the image by comparing the relative average width and relative average height of each triangle within the image to the distribution of widths and heights for the same triangle over an entire dataset.

FIGURE 3.20: Via our per-pixel triangular closure in barycentric coordinates, we can efficiently manipulate triangles derived from the mesh points fit by the AAM.

The separation of horizontal and vertical scaling requires a generalization of the GRAB operator so that it is no longer confined to square regions, but rather to generalized rectangular regions defined by two distinct GRAB scales – a horizontal GRAB scale and a vertical GRAB scale. Under this definition, a canonical GRAB-k extraction becomes a GRAB-$k-k$ extraction and an extraction of horizontal scale i and vertical scale j becomes a GRAB-$i-j$ extraction. We calibrate our piecewise GRAB scale extraction, in which a combination of scales is used per image, against a "GRAB bag" of $i-j$ combinations. Our "GRAB bag" consists of a grid over an $i-j$ range. Images for which the AAM failed to detect a face, we simply assign to the most accurate scale (as measured with respect to some reference dataset). Likewise, we assign all pixels *not* in any of the triangles to the most accurate scale.

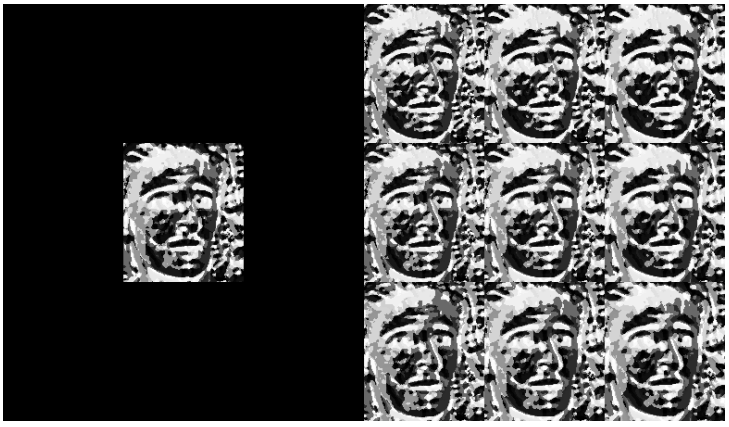

FIGURE 3.21: Left: A piecewise GRAB image composed of vertical scales 7,9,11 and horizontal scales 5,7,9. Right: The corresponding GRAB components. From left to right, top to bottom are GRAB images from scales 5-7, 5-9, 5-11, 7-7, 7-9, 7-11.

We could use any number of mappings between triangle widths/heights to different horizontal/vertical GRAB scales. In our experiments, we simply experiment with different n-tiles with respect to the width and height distributions for each triangle over the dataset. When using three GRAB scales, for example, and dividing the widths and heights into tertiles, we would assign pixels in longitudinal triangles, whose widths were in the first tertile the lowest horizontal GRAB scale, pixels in triangles whose widths were in the second tertile the middle GRAB scale, and pixels whose widths were in the third tertile the highest GRAB scale. An analogous procedure would be performed for latitudinal triangles and vertical GRAB scales. Using n-tiles is only one way to assign GRAB scales based on triangle width and height distributions. Others could include clustering or Gaussian mixture models.

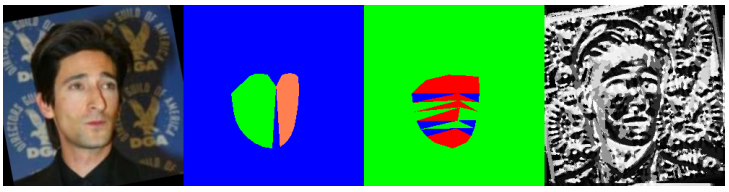

FIGURE 3.22: From left to right we see the raw LFW image, a map of the horizontal GRAB scales, a map of the vertical GRAB scales, and finally the piecewise GRAB image. In the scale maps, orange corresponds to scale 5, blue corresponds to scale 7, green corresponds to scale 9, and red corresponds to scale 11. Scales increase and decrease as we would expect from the pose of the image, at least qualitatively.

Chapter 4

Experiments, Results, and Discussion

4.1 Dataset Description

For our experiments we use the Labeled Faces in the Wild (LFW) dataset [10]. LFW is one of the largest and most well known unconstrained face data sets in the world. The dataset was tailored specifically for the problem "Given two pictures, each of which contains a face, decide whether the two people pictured represent the same individual" [10]. This problem is commonly referred to as *pairwise face verification* or *face authentication*. However, subsets of the dataset have been applied to other problems, including generalized recognition (e.g., [12] – given a closed-set gallery of faces determine to which of the gallery members the probe image belongs).

The images in LFW consist mostly of color images with a few grayscale images. The file names of the images contain labeled identity information corresponding to the individual at the center of each image. All individuals in LFW are public figures, the images of whom were obtained predominantly from online media sources. Each raw LFW image is 250x250 pixels. The LFW faces were detected by a Viola-Jones algorithm and aligned by bounding box to the center of the image, translating into a very loose alignment. In total, the dataset consists of 13,233 images with 5,749 individuals, 1,680 in two or more images and 610 in four or more images.

Two protocols are recommended for running LFW: The View 1 protocol for parameter selection and the View 2 protocol for publishing official results. View 1 consists of 1100 match pairs and 1100 non-match pairs for training and 500 match and 500 non-match pairs for testing. View 2 consists of 6000 pairs of images; half match and half non-match.

The protocol dictates a split into 10 pre-assigned folds, where each fold consists of 300 match images and 300 non-match images. The View 2 classification experiment consists of 10 fold cross validation in a leave-one-out train/test regimen; i.e., nine folds used in training; nine in testing. Though all pairs are unique, some images are duplicated. However, the 9-fold split is made so that no two images overlap between train and test.

FIGURE 4.1: Sample images from the LFW dataset. The pairwise matching problem is to determine whether pairs of images correspond to matches (e.g., left), or non-matches (e.g., right). Note that images of the same person vary dramatically in terms of all unconstrained factors including pose. Images courtesy of [10].

For accuracy reporting across folds, the mean accuracy and standard error in the mean ($S_E = \frac{\hat{\sigma}}{\sqrt{10}}$)are used as measurements. By the rules of the View 2 protocol, an algorithm may not set its parameters to maximize performance on each fold.

The LFW dataset has several variations including sophisticated pre-alignments, using funneling [25], deep funneling [26], and state-of-the-art commercial face alignment software [27]. Additionally, sophisticated metadata has been collected. This metadata includes image segmentation ground truth and manually acquired ground truths of fiducial coordinates [28].

4.2 Classification Experiments

4.2.1 Dataset Preprocessing and Alignment

Many intricate normalization schemes exist including [25], [26], and [27]. However, our research goal is not to obtain the best performance by using the most sophisticated alignment strategy in existence; Rather, our goal is to establish potential performance gains over a baseline algorithm by incorporating pose information described in chapter 3. As this entails transforming model points consistent with the alignments on baseline images themselves, we chose to pursue a linear normalization scheme: Given ground truth LFW fiducial coordinates in [28], we take the mean left and right eye coordinates. For each image, we then obtain rotation and scaling parameters to line up the eye coordinates on that image with the mean eye coordinates. Applying this transformation to each image yields an approximately aligned set of images. We then crop the images to 130x150, to remove background noise.

Note that better alignment is easily obtained even with a linear model using 3 fiducial points as opposed to 2, or doing a least-squares fit of all fiducial points to the mean to obtain an affine matrix. Our choice of similarity transform on just the eye coordinates over a more generalized affine transform is based on the fact that a least squares approximation introduces distortion of the eye alignments which are very important for preserving identity information due to the rigidity of the periocular area of the face [29]. Our choice is also based on the fact that similarity transformations maintain the ratios and appearance of the face, while an affine transform that shears the image causes uneven interpolation to occur and distorts the appearance of the face, even if fiducial points are better aligned with the average of the dataset.

4.2.2 Classifier and Parameter Choice

For classification, we chose Gaussian radial basis function kernel support vector machines (RBF SVMs)due to their good "off the shelf" performance. RBF SVMs have two parameters which require estimation: C and γ. The γ parameter is the standard deviation of the kernel used to lift the dimensionality of the features space, the optimal value of which depends on the relative proximities of the training data points. The C parameter is a cost coefficient, whose selection can be thought of as regularizing generalization versus goodness of fit to the training data.

Parameter estimation is an inexact science, but we loosely follow the guidelines set forth by Chappelle and Zien [30] and Hsu et al. [31], by uniformly normalizing all

FIGURE 4.2: A similarity transform (top) and a shearing affine transform (bottom) on two LFW images. Note that for the similarity transformation, identity information is maintained, whereas for the shearing affine transform identity information is distorted due to the uneven interpolation.

FIGURE 4.3: Top: Raw LFW images. Bottom: The same images geometrically normalized about the eye coordinates via similarity transform and converted to grayscale. Note that even with alignment there exists considerable variance between images due to pose and other exogenous factors (expression, illumination, accessories, hairstyle, occlusion).

feature vectors so that their respective maxima are no greater than 1 in any dimension (by dividing by the total number of pixels considered for each GRAB histogram), then performing an order of magnitude grid search. Our grid consists of $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$.

During classification, we ran the grid search in parallel, performing cross-validation on the first view of the training data. We then used the RBF parameters which yielded best performance on the cross-validation experiment to train classifiers for each of the ten folds of the view 2 LFW protocol.

4.2.3 Baseline Performance: Conventional GRAB-k

The baseline performance of GRAB is shown in table 4.2.3, for GRAB scales 3,5,7, and 9. No threshold was used and a 3x3 blur was applied during denoising in all cases. Note that these accuracies are far below state of the art and are even below some in published work for GRAB, in which multiple scales were concatenated and dimensionality was reduced via uniform patterns and other reduction techniques (e.g., PCA, LDA). Our reasons for not pursuing these approaches in this paper are twofold: First, computational overhead is a consideration. Second, the objective of the baseline is to serve as a comparison for our proof-of-concept implementations: If our new features achieve superior performance to the highest performing individual GRAB scale using either a piecewise combination of scales in one image, or different weight representations of scales, then we say that our proof-of-concept implementation offers superior recognition performance, answering the underlying research question. The proof-of-concept implementation could then be extended to select weights or scales across a pyramid of GRAB scales instead of one.

Scale	**Mean Accuracy**	**C**	γ
GRAB-3	0.735 ± 0.006	2	2^{-3}
GRAB-5	0.739 ± 0.006	2	2^{-3}
GRAB-7	0.751 ± 0.005	2	2^{-3}
GRAB-9	0.755 ± 0.005	2	2^{-3}
GRAB-11	0.752 ± 0.005	2	2^{-3}
GRAB-13	0.754 ± 0.004	8	2^{-3}
GRAB-15	0.751 ± 0.003	8	2^{-3}

TABLE 4.1: The baseline performance of raw GRAB features on the View 2 LFW protocol over several scales along with parameter choices derived from cross validation on View 1. For each scale, the blur window was 3×3, image size was 130×150. Histograms were taken over an 8×8 grid.

4.2.4 Baseline Performance: Separating Horizontal and Vertical Scales

Our piecewise GRAB experiments require the separation of GRAB scales into horizontal and vertical components, this raises an interesting question: Is using equal horizontal and vertical scales the best way to formulate grid-based features in the first place? This question has not been raised to our knowledge in the formulation of LBP, MB-LBP, GRAB, and several other grid-based feature types, which use operators based on square or isotropic neighborhoods as opposed to rectangular/ellipsoidal ones. Particularly for rectangular neighborhoods, different widths and heights can be trivially accommodated with little computational overhead or alteration to the GRAB operator. Moreover, since the human face is taller than it is wide the apriori assumption that vertical and horizontal scales *should* be symmetric is a strange one.

Scale	**Mean Accuracy**	**C**	γ
GRAB-9-5	0.737 ± 0.006	2^3	2^{-7}
GRAB-9-7	0.735 ± 0.006	2^9	2^{-13}
GRAB-9-9	0.755 ± 0.005	2	2^{-3}
GRAB-9-11	0.751 ± 0.005	2^3	2^{-3}
GRAB-9-13	0.7572 ± 0.0037	2^3	2^{-3}

TABLE 4.2: Vertical baseline performance.

Scale	**Mean Accuracy**	**C**	γ
GRAB-5-9	0.752 ± 0.005	2^3	2^{-7}
GRAB-7-9	0.7573 ± 0.0044	2	2^3
GRAB-9-9	0.755 ± 0.005	2	2^{-3}
GRAB-11-9	0.750 ± 0.006	2^3	2^{-3}
GRAB-13-9	0.746 ± 0.005	2	2^{-3}

TABLE 4.3: Horizontal baseline performance.

As we alter horizontal and vertical GRAB scales, adjusting one, while keeping the other fixed at the winning scale from section 4.2.3, we notice that different horizontal and vertical scales offer both better and worse classification performance than the winning scale (GRAB-9) from section 4.2.3. This result suggests that separate horizontal and vertical GRAB scales should be used. Consistent with intuition from the vertical elongation of the face, we notice that the top two performing scales from our separate vertical-horizontal scale experiments are GRAB-$7-9$ and GRAB-$9-13$ respectively, suggesting that vertical scale should be greater than horizontal scale. This result also suggests that perhaps the consistency of ratio between vertical and horizontal scales with respect to the aspect ratio of the face has more impact on classification accuracy for small GRAB scale variations than does the absolute scale difference. We select GRAB-7-9 as the winning baseline for separate horizontal and vertical scales. Unfortunately, due to the magnitudes of standard error in the mean and our coarse grid search for RBF parameters, this claim is not without some statistical uncertainty.

4.2.5 Using Mesh Point Densities to Alter GRAB Feature Vectors

The experimental results on LFW of introducing mesh point density information from our generic 3D face model into GRAB-9 feature vectors are shown in table 4.2.5 along with a baseline accuracies obtained from GRAB-9 features. In all cases, our changes result in *inferior* performance. We discuss potential reasons for this decrease in accuracy in section 4.3, but we believe from visual observation that it is primarily because

alignment discrepancies between an average rigid face and real non-rigid faces introduce more noise than useful scale information.

The table can read as follows: rows with feature type $f(x) = 1 + cx$ correspond to each GRAB region (histogram) weighted by $1 + cx$, where c is a constant and x is the ratio of mesh points within the region to total mesh points. Interestingly, altering the value of c from 0.5 to 2.5 in increments of 0.5 does not noticeably change results. The result of concatenating all weights to each feature vector (Corresponding to the row of type "Concatenated" in the table) also decreases accuracy by a similar amount to weighting feature vector elements by $f(x) = 1 + cx$. When appending, in contrast to weighting, the unappended elements of the feature vector remain the same as base GRAB-9, which indicates that appended noise induces a tilt on the hyperplane, resulting in worse classification performance. The row labeled "No Offset" refers to simply weighting each feature vector region by the relative number of binned mesh points with respect to the image. It is not surprising that this feature type yields significantly worse performance because feature vector elements corresponding to parts of each face not subsumed by the generic face zeroed out and hence not seen by the classifier.

Feature Type	**Mean Accuracy**	**C**	γ
$f(x) = 1 + cx; c = 0.5$	0.737 ± 0.007	2048	2
$f(x) = 1 + cx; c = 1.0$	0.737 ± 0.007	2048	2
$f(x) = 1 + cx; c = 1.5$	0.737 ± 0.007	2048	2
$f(x) = 1 + cx; c = 2.0$	0.737 ± 0.007	2048	2
$f(x) = 1 + cx; c = 2.5$	0.737 ± 0.006	2048	2
Concatenated	0.737 ± 0.007	2048	8
No Offset($f(x) = x$)	0.681 ± 0.008	2^{-5}	8
GRAB-9	0.755 ± 0.005	2	2^{-3}

TABLE 4.4: LFW classification results of augmenting GRAB-9 feature vectors with scale information from a rigid 3D model.

4.2.6 Using Localized Scale Estimation for GRAB Scale Selection

The classification results obtained from synthesizing piecewise GRAB images of the LFW dataset are shown in table 4.2.6, along with baseline results on GRAB-$7-9$ and GRAB-$9-9$. Piecewise GRAB images were stitched together from scales of $\{5, 7, 9\} \times \{7, 9, 11\}$, where $\times$ denotes the cross product between sets. Pixels in the longitudinal triangles obtained from the AAM fit (cf. figure 3.16) were assigned horizontal scales of 5 and 9 respectively if their widths were in the first and last n-tiles for the corresponding triangles across the entire LFW dataset. Otherwise pixels in the longitudinal triangles were assigned scale 7. Similarly, pixels in the latitudinal triangles obtained from the AAM fit (cf. figure 3.15) were assigned vertical scales of 7 and 11 respectively if their

heights were in the first and last n-tiles for the corresponding triangles across the entire LFW dataset. Otherwise pixels in the latitudinal triangles were assigned scale 9. As in section 4.2.5, classification performance on LFW using feature vectors derived from piecewise GRAB images is inferior in accuracy to that obtained using feature vectors derived from both homogeneous GRAB-$9-7$ and GRAB-$9-9$ images. We discuss these results further in section 4.3 but suspect the performance degradations are due both to poor AAM fit and poor correspondence between GRAB scale and image scale.

Scale	**Mean Accuracy**	**C**	γ
GRAB-7-9	0.757 ± 0.004	2	2^3
GRAB-9-9	0.755 ± 0.005	2	2^{-3}
Piecewise Tertile Selection	0.746 ± 0.005	2^3	2^{-3}
Piecewise Quartile Selection	0.747 ± 0.005	2^3	2^{-3}

TABLE 4.5: LFW classification results using piecewise GRAB images.

4.3 Analysis and Discussion

Upon inspection of the fits in figures 3.10 and 3.9 it is not surprising that our technique of weighting the GRAB feature vector with mesh point densities taken from the average face did not work well: Visually speaking, a generic face model does not characterize individual faces well due to large variances in individual face structure. Because not all faces overlap with a generic model, when weighting histogram bins by the function $f(x) = cx$, for some images, parts of the face inevitably get left out. Even with the more forgiving function $f(x) = 1 + cx$, the poor quality of fit feasible with a rigid model likely introduces noise to the classifier. Possibly we could have obtained better results using a more sophisticated pose estimation technique which included pitch information and offered a more sophisticated treatment of the pose axis of rotation, but using only a single rigid 3D model, the potential for improvement from better pose estimation alone is extremely limited, not only due to nonrigid geometric aspects of the face, but also due to the variance in rigid face geometries. The approach might be useful for scale estimation given a 3D morphable model of better fit, but again, this raises time complexity and 3D shape/texture data availability concerns.

Assuming we could obtain a perfect fit of the model to the face, it is also questionable whether incorporating mesh point density information into the GRAB feature vectors even imbues the feature vector with useful scale information in the first place, since the feature vectors have already been extracted at a particular GRAB scale. Theoretically, if the classifier were "smart enough" to rely more on histogram elements that were closer to the correct scale, then it could "learn" when to rely more and when to rely

less on given feature vector elements when classifying images, especially if a multiscale GRAB pyramid were involved. However, using a small number of GRAB scales, this approach likely results in scale information loss compared to feeding the classifier feature vectors with *the right* scale(s) to begin with, making the latter approach "better", on the hypothetical that we know the optimal scales in advance. The reader should bear in mind, however, that *if* we could perfectly fit a 3D face to an image in the first place, there would be little point to estimating scale of our image to compensate for unconstrained factors, since we could constrain our model however we wanted.

Note that even when we append scale information to our feature vector, rather than use it for weighting, thereby increasing the dimensionality of our feature vector and removing no information, we see decreased performance, meaning that the added dimensions imbue a tilt on the hyperplane learned in RBF kernel space that causes the classifier to be overfit. This behavior is caused by a combination of two factors: random noise introduced due to errors in the appended scale information and too little training data to meaningfully sample the space of pose differences for match/non-match pairs. Which of these two factors dominates is unknown without further experimentation and analysis. We note that for the first factor, even if the fit of the generic face to the image were of high quality, the relatively small number of GRAB histograms could considerably degrade the accuracy of appended scale information. Increasing the number of histograms, and even interpolating (e.g., via kernel density estimation) could compensate for this type of degradation but would result in an very long feature vector, requiring some form of dimensionality reduction and would not compensate for poor quality of fit.

Unlike our mesh point binning technique, when estimating scales with our AAM, we did not alter the GRAB histograms, but rather synthesized piecewise GRAB scale images with scales derived from triangle sizes. These triangle-derived estimates of local face scale from AAM fits were qualitatively consistent with respect to face shape and orientation in a relative sense when the AAM fit was correct. However, classification using synthesized piecewise GRAB scales yielded slightly worse performance than when using homogeneous (GRAB-$7-9$) scales. We suspect that this decrease in performance is due predominantly to AAM fitting errors and improper correspondences between scale estimates and GRAB scales.

4.3.1 AAM Fitting Errors

From qualitatively examining the AAM fits, we found that, although the AAM found a face in over 12,000 of 13,233 LFW images, noticeable misalignment errors in at least one dimension (vertical or horizontal) were common, suggesting that we should have used a

more rigorous confidence threshold on our AAM, even if it would have resulted in fewer face detections. Although our choice of triangles did not account for expression when it could have, given the alignment errors we witnessed when inspecting our results, it is unlikely that choosing a set of triangles which incorporated expression information would have improved our results, since non-rigid parts of the face were more often mis-classified by the AAM than rigid parts of the face.

The most prevalent issue that we noticed in our AAM fits was the failure to detect in-plane yaw rotations, fitting the most visible half of the face well, but picking up background for the other half of the face, resulting in very wide horizontal scale, even for the compressed half of the face. Although the MUCT dataset used to train the AAM contains different poses for each subject, an inspection of figures 3.12 and 3.13 suggests that the range of poses in the subset of the MUCT dataset used to train the AAM is too narrow compared to the vast range of face shapes and sizes, causing the dominant modes of variation to be explained more by structural differences among faces than by in-plane pose rotations. Recall, from Chapter 2 that AAMs, and generally 2D and 3D morphable models often employ a coarse-to-fine strategy in their fit. The AAM implementation that we use [3] is no exception, giving optimization priority to the modes of highest explained variance in the initial stages of the fit, largely for efficiency reasons. This optimization technique combined with an inadequate yaw range throughout the training set likely explain the center-pose bias.

Although quite apparent, it might not be immediately obvious that dramatic AAM fitting errors can actually cause false-positive classifications in a homogeneous GRAB feature space to become true negative classifications in a piecewise GRAB feature space if mis-fit AAMs sufficiently distort the GRAB scales for two similar looking images of different identities. We witnessed this several times when examining which images were classified correctly by piecewise GRAB implementations and incorrectly by GRAB-$7-9$. This means that *poor fit cannot be directly related to inferior performance, even if poor fit were the sole reason for inferior performance.* The same can be said of our histogram binning technique.

4.3.2 Correspondence Between Scale Estimates and GRAB Scales

Our scale estimates based on n-tiles of triangle width and height distributions were relatively correct for good AAM fits, based on visual observation. However, little basis exists for the choice of GRAB scale that we assign to different triangles based on the n-tiles within which the different triangles lie with respect to some reference distribution. For example, variations of per-pixel scale due to pose differences within an image might span

multiple GRAB scales, or they might span less than one GRAB scale, e.g., depending on the depth of field of the lens used by the photographer. It is entirely possible that across the LFW dataset, there exists so little intra-image scale variation that our piecewise GRAB implementation was doomed to poorer performance than a homogeneous GRAB implementation regardless of the technique used to assign piecewise GRAB scales.

Likewise, our choice to assign scales based on n-tiles of distributions is largely predicated on the assumption of Gaussian scale variations. As illustrated in the appendices, for some triangles a normal distribution of widths and heights is a good assumption, while for others it is not. Ultimately the problem of obtaining correspondence between GRAB scales and scale estimates from an AAM is a regression problem.

On a related note, there also exists the possibility that our GRAB scales and our vertical/horizontal scale transitions are far too granular. To address scale transitions, we could 1.) use a superposition of many more triangles across the face and 2.) apply a smoothing function (e.g., Gaussian convolution). To address the granularity of individual GRAB scales, we could further extend the GRAB operator to draw on MB-LBP so that offsets of *any* radius could be used for the GRAB comparison: rather than relying only on actual pixels, we could generate virtual pixels for whatever choice of offset. For GRAB-4, for example, we could create virtual pixels, consisting of the average of pixels offset 3 from the center and 5 from the center respectfully. Virtual pixels corresponding to GRAB 3.5 would be $p_{iGRAB3.5} = \frac{3}{4} \times ofs_{iGRAB3} + \frac{1}{4} \times ofs_{iGRAB5}$. Generally, virtual pixels for the GRAB comparison of offset $k + m$ would be

$$p_{iGRAB(k+m)} = (2 - \frac{m}{2}) \times ofs_{iGRABk} + \frac{m}{2} \times ofs_{iGRAB(k+2)}; k \in \mathbb{Z}^+; m \in \{\mathbb{R}|0 \leq m \leq 2\}, \tag{4.1}$$

under a simple linear weighting scheme, although with slightly more overhead, non-linear basis interpolants such as Lagrange polynomials or cubic splines could also be used.

4.3.3 Effects of Parameter Selection on Results and Analysis

Although Gaussian RBF kernel SVMs yield superior classification performance to their linear counterparts, this performance comes at the costs of additional computational complexity, an additional parameter to estimate, and a far less intuitive classifier – a hyperplane in an infinite dimensional RBF inner product space or a very non-linear classifier in finite dimensional feature space depending on your perspective – versus a hyperplane in finite dimensional feature space.

Unfortunately, due to computational considerations, we cannot obtain optimal classifier parameters, but can only select those that perform best over a grid search on the validation set. Therefore, there is a chance that our results are incorrect – that we simply "stepped over" the optimal location in parameter space during our grid search. Unfortunately, there is no way to avoid this possibility, although one approach to give a better degree of certainty that one set of features outperforms another would be to achieve superior classification performance in the "winning" feature space with the parameters that yielded highest accuracy for the other feature space. Alternatively, a linear classifier could be used with a denser sampling of the the mis-classification cost parameter. Unfortunately, this introduces an increased chance of missing non-linear class boundaries.

We found that even though a classifier for one feature type often yielded superior performance to a classifier of another feature type, even the poorer performing classifier classified some pairs of images correctly that the winning classifier scored incorrectly. Unfortunately, why this occurred is often unknown, i.e., whether it is due to a noticeable difference in location of the point in feature space with respect to the others or whether of that class or whether it is due to a difference in RBF parameters.

4.3.4 Choice of LFW Dataset and Technique

In retrospect, our choice of LFW dataset was a poor one for testing our GRAB feature vector augmentations because it contains a random sampling of different unconstrained factors throughout the images. Likewise, our technique of using an AAM to estimate scale depends on the goodness of fit of the AAM, and does not assess directly how well the scale estimation technique works. Our reason for choosing to experiment on the LFW dataset was ironically due to its popularity for unconstrained problems and with readily available metadata. Unfortunately, few datasets offer associated ground truth for even a few fiducial points. The MUCT dataset is an exception we learned about after formulating much of our code base, although there is little established protocol for the MUCT dataset. The PIE [32] and Multi-PIE [33] datasets offer different poses, lightings, and expressions, keeping all but one of these factors constant across different partitions. Although the MULTI PIE dataset has associated fiducial points it is expensive to obtain. Likewise, we would have synthesized a 3DMM, but for the dearth of publicly available data. Fiducial points with corresponding ground truth according to an established protocol are a necessary and scarcely available requirement for research in unconstrained face recognition.

4.3.5 Decoupling of Horizontal and Vertical GRAB Scales

We found that decoupling horizontal and vertical GRAB scales offers superior matching results when the ratio between scales is closer to the natural aspect ratio of the face. Due to the relatively high uncertainties on our LFW dataset classification experiments, this claim should be substantiated across additional datasets, and across additional classes of objects other than faces.

Surprisingly, no research that we could find has been conducted on using different scales for vertical and horizontal components of grid-based operators. Possibly, this is due to the added difficulty, or more likely the foreseeable computational overhead required to extract features for non-rectangular patterns, e.g., incorporating the anisotropies in an elliptical version of MB-LBP could slow down feature extraction considerably. One of the strengths of GRAB features is their regularity, which is maintained under rectangular patterns.

A related issue which we did not address in our research is the difference in texture sampling with respect to radius between GRAB and MB-LBP as it is conventionally applied. In MB-LBP implementations, the number of sample points generally scale with the radius of the circle, while in GRAB implementations, they have remained constant. This maintains the regularity of GRAB, but causes the sampled texture information to fall as the inverse square of the offset. The degree to which this approach affects GRAB classification performance has not yet been addressed.

Chapter 5

Conclusion

Our experimental results were inconsistent with our expectations that the changes we made to incorporate scale information into the GRAB feature vector would yield enhanced recognition performance. Due to the degree to which facial recognition performance depends on proper alignment, and the common misalignments of our rigid 3D model and AAM, one key problem in our experiments was improper alignment. From our current results, we have no way of knowing whether our our underlying approach of trying to account for pose variations by incorporating scale information into the features is theoretically sound.

Should our approach be pursued in future research, we recommend that proper alignment be enforced via manual ground truth and that a data set with constrained variations in pose, illumination, and expression be used to ascertain the degree to which the technique works on a theoretical level. Validating the technique on a theoretical level, is a separate question from whether the technique can be practically be implemented in realistic applications of facial recognition, which likely hinges on goodness of fiducial point detections. From our observations on the LFW dataset, AAM fits work well in most cases but can fail dramatically in others. One possible way of detecting failures would be to use different types of fiducial detectors with different failure characteristics for the same points and measure the distance between detections. Another solution would be to bag or bootstrap the AAM training data to train multiple AAMs and use a voting method, on fiducial point location and detection consistency, similar in concept to a random forests classifier.

Though our current AAM implementation does not seem to fit with sufficient precision to estimate intra-face scale, perhaps it could be used with better results to estimate inter-face scale, e.g., over multiple faces at different distances in a particular frame. By measuring the divergence of landmarks, perhaps better global scales could be selected

over the entire face image. However, such an application would depend on the ability of the AAM to fit face images over many scales.

Although we did not derive new features well suited to unconstrained recognition as we had hoped, we did make several discoveries which may be useful in future research devoted to the design of pose-invariant features for facial recognition. Arguably the most important research discovery to draw from this work is that superior performance can be obtained with grid based features of different horizontal and vertical scale. This observation could prove useful in designing improved features for face recognition. Perhaps an even more suitable application which could leverage the capability of GRAB to accommodate multiple thresholds and asymmetric scales would be a cascaded object detector based on GRAB features. Detectors based on LBP or HOG cascades are ubiquitous, and GRAB features would offer a broader range of weak classifiers from which to fuse.

Bibliography

[1] Face recognition via sparse representation. URL http://perception.csl.illinois.edu/recognition/Robust_face.html.

[2] The INFace toolbox. URL http://luks.fe.uni-lj.si/sl/osebje/vitomir/face_tools/INFace/techs.html.

[3] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6): 681–685, 2001.

[4] Xiujuan Chai, Shiguang Shan, Xilin Chen, and Wen Gao. Locally linear regression for pose-invariant face recognition. *Image Processing, IEEE Transactions on*, 16 (7):1716–1725, 2007.

[5] Laurenz Wiskott, J-M Fellous, N Kuiger, and Christoph Von Der Malsburg. Face recognition by elastic bunch graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):775–779, 1997.

[6] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[7] Carlos Eduardo Thomaz and Gilson Antonio Giraldi. A new ranking method for principal components analysis and its application to face image analysis. *Image and Vision Computing*, 28(6):902–913, 2010.

[8] M Cox, J Nuevo-Chiquero, JM Saragih, and S Lucey. Csiro face analysis sdk. *Brisbane, Australia*, 2013.

[9] S. Milborrow, J. Morkel, and F. Nicolls. The MUCT Landmarked Face Database. *Pattern Recognition Association of South Africa*, 2010. http://www.milbo.org/muct.

[10] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[11] Archana Sapkota, Brian Parks, Walter Scheirer, and Terrance Boult. Face-grab: Face recognition with general region assigned to binary operator. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, page 82–89, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5544597. Cited by 0008.

[12] Archana Sapkota and Terrance E. Boult. GRAB: generalized region assigned to binary. *EURASIP Journal on Image and Video Processing*, 2013(1):35, 2013. URL http://jivp.eurasipjournals.com/content/2013/1/35/abstract. Cited by 0000.

[13] David G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, page 1150–1157, 1999. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=790410. Cited by 6297.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, page 404–417. Springer, 2006. URL http://link.springer.com/chapter/10.1007/11744023_32. Cited by 3228.

[15] Haitao Wang, Stan Z Li, and Yangsheng Wang. Face recognition under varying lighting conditions using self quotient image. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 819–824. IEEE, 2004.

[16] Masashi Nishiyama, Tatsuo Kozakaya, and Osamu Yamaguchi. Illumination normalization using quotient image-based techniques. *Recent Advances in Face Recognition, I-Tech, Vienna, Austria*, page 97–108, 2008. URL http://www.intechopen.com/source/pdfs/5895/InTech-Illumination_normalization_using_quotient_image_based_techniques.pdf.

[17] Qi Yin, Xiaoou Tang, and Jian Sun. An associate-predict model for face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, page 497–504, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995494.

[18] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[19] Tim Cootes, ER Baldock, and J Graham. An introduction to active shape models. *Image Processing and Analysis*, pages 223–248, 2000.

[20] Daniel González-Jiménez and José Luis Alba-Castro. Toward pose-invariant 2-d face recognition through point distribution models and facial symmetry. *Information Forensics and Security, IEEE Transactions on*, 2(3):413–429, 2007.

[21] Dong Yi, Zhen Lei, and Stan Z. Li. Towards pose robust face recognition. URL http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Yi_Towards_Pose_Robust_2013_CVPR_paper.pdf. Cited by 0000.

[22] FaceGen modeller: 3d face generator. URL http://www.facegen.com/modeller.htm.

[23] Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao, and Stan Z. Li. Face detection based on multi-block lbp representation. In *Advances in Biometrics*, page 11–18. Springer, 2007. URL http://link.springer.com/chapter/10.1007/978-3-540-74549-5_2. Cited by 0120.

[24] P. Sankaran, S. Gundimada, R. C. Tompkins, and V. K. Asari. Pose angle determination by face, eyes and nose localization. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, page 161–161, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1565479. Cited by 0013.

[25] Gary B Huang, Vidit Jain, and Erik Learned-Miller. Unsupervised joint alignment of complex images. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[26] Gary Huang, Marwan Mattar, Honglak Lee, and Erik G Learned-Miller. Learning to align from scratch. In *Advances in Neural Information Processing Systems*, pages 764–772, 2012.

[27] Lior Wolf, Tal Hassner, and Yaniv Taigman. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):1978–1990, 2011.

[28] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *Computer Vision and*

Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2578–2585. IEEE, 2012.

[29] Terry Riopka and Terrance Boult. The eyes have it. In *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pages 9–16. ACM, 2003.

[30] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. 2004.

[31] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[32] Terence Sim, Simon Baker, and Maan Bsat. The CMU pose, illumination, and expression (PIE) database. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, page 46–51, 2002. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1004130. Cited by 0775.

[33] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010.

Appendix A

Appendix: AAM Triangle Width and Height Distributions and Tertiles across the LFW Dataset

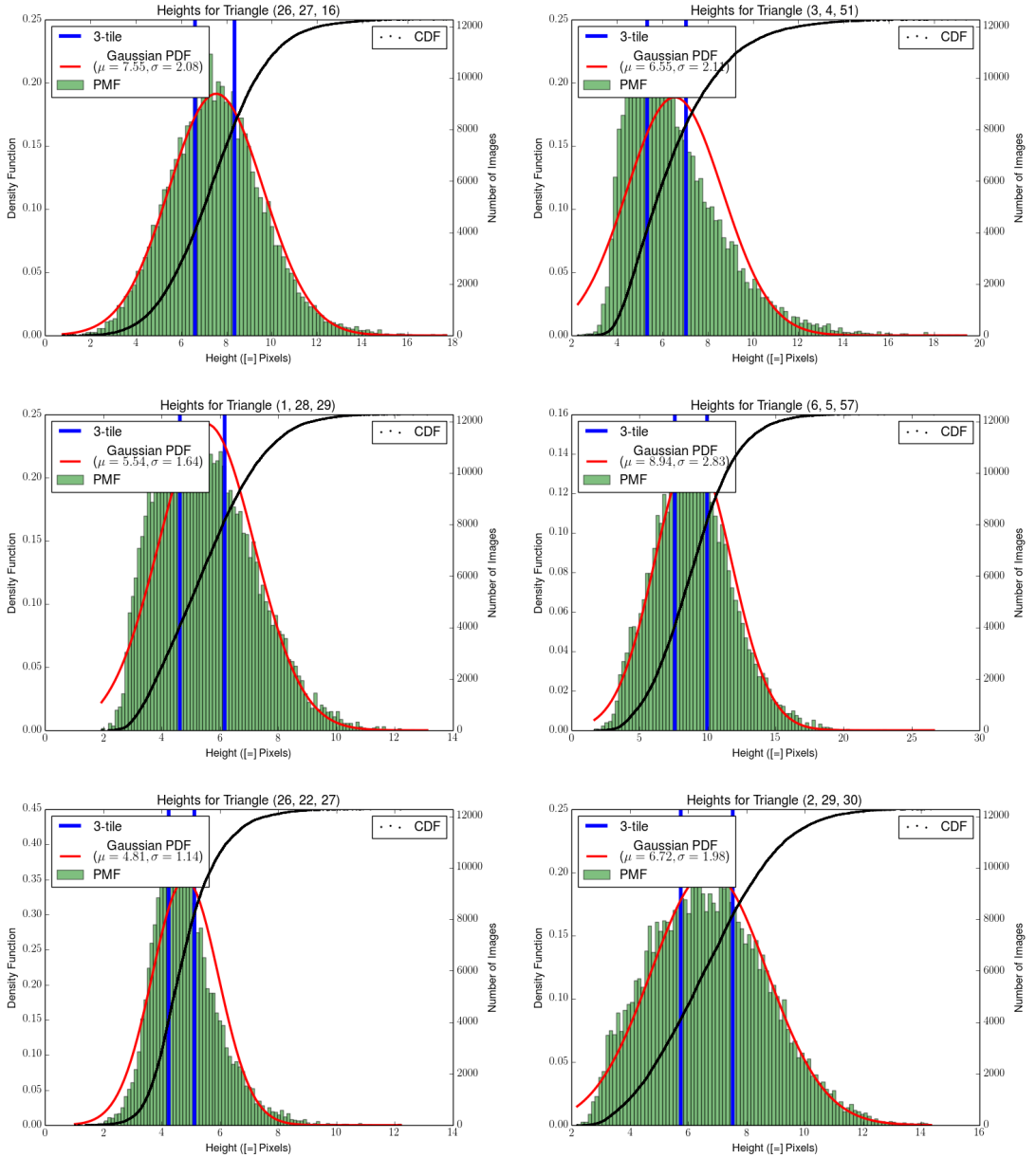

FIGURE A.1: Distributions for horizontal triangles defined by points in figure 3.11.

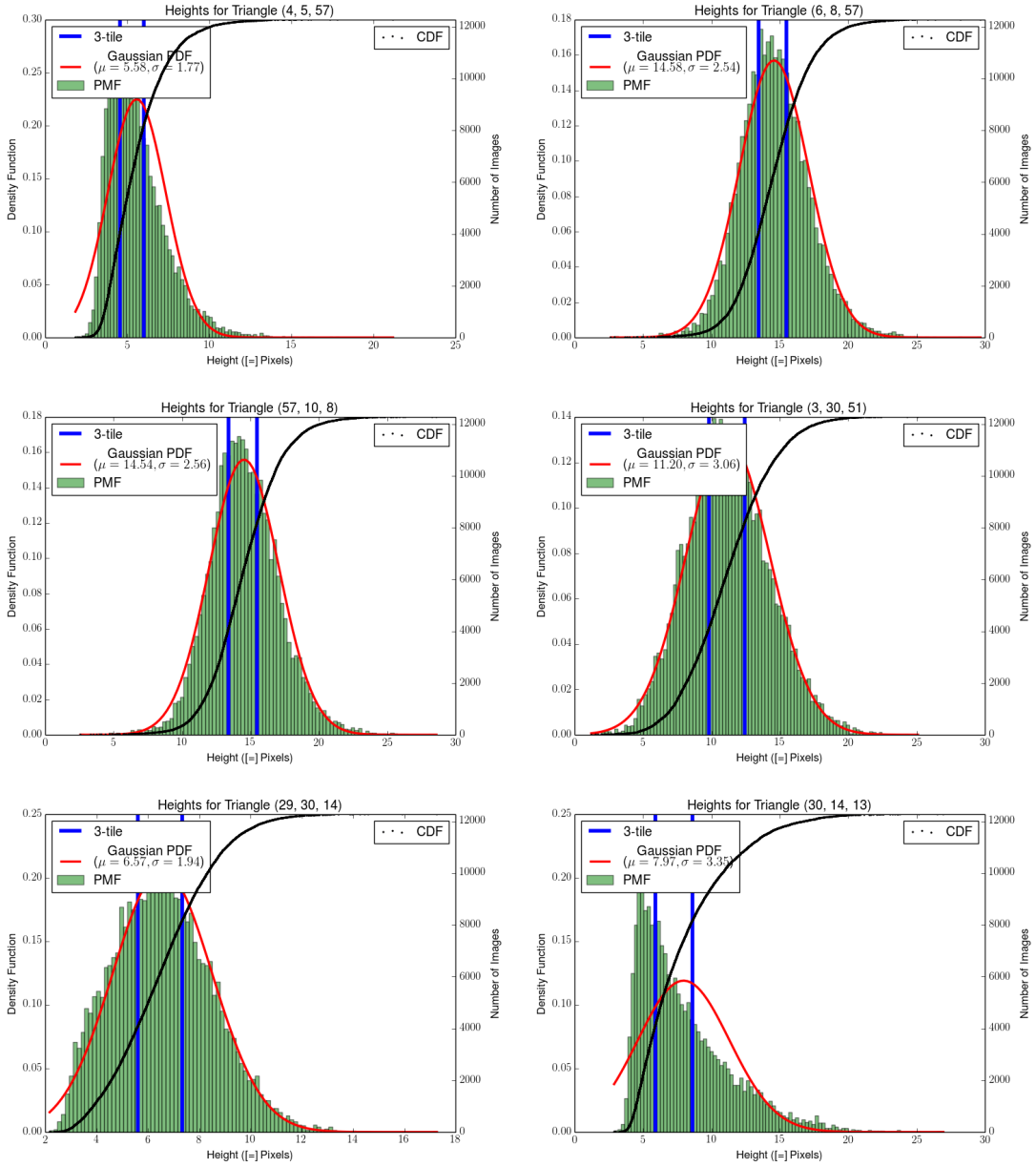

FIGURE A.2: Distributions for horizontal triangles defined by points in figure 3.11.

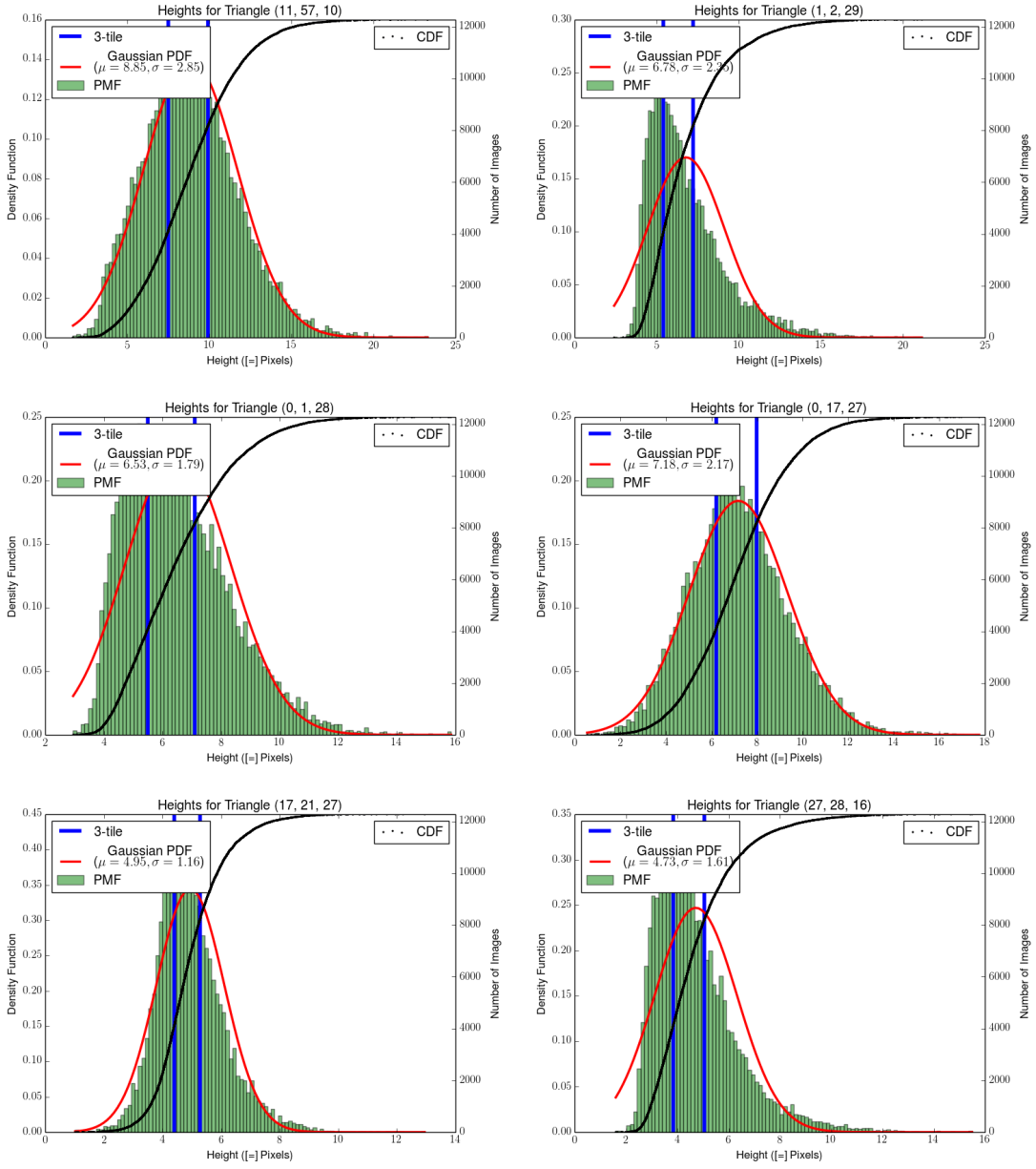

FIGURE A.3: Distributions for horizontal triangles defined by points in figure 3.11.

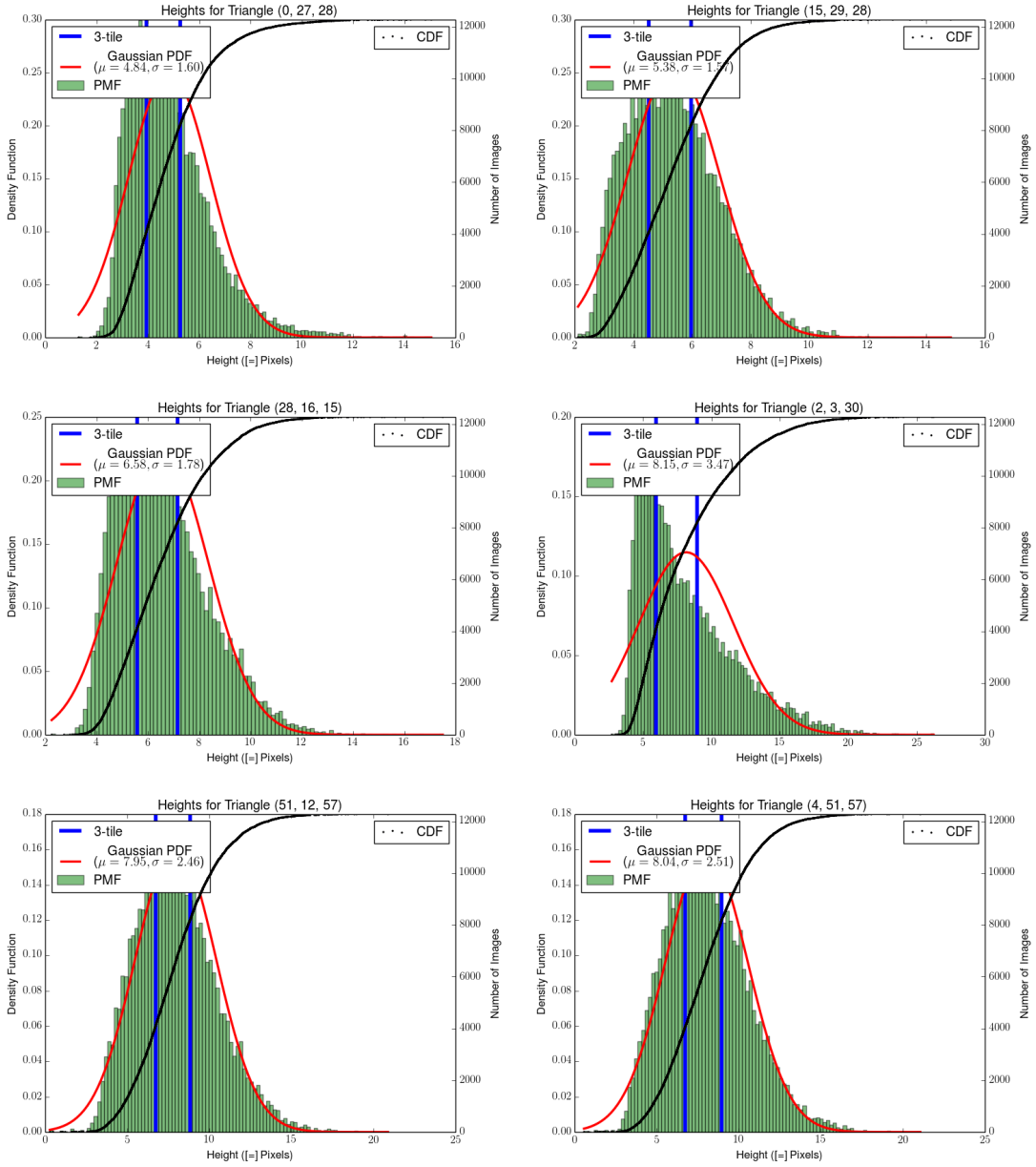

FIGURE A.4: Distributions for horizontal triangles defined by points in figure 3.11.

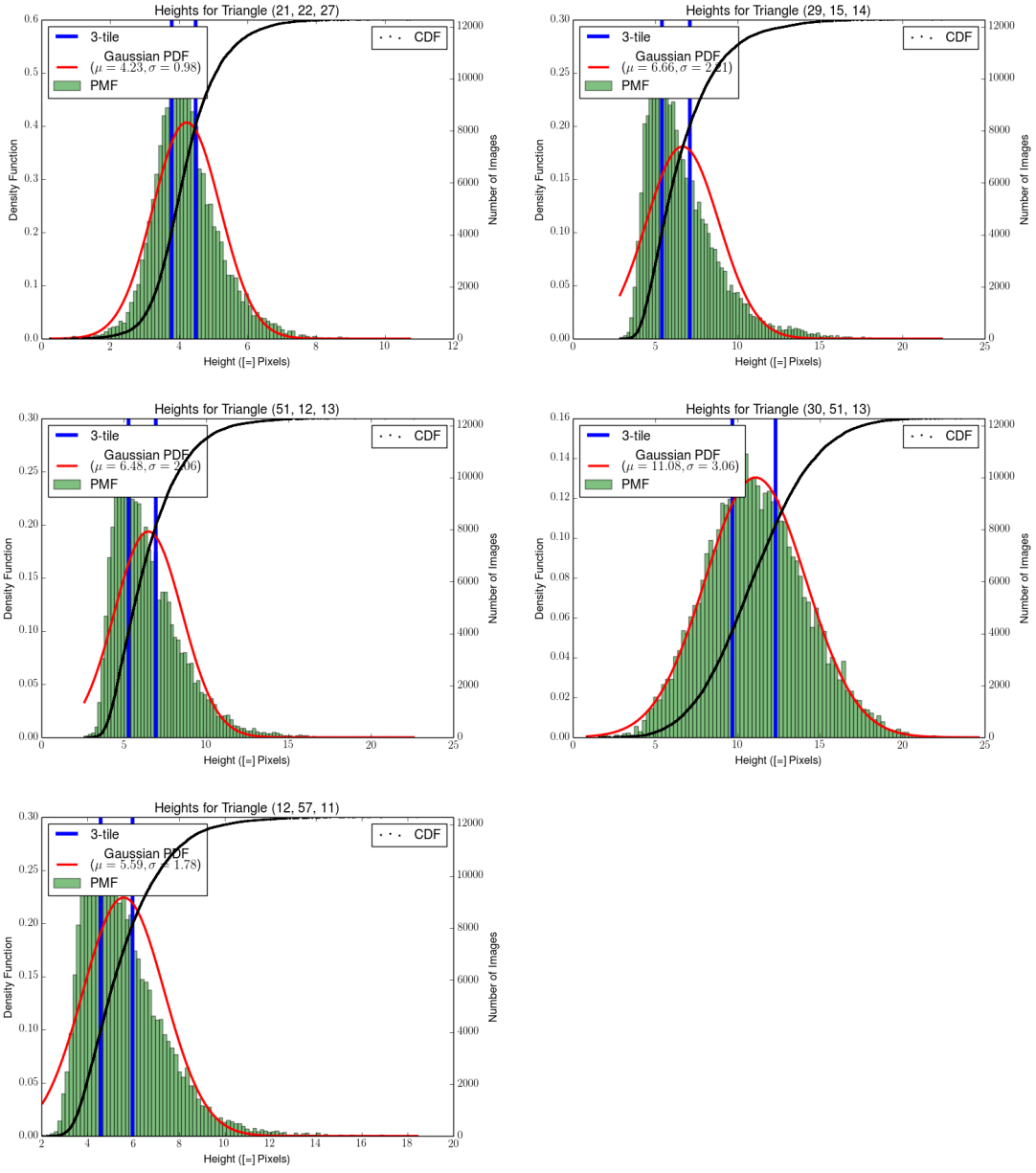

FIGURE A.5: Distributions for horizontal triangles defined by points in figure 3.11.

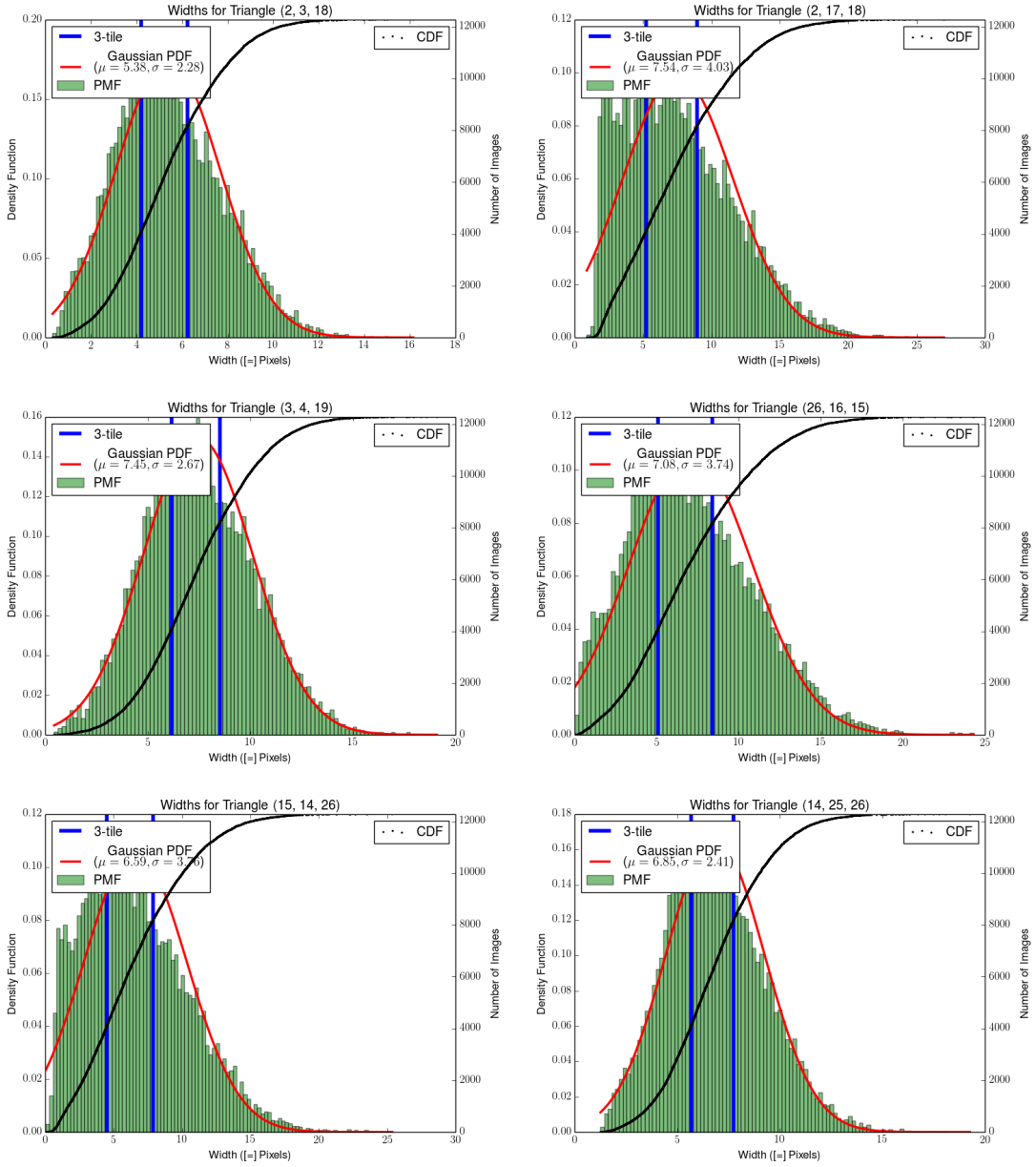

FIGURE A.6: Distributions for vertical triangles defined by points in figure 3.11.

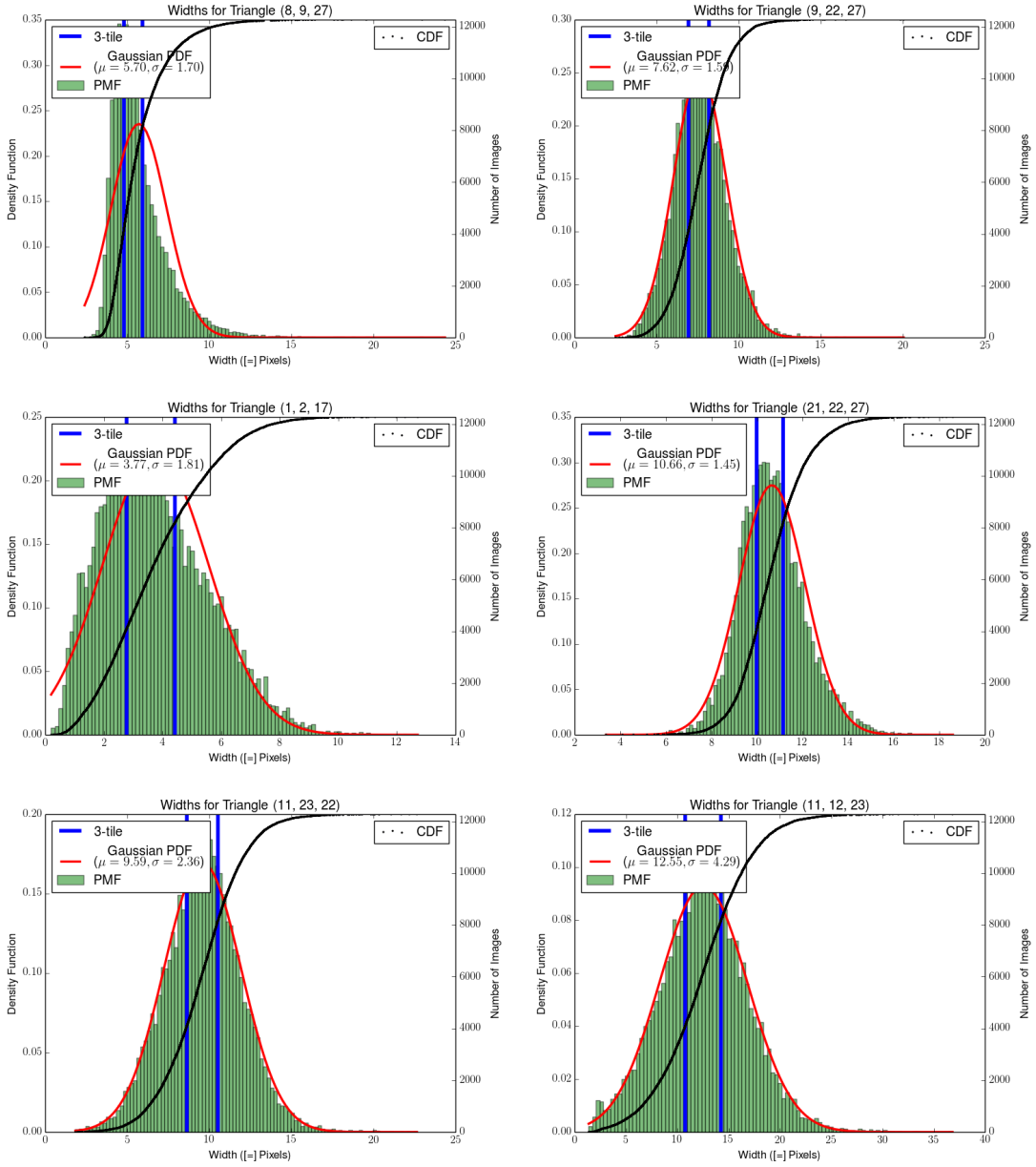

FIGURE A.7: Distributions for vertical triangles defined by points in figure 3.11.

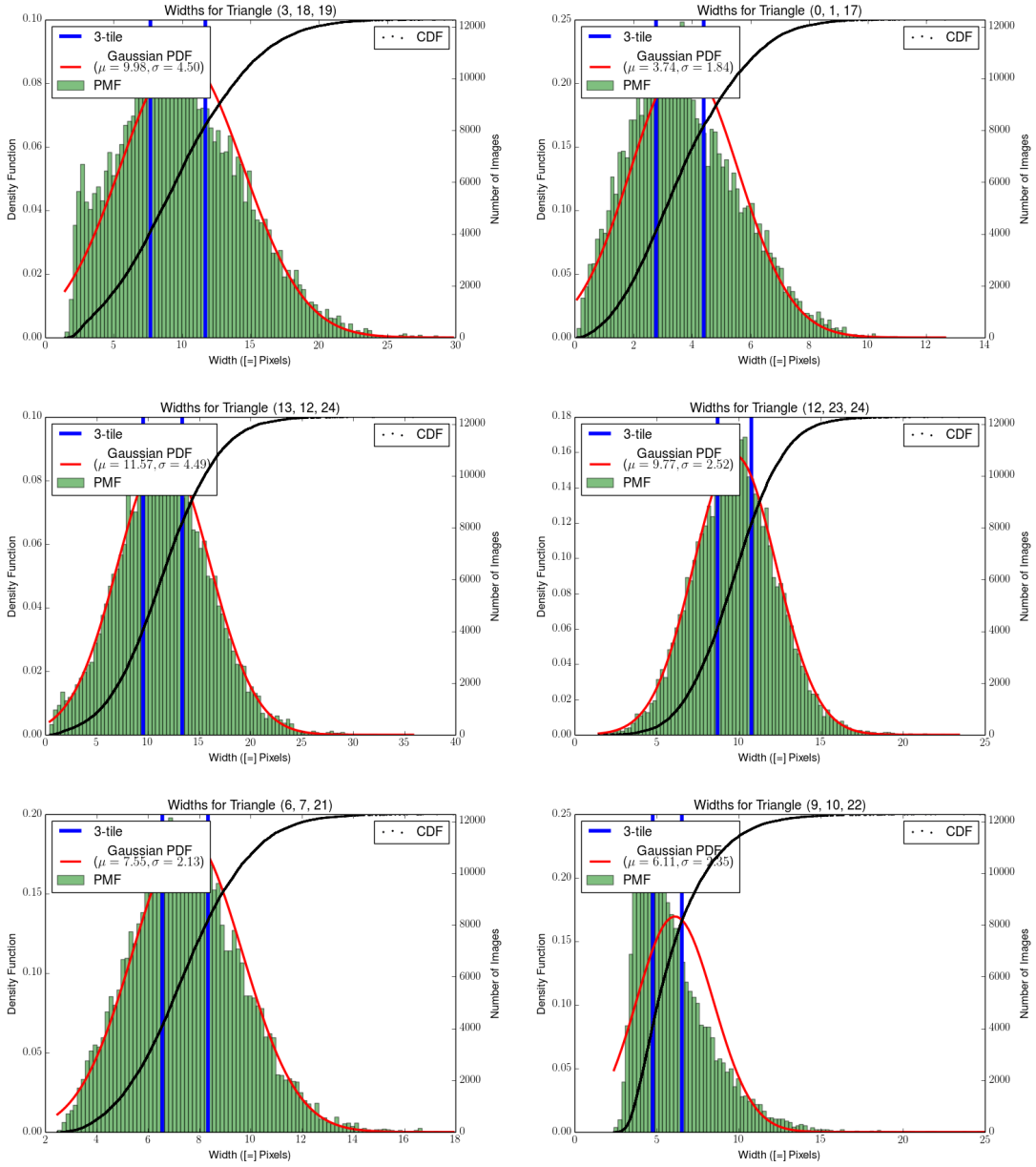

FIGURE A.8: Distributions for vertical triangles defined by points in figure 3.11.

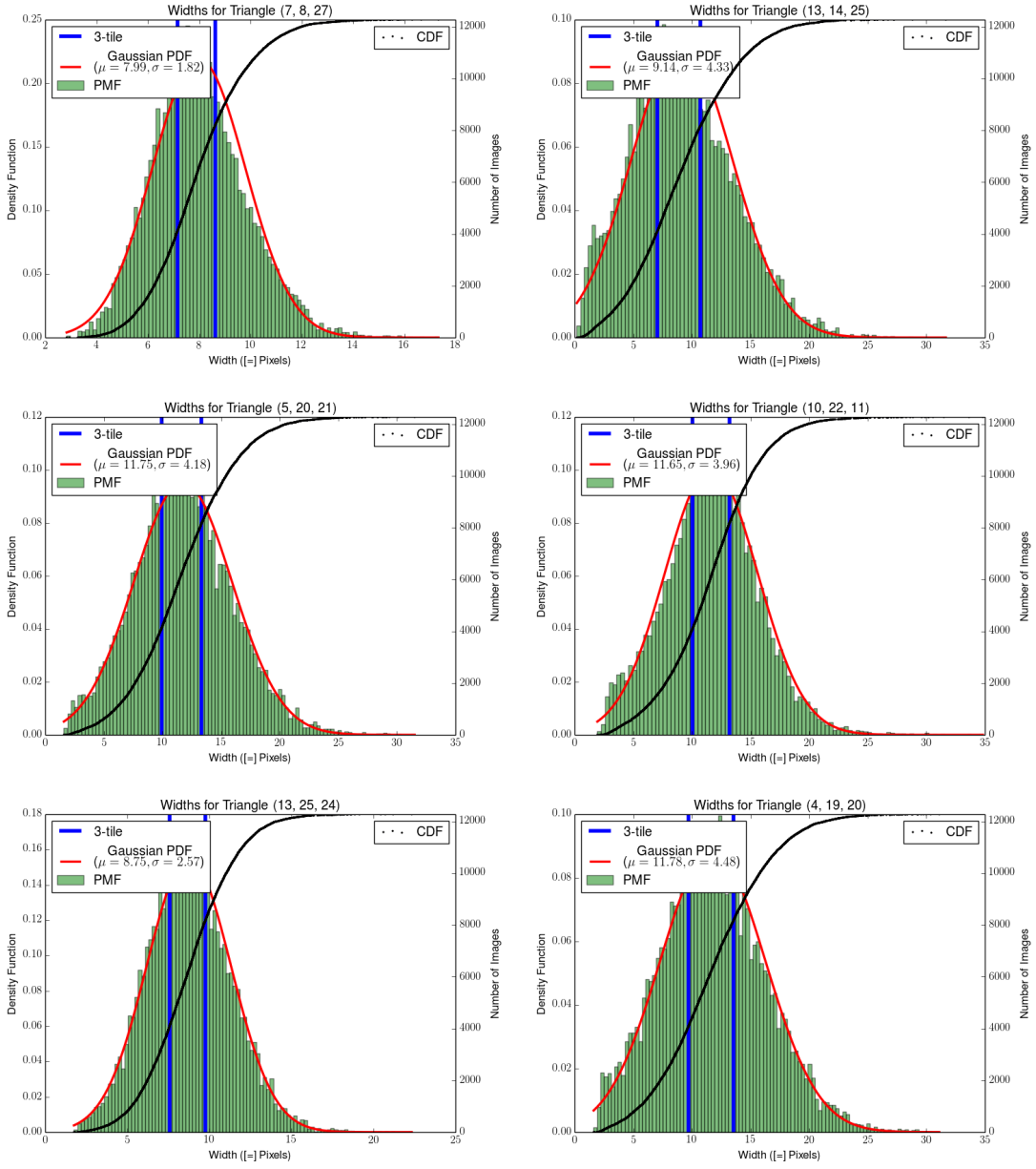

FIGURE A.9: Distributions for vertical triangles defined by points in figure 3.11.

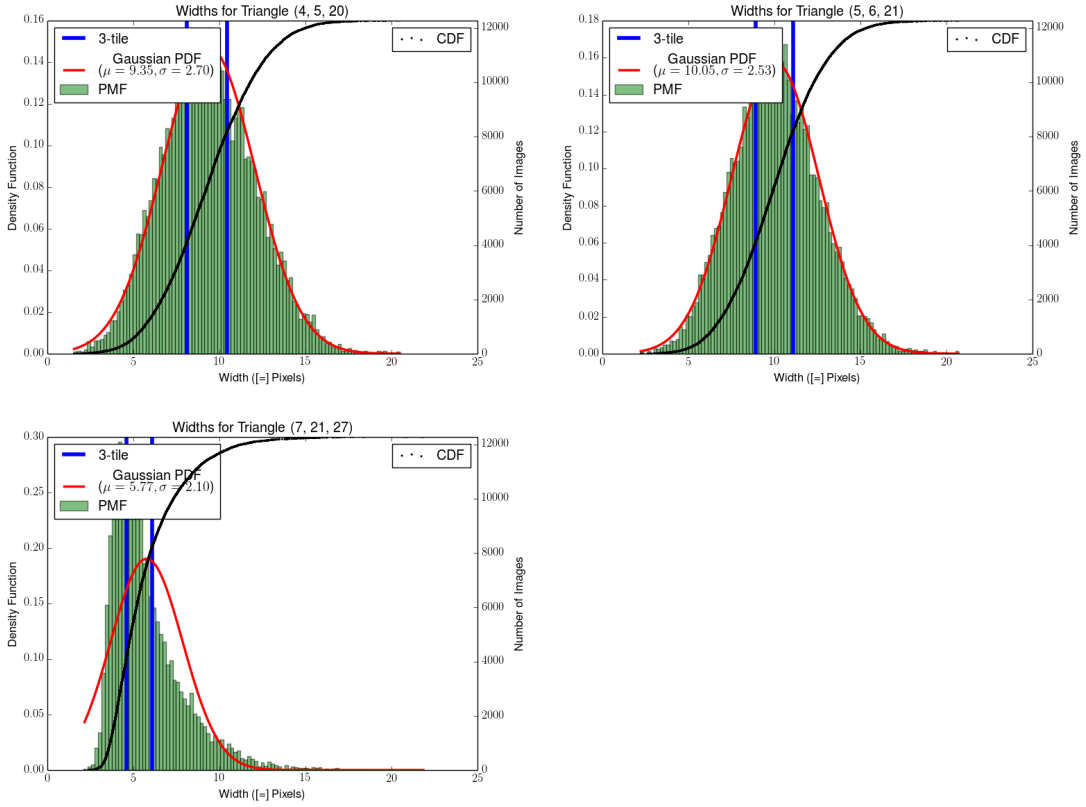

FIGURE A.10: Distributions for vertical triangles defined by points in figure 3.11.

Appendix B

Appendix: Triangle Width and Height Distributions and Quartiles

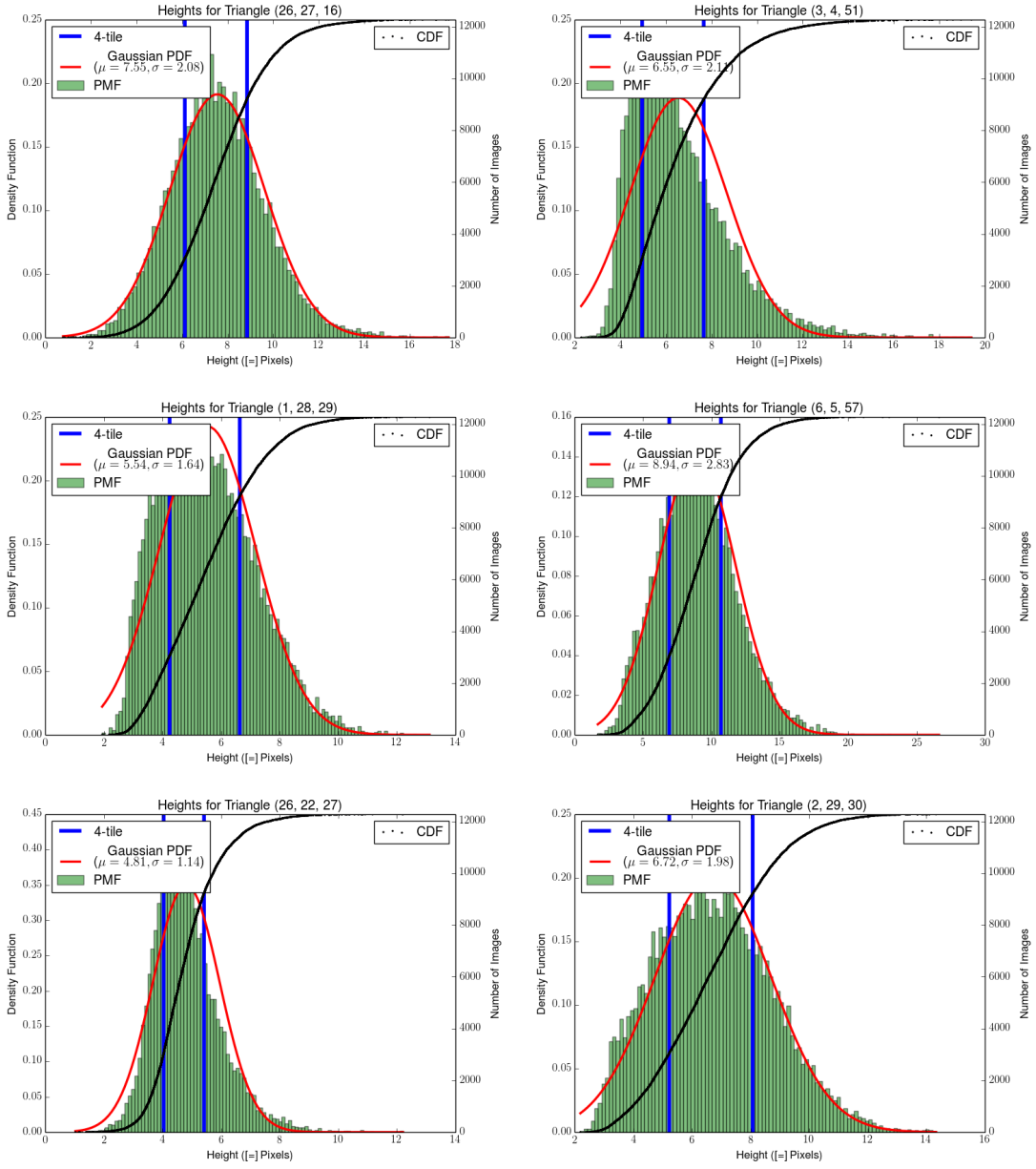

FIGURE B.1: Distributions for horizontal triangles defined by points in figure 3.11.

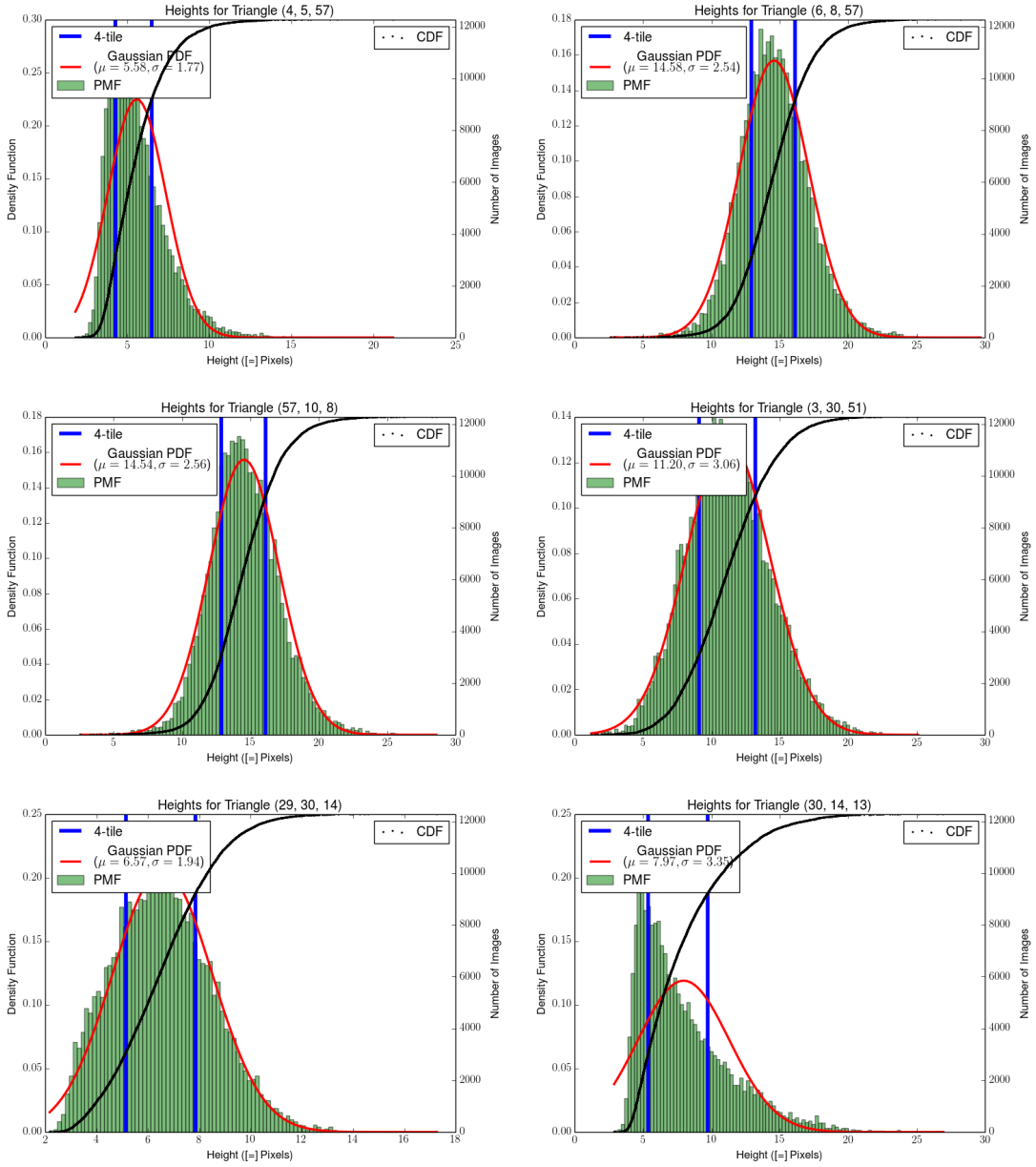

FIGURE B.2: Distributions for horizontal triangles defined by points in figure 3.11.

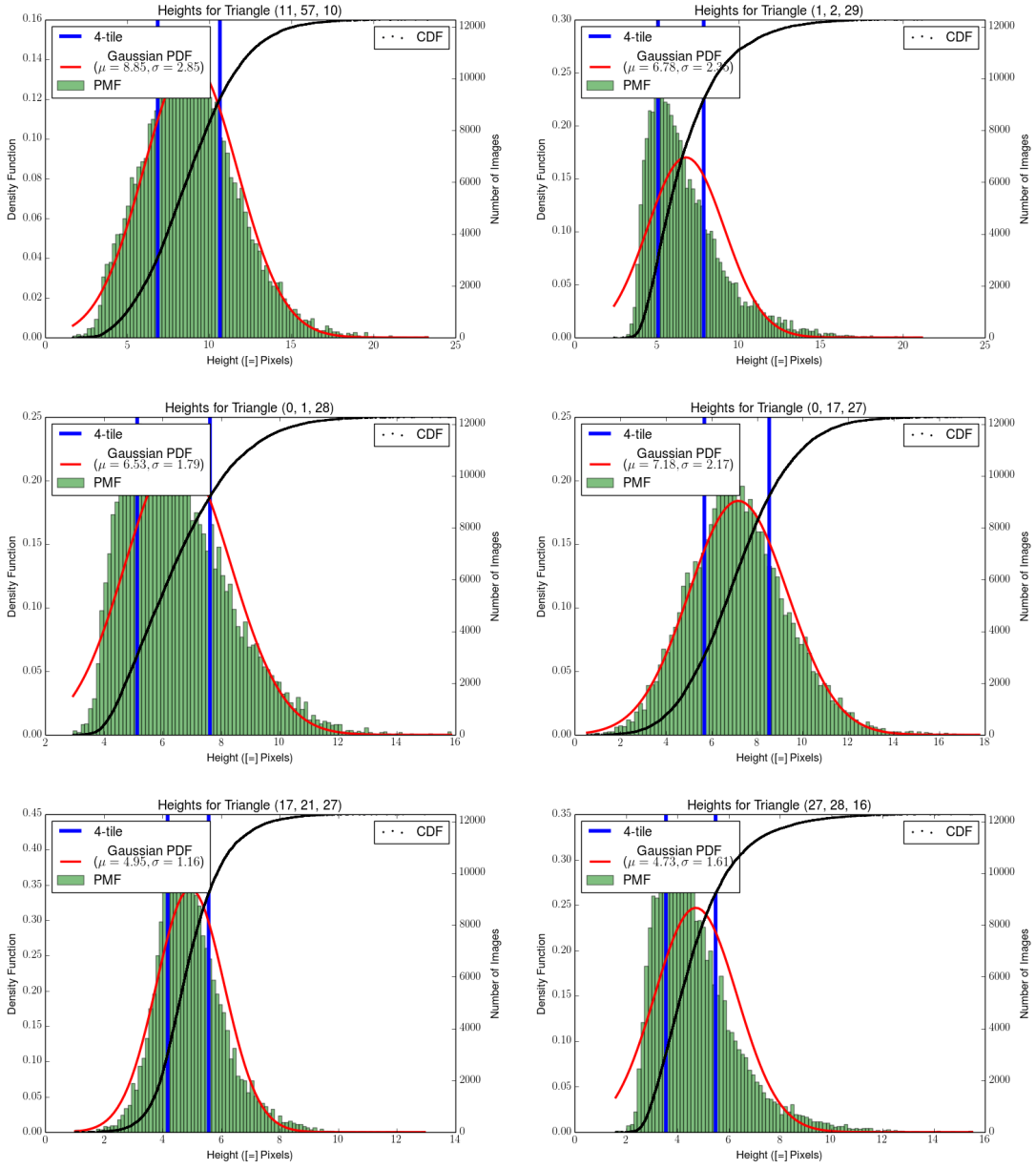

FIGURE B.3: Distributions for horizontal triangles defined by points in figure 3.11.

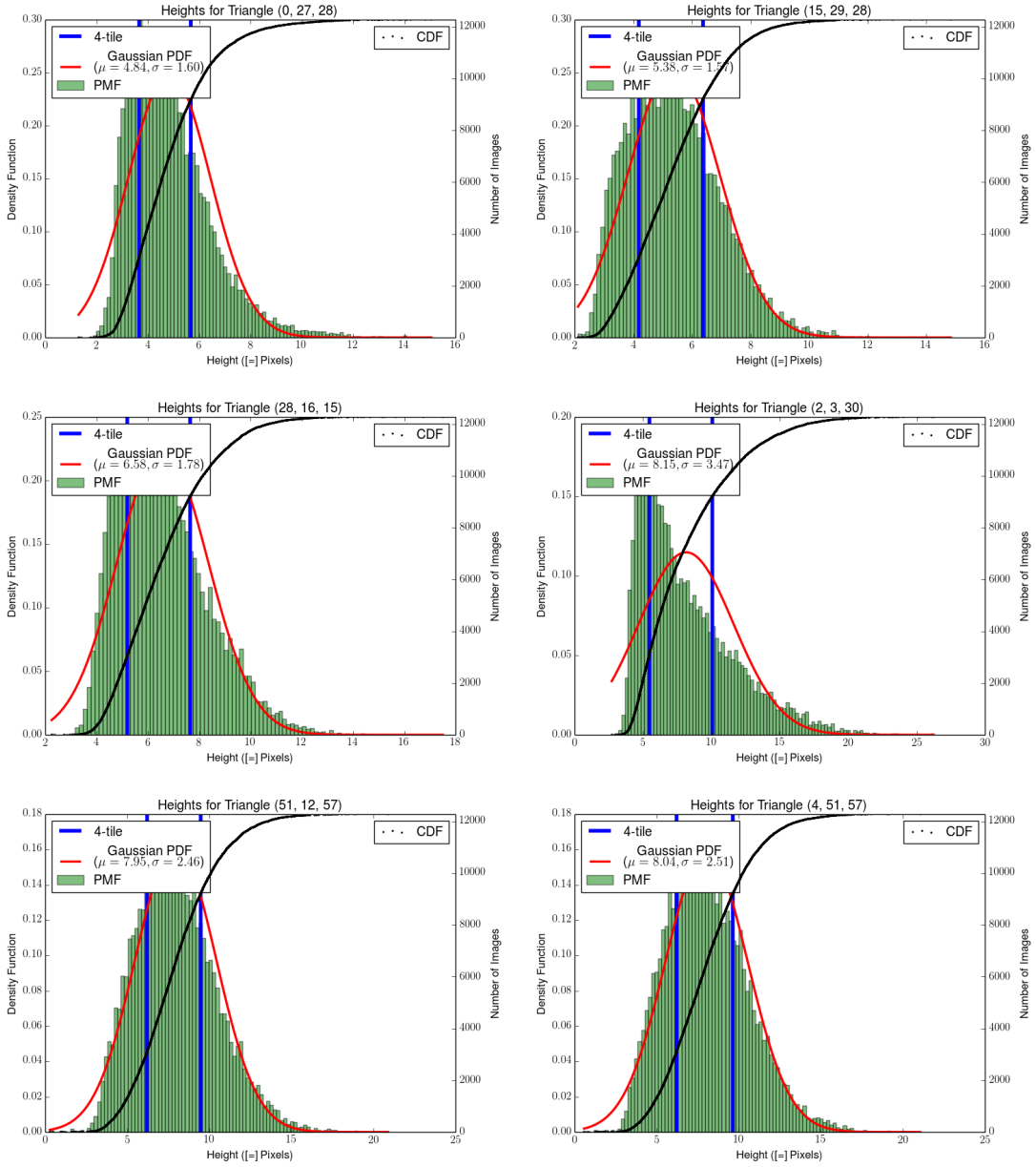

FIGURE B.4: Distributions for horizontal triangles defined by points in figure 3.11.

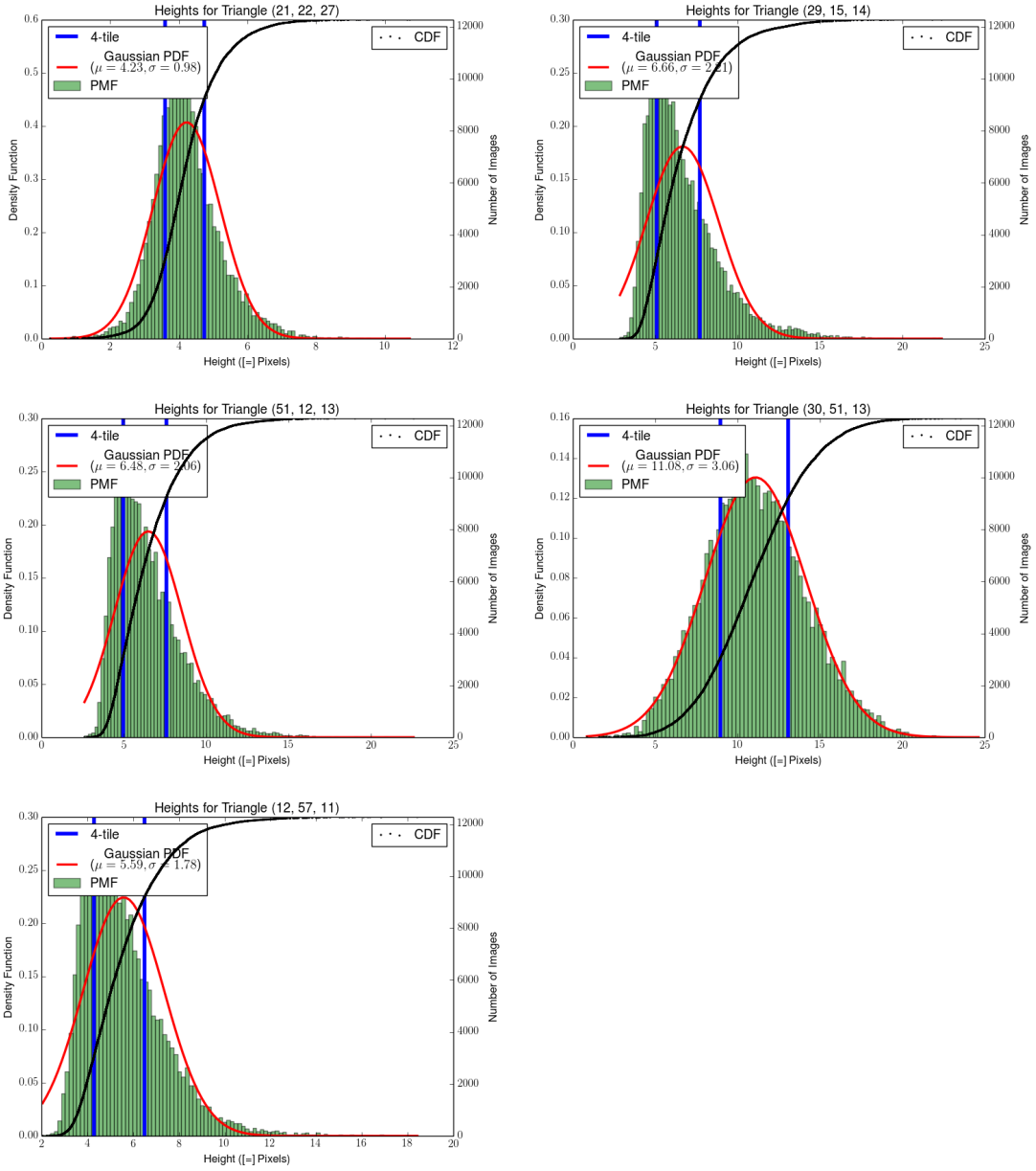

FIGURE B.5: Distributions for horizontal triangles defined by points in figure 3.11.

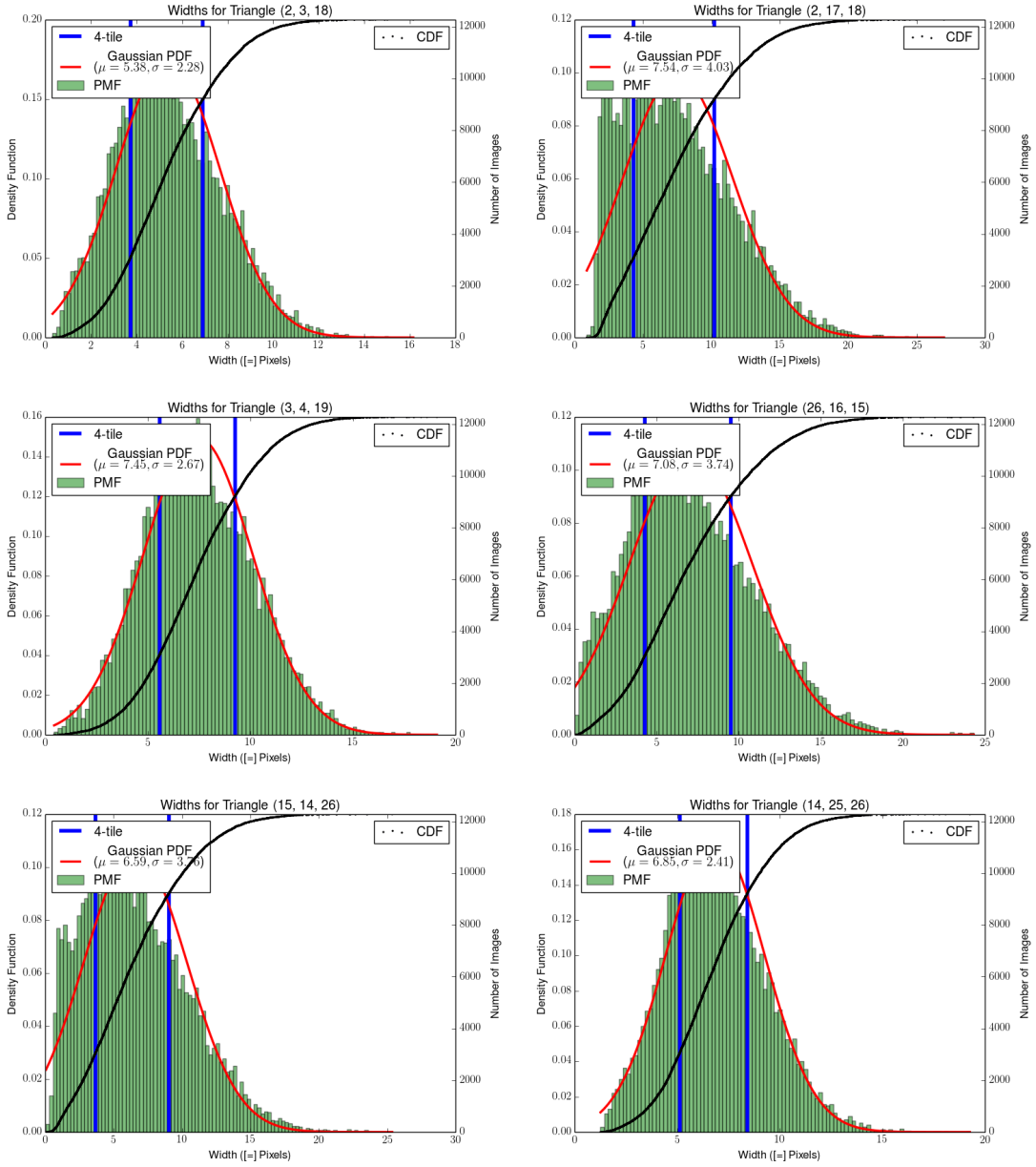

FIGURE B.6: Distributions for vertical triangles defined by points in figure 3.11.

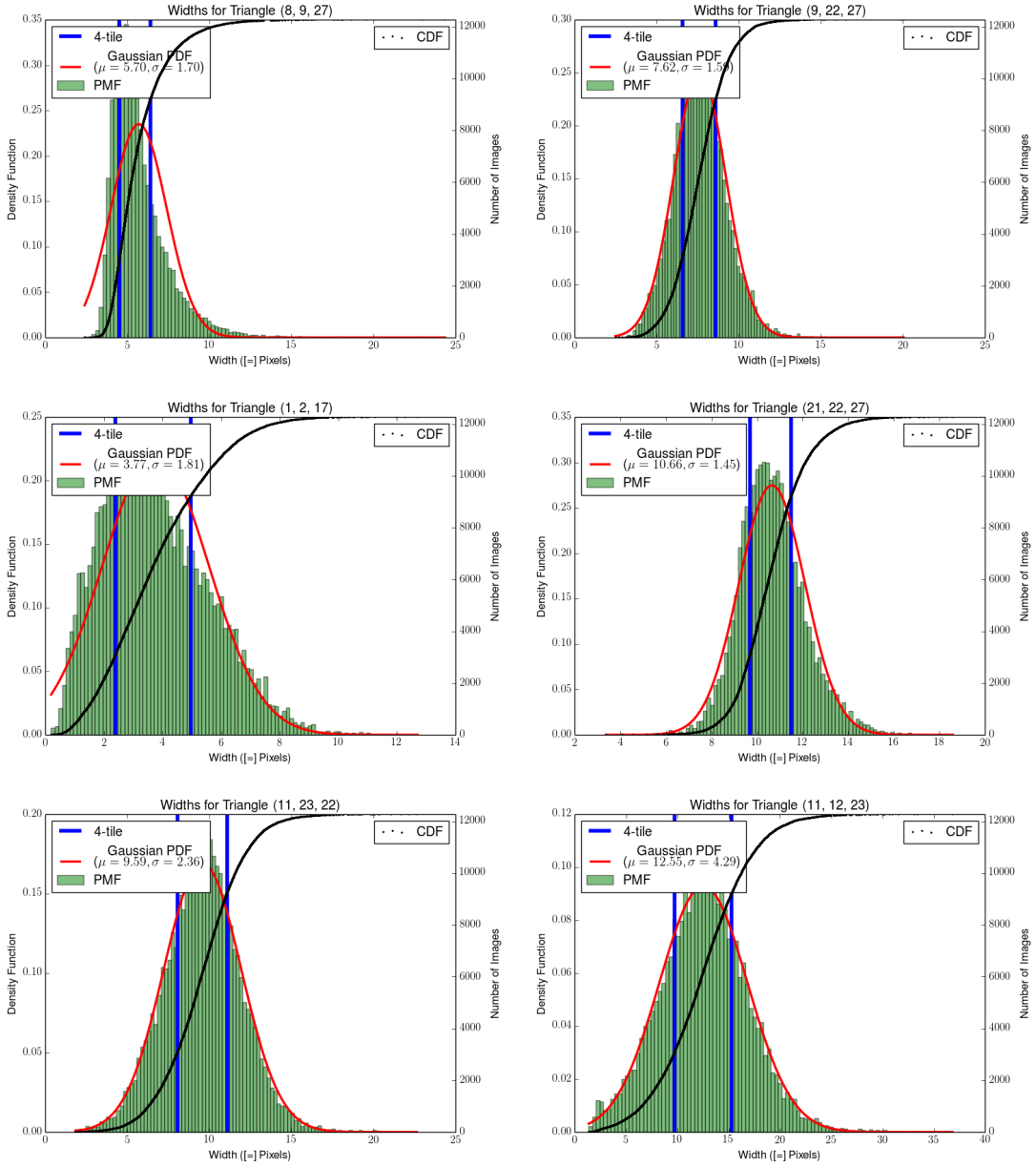

FIGURE B.7: Distributions for vertical triangles defined by points in figure 3.11.

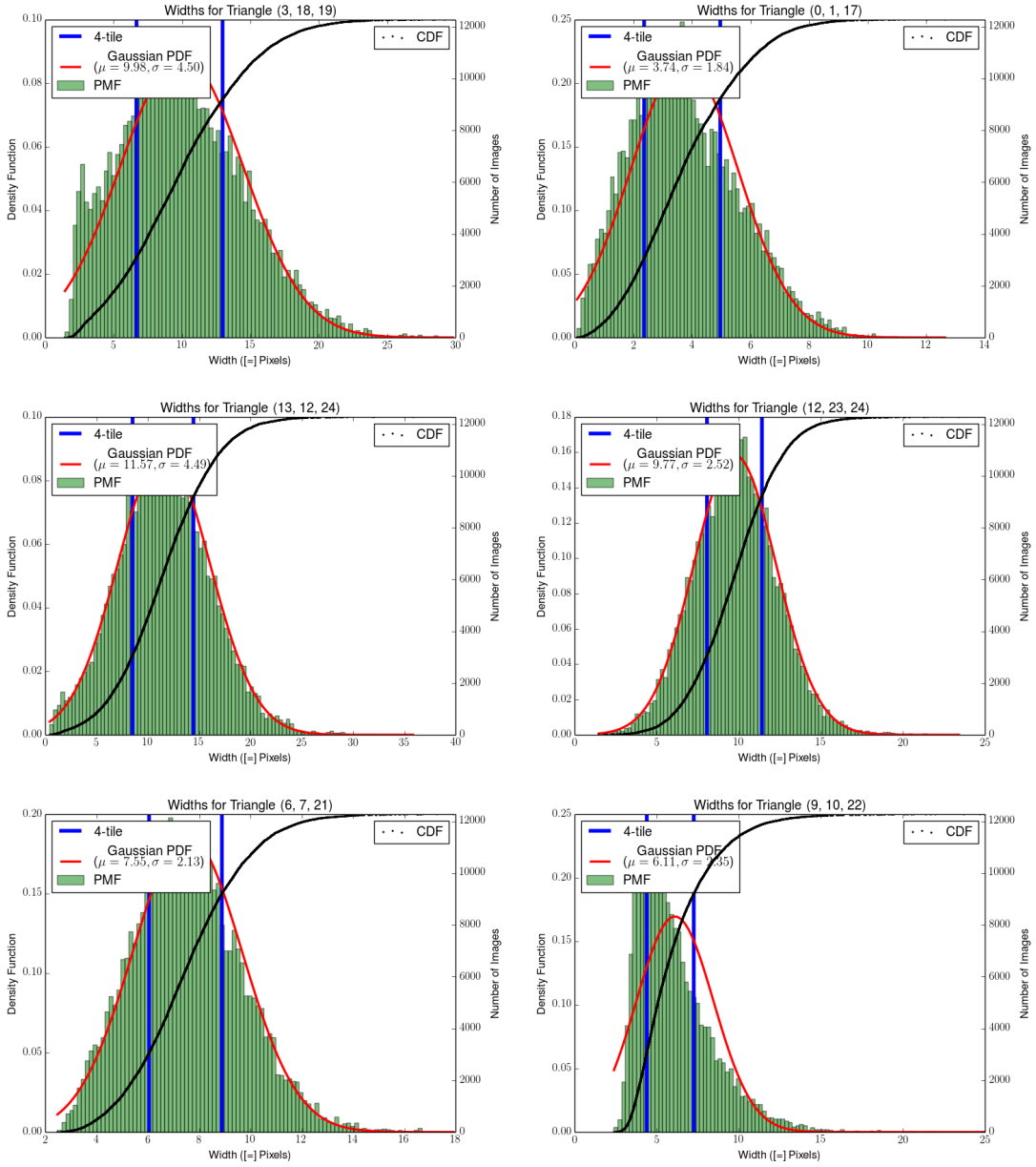

FIGURE B.8: Distributions for vertical triangles defined by points in figure 3.11.

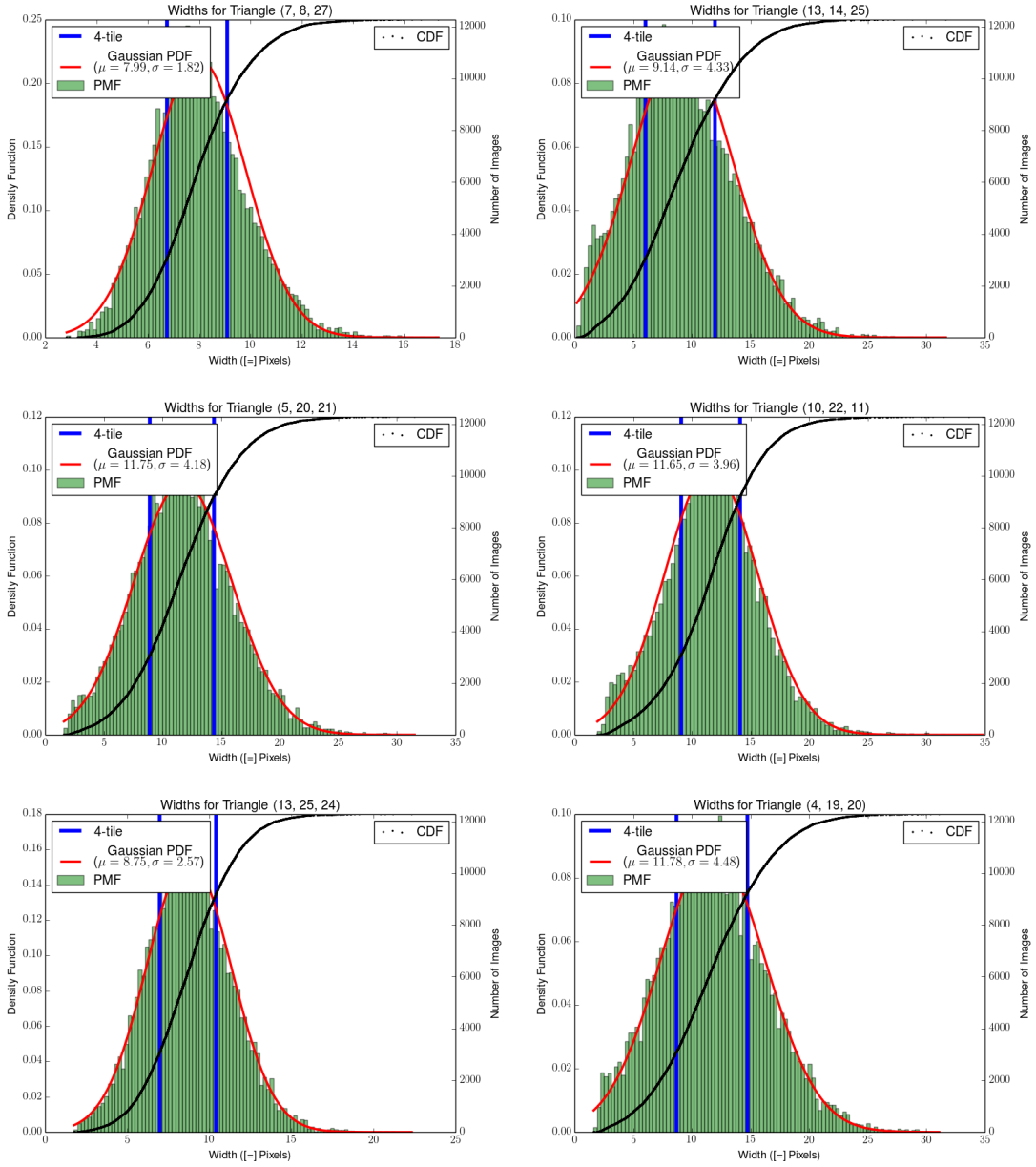

FIGURE B.9: Distributions for vertical triangles defined by points in figure 3.11.

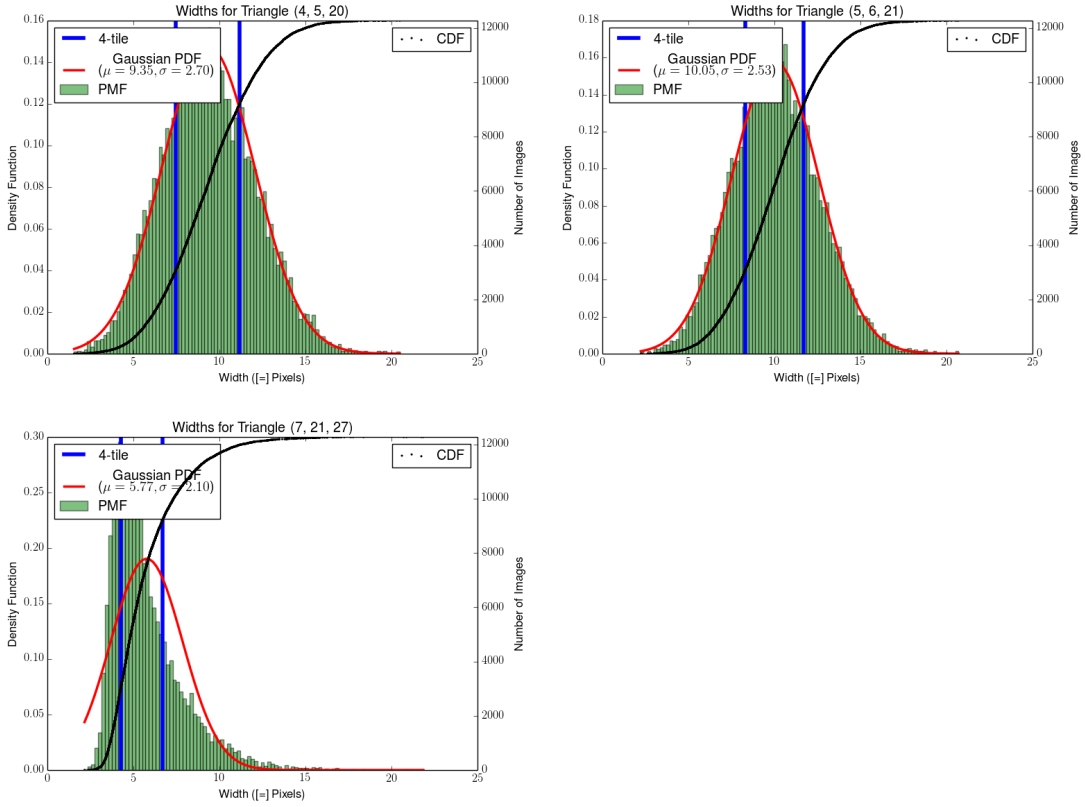

FIGURE B.10: Distributions for vertical triangles defined by points in figure 3.11.