DISSERTATION

FACE DETECTION USING CORRELATION FILTERS

Submitted by

Mohammad Nayeem Teli

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2013

Doctoral Committee:

Advisor: J. Ross Beveridge

Bruce A. Draper
Adele Howe
Geof H. Givens

ABSTRACT

FACE DETECTION USING CORRELATION FILTERS

Cameras are ubiquitous and available all around us. As a result, images and videos are posted online in huge numbers. These images often need to be stored and analyzed. This requires the use of various computer vision applications that includes detection of human faces in these images and videos. The emphasis on face detection is evident from the applications found in everyday point and shoot cameras for a better focus, on social networking sites for tagging friends and family and for security situations which subsequently require face recognition or verification.

This thesis focuses on detecting human faces in still images and video frames using correlation filters. These correlation filters are trained using a recent technique called Minimum Output Sum of Squared Error (MOSSE) developed by Bolme et al. [BBDL10]. Since correlation filters identify only a peak location, it only helps in localizing a single target point. In this thesis, I develop techniques to use this localization for detection of human faces of different scales and poses in uncontrolled background, location and lighting conditions.

The goal of this research is to extend correlation filters for face detection and identify the scenarios where its potential is the most. The specific contributions of this work are the development of a novel face detector using correlation filters and the identification of the strengths and weaknesses of this approach.

This approach is applied to an easy dataset and a hard dataset to emphasize the efficacy of correlations filters for face detection. This technique shows 95.6% accuracy in finding the exact location of the faces in images with controlled background and lighting. Although, the results on a hard dataset were not better than the OpenCV Viola and Jones face detector, it showed much better results, 81.5% detection rate compared to 69.43% detection rate by the Viola and Jones face detector, when tested on a customized dataset that was controlled for location change between training and test datasets. This result signifies the strength of a correlation based face detector in a specific scenario with uniform setting, such as a building entrance or an airport security gate.

ACKNOWLEDGEMENTS

Thank you ALLAH, for all the blessings. I would like to follow it up by thanking my advisor, Professor J. Ross Beveridge for giving me the opportunity to work on exciting projects, guiding me through Ph.D. and encouraging me whenever it seemed to be getting difficult. I would also like to thank my co-advisor Professor Bruce A. Draper for his valuable insights in our meetings and pulling me back on the right track whenever I went off-track during this research.

Thanks to Professor Adele Howe for valuable comments during my proposal defense which were very insightful. I would also like to thank my other committee member, Professor Geof H. Givens for teaching me statistical analysis techniques and being there on my committee.

I extend my gratitude to my friends David Bolme, Steve O'Hara and Yui Man Lui. I have learned a lot from all of you during all those discussions which we had over our Pizza lunches and in the cubicles. In particular, I would acknowledge Dave for teaching me research techniques and helping me in my initial days.

I am indebted to my parents for always having faith in me and training me to face the challenges of life with patience. For always being there to support me. Mom and dad this would not have been possible without you. I thank my siblings for always encouraging me and believing in me when I would not. Last but not the least I would like to thank my wife for being patient with me because the path to Ph.D. was long.

DEDICATION

This thesis is dedicated to my parents, Mr. Mohammad Yehya and Razia Yehya and to my two lovely daughters, Jennah and Nihaya. You are my world and I love you a lot.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

One of the tools to enable human-computer interaction (HCI) is cameras. The images captured through cameras (still or video frames) can be analyzed through various computer vision techniques. Many biometric and HCI applications involve performing some analysis on human faces such as in face alignment, 3D modeling, recognition, verification and authentication. Before any such analysis can occur, faces must be detected in these images [ZZ10b].

Yang et al. [YKA02] defined face detection as: "given an image, the face detection algorithm should determine whether or not there are any faces in that image and if there are, locate all the faces in that image". Yang et al. surveyed the early face detection work and divided it into four main categories: knowledge-based methods, feature invariant approaches, template matching methods, and appearance based methods. A lot of work has been done in this direction resulting in the development of many face detectors. One of the seminal papers in face detection is by Viola and Jones [VJ01]. This face detector is very commonly used. However, it has some limitations such as a high false positive rate and difficulty in training. I will discuss this technique and many others in detail in our chapter on face detection. In this research I propose to develop a novel face detector based on a recent technique for training correlation filters, Minimum Output Sum of Squared Error (MOSSE) filter by Bolme et al. [BBDL10].

The contribution of this work is three-fold:

- To the best of my knowledge it is the first face detector based on correlation filters. I have explored the efficacy of using a correlation based filter to detect faces in still images and video frames.

- I present the scenarios where such a face detector would be successful and where it may not be an approach of choice.

- This work presents the first study to detect faces in a newly released dataset, The Point-and-Shoot Face Recognition Challenge (PaSC) [BPB+13], since its release to the public .

1.1 Face Detection is not a Solved Problem

Face detection is an important challenge because of its many applications such as face tracking, particularly if it is efficient and fast. It may be used to initialize tracking, for e.g., when a face enters frame or appears from an occluded position. It is also a challenging case of the more general problem of object detection, which has many applications [Kin03]. Due to its importance face detection has been one of the most popular areas of computer vision research and the community has been very successful in detecting faces in images taken under controlled conditions.

Consider an image like the one shown in Figure 1.1 where it is very easy to detect the face.

Figure 1.1: Easy to detect face.

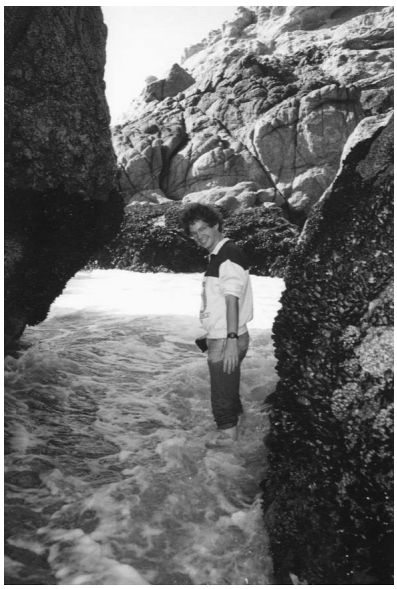


Figure 1.2: Difficult to detect face.

Most of the state of the art algorithm like Viola and Jones cascade detector [VJ01] are easily able to locate the face. This face is frontal, centered and the lighting is uniform. In images like these detecting faces is a trivial task. However, I can't say the same about images, such as in Figure 1.2, taken under unconstrained settings. Unconstrained settings include uncontrolled lighting, background, pose, age, race, gender, clothing, etc. Some examples of the images from the Labeled Faces in the Wild (LFW) [HRBLM07] dataset are taken under unconstrained settings are presented in Figure 1.3. According to Jain and Learned-Miller [JLM10], face detection in completely unconstrained settings remains a very challenging task due to the significant pose and lighting vari-

Figure 1.3: Examples of Labeled Faces in the Wild Images

ations. In general some of the reasons that make it difficult to detect faces in images taken under uncontrolled settings can be ascribed to the following reasons: 1) different pose such as frontal, 45 degrees, profile and upside down, 2) presence or absence of beards, glasses, moustache and occlusions because the appearance of a person is directly influenced by the facial expressions which can be varied, and finally, 3) lighting conditions and camera characteristics.

Recently Zhang and Zhang [ZZ10a] presented the results of the state-of-the-art face detectors and reported a detection rate of 50-70% with about 0.5-3% of the detected faces being false positive. From these results they believe that face detection requires a lot more work. Degtyarev and Seredin [DS10] presented an objective comparison of seven face detection algorithms. These algorithms were tested on 9 different face databases. Degtyarev and Seredin show that 64% of the images, referred to as the easy images, were correctly processed by all the algorithms. There was still a 5-6% False Rejection Rate (FRR) although only 0.14% of the images are referred to as challenging images by the authors. With such a small number of challenging examples we should expect a higher true detection and a much lower false rejection rate. This indicates there is a potential to improve face detection algorithms.

While there is clearly a need to improve face detection for purely scientific reasons which fuels the interest of researchers, there is also a commercial reason to improve face detection. The commercial importance is boosted by the interest of the world's leading camera manufacturers through their interest in including it as one of the features of their products.

The computer vision community in general acknowledges the role and importance of face de-

tection which is displayed in the need for organizing a workshop on face detection[1] in conjunction with the European Conference on Computer Vision (ECCV), 2010. The two main objectives of this workshop were to establish the current state-of-the-art in face detection and identify new frontiers of research in that direction. This workshop emphasized that face detection in unconstrained settings is still a challenging problem and needs to be addressed. In line with this interest, face detection on hard datasets is also a focus of the competition involving Face Detection on Hard Datasets[2] organized in conjunction with the International Joint Conference on Biometrics (IJCB) 2011. The motivation for this competition was that although face detection algorithms do well on many applications, they often fail or return false positives when presented with a challenging image taken from a distance or in low light. I took part in this competition using a correlation filter based face detector. However, before I get into the details of my face detector and the motivation of using this approach I will begin by presenting the theory behind correlation.

1.2 Correlation

Correlation is a single number that describes the degree of relationship between two variables. In signal processing, cross correlation is a measure of similarity of two signals as a function of a time-lag applied to one of them. This is also known as a sliding dot product or inner-product. It can be represented as:

$$(g * h)(t) = \int_{-\infty}^{\infty} g^*(\tau)h(t+\tau)d\tau, \tag{1.1}$$

where g and h are continuous functions and g^* is the complex conjugate of g. It seems pertinent here to present a relation between cross-correlation and convolution since the two are very similar concepts. However, the difference is that while convolution requires signal reversal followed by shifting and multiplication with another signal, correlation only requires shifting a signal and multiplying without reversing it. As such, cross correlation of functions $g(t)$ and $h(t)$ is equivalent to

[1] `http://vis-www.cs.umass.edu/fdWorkshop`

[2] `http://vast.uccs.edu/FDHD`

the convolution of $g(-t)$ and $h(t)$, i.e.,

$$fcorr = g(t) * h(t), \qquad fconv = g(-t) * h(t) \tag{1.2}$$

where $fconv$ of Equation 1.2 is the expression of convolution equivalent of the correlation operation, $fcorr$. At this stage it is also important to point out that when symmetric masks / images are used, reversal does not matter and therefore image processing descriptions often ignore the distinction between correlation and convolution and use convolution for operations such as smoothing. Both convolution and correlation have two key features: shift invariance and linearity. Shift invariance means that the same operation is performed at every point in the image and linearity means that every pixel is replaced with a linear combination of its neighbors. Mathematically, linearity is represented as: $g * kh = k(g * h)$ and $g * (h1 + h2) = g * h1 + g * h2$.

In order to demonstrate correlation through a simple example, consider a row of pixel values. This row of pixels can be regarded as a 1D image. One of the simplest convolution operations is local averaging. In this operation every pixel in a 1D image is replaced with the average of that pixel and its two neighbors. For example, consider an image I with the following pixel values:

2	3	6	5	5	1	8	9	7

Value in cell 2, $I(2) = 3$, is replaced with the average of values in cells 1, 2 and 3 such that the value 3 changes to $I(2) = \frac{(2+3+6)}{3} = \frac{11}{3}$ and therefore image I changes to the output of step 2 of Figure 1.4. Similarly, steps 6 and 8 are shown in the same figure with their outputs. Other cells

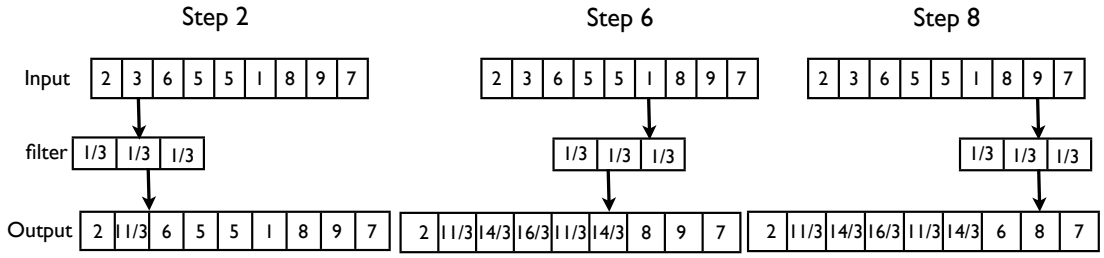


Figure 1.4: Correlation by Averaging Filter

are also modified by a similar averaging operation. However, it doesn't work on the edges such

as cells 1 and 9 and therefore their outputs haven't changed in Figure 1.4 . One of the ways to get around that problem is to assume a cell 0 with a value 0 to find the average value for cell 1 and a value 0 in cell 10 to find the averaged value of cell 9. Therefore, $I(1) = \frac{(0+2+3)}{3} = \frac{5}{3}$ and $I(9) = \frac{(9+7+0)}{3} = \frac{16}{3}$. Steps 1 and 9 are presented after these modifications in Figure 1.5 with the latter output being the final output.

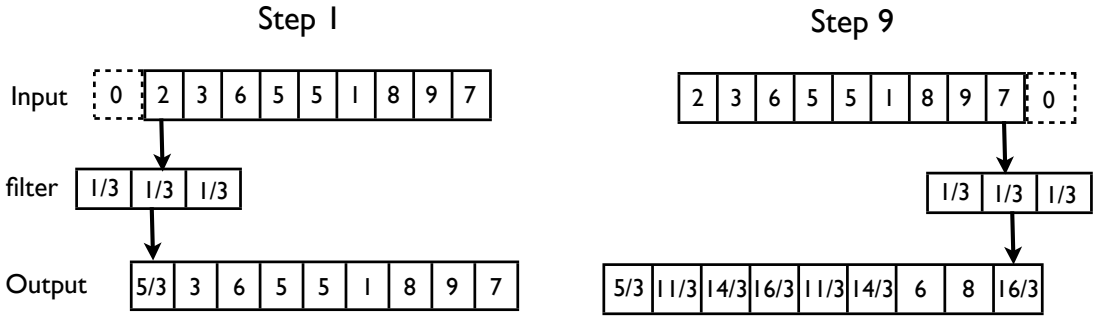


Figure 1.5: Correlation by Averaging Filter on the edges.

This can also be viewed as a sliding window operation such that each pixel value and its neighbors are multiplied by $\frac{1}{3}$ each and then the three numbers are added up to find the output. The numbers I multiply, $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ form a filter as shown in Figures 1.4 and 1.5. At each pixel I multiply the pixel value with the filter values and add the result up, to get the new value for the pixel. This forms a very simple correlation filter. Averaging like this makes this operation shift invariant since the same operation is performed at every pixel.

One of the more important and popular image smoothing functions that are used as a filter is a Gaussian filter. A one-dimensional Gaussian is defined as:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{1.3}$$

where μ is the mean value and σ is the variance. Figure 1.6 shows an example of a Gaussian plot. Instead of using the averaging to determine the pixel value as previously discussed the Gaussian, is better for smoothing since there is a continuous drop-off in the effect of the pixels on the result, rather than a sudden change. However, when the Gaussian is used for smoothing the mean is set to, $\mu = 0$ such that the pixel has the biggest effect on its smoothed value. The variance factor

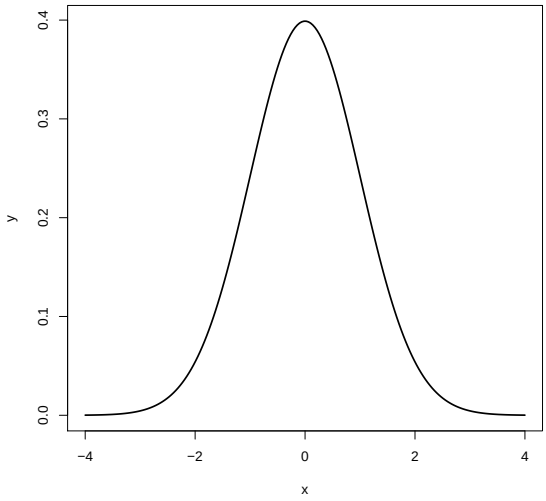


Figure 1.6: Gaussian Distribution

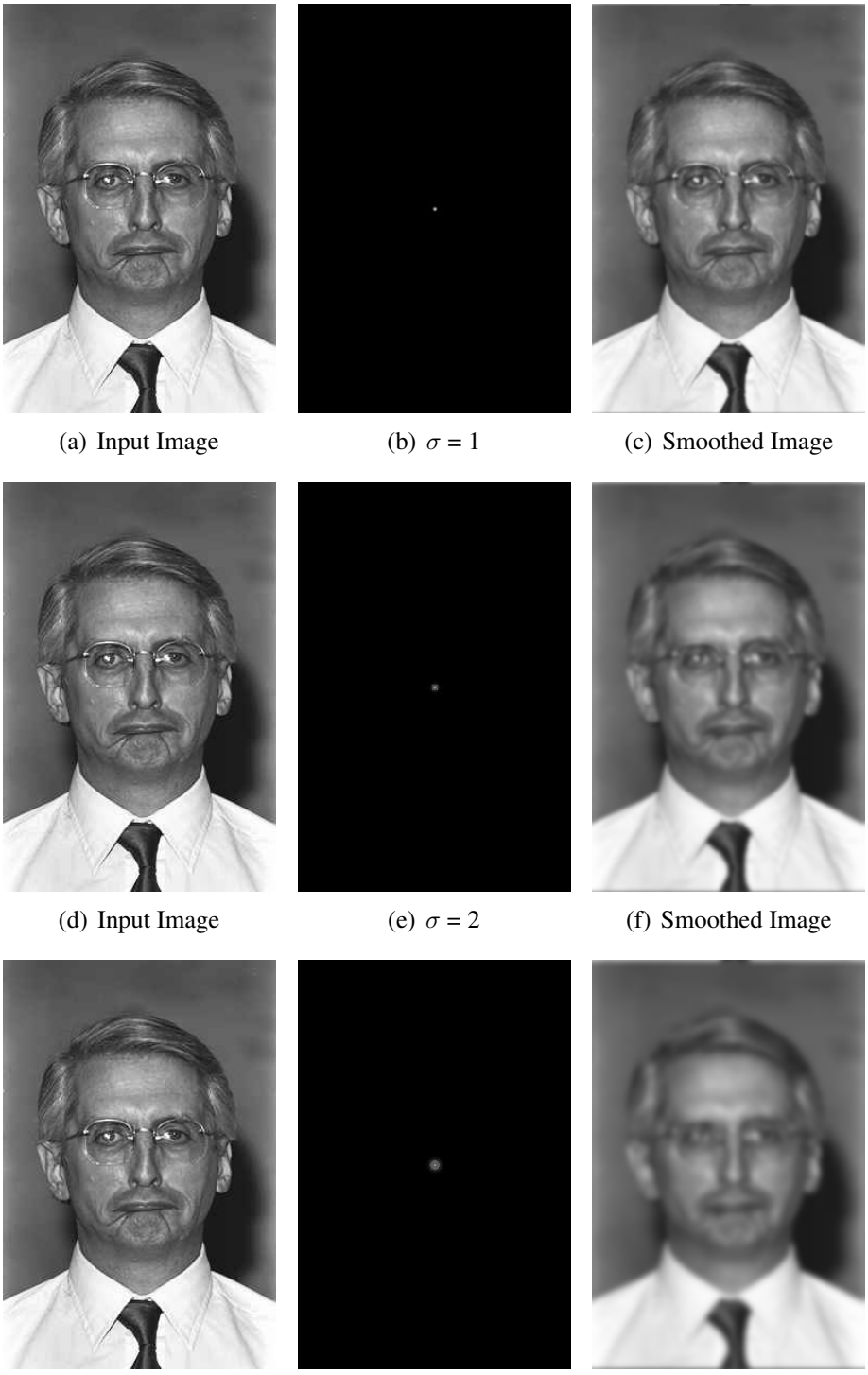
σ determines the neighborhood used to smooth out an image. The bigger the σ, the more we smooth the image. This is demonstrated with three different radii of the Gaussian and their output in Figure 1.7.

One of the important benefits of correlation is that it can be used to find parts of an image that match a template (or filter). Since correlation gives a measure of the similarity, as the filter is slid across the image it results into a high similarity value where the filter matches the image, the most. However, there is a disadvantage to using correlation for matching because it can give a high value where the image intensity is high even though the image location and the filter may not be as similar as at some other location. This can be overcome by using sum of square differences between the signals instead of simple averaging. One needs to keep in mind that using correlation involves normalizing each signal through zero mean and unit length.

The relationship between the sum of squared differences and the correlation can be described starting with the definition for correlation in equation 1.4,

$$\text{corr} = \frac{\frac{1}{N}\sum(x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y} \tag{1.4}$$

where N is the length, μ_x and μ_y the means, and σ_x and σ_y the standard deviations of the vectors x and y, respectively. Since each signal is normalized to have zero mean and unit standard deviation,



(a) Input Image (b) $\sigma = 1$ (c) Smoothed Image

(d) Input Image (e) $\sigma = 2$ (f) Smoothed Image

(g) Input Image (h) $\sigma = 3$ (i) Smoothed Image

Figure 1.7: The effect of Gaussian smoothing of an image using different σ.

the correlation equation 1.4 gets reduced to equation 1.5.

$$\text{corr} = \frac{1}{N} \sum x_i y_i \tag{1.5}$$

The sum of squared differences is represented by the equation 1.6.

$$\begin{aligned} \text{ssd} &= \sum (x_i - y_i)^2 \\ &= \sum {x_i}^2 + \sum {y_i}^2 + 2 \sum x_i y_i \end{aligned} \tag{1.6}$$

With zero mean and unit standard deviation the first two summations of equation 1.6 are both equal to N, and the third term is $N \times correlation$. Therefore, combining equations 1.5 and 1.6, I get the relationship between the sum of squared differences and correlation.

$$\begin{aligned} \text{ssd} &= N + N - 2N \times corr \\ \text{corr} &= 1 - \frac{\text{ssd}}{2N} \end{aligned} \tag{1.7}$$

Until now this approach has been explained using 1D images and filters. However, images are 2D and so are the filters used here. Computationally, 2D correlation is more expensive than 1D correlation. For a filter of size $M \times M$ and an image of size $N \times N$, the total number of multiplications that must be performed is N^2M^2. However, this complexity can be greatly reduced if the computation is carried out in the Fourier domain instead of the spatial domain. In the Fourier domain the correlation is simply a multiplication between an image and a template, after taking their Discrete Fourier Transform (DFT). In this research the DFT of an image and a template is computed using the Fast Fourier Transform (FFT) algorithm. As a result of such a transformation, the number of operations gets reduced to $N \log N$, which leads to a significant decrease in the computation time.

1.2.1 Correlation and Discrete Fourier Transform

Convolution, correlation and autocorrelation are operations commonly made much faster through the application of the Fast Fourier Transform (FFT) algorithm. The correlation theorem like the convolution theorem says that DFT of the correlation (or convolution) of two functions is equal to the element wise product of the Discrete Fourier transform of one function with the complex

conjugate of the Discrete Fourier transform of the other as shown below.

$$Corr(g, h) \leftrightarrow G(f)H^*(f) \tag{1.8}$$

where $G(f)$ and $H^*(f)$ are the Discrete Fourier Transform of $g(t)$ and the complex conjugate of the Discrete Fourier Transform of $h(t)$, respectively.

This process can also be explained as follows: Take a DFT of two datasets, multiply one resulting transform by the complex conjugate of the other, and inverse transform the product. Since a Gaussian filter was discussed earlier I would like to point out that the Discrete Fourier Transform of a Gaussian is also a Gaussian. Smoothing with a Gaussian reduces high frequency components of a signal.

1.3 Face Detection and Correlation Filters

The computational efficiency of using correlation filters in the frequency domain makes it a viable solution for face detection particularly in real time applications. It is also very easy to train correlation filters, unlike the Viola and Jones cascade face detector which is tedious to train and takes a long time. In addition, Viola and Jones face detector results in a lot of false positives. Although speed and ease of training are very good features to have in a face detector, those are not the only reasons for us to invest in this research for a correlation filters based face detector. It is also an exploration of the performance of a correlation based approach on images in unconstrained settings. The results on the FERET dataset are very competitive. I will present these results in detail in later chapters. Here I want to show the promise of success that these filters display when tested on hard datasets.

Recently I participated in a competition to detect faces in hard datasets organized as part of the International Joint Conference on Biometrics[3] [PWH+11]. There were eight academic and four commercial participants that submitted their results to this competition. On the hardest dataset of this competition my solution was regarded as the most practical approach. An example image of

[3] http://vast.uccs.edu/FDHD

this dataset is shown in Figure 1.8. Although I only got 27 true positives in 200 images and the

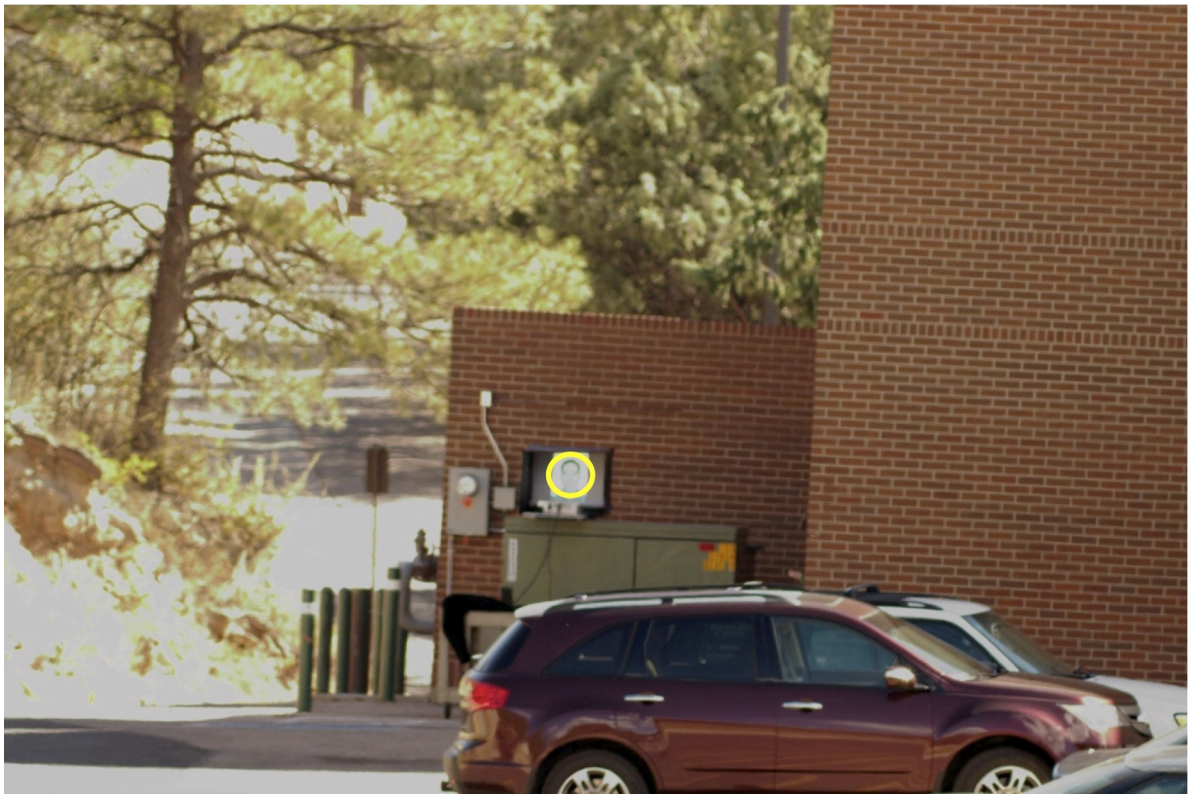


Figure 1.8: Example of an image from a hard dataset with the face marked by the yellow circle.

best method got 100 and the second best method 80, my approach is encouraging because not only did it perform better than 9 other submissions but also had only 147 false positives compared to 505 and 280 in the first two performers [PWH+11]. It is important to point out that both of these methods use cascade detectors for face detection and the training is cumbersome.

1.4 Research Goals

In this thesis I propose to develop a novel face detector based on correlation filters. My goal is to have a face detector which is very easy to train on any new dataset and is very fast in testing new images. The approach is based on identifying a point that corresponds to a face. This point is chosen to be the peak of a Gaussian function centered on a face during training such that the values around this point drop off gradually. It makes it easier to associate the peak with a face whenever the algorithm finds one in a test image. This requires creating a Gaussian filter for each

training image and centering the Gaussian at the point that is identified as a target location. This is followed by finding a Discrete Fourier Transform using the FFT algorithm of each image and its Gaussian filter and element-wise product of the two to give a resulting correlation surface. The goal is that in this correlation surface I get a peak at the location identified to be the target location and the peak identifies the face in the image.

There are many challenges to this approach. One of the biggest is the scale of the face in an image. In this approach the size of the filter is the same as the size of the image. In order to get around the scale issue I use a bank of filters of different scales to identify faces of different sizes in the test images. Each image is correlated with a bank of filters, and the filter that gives the best correlation output identifies the location of the face in the image. The methodology to identify the best filter will be discussed in detail in the later chapters.

The other important challenge is pose. This also requires training filters with multiple poses for each scale so that I get a better match at the right scale of the face. The other important challenge is the number of faces in an image. In this research I will restrict our approach to images having a single face because the focus is on datasets of images and video frames where the ultimate goal is face recognition. This means that very often the goal will be to match only one detected face with another for the next step of recognizing that face.

1.5 Dissertation Outline

The rest of the dissertation is organized as follows: Correlation filters will be discussed in the second chapter followed by a detailed account of the MOSSE filter in the third chapter. In the fourth chapter I discuss face detection research. The fifth chapter will explore the approach to determine the size of the face and present results on FERET database. In Chapter 6 I will present face detection results in still images and videos in the recently released Point-and-Shoot Challenge (PaSC) dataset [BPB$^+$13]. Chapter 7 presents a case study of filters tested on a specific scenario where the location is held constant between the training and the test images. Finally, this document ends with a conclusion chapter.

Chapter 2

Correlation Filters

Correlation Filters [Kum92] have been used in face verification [MSK02], face recognition [KSX07], target detection, and noise and clutter rejection [Goo96]. The process of using correlation filters in computer vision applications is demonstrated in Figure 2.1. The peak in the correlation output

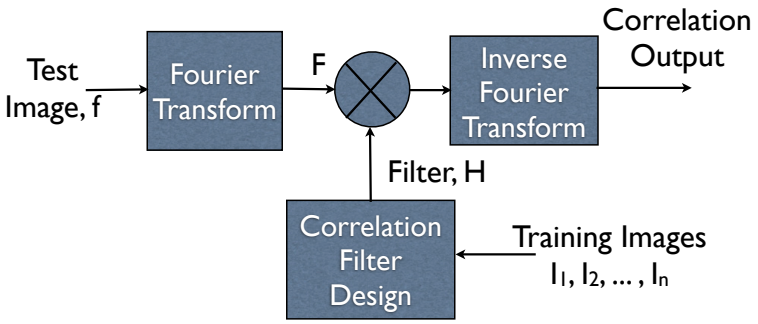


Figure 2.1: Correlation Filter Process

is used to determine the target location. This is a general approach in the use of correlation filters which is similar to how I use it, except for some missing details that will be elaborated on in the subsequent chapters. It needs to be pointed out here that the correlation filters approach described here is based on a Fourier approach. There are advantages to working in the Fourier domain such as shift-invariance, graceful degradation, and closed form solutions [KSX07]. Some of these filters can be described as follows:

2.1 Matched Filter

A matched filter [Nor63] is an optimized linear filter that maximizes the Signal to Noise Ratio (SNR) in the presence of additive stochastic noise. It is obtained as a result of correlating a known signal with an unknown signal to determine whether a template exists in the unknown signal. It involves convolving the unknown signal with the conjugated time reversed copy of the template. It is commonly used in RADAR. One of its applications in image processing is to improve SNR in X-rays. Its role in other image processing applications such as face recognition is very limited. Since a test face image will differ from the reference image in expression, pose, age, illumination etc., optimal matched filter solution would be to create a filter for each one of these variations. This renders it unusable due to the huge number of filters that would be required [KSX07]. This also means that when these variables match or the two images differ only in white additive noise, one can get a good response using matched filters.

2.2 Synthetic Discriminant Function (SDF) Filters

In order to overcome the explosion of the number of matched filters that are required to be used, Hester and Casseant introduced Synthetic Discriminant Function (SDF) filters [HC80]. They are based on a concept of setting a correlation value of 1 at the origin for a set of training images corresponding to an authentic subject and a zero value at the origin of the impostor subject. This approach requires a training set of images and the SDF filter is the weighted sum of matched filters where the weights are chosen so that the correlation outputs corresponding to the training images would yield a pre-specified correlation value at the origin. As such, in face verification applications, it is expected that the resulting correlation filter has a peak value close to one for authentic images and close to zero for impostor images. In face recognition application of SDF, the approach is similar to the one described in Figure 2.1. The correlation output peak is used to determine whether there is a match or not depending on its value.

SDF is a linear combination of a set of matched filters [D.84] [Kum92]. The SDF filter function $h(x, y)$ can be expressed as a linear combination of the set of reference images $f_i(x, y), i =$

$1, 2, ..., N$ [GFDRKAB05], such that,

$$h(x, y) = \sum_{i=1}^{N} a_i f_i(x, y) \tag{2.1}$$

where a_i are weighting coefficients chosen to satisfy the following conditions:

$$f_i \circ h = u_i \tag{2.2}$$

where $\circ$ denotes correlation and u_i is a pre-specified value in the correlation output at the origin for each training image. Let **R** denote a matrix with N columns and d rows (number of pixels in each training image), where its ith column is given by the vector version of $f_i(x, y)$. Let **a** and **u** represent column vectors of the elements a_i and u_i, respectively. The equations 2.1 and 2.2 can be rewritten as:

$$\mathbf{h = aR}, \tag{2.3}$$

$$\mathbf{u = R^{+}h} \tag{2.4}$$

where $\mathbf{R}^{+}$ is a complex conjugate transpose of **R**. By substituting equation 2.3 into equation 2.4, one obtains

$$\mathbf{u = (R^{+}R)a}. \tag{2.5}$$

The element (i,j) of the matrix $\mathbf{S = R^{+}R}$ is the value at the origin of the cross correlation between the images $f_i(x, y)$ and $f_j(x, y)$. If the matrix **S** is non-singular, the solution of the equation system is

$$\mathbf{a = (R^{+}R)^{-1}u}, \tag{2.6}$$

and the filter, expressed as a vector is given by

$$\mathbf{h = R(R^{+}R)^{-1}u}. \tag{2.7}$$

However, one of the drawbacks of SDF filters is that they control the output at the origin only. Also, they require the training images be centered on the target of interest. Since the test images are not necessarily centered one will have no idea where the controlled values in the output are located, unless one can control the rest of the values to be very small [KSX07].

2.3 Minimum-Variance Synthetic Discriminant Functions (MVSDF) Filter

One of the requirements of SDF filters [HC80] is a linear combination of the training images such that they satisfy some deterministic constraints. Kumar introduced an MVSDF filter [Kum86] such that the deterministic constraints are still satisfied but at the same time the output variance due to white noise is minimized. Kumar suggested an approach to minimize this noise by removing the condition of linear combination of training images in the design of SDF filters.

In SDF design it is assumed that the correlation value of the origin will be a pre-specified value whenever the input image is a training image. This may not always be true and if the input image is a noisy training image, the output would differ from the pre-specified value. In the design of MVSDF, Kumar [Kum86] report the minimization of any such variance.

In order to design an MVSDF let us assume that a training image is corrupted by a zero-mean, additive, stationary noise vector n. The constraint equation 2.4 of SDF is replaced by

$$|u|^2 = |R^+h|^2 \tag{2.8}$$

In the absence of noise, the $|R^+h|$ term will yield the constraint vector u. However, because of the noise n, the cross-correlation output will have an additional random contribution $y = h^+n$. One can show that:

$$E\{y\} = 0 \tag{2.9}$$

and

$$\begin{aligned} Var\{y\} =& E\{h^+nn^+h\} \\ =& h^+E\{nn^+h\} \\ =& h^+Ch, \end{aligned} \tag{2.10}$$

where $E\{.\}$ and $Var\{.\}$ denote the expected value and the variance, respectively, and C is the $d \times d$ covariance matrix of the noise vector n. The goal of MVSDF is to minimize the variance. In order to minimize variance, Kumar introduces a function ϕ, such that

$$\phi = h^+Ch - 2\lambda_1(h^+x_1 - u_1) - \ldots - 2\lambda_N(h^+x_N - u_N), \tag{2.11}$$

where $\lambda_1, \ldots, \lambda_N$ are Lagrange multipliers introduced for the constrained minimization. Taking a derivative of ϕ w.r.t. h and setting it to zero one gets,

$$Ch_{opt} = \lambda_1 x_1 + \ldots + \lambda_N x_N, \tag{2.12}$$

where h_{opt} is the optimal filter. It is assumed that the covariance matrix C is invertible and therefore h_{opt} can be rewritten as

$$\begin{aligned} h_{opt} =& C^{-1} \sum_{i=1}^{N} \lambda_i x_i \\ =& \sum_{i=1}^{N} \lambda_i C^{-1} x_i, \end{aligned} \tag{2.13}$$

2.4 Minimum Average Correlation Energy (MACE) Filter

A MACE filter [MKC87] minimizes the average energy of the correlation outputs due to the training images. It is designed such that the correlation value is the highest at the object location in the training images and zero everywhere else. The peaks of the correlation output are very distinguishable because of their sharpness. In order to construct a MACE filter one needs to minimize average correlation energy, E_{av}, and this can be done directly in the frequency domain by averaging the correlation plane energies E_i [MKC87] as follows:

$$\begin{aligned} E_{av} =& (\frac{1}{N}) \sum_{i=1}^{N} E_i \\ =& (\frac{1}{N}) \sum_{i=1}^{N} H^+ D_i H \\ =& (\frac{1}{N}) H^+ (\sum_{i=1}^{N} D_i) H, \end{aligned} \tag{2.14}$$

The variables in the equations can be explained as follows: N is the number of training images, each of size $d \times d$, H is the Fourier transform of the filter, H^+ is its complex conjugate transpose and D_i is a diagonal matrix of size $d^2 \times d^2$ whose diagonal elements are the magnitude square of the associated elements of X_i, a column of a complex valued matrix X which is formed by taking

the 2-D FFT of the training images followed by vectorizing them to form the columns of X whose size is $d^2 \times N$. We can define D as

$$D = \sum_{i=1}^{N} \alpha_i D_i, \tag{2.15}$$

where α_i are all constants. If all $\alpha_i = 1$, we may rewrite equation 2.14 as

$$E_{av} = (\frac{1}{N}) H^+ D H, \quad \alpha_i = 1, i = 1, 2, \ldots, N. \tag{2.16}$$

Therefore, $\alpha_i = 1$, makes sure that all correlation planes get equal weight. The average energy is now represented by equation 2.16. In order to create a filter with minimum energy one needs to minimize H^+DH subject to the linear constraints, $X^+H = u$, where u is the constraint vector containing the pre-specified correlation values. This problem can be solved using Lagrange multipliers and the final filter, H, is given by

$$H = D^{-1}X(X^+D^{-1}X)^{-1}u. \tag{2.17}$$

A detailed proof of this result can be found in the paper that introduced MACE filters [MKC87].

MACE filters are very sensitive to input noise and deviation from the training images because such filters emphasize high spatial frequencies for sharp correlation peaks [KSX07]. It is important to point out, however, that a MACE filter is preprocessing invariant. This implies that even if the training images are linearly pre-processed, the correlation energies E_i do not change. Therefore, the use of a high, low or a band pass filter on the data does not influence the outcome [MKC87]. These filters have been successfully used in applications like face recognition and face verification [SVVK03] [SVK03] [MSK02] [VSVX02].

2.5 Optimal Trade-off Filters (OTF)

Most of the filters discussed thus far try to focus on one performance measure, for example, minimization of noise sensitivity in MVSDF and maximization of peak sharpness in MACE [Kum92]. When focus is on one factor, a filter may not be able to provide better performance on other factors, therefore, it would be desirable to obtain filters that optimize a combination of factors. Refregier [Ref91] designed the optimal trade-off filter in order to detect an undistorted image.

The goal is to improve three performance measures: SNR to measure noise tolerance, the peak-to-correlation energy to measure peak sharpness, and the Horner efficiency [Hor82] [Cau82] to measure the light-throughput eficiency of the filter. Refregier demonstrated that we can optimize any one of these three measures while holding the other two at specific values. The filter is said to be optimal because no other filter can be designed to yield a smaller value of the performance measure while holding the other two at specified levels.

In the correlation output, the variance of the noise is given by equation 2.10. Minimizing this output variance h^+Ch for the MVSDF filter results in a filter emphasizing low spatial frequencies while minimizing h^+Dh (Section 2.4) for the MACE filter results in a new filter emphasizing high spatial frequencies [KSX07]. While satisfying the constraints of equation 2.4, and optimally trading off between h^+Ch and h^+Dh, Refregier [Ref91] designed the following Optimal Trade-Off filter:

$$h = T^{-1}X(X^+T^{-1}X)^{-1}u \tag{2.18}$$

where $T = (\alpha D + \sqrt{1-\alpha^2 C})$, and $0 \leq \alpha \leq 1$ is a parameter that controls the tradeoff. When $\alpha = 0$, we get a maximally noise tolerant filter, and $\alpha = 1$ results in a MACE filter, producing very sharp correlation peaks.

2.6 Unconstrained Minimum Average Correlation Energy (UMACE) Filter

Most of the filters described thus far assume a pre-specified correlation value at the origin. Mahalanobis et al. [MKS+94] removed that constraint and developed UMACE filters to optimize distortion tolerance. These filters are designed to yield good performance in the presence of noise and background clutter and at the same time result in sharp correlation peaks for easy output detection. In order to present a design for UMACE filters, let us assume the training set consists of N images, of two classes called a true and a false class, which don't have to be of the same size. Each image is assumed to be of size $d_1 \times d_2$. The ith training image for the true class is denoted by $x_i(m.n)$ and is represented in the frequency domain by a $d\times 1$ vector x_i. The false class training images, $y_i(m,n)$ are represented in vector notation in the frequency domain as y_i.

The filter is represented by H, of size $d\times 1$. The correlation of the ith training image and the filter can be represented as

$$g_i = X_i H, \tag{2.19}$$

where X_i is a $d \times d$ diagonal matrix containing the elements x_i and g_i denotes the Fourier transform of the ith correlation output. Mahalanobis et al. quantify the deviation of a correlation plane from some ideal vector f using Average Squared Error (ASE), represented as

$$ASE = \frac{1}{N}\sum_{i=1}^{N}(g_i - f)^{+}(g_i - f). \tag{2.20}$$

Therefore, ASE is a measure of distortion with respect to the reference shape f. One is interested in an f that minimizes ASE. Therefore, the optimum shape f_{opt}, is obtained by dividing ASE with respect to f and setting it to zero and one gets,

$$f_{opt} = \frac{1}{N}\sum_{i=1}^{N} g_i = \overline{g} \tag{2.21}$$

where

$$\overline{g} = \frac{1}{N}\sum_{i=1}^{N} g_i = \frac{1}{N}\sum_{i=1}^{N} X_i h = \overline{X}h \tag{2.22}$$

is the average correlation plane and $\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$ is the average training image expressed as a diagonal matrix. Therefore, the minimal ASE is obtained by using the average correlation plane $\overline{g}$, resulting in the least distortion. Substituting $f = \overline{g}$ in the ASE expression defines the Average Similarity Measure (ASM), as

$$\begin{aligned} ASM =& \frac{1}{N}\sum_{i=1}^{N}(g_i - \overline{g})^{+}(g_i - \overline{g}) \\ =& \frac{1}{N}\sum_{i=1}^{N}(X_i h - \overline{X}h)^{+}(X_i h - \overline{X}h) \\ =& h^{+}\left[\frac{1}{N}\sum_{i=1}^{N}(X_i - \overline{X})^{*}(X_i - \overline{X})\right] h \\ =& h^{+}\left(\frac{1}{N}\sum_{i=1}^{N} X_i X_i^{*}\right) h - h^{+}\overline{XX^{*}}h \\ =& h^{+}D_x h - h^{+}\overline{XX^{*}}h. \end{aligned} \tag{2.23}$$

where $h^+ D_x h$ is called the Average Correlation Energy (ACE) for the true class with $D_x = (\frac{1}{N}) \sum_{i=1}^{N} X_i X_i^*$. SImilarly $D_y = (\frac{1}{N}) \sum_{i=1}^{N} Y_i Y_i^*$ is the diagonal matrix containing the average power spectrum of the false class. It is used to define a filter called the Maximum Average Correlation Height (MACH) filter, defined as

$$h = c(D_y + S_x)^{-1} \overline{x}, \tag{2.24}$$

where $c = \frac{\alpha}{\lambda}$ and $S_x = \frac{1}{N} \sum_{i=1}^{N} (X_i - \overline{X})^* (X_i - \overline{X})$. A MACH filter maximizes the relative height of the average correlation peak with respect to the expected distortions. If the correlation energy term D_y is deleted from the MACH filter equation, we get

$$h_{mach} = S_x^{-1} \overline{x}. \tag{2.25}$$

This special MACH filter without the correlation energy is called unconstrained Mean Squared-Error Synthetic Discriminant Function (MSE SDF). Now representing the correlation energy of the X data, the special MACH filter can be represented as,

$$h_{mach} = D_x^{-1} \overline{x}. \tag{2.26}$$

It is obtained by dropping the ASM term from the MACE filter (section 2.4) expression and is referred to as the unconstrained MACE (UMACE) filter. A detailed analysis of this approach can be found in the paper on UMACE [MKS+94].

2.7 Other Correlation Filters

As just illustrated, there have been different approaches to correlation filter design, while some of them have been designed with the constraint of a pre-specified value at the origin, there are others designed to maximize the average correlation height [MKS+94]. A set of filters called the Distance Classifier Correlation Filter (DCCF) [MKS96] are aimed at designing a filter such that the training images from different classes are mapped to well-separated clusters. Mahalanobis and Kumar [MK97] presented Polynomial correlation filters, which use the original image and its non-linear versions as input. A correlation filter is designed for each non-linear version and are then

fused together. Recently, Kerekes and Kumar [RK06] extended the correlation filter designs using Mellin radial harmonic functions [RS89].

Chapter 3

Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE) Filter

In the Computer Vision lab at Colorado State University, my friend and colleague, David Bolme, invented two correlation filters, ASEF [BDB09] and MOSSE [BBDL10] . This chapter presents the details of these correlation filters, particularly MOSSE, which forms the core of the face detector introduced in this dissertation.

3.1 Average of Synthetic Exact Filters (ASEF)

Bolme et al. [BDB09] proposed a new class of filters called the Average of Synthetic Exact Filters (ASEF) that have two main differences from the previous methods, discussed in Chapter 2: firstly, for each training instance, an entire correlation response surface is specified during filter construction, and secondly, the resulting filters, one per training image, are then simply averaged. According to the authors, ASEF filters are less susceptible to over-fitting the training data and can therefore be trained over more inclusive and larger training sets [BBD10].

In ASEF filters the convolution theorem is simplified to map the input training image and the output correlation plane. As previously explained, the correlation operation in the Fourier domain becomes an element-wise multiplication. ASEF filters are trained using response images specifying the ideal correlation output for an image. This target correlation surface specifies a desired response at every location. This approach differs from SDF, since SDF specifies only a single correlation value per training image. In the ASEF approach, for each training image there is an exact filter. This results in a balance between constraints and degrees of freedom. The final filter is an average of the exact filters. This averaging of filters avoids over-fitting [BDB09]. UMACE filters also average to avoid over-fitting, but they differ from the ASEF filters because the averaging is done over the training images and not over the filters.

ASEF has performed well at both eye localization [BDB09] and pedestrian detection [BLDB09], however, it has not been as successful in visual tracking, because it requires a large number of training images [BBDL10]. This is overcome by using fewer images in Minimum Output Sum of Squared Error (MOSSE) [BBDL10] filters, which are ASEF-like filters constructed from fewer training images.

3.2 Minimum Output Sum of Squared Error (MOSSE) Filter

MOSSE [BBDL10] filters are like ASEF filters but can be trained with very few images. They are more robust with respect to changes in the appearance of a foreground object and discriminate well between the target and the background. Bolme et al. successfully used these filters for tracking. Such a tracker is robust to changes in lighting, scale, pose and non-rigid deformations and operates at 669 frames per second [BBDL10]. The design of the filter is carried out in the Fourier domain to exploit the convolution theorem. According to this theorem correlation in the Fourier domain is an element-wise multiplication. Therefore correlation is defined as:

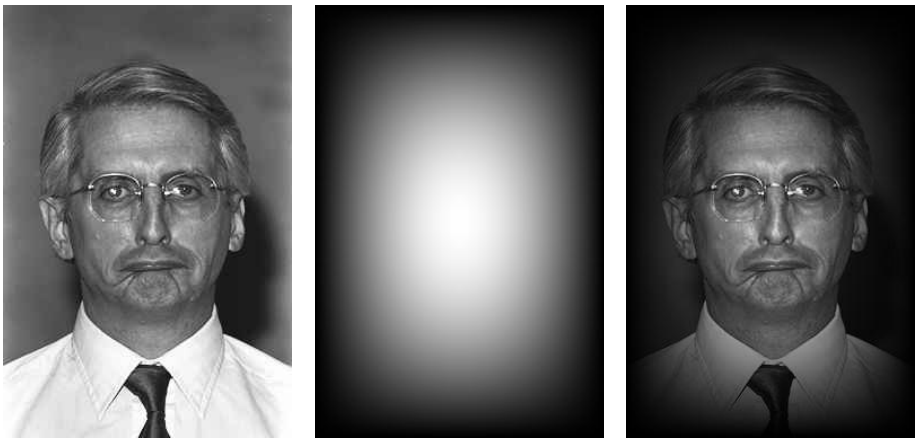
$$G = F \odot H^* \tag{3.1}$$

where $\odot$ denotes the element-wise multiplication and $*$ is the complex conjugate. G, H and F are the Fourier transform of the Gaussian distribution g, image f and filter, h. The correlation output is transformed back into the spatial domain using the inverse FFT. In this process the time complexity bottleneck is forward and the inverse FFT and the complexity of the entire process is $O(N \log N)$, where N is the number of pixels in the tracking window. I will now describe the design of the MOSSE filter starting with the pre-processing of the images.

3.2.1 Preprocessing

The use of FFT in convolution greatly enhances the speed of computation, however, it comes with some challenges that need to be overcome. One of them being the mapping of the images and the filter to the torus topological structure. This means the image left is connected to the image right and the image top is connected to the image bottom. The reason for this is because

during the convolution process the images rotate through toroidal space rather than translating like they would, in the spatial domain. This artificial connection of image edges affects the correlation output. This can be avoided by following the preprocessing steps in [BDB09]. First, the pixel values are transformed using the natural log function to help with low contrast lighting situations. In order to avoid taking the log of a pixel with an intensity value of zero, all such pixels are added a value of one before applying the log function. The second step is to normalize the pixel values such that the image has a mean of zero and a unit norm. In the final step, the image is multiplied by a cosine window. Such a window gradually reduces the pixel values near the edges to zero, putting more emphasis on the center of the target. An example of such a cosine window is displayed in Figure 3.1.



(a) Input Image (b) Cosine Window (c) Processed Image

Figure 3.1: Cosine Window and a Processed Image after multiplying an image with a cosine window.

3.2.2 MOSSE Filter Design

A MOSSE filter is constructed such that the output sum of squared error is minimized. Figure 3.2 displays the process of creating such a filter.

The pairs f_i, g_i are the training images and the desired correlation output, respectively. This desired output image g_i is synthetically generated such that the point between the eyes in the training image f_i has the largest value and the rest of the pixels have very small values. More specifically, g_i is generated using a 2D Gaussian centered at the target point (x_i,y_i) between the

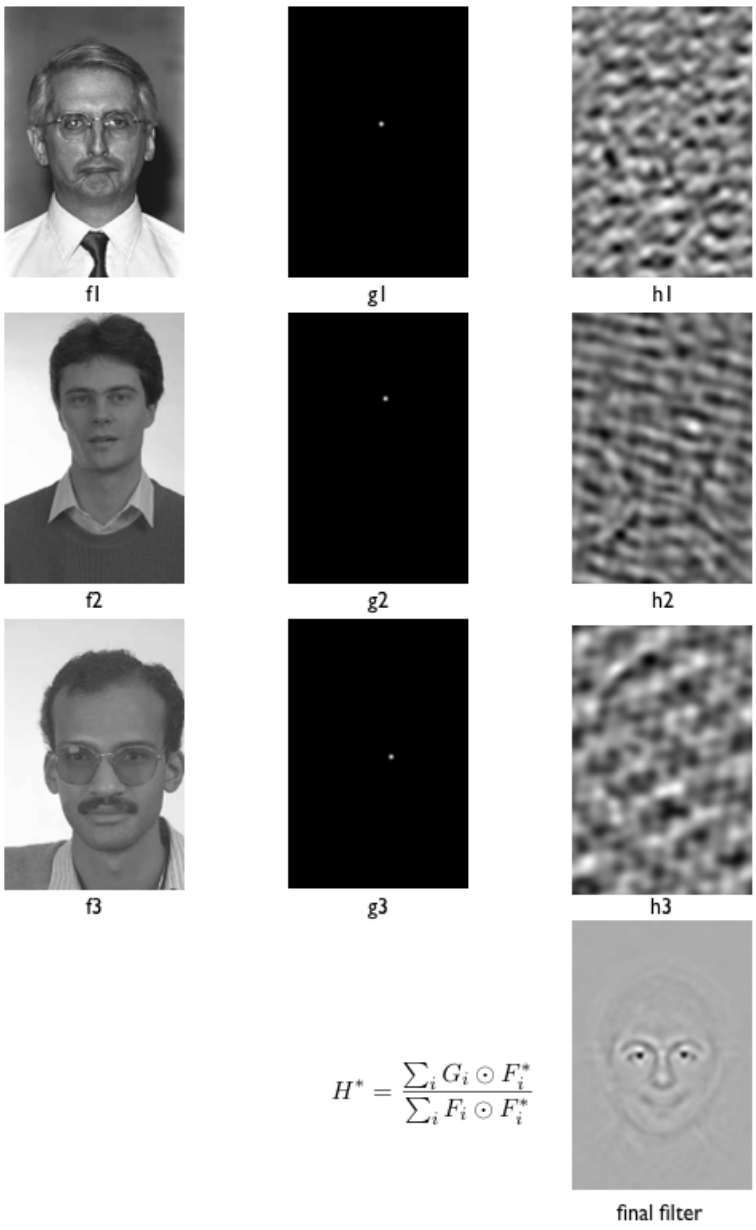


Figure 3.2: MOSSE Filter Process

eyes with radius σ, such that

$$g_i(x, y) = e^{-\frac{(x-x_i)^2+(y-y_i)^2}{\sigma^2}} \tag{3.2}$$

The construction of the filter requires transformation of the input images and the Gaussian images into the Fourier domain to take advantage of the simple element-wise relationship between the input and the output. Let F_i, G_i be the Fourier transform of the lower case counterparts, then the exact filter H_i is defined as,

$$H_i^* = \frac{G_i}{F_i} \tag{3.3}$$

where the division is performed element-wise. The filters like the ones shown in Figure 3.2 such as h_1, h_2 and h_3 are specific to their corresponding images f_1, f_2 and f_3, respectively. In order to find a filter that generalizes across the dataset, the MOSSE filter H, is generated such that it minimizes the sum of squared error between the actual output of the convolution and the desired output of the convolution. This minimization problem is represented as:

$$min_{H^*} \sum_i |F_i \odot H^* - G_i|^2, \tag{3.4}$$

where F_i and G_i are the input images and the corresponding desired outputs in the Fourier domain. This equation can be solved to get a closed form solution for the final filter H [BBDL10]. Since the operation involves element-wise multiplication, each element of the filter H can be optimized independently. Therefore, the equation 3.4 can be re-written as

$$H_{wv} = min_{H_{wv}} \sum_i |F_{iwv} {H^*}_{wv} - G_{iwv}|^2 \tag{3.5}$$

where w and v index the elements of H. This function is real valued, positive, and convex implying the presence of a single optimum. This optimum is obtained by taking the partial derivative of H_{wv} w.r.t. ${H^*}_{wv}$ and setting it to 0. Therefore, the equation becomes,

$$0 = \frac{\partial}{\partial H^*_{wv}} \sum_i |F_{iwv} H^*_{wv} - G_{iwv}|^2 \tag{3.6}$$

By solving for H^* a closed form expression for the MOSSE filter is

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*} \tag{3.7}$$

where H^* is the complex conjugate of the final filter H in the Fourier domain. A more complete derivation of this expression is in the appendix of the MOSSE paper [BBDL10]. The terms in equation 3.7 can be interpreted as the correlation between the input and the desired output in the numerator and the energy spectrum of the input in the denominator. It is useful to mention here that ASEF filter has a similar expression except for the averaging, which can be represented as

$$H^*_{asef} = \frac{1}{N}\frac{\sum_i G_i \odot F^*_i}{\sum_i F_i \odot F^*_i}. \tag{3.8}$$

In the case of using only one image for training both MOSSE and ASEF produce an exact filter. It is also worth noting that the MOSSE filter is in the Fourier domain and needs to be converted back to the spatial domain through an inverse Fourier transform.

Chapter 4

Face Detection Background

The first step of a face processing system is to detect faces. Face detection aims to identify whether a face exists or not in an image and if it does, report the location of each face [YKA02]. We will now describe some of the techniques in face detection research.

4.1 Overview and Early Work in Face Detection

According to Yang et al. [YKA02], face detection can be divided into four broad categories based on their approach: knowledge-based, feature invariant, template matching and appearance-based method. Knowledge-based methods are based on rules that encode a typical face. They have been primarily used for face localization. Yang and Huang [YH94], used this approach to locate human faces in a complex background. Their approach consists of three knowledge based levels. At level 1, the whole image is scanned to locate all the faces and in the next level, 8×8 cell window screens locate faces obtained in the first level. At level 3 further screening is done and if eye and mouth regions match with the established characteristics then the presence of the face is established.

Feature invariant approaches use structural features to identify faces in an image. This approach makes them robust to variations in pose, viewpoint, or lighting conditions. Leung et al. [LBP95] used this approach to locate faces in cluttered scenes. They used a set of local feature detectors and coupled them with a statistical model of the mutual distances between the facial features. This approach is robust to variations in scale, translation, rotation (in the plane) and can handle partial occlusions.

Yow and Cipolla [YC97] proposed a feature based algorithm that detects feature points from the image using spatial filters and groups them into face candidates using geometric and gray level constraints. In order to establish the presence of a face, they use a probabilistic framework. Dai and Nakano [DN96] developed a method to detect faces in cluttered color scenes. They de-

rived a set of inequalities to form a face-texture model based on the feature parameters present in a Space Gray-level Dependence matrix (SGLD). These inequalities helped define the face location. They enhanced face areas by utilizing the I component of the YIQ color system. Yang and Waibel [YW96] developed a real-time face tracker. They characterized human faces using a skin color model in chromatic color space. They also proposed a motion model to track head motion and a camera model to predict camera motion. McKenna et al. [MGR98] used Gaussian mixture models to detect human faces.

Template matching methods use correlation between an image and stored patterns for detection as well as localization. Craw et al. [CTB92] presented a method to locate individual face features like the eyes and mouth in a gray scale face image. Face detection is carried out in two steps beginning with general localization of the faces in the image followed by further refinement and assessment to find the face. Lanitis et al. [LTC95] present a model based representation of the shape and grey-level appearance of human faces. These models are subsequently used to classify images.

Appearance-based methods are primarily designed for face detection. The models are generated from the images to capture their variability. Turk and Pentland [TP91] carried out face recognition and detection using principal component analysis. Principal Component Analysis is used to generate eigen faces that span a subspace. After projecting these images to the subspace, they are clustered. The face images do not change radically when projected to the subspace while the non-faces appear differently when projected onto the subspace. The distance between each location in the image and the face space is computed to detect the presence of a face. The distance from the face space is used as a measure of closeness to a face and the result of this distance is a face map. A face can then be detected from the local minima of the face map.

Sung and Poggio [SP98] presented a system to learn about the image patterns of a class from the positive and the negative examples of that class. They use a two component approach to distinguish between faces and non faces. The first component consists of distribution-based models for face/nonface patterns and the second component is a multilayer perceptron classifier.

Rowley et al. [RBK98] have done the most significant work in face detection using neural

networks. They use a multilayer neural network to differentiate between the face and nonface patterns from the face and nonface images. However, they can only detect upright frontal faces. One of the most widely used face detection databases has been created by Rowley et al.[1]. Osuna et al. [OFG97] were the first to use an SVM for face detection. Their system has lower error rates and runs 30 times faster than the system by Sung and Poggio.

Schneiderman and Kanade [SK98] developed an approach using a naive Bayes classifier to compute the joint probability of local appearance and position of the subregions of the face at different resolutions. The emphasis on local patterns or subregions is because they may be more distinctive. For example, there is a clear difference between the patterns around the eyes and those around the cheeks. In this procedure a face is divided into four rectangular subregions at each scale. These subregions are projected to a lower dimensional space using PCA and quantized into a finite set of patterns. The statistics from each subregion are used to encode local appearance. Therefore, they report the presence of a face when the likelihood ratio is larger than the ratio of the prior probabilities. This approach is able to detect some rotated and profile faces. This work was later extended by Schneiderman and Kanade [SK00] using wavelet representations to detect profile faces and cars.

Rajagopalan et al. [RKK$^+$98] proposed two probabilistic methods for face detection. In the first method they use higher order statistics (HOS) for density estimation. In the second method faces and nonfaces in an image are defined through the use of a Hidden Markov Model(HMM). Although these approaches have a high detection rate, there are also some false positives.

Lew [Lew96] carried out face detection by associating a probability function $p(x)$ to the event that the template is a face and $q(x)$ to the event that the template is not a face. A face training database consisting of nine views of 100 individuals is used to estimate the face distribution and a set of 143,000 nonface templates is used to estimate the nonface probability density function.

One of the seminal papers in face detection that has made a huge impact is by Viola and

[1] http://vasc.ri.cmu.edu/NNFaceDetector

Jones [VJ01] [VJ04]. I will now describe this face detector in the next section.

4.2 Viola and Jones Face Detector

Viola and Jones [VJ01] presented a face detection framework that can detect faces at 15 frames per second. The authors distinguished their work through three key contributions in the development of this face detector: An image representation based on the integral image presented by Crow [Cro84], a learning algorithm, based on AdaBoost and combining complex classifiers in a cascade. These key components of this face detector will now be described.

4.2.1 Integral Image

An integral image is based on Haar like features [POP98] which are shown in Figure 4.1. These are the three kinds of features that Viola and Jones used in their research. The value of a

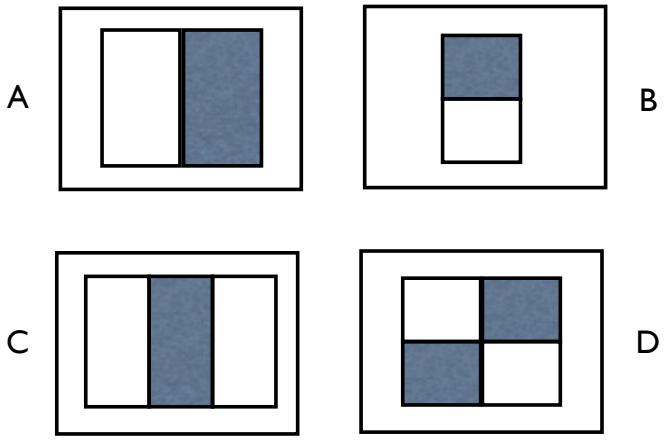


Figure 4.1: Integral Image Features: A and B: Two-rectangle features, C: Three rectangle features, D: Four rectangle features

two-rectangle feature is the difference between the sum of the pixels within the two rectangular regions. These regions are horizontally or vertically adjacent and of the same size. The three-rectangle feature computes the difference between the sum of the two outside regions and the center rectangle. The four-rectangle features find the difference between the diagonal rectangle

pairs. The base resolution of the detector is 24 $\times$ 24 which makes the number of rectangular features to be 160,000.

In order to find these rectangular features, Viola and Jones [VJ04] used an intermediate image representation which they called an integral image. At a location x,y an integral image contains the sum of the pixels above and to the left of x,y, inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{4.1}$$

where $ii(x,y)$ is the integral image of the original image i.

Rectangular features are somewhat primitive and sensitive to the presence of edges, bars, and other simple image structure. At the same time they are quite coarse. Viola and Jones generated a very large and varied set of rectangular features. This set of overcomplete rectangular features provide a rich image representation supporting effective learning. Therefore the limitations of the rectangle features are largely overcome by their extreme computational efficiency.

This face detector scans an input image at many scales, starting at the base scale in which faces are detected at a size of 24 $\times$ 24 pixels, a 384 $\times$ 288 pixel image is scanned at 12 scales, each a factor of 1.25 larger than the last. In practice, however, a pyramid of 12 images, each 1.25 times smaller than the previous image is computed. A fixed scale detector is then used to scan across each of these images.

4.2.2 AdaBoost Learning

Viola and Jones use a variant of AdaBoost [FS95] learning to train the classifier and select the features. AdaBoost is a greedy selection process. A weighting system is used to associate weights to classifiers with large weight associated with each good classification function and a small weight with poor functions.

AdaBoost is an effective technique to select a small set of good functions. The goal is to separate positive and negative examples such that a minimum number of examples are misclassified. In order to achieve this, a weak learning algorithm is designed to select a single rectangular feature that best achieves this separation between the positive and the negative examples. This weak classifier determines the optimal threshold function to minimize misclassification. A weak classifier

can be presented as

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

where $(h(x, f, p, \theta))$ is a weak classifier consisting of feature f, a threshold θ and a polarity p and a 24 $\times$ 24 pixel sub-window, x, of an image.

The number of weak classifiers is in the order of $K \times N$, where K is the number of features and N is the number of examples. In order to get an idea about the number of distinct weak classifiers we will cite an example from Viola and Jones [VJ04]. Given a task with $N = 20000$ and $K = 160000$ there are 3.2 billion distinct binary weak classifiers. Despite this extraordinarily large set of classifiers, AdaBoost is really very efficient and fast. A 200 feature classifier can be learned in $O(MNK)$ or about 10^{11} operations. In each stage the dependence on previously selected features are efficiently encoded using example weights which can then be used to evaluate a given weak classifier in constant time.

The process of generating a weak classifier can be explained as follows. The examples are sorted on the basis of each feature value for each feature. The optimal threshold for AdaBoost for a feature can be computed in a single pass over this sorted list. For each element in the sorted list , the following four sums are maintained and evaluated: the total sum of positive example weights T^+, the total sum of negative example weights T^-, the sum of positive weights below the current example S^+ and the sum of negative weights below the current example S^-. The error for a threshold which splits the range between the current and previous example in the sorted list is:

$$e = min(S^+ + (T^- - S^-), S^- + (T^+ - S^+)), \tag{4.3}$$

or the minimum of the error of labeling all examples below the current example negative and labeling the examples above positive versus the error of the converse. These sums are easily updated as the search proceeds.

The initial features selected by AdaBoost are very useful for face detection. It establishes that the region of the eyes is often darker than the region of the nose and the cheeks. However, this feature is not sensitive to the location and the size of the face. The second feature is slightly more specific and relies on the property that the eyes are darker than the bridge of the nose. Therefore,

the features become an effective tool in the process of face detection.

4.2.3 Cascade of Classifiers

The goal of building a cascade of filters is more efficiency and so a reduction in computation time. Viola and Jones suggested that a smaller number of efficient and boosted classifiers can be constructed. These boosted classifiers will be more efficient in rejecting many of the negative sub-windows while detecting almost all positive instances. The majority of the sub-windows are rejected by simpler classifiers, and the reduction in false positive rate is achieved by using more complex classifiers.

AdaBoost is used to create stages for training classifiers. The first stage is comprised of a strong two-feature classifier. In order to build an effective face filter the strong classifier threshold is adjusted to minimize false negatives. A lower error rate on the training data is obtained by using the initial AdaBoost threshold, $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$. This lower threshold yields higher detection rates and higher false positive rates. Viola and Jones obtained an accuracy of 100% with a false positive rate of 50% using the two-feature classifier. A cascade of filters works such that a positive result from the first classifier triggers a second classifier and a positive result from the second classifier triggers a third classifier, and so on. A negative result at any stage leads to the rejection of the sub-window. This process is described in the Figure 4.2. Therefore, a cascade is designed such that more and

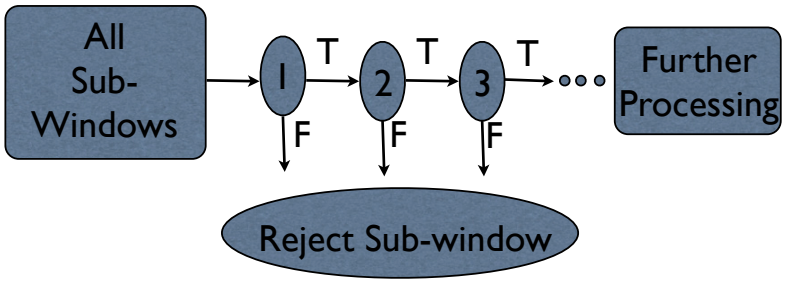


Figure 4.2: An example of a cascade of classifiers used by Viola and Jones

more negative examples are rejected and the positive examples pass through subsequent classifiers. Viola and Jones based the design of this cascade of filters on an optimization framework such that the number of classifier stages, the number of features in each stage and the threshold of each stage,

are traded off to minimize the expected number of evaluated features. Viola and Jones used a total of 38 stages with over 6000 features in their cascade detector and still had fast average detection times.

4.3 Recent Face Detection Techniques

In section 4.1, we discussed the face detection techniques prior to the year 2002 followed by the Viola and Jones face detector because of its influence in face detection research. I will now discuss recent face detection research in the context of some of the challenges, including changes in pose, occlusion, facial expressions, speed, image orientation and image conditions as listed by Yang et al. [YKA02].

4.3.1 Pose and Image Orientation

Detecting frontal poses under controlled conditions is more or less a solved problem. However, variations in image conditions make face detection much harder. As a result many researchers have been pushing different techniques to overcome such challenges. Viola and Jones extended the frontal face detection framework to handle profile and rotated faces [JV03]. They trained different detectors for different views of the face and used a decision tree to determine the profile and the rotation. However, they got a lot of false rejects and false positives.

Wu et al. [WAHL04] used a parallel cascade for multi-view face detection. They learned a classifier for each view. Although the performance was good, the set up is too slow for practical use. Li et al. [LZZ+06] used a 3-level pyramid cascade to improve the design of Wu et al. for better speed and performance. In the pyramid, the first level works on all poses, the second level detects left profiles between -90° and -30°, between -30° and +30° for the frontal views and the right profiles between +30° and +90°. The third level of the pyramids detects faces at 7 finer angles. Although this design shows much better performance, it still requires a lot of improvement.

Huang et al. [HALL05] designed an efficient tree-structured Multi-View Face Detector (MVFD), that divides the entire face space into smaller and smaller subspaces. Using this approach they can deal with pose changes that cover $\pm 45^\circ$ rotation in plane (RIP) and $\pm 90^\circ$ rotation off plane. This

approach requires only four such detectors to take care of a rotation independent MVFD. However, this method requires manual categorization of the training data and can be a tedious task.

Anoop et al. [AARK08] presented an approach for multi-view face detection in videos. They performed this multiview face detection at certain key positions in the video instead of at every position in a frame. These key positions are based on statistical samples drawn from a density function whose estimation is based on color cues, past detection results, meanshift tracker results and a temporal continuity model.

4.3.2 Image Conditions and Facial Expressions

As has been previously discussed in Chapter 1, face detection under uncontrolled lighting is very difficult. On top of that, changes in expressions further complicate detection. Although the major focus of the face detection research has been on pose invariance, some work has been carried out to reduce the impact of the image lighting conditions and incorporate different facial expressions. Hsu et al. [HAMJ02] proposed a face detector using a novel lighting compensation technique and a nonlinear color transformation to detect skin regions over the entire image. This was followed by generating face candidates based on the spatial arrangement of these skin patches. The skin color is modeled using a parametric ellipse in a two dimensional transformed color space and facial features are extracted by constructing feature maps for the eyes, mouth and face boundary. However, the number of false positives using this approach is high.

Xiao et al. [XLZ04] used a three step approach to detect faces in images with complex backgrounds, views, illuminations and facial expressions. They use a linear-filtering algorithm to remove most of the non-faces, followed by a chain boosting algorithm to enhance efficiency. Finally, they use image preprocessing, SVM-filter and color filter to refine the final prediction. They also used an in-plane pose estimator for this real-time system for multiview face detection in photos.

Although these approaches do very well on certain aspects of face detection, most of the systems fail when all these challenges are presented in the same database resulting in lots of false positives and a high number of false rejects. Even if such challenges are overcome, the result is often a system with large computational demands that limit real-time application.

4.3.3 Further Work on Cascade Based Face Detectors

After the success of the Viola and Jones' cascade based face detector many researchers have pushed the idea in different directions. Li and Zhang [LZ04] introduced a new procedure called FloatBoost for learning a boosted classifier for achieving the minimum error rate. FloatBoost is used to bridge the gap between the goal of a conventional boosting algorithm (maximizing the margin) and that of many applications (minimizing the error rate) by incorporating floating search into AdaBoost. This technique allows deletions of weak classifiers that are ineffective in terms of the error rate. It is then effectively used for achieving real-time multiview face detection by incorporating the idea of the detector pyramid learned using FloatBoost.

However, one of the disadvantages of cascading methods is that reducing false positives requires training new stages which reduces the detection rate. In order to overcome such weaknesses of the cascade based systems, Bourdev and Brandt [BB05] proposed a new cascade based system called the "soft cascade". According to the authors cascade systems have several disadvantages that their system addresses. Some of these disadvantages have been reported to be as follows: firstly, at each stage a decision to accept or reject an instance is taken without considering how well the instance performed in the prior stages; secondly, there is a severe training requirement at each stage. A positive classification is required to pass every stage and the final detection rate is the product of the detection rates of all stages, for example, in a cascade with 10 stages, a final detection rate of 90% requires that at each stage the face detection rate should be 99%. This makes it very difficult for the later stages; thirdly, the training parameters such as target detection and false positive rates provide only an indirect control over total execution speed, which requires us to retrain the cascade to test different parameters; and finally, there is no way to pick the parameters like the number of stages, the ordering, the target detection rate and the false positive rate at each stage.

Bourdev and Brandt used a two step approach to overcome some of these disadvantages of the cascade detectors. Firstly, they generalized the decision making function for each stage to be scalar-valued, rather than binary-valued. Secondly, the decision whether a positive instance passes a given stage depends on the values of each of the prior stages, rather than just the value of the

stage under consideration. In this approach, instead of training a sequence of consecutive stages like in the Viola and Jones' approach, Bourdev and Brandt trained a single, potentially very long stage consisting of T features. The resulting classifier can be represented as:

$$H(x) = \sum_{t=1,\ldots,T} c_t(x), \tag{4.4}$$

where $c_t(x) = \alpha_t h_t(x)$ are the set of thresholded Haar based classifiers selected during AdaBoost training scaled by the associated weights. However, in order to reduce the number of false negatives, Bourdev and Brandt proposed a solution that uses a parameterized curve. In an attempt to reduce the number of false negatives this solution also gives up on the positive examples [ZV08].

Mita et al. [MKH05] introduced a new Haar-like feature for face detection. In this approach AdaBoost is used to learn a face detector through stagewise selection of the joint Haar-like features. The authors claim that this detector yields higher classification performance than the Viola and Jones' detector. This detector is 2.6 times faster than Viola and Jones face detector. However, this algorithm has been tested only on the MIT-CMU dataset [RBK98].

Heisele et al. [HSP07] developed a part-based framework by looking for prominent facial components such as eyes, nose etc., and then using their spatial relationship to detect faces. This method is very robust to image deformations and occlusions. However, the choice of feature representations and accurate characterization of the relationships between the facial components is still a challenge.

Nilsson et al. [NNC07] use the local Successive Mean Quantization Transform (SMQT) technique for face detection. Local SMQT is robust to changes in illumination. This approach is an extended version of the Sparse Network of Winnows (SNoW) [FE04] classifier. SNoW is a sparse network of linear units over the feature space [RMA00]. One of the drawbacks of the SNoW-based methods is that they require a large number of face and non-face patches during training.

Jang and Kim [JK08] used an AdaBoost-based cascade detector for face detection. They used an evolutionary pruning to reduce the number of weak classifiers. In order to construct a multiview face detector the authors incorporated frontal, left profile and a right profile face detector. However, they only tested it on the CMU-MIT face dataset and they also suffer from the same problems with

more false positives as other cascade detector based approaches. For such detectors, Brubaker et al. [BWS+08] presented a way to determine the number of hypotheses to include in an ensemble and the appropriate balance of detection and false positive rates in the individual stages. Pakazad et al. [PHF11] use a cascade based face detector on the pattern of the Viola and Jones face detector. At each stage of the cascade a best minimal set of features are selected using a feature selection algorithm. This approach reduces the average number of operations per location. The features that they used are Central Geometrical Moments (CGMs) of face components and their horizontal and vertical gradients. These features were calculated using the Kahan [Kah71] summation algorithm to decrease the effects of round-off error.

4.3.4 Some Other Approaches to Detect Faces

Sznitman and Jedynak [SJ10] used an information theory based approach for face detection and localization. In this approach, general and specific questions are asked to determine the face pose while spanning different face spaces. This approach strictly focuses on frontal faces only.

Tsao et al. [TLL+10] applied a database based approach to face detection. The face detector consists of three cascaded classifiers to prune non faces. It involves data mining the feature patterns of human faces automatically and efficiently. Each training image is converted to an edge image through the application of Sobel edge detection operator, a morphological operator and a threshold. From these edge images they obtain the positive feature pattern by mining the maximal frequent patterns using a database approach called the MAFIA [BCG01] algorithm. This approach not only pays attention to the positive patterns like eyes and mouth but also some of the negative features like the cheeks that do not contain any facial feature patterns.

Wang et al. [WQXL10] used a combination of Haar features and features of skin colors for face detection. This combination approach helps the two methods to draw on each others strengths. In addition, this enables not only an increase in the speed of detection but also enhances its accuracy and minimization of false positives.

Paisitkriangkrai et al. [PSZ11] suggest that instead of using Haar like features, as used in most of the cascade based face detectors, they can use simpler features and still obtain discriminative

information in a more efficient way. They create a feature set using sparse least square regression by simplifying a combination of Histograms of Oriented Gradients (HOG) and Local Binary Pattern (LBP) proposed by Wang et al. [WHY09]. This new block descriptor retains the properties of HOG and LBP descriptors, such as, robustness to illumination changes and image noise, in addition to computational simplicity. Their results are comparable to the ones obtained using Haar features with the advantage that the new features are simpler.

Chapter 5

Batch of Filters Approach to Face Detection

In this chapter I am addressing two challenges of face detection using correlation filters. The first challenge is due to the way correlation works. The output of a correlation between any two images results in a correlation surface and each pixel value in this surface is a correlation value. In this correlation surface there is no inherent information about the location of a face. We start from the location in the correlation surface that has the highest correlation value and because of the way the correlation filters are designed (see Chapter 3), I expect this peak to be the point between the eyes. However, correctly locating the peak only localizes a face. We are interested in detecting a face which entails finding the coordinates of a rectangle that encompasses a face. The correlation peak does not give us any idea about the size of a face rectangle (and hence the scale of the face). The challenge is to go from face localization to face detection.

This challenge of face detection starting from the target localization to face detection is related to our second challenge. The challenge of determining the size of the face. Even though we are experimenting with images containing only a single face, face size varies between images. Therefore, we not only need to find the location of the face in the image but also the scale of the face from one image to another. Rest of the chapter presents the experiments to localize a face and find the scale of the face in the image.

5.1 Face Localization

Face localization is defined as the process of finding a target location in an image. In this dissertation this target is identified as the point between the eyes of a face and its localization acts as a first step for face detection using correlation filters. This approach begins with the convolution of an image with a face and a filter in the Fourier domain. The result of this convolution is a correlation surface in which we identify the coordinates of the highest correlation value i.e., peak. The goal is to obtain a correlation surface that has a high peak value at the target location on the

face and low values everywhere else. Figure 5.1 displays convolution between an image and a filter. This results in a correlation output surface displayed in the last graphic of this figure. It is

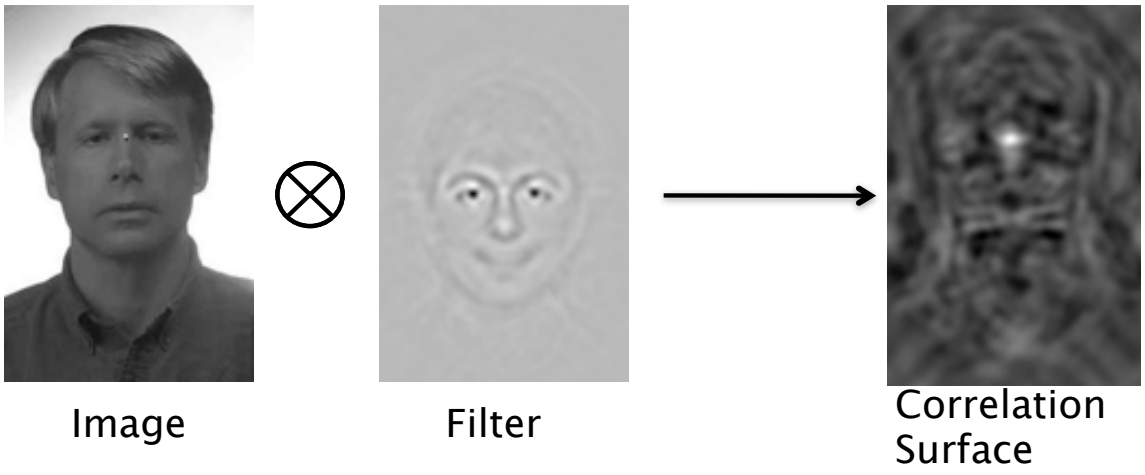


Figure 5.1: Face Localization using convolution of an image with a filter.

clear that there is a bright spot in the correlation surface and the rest of the image is darker than this location. In order to understand this result, I will start with the graphic labeled, Image, in this figure. The target is the location between the eyes. For demonstration, this target is identified by a white dot in the Image. The dimensions of the input image, filter and the output correlation surface are the same. Therefore, if the coordinates of this white spot in the input image matches with the bright spot in the correlation surface, we can confirm that the target has been localized. However, they may not always correspond to exactly the same location. For example, in Figure 5.1, the bright spot location of the correlation surface corresponds to the black spot in the Image which is very close to the targeted white spot. In my experiments if the actual location identified by the black spot in the image lies within a certain range of pixels from the target location it can still be considered a successful localization. I will discuss this in more detail in the experiment design section.

5.2 Experimental Setup

As previously discussed, the correspondence of identifying the target and the actual location is localization of a part of the face that I am trying to detect. This target identification is the first step

to detect a face. The next step is to get the coordinates of a rectangle that contains the face. This transition from localization to face detection, identified by the face rectangle, is not straightforward because the width and the height of a face cannot be determined directly from the peak correlation value. The reason being, there could be multiple rectangles encompassing the face fully or partially as displayed in Figure 5.2. The question is, starting from the target location, which is correct, the

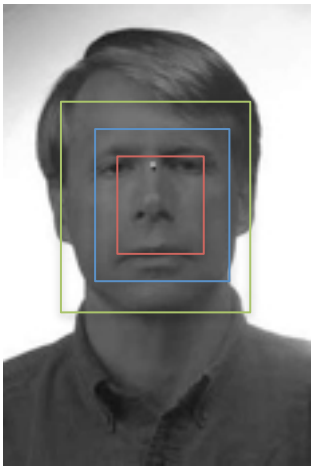


Figure 5.2: Different rectangles that could represent a face after locating the target point between the eyes.

red rectangle, the blue rectangle or the green rectangle. In order to answer this question I will have to find the size of the face which may also be referred to as the scale of the face. While answering this question I am trying to handle just the scale of a face among many other challenges in detecting faces, like pose, lighting, complex background etc. In order to tackle just the face scale issue I will begin with considering only frontal face images with a controlled background.

5.3 Dataset

The images for the scale experiments are taken from the FERET [PMRR00] database. All the images in this database have been taken in the same physical setup controlling for background and lighting. There are 2413 still facial images representing 856 individuals. I control for the pose and only include the images with frontal face views with different expressions. This is done so that we have a controlled experiment to study the effect of scale variation between the images without

having to worry about other factors. Some of the representative pictures from the database are displayed in Figure 5.3.

(a) Image 1 (b) Image 2 (c) Image 3

(d) Image 4 (e) Image 5 (f) Image 6

Figure 5.3: Representative images from the FERET database.

I have used 250 training and 250 test images with no overlap for people between these two datasets. These two datasets consist of images with faces of different scales determined by the interocular width. These two datasets will be referred to as the unprocessed training set and the unprocessed test set, respectively. As described in Section 3.2.2, training involves the generation of an exact filter for each training image and minimizing the sum of squared error to obtain the final filter. This correlation filter is used to correlate with the test images, one at a time in the frequency domain, while exploiting the convolution theorem. The coordinates of the eyes, nose and the mouth for each image are known. I begin with localizing a target on the face and in all

the experiments this target is the point between the eyes. The goal is that correlation surface has the highest peak in each of the test images. We can compare the coordinates of this located point with the ground truth coordinates of the point between the eyes to measure accuracy of this target localization. The target is accurately located if it lies within five pixels from the ground truth coordinates in both directions.

5.4 Experiments

In order to understand the effect of scale on the accuracy of finding the target point in an image I have designed three different experiments which are described in this section. The results of these experiments eventually lead from face localization to the face detection.

5.4.1 Same Interocular Width Between the Test and the Training Datasets.

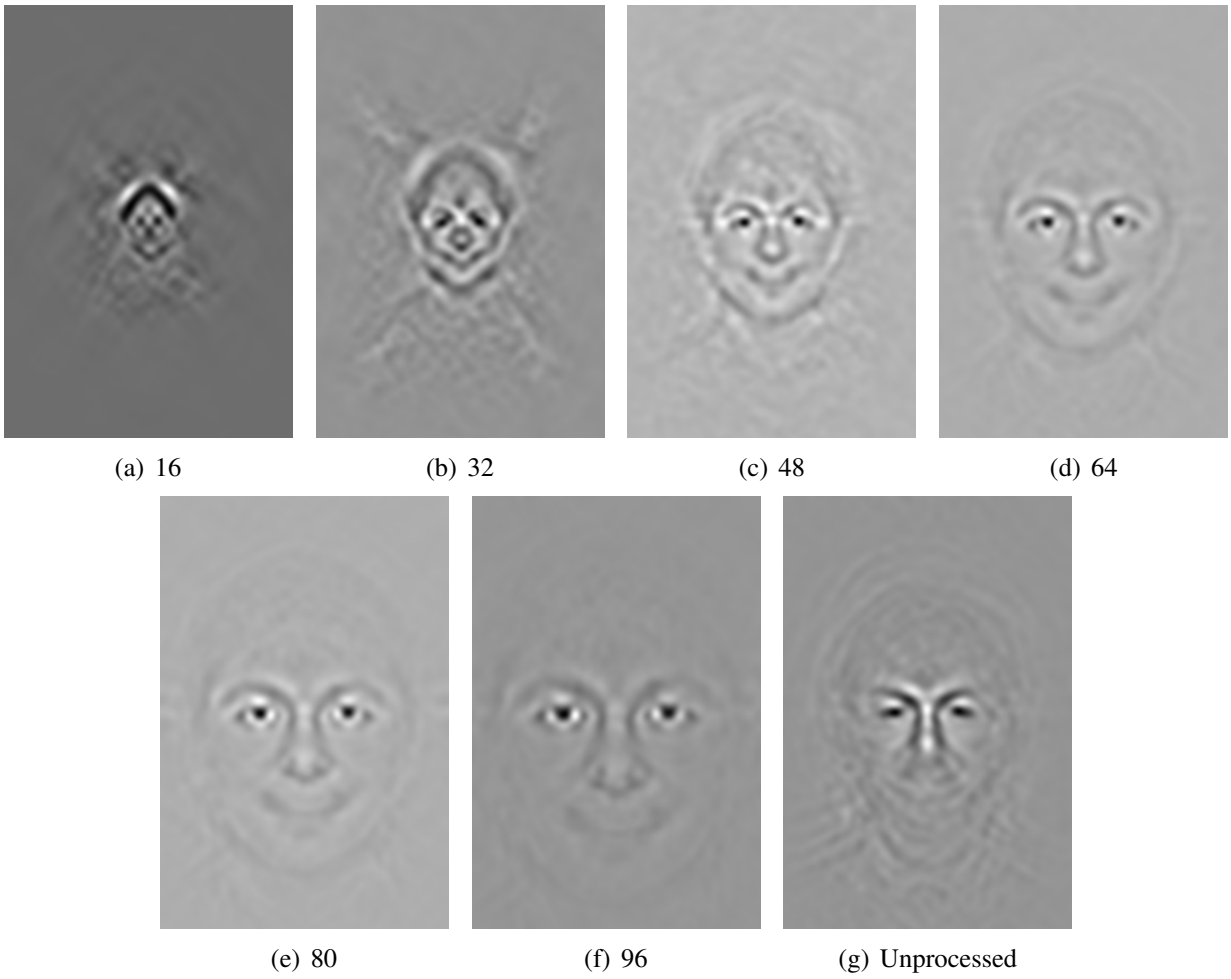
My first experiment is very simple. The training and the test datasets have the same scale defined by the interocular width. This means the interocular width between the training and the test images is the same. The goal of this experiment is to test the filters trained on images with the same interocular width. The hypothesis is that if the training and the test images have the same face sizes, we will have a high accuracy.

5.4.1.1 Training and Test Datasets

To be more specific I use the 250 training images from the unprocessed training set. Since the images in this dataset have different interocular widths, I process them to create new datasets of specific scales. More specifically I created six new training sets with 16, 32, 48, 64, 80 and 96 pixels between the eyes. These training sets were created from the unprocessed training set and as such each of them had the same number of images, 250.

The way to create each of these datasets is by sampling each image such that the number of pixels corresponds to a given pixel width. For e.g., the training dataset consisting of images with 16 pixels between the eyes is created from the same unprocessed training dataset of 250 images such that the pixel width of each of these 250 images is 16 pixels between the eyes. The same

procedure is repeated to create datasets for 32, 48, 64, 80 and 96 pixels between the eyes. The filters trained from these datasets are presented in Figure 5.4. The filter labeled unprocessed is the



(a) 16 (b) 32 (c) 48 (d) 64

(e) 80 (f) 96 (g) Unprocessed

Figure 5.4: Each filter is labeled as the number of pixels between the eyes in the training dataset .

one trained on the images in the unprocessed training dataset.

5.4.1.2 Results

The results of Experiment 1 are presented in Table 5.1. In each of these tests a target is considered to be accurately located if the difference between the target and the actual location is within five pixels in both x and y coordinates. As we can see each of these numbers in column 2 of Table 5.1 display good accuracy. However, this simple experiment confirms my hypothesis that correlation filters trained and tested on images of similar interocular widths will yield good accuracy results. This also means that if the scale of faces between the test and the training images is

Table 5.1: Accuracy of locating point between the eyes with filters trained and tested on the images with the same number of pixels between the eyes.

Pixels between Eyes	% Accuracy (within 5 pixels)
16	78.8
32	72.4
48	87.6
64	95.2
80	96.4
96	97.2
Unprocessed	84

the same there is always a very good chance of localizing a face accurately. After localizing a face, the scale of the face can be easily determined because the filters have been trained for a particular scale. However, having verified the hypothesis it is highly unlikely that a test dataset will have images with the same number of pixels between the eyes as the ones in the training set. Therefore, it would be useful to determine the response of the filters that are tested on a dataset which could have any number of pixels between the eyes. However, before I proceed with that experiment, it is important to determine how far can a single filter stretch. I will discuss this in detail in the next experiment.

5.4.2 Single Filter is not Enough

In the previous experiment it has been established that a single filter is good when the training and the test face scales are the same. However, since that situation is not likely, I want to run an experiment to determine the range of scales over which a single filter would correctly localize a face and hence provide its scale. This is being done for two reasons: firstly, to determine whether a single filter can be used across a dataset containing a range of interocular widths, and secondly, if it can't be used, how many filters do we really need for a test set to cover a range of face sizes.

The process of creating datasets that we are using for this experiment are created in the same manner as the ones in the previous experiment. The accuracy of the results are still computed based on the difference between the target and the actual located point and for a correct localization this difference in both directions should still lie within 5 pixels. In line with the scale space theory of

the linear filters [FP02], the scale of the images in my experiments will be represented as octaves. Each octave is a power of 2 and hence a filter trained on a dataset with each image having 16 pixels between the eyes will be referred to as an octave 4 (=2^4) filter.

5.4.2.1 Training and Test Datasets

In order to study the range of face sizes that a particular filter trained for given scale can handle with an accuracy of over 90%. I have used six different training sets, comprising of octaves 4, 5, 5.58, 6, 6.32, and 6.58. Each of these training sets is trained on 250 images. The interocular width between the six training sets is different but the images have the same interocular width within a training dataset. Each of these filters is tested on a range of octaves around the training octave. For example, an octave 4 trained filter is tested on 13 different datasets corresponding to octaves around the training octave. Each of these 13 test datasets have the same number of images, 250. The range of octaves tested using an octave 4 filter varies from octave 3.32 (10 pixel) interocular width to octave 4.46 (22 pixel) interocular width. Each test octave has images that have the same interocular width. The same approach of training and testing is applied to rest of the octaves.

5.4.2.2 Results

The results for various training and test set ups using a single training filter are presented in 6 tables and 6 figures. Each table and figure corresponds to a given training set and 13 test sets around the training set interocular width. The tables present the numbers corresponding to the graphs showing the visual changes in the accuracy results. I will start with Table 5.2 and Figure 5.5. The latter clearly displays that for a filter trained for octave 4 (see Filter in 5.4(a)) interocular width, the range of test image interocular widths over which it can get an accuracy greater than 90% is limited to the test images of the same interocular width. That does not make it very useful. So that can't be our single filter of choice for the entire range of scales.

Now, let me consider the filter trained for octave 5(see filter in Figure 5.4(b)) interocular width training dataset. It is tested over test datasets ranging between octaves 4.7 and 5.25. Table 5.3 represents the empirical accuracy values of the test sets and Figure 5.6 displays these results graphically.

Table 5.2: Accuracy of locating point between the eyes around octave 4 interocularwidth.

Test Octave	% Accuracy (within 5 pixels)
3.32	2
3.45	3.6
3.58	16
3.70	28.4
3.80	52.8
3.90	70.8
4.0	91.2
4.08	72
4.16	62.8
4.24	40.8
4.32	28.8
4.39	14.8
4.46	5.2

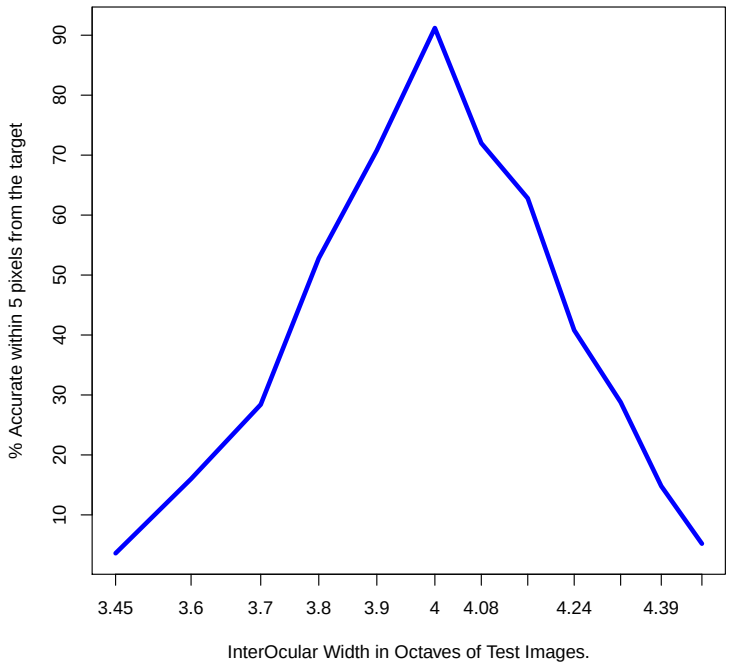


Figure 5.5: Accuracy of locating the target within 5 pixels when a filter is trained on 4 octave and tested on 7 datasets containing images between octaves 3.45 and 4.46.

The accuracy results are not high but that is not what I am interested in this experiment. The most important aspect that I want the readers to look at, is the change in accuracy values that happens as the scale of the faces changes from the training dataset scale. Contrary to the rate of change of accuracy for octave 4 results, here one can notice that the accuracy values for octave 5 filter does not drop so drastically and there seems to be somewhat gradual drop. Neverthless, the change is still drastic enough to continue with the same story line that even this filter cannot be used over a large range of scales. One can argue that we could use it to test datasets for a scale change of 0.05 octaves from the training octave, but it is still not large enough to justify the use of a single filter over a larger range of scale change.

Table 5.3: Accuracy of locating point between the eyes around octave 5 interocularwidth.

Test Octave	% Accuracy (within 5 pixels)
4.7	10.4
4.75	18.8
4.8	34.4
4.85	51.2
4.9	62
4.95	71.2
5	77.6
5.04	69.2
5.08	64.8
5.13	59.2
5.17	48.4
5.2	40
5.25	28

The next transition for the training dataset to study the effect of scale is not one octave like we did by going from octave 4 to octave 5. This training filter (see Figure 5.4(c)) is trained using images of octave 5.58 interocular width. The reason is that the trained filters don't show good results over even small change of scale in the test images. The results for test datasets with scales between the range of 5.39 and 5.75 octaves are presented in Table 5.4 and Figure 5.7. It becomes quite apparent that although the range of scales over which the same filter can be used without a drastic drop in the accuracy values, this range of scales is still not large enough that it would be

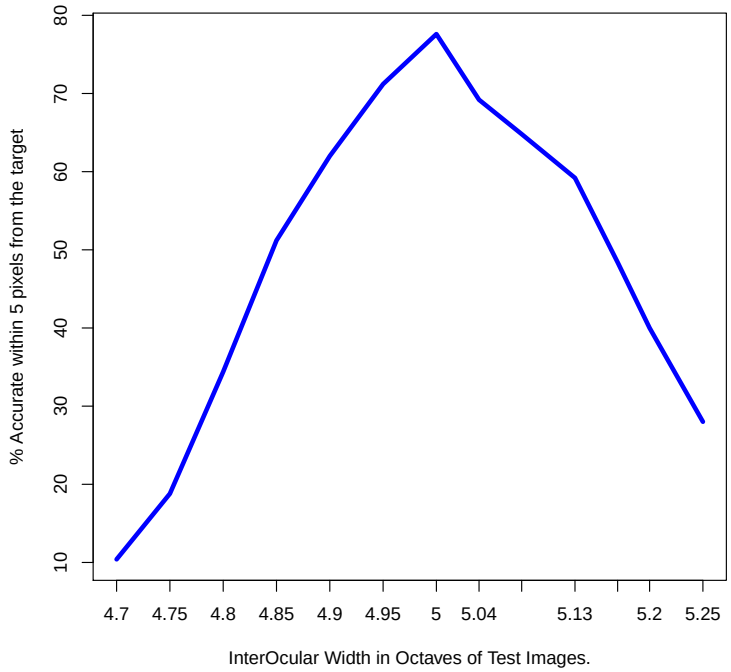


Figure 5.6: Accuracy of locating the target within 5 pixels when a filter is trained on 5 octave and tested on 13 datasets containing images between octaves 4.7 and 5.25.

used for the entire dataset.

Table 5.4: Accuracy of locating point between the eyes around octave 5.58 interocularwidth.

Test Octave	% Accuracy (within 5 pixels)
5.39	41.2
5.42	58.4
5.45	66
5.49	76.4
5.52	83.2
5.55	86
5.58	87.6
5.61	86
5.64	84.4
5.67	78.4
5.70	74
5.72	66
5.75	61.2

That brings us to filter trained on a dataset with octave 6 (see Figure 5.4(d)) interocular width images. It is used on test datasets ranging between 5.85 and 6.13. Between octave 5.90 and 6.13, which is presented in Table 5.5 and Figure 5.8, there is a drop of 8% in accuracy which is not a significant drop over a quarter octave. This also means that I could use a single filter to test images within a scale of quarter octave of the training dataset.

For images trained on octave 6.32 (see Figure 5.4(e)), the test datasets are spread over a scale of 6.20 to 6.42 octaves. The results for these datasets are presented in Table 5.6 and graphically represented in Figure 5.9. In almost a quarter of an octave in the change of scale between the test sets and the trained filter there is only a drop of 2.8% in accuracy. This result also confirms that a filter can be used over a quarter of scale change. This kind of result helps determine the number of filter that should be used to cover the entire range of test datasets.

For my final filter trained on images with 6.58 octave (see Figure 5.4(f)) interocular width the results are presented in Table 5.7 and Figure 5.10. The drop in accuracy is only about 2 % over one sixth of an octave. This confirms the use of the same filter over a range of scales around the training dataset scale.

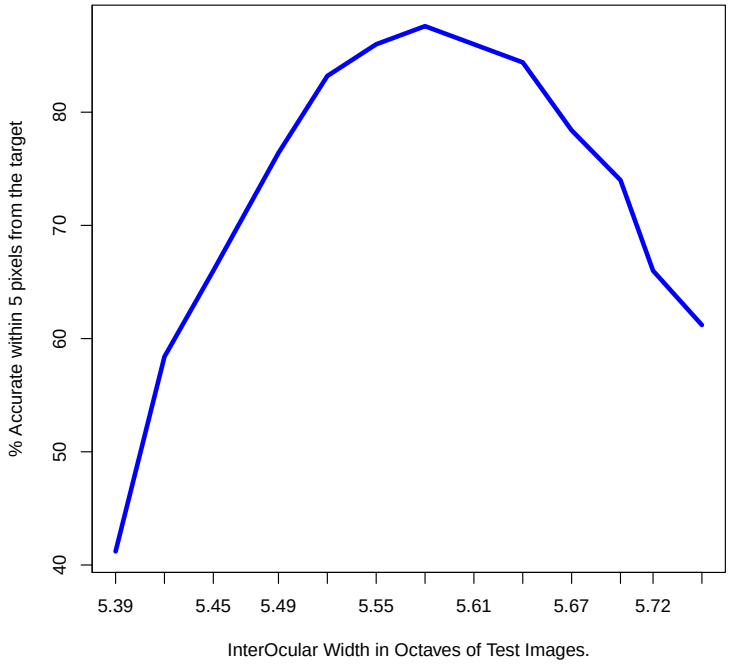


Figure 5.7: Accuracy of locating the target within 5 pixels when a filter is trained on 5.58 octave and tested on 13 datasets containing images between octaves 5.39 and 5.75.

Table 5.5: Accuracy of locating point between the eyes around octave 6 interocularwidth.

Test Octave	% Accuracy (within 5 pixels)
5.85	73.2
5.88	80.8
5.90	87.2
5.93	91.2
5.95	92.8
5.97	93.6
6.0	95.2
6.02	95.2
6.04	94.4
6.06	93.2
6.08	91.6
6.10	90
6.13	87.6

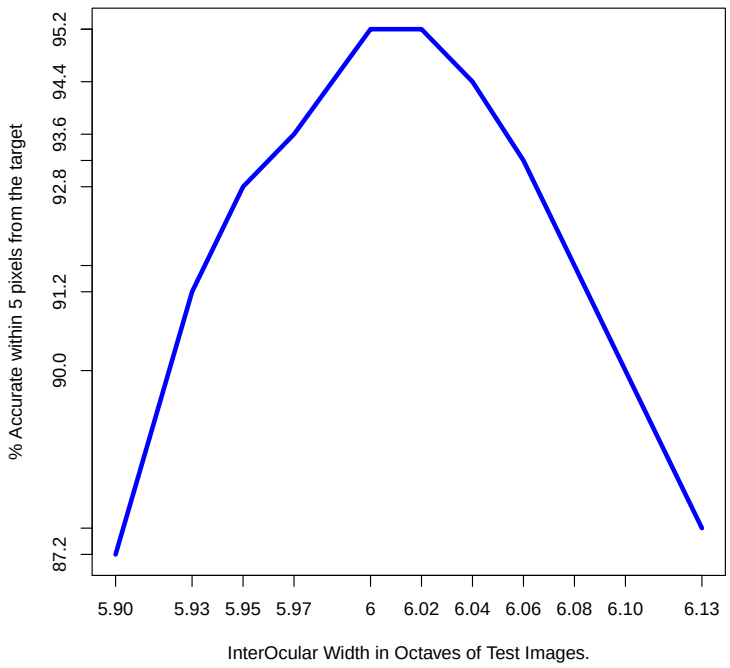


Figure 5.8: Accuracy of locating the target within 5 pixels when a filter is trained on 6 octave and tested on 13 datasets containing images between octaves 5.85 and 6.13.

Table 5.6: Accuracy of locating point between the eyes around octave 6.32 interocularwidth.

Test Octave	% Accuracy (within 5 pixels)
6.2	93.6
6.22	94.8
6.24	95.6
6.26	96
6.28	96.4
6.30	96
6.32	96.4
6.34	95.6
6.35	95.6
6.37	95.2
6.39	94
6.40	94.4
6.42	94

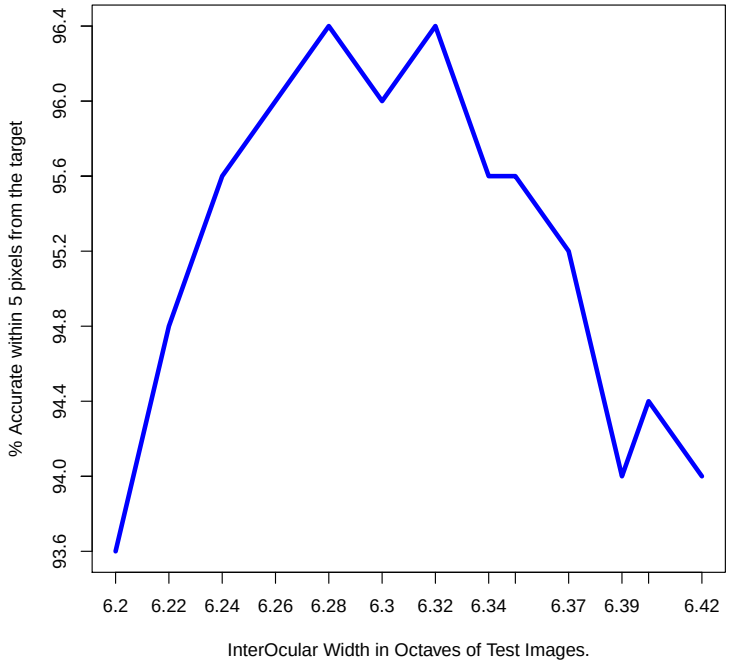


Figure 5.9: Accuracy of locating the target within 5 pixels when a filter is trained on 6.32 octave and tested on 13 datasets containing images between octaves 6.2 and 6.42.

Table 5.7: Accuracy of locating point between the eyes around octave 6.58 interocular width.

Test Number of Pixels	% Accuracy (within 5 pixels)
6.49	94.8
6.5	96
6.52	97.2
6.54	96.4
6.55	97.2
6.57	97.2
6.58	97.2
6.60	96.8
6.61	96.8
6.63	96.4
6.64	96
6.65	96.8

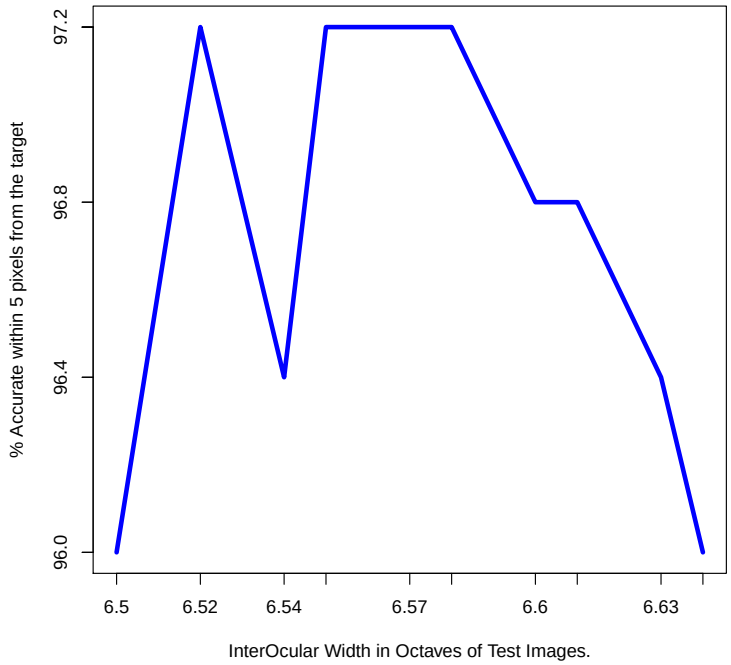


Figure 5.10: Accuracy of locating the target within 5 pixels when a filter is trained on 6.58 octave and tested on 13 datasets containing images between octaves 6.5 and 6.64.

5.4.2.3 Conclusion

All the plots in Figures 5.2 to 5.7 show a drop in the accuracy as the interocular width in the test images drops beyond a certain scale. Smaller scales show a big drop in the accuracy when testing images within a quarter of an octave while the larger ones show a smaller change. However, that is not the most important learning from the experiment. These experiments were conducted over many scales to verify the hypothesis that no single filter is good enough to maintain a consistent accuracy over a large range of scales in the test dataset. As such there is no single filter that can be used across a dataset. We will need more than one filter. The study of the results for the scales does indicate that the filters trained over larger scales can be used over test scales of a quarter of an octave around the training interocular width but we will need to train more filters for smaller scales.

5.4.3 Test Images of Unprocessed Test Data

From the previous experiment it has been established that more than one filter is needed to cover all the scales of a test dataset. It also became clear that we need more filters for test images containing faces of smaller scales for e.g., 4 to 5.25 octave and slightly fewer for the larger scales for which a filter could be trained to scale a quarter of an octave change in the test images. However, using more than one filter for a particular image poses another challenge. Out of many filters that we use on an image, how do we pick the one that best matches a test image and hence find the scale of the face in an image. In this section my first goal is to establish a methodology to pick a single best match filter among many filters that will help determine the scale of the face in an image. The second goal of this system is to test filters on an unprocessed dataset where I will not control the interocular width of the images, unlike in my previous two experiments.

5.4.3.1 Training and Test Datasets

Having noticed that filters cover a limited range of scales in the test datasets, I will be using filters trained on datasets whose interocular width ranges between 3 and 6.7 octaves. There are a total of 13 filters being used, which are listed in the first column of Table 5.10. Each of these filters

is trained on a corresponding dataset having all the images of the same interocular width just like the way we have used them in the previous experiments. Some of the trained filters are presented in Figure 5.4. The test dataset for this experiment is different from the previous experiments. I use the unprocessed test dataset. Just to reiterate, this dataset contains 250 images between interocular widths of 5.32 to 6.7 octave.

5.4.3.2 Peak-to-Sidelobe-Ratio (PSR)

In order to find a filter that best matches an image we use PSR [SKK02] as an evaluation metric. Before getting into the details of this metric let me use an example to explain its need. Consider the results presented in Table 5.8. In this table the test dataset is unprocessed which means we

Table 5.8: Accuracy of locating point between the eyes with filters trained on images with different octaves of pixels between the eyes and tested on a dataset with random number of pixels between the eyes.

Training Pixel Octave	% Accuracy (within 5 pixels)
4	0.4
5	2.8
5.58	9.2
6	81.2
6.32	62.4
6.58	29.2

do not control the interocular width. Using filters between octave 4 and 6.58 the accuracy values

Table 5.9: Accuracy of locating point between the eyes with filters trained on images with different number of pixels between the eyes and tested on a dataset with random number of pixels between the eyes.

Training Number Pixels	% Accuracy (within 5 pixels)
Unprocessed	84
Best	92.4

vary from 0.4% for filter trained on images with 4 octave pixels between the eyes and 62.4% for the filter trained on unprocessed dataset. If I use filter 5.4(g) trained on images that have not been

controlled for a specific interocular width the accuracy is 84% as shown in Table 5.9. However, the most interesting row in this table is the one called the 'Best' with an accuracy of 92.4%. In order to determine the best filter one has to understand that each image is tested using a batch of filters. This implies that each test image is tested against all the trained filters that have different octave pixels between the eyes and they are listed in the first column of Table 5.8 plus the unprocessed filter from Table 5.9. Out of this batch of filters the filter that has the highest Peak-to-Side-lobe Ratio (PSR) is picked to be the best filter and as such gives better accuracy than any of the filters used individually across the whole test set.

PSR measures the peak to the sharpness of the image and can be demonstrated using Figure 5.11. The procedure to compute PSR of the correlation output is to first find the peak value

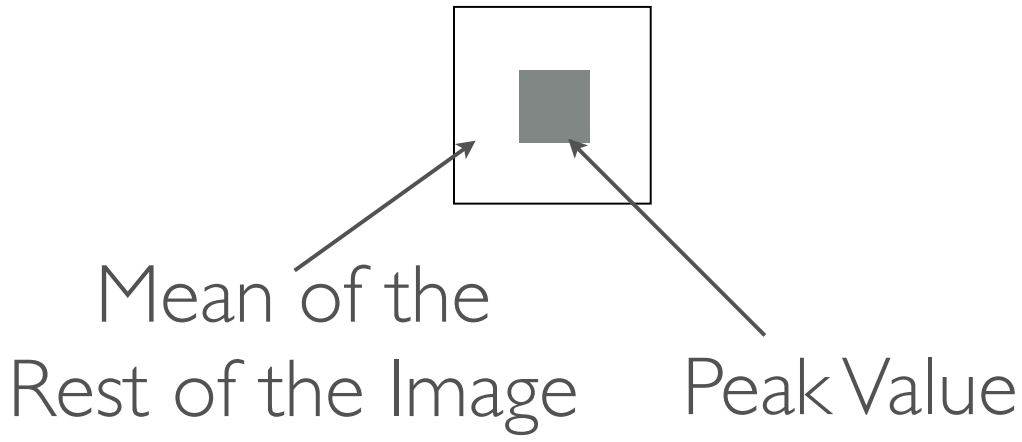


Figure 5.11: Peak to Side Lobe Ratio

and its coordinates in the correlation output image between a filter and the test image. Secondly, compute the mean and the standard deviation of the rest of the image leaving a window of 5×5 pixels around the peak and then determine the Peak-to-Sidelobe (PSR) to be defined as:

$$PSR = \frac{peak - mean}{\sigma}, \tag{5.1}$$

where $peak$ is the maximum value in the correlation surface, $mean$ and σ are the mean and the standard deviation of the correlation surface image when a 5×5 pixel window around the maximum value is left out. The 5×5 pixel window around the peak value is shown to be the dark rectangle in

Figure 5.11. It is pertinent to mention here that the authors in [SKK02] did not use the entire image outside of the central window to measure the mean and the standard deviation to compute the PSR value. Rather, they used another window around the central window of 64×64 pixels, while still not using the central 5×5 pixel to compute the mean and the standard deviation. In this research, I tried the approach of [SKK02] , but the accuracy results were better when the full image is used instead of just the 64×64 pixel window. The 5×5 pixel window around the correlation peak is still not used for the computation of mean and standard deviation.

5.4.3.3 Results

Having established the metric to find the best filter for an image out of a batch of filters, I now used some more filters to find whether it has any affect on the accuracy results. These results are presented in Figure 5.10. While the first and the second column have been used before the third column gives an idea about the percentage of images in the dataset that have the same interocular width as the corresponding training dataset. It turns out that only 16.8% of the 250 test images actually have an interocular width that matches the interocular width of any of the training datasets.The rest of the images, 83.2%, in the test dataset have an interocular width between these octaves. In this example, interocular width of the trained filters varies from 3 to 6.7 octaves.

Using PSR values to determine the filter that gives the best match from among the batch of filters in Table 5.10 and the unprocessed filter from Table 5.11, I can get an accuracy of 95.2% up from 92.4% using the filters from Tables 5.8 and 5.9.

5.4.3.4 Discussion

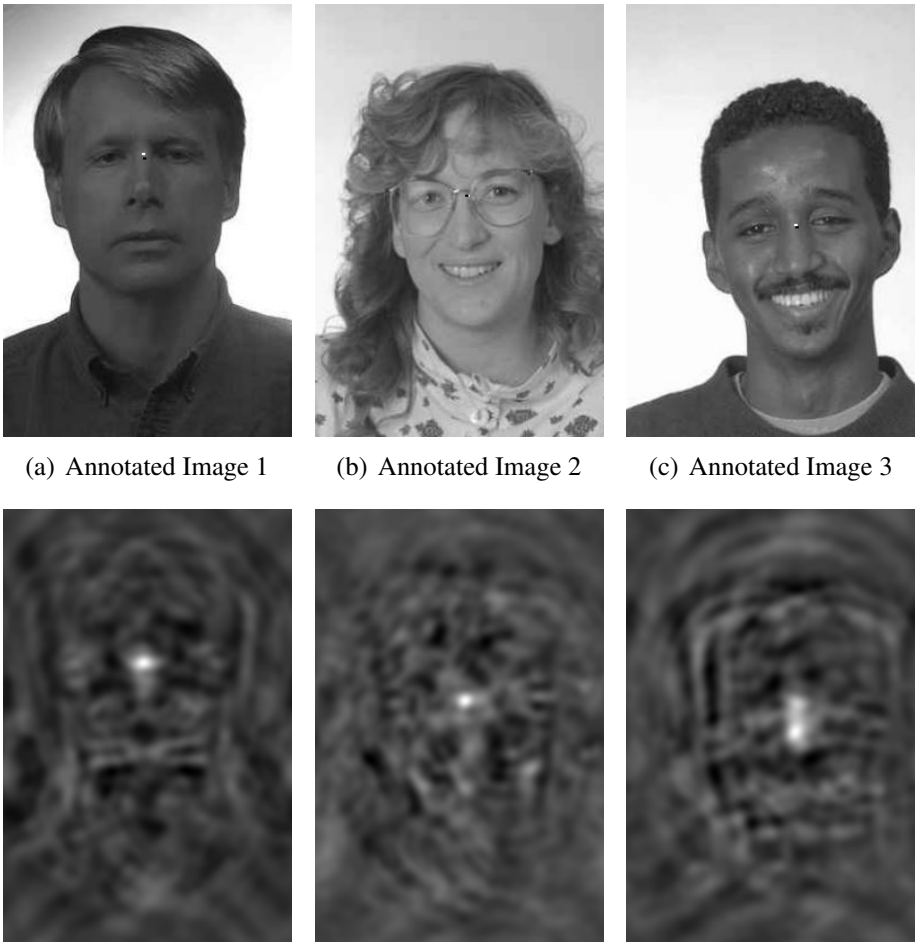
So one of the filters used in this experiment finds the target correctly in 95.2% of the images. This is the location of the peak value resulting from the correlation between an image and the best filter.For some of the images this accuracy of localization is shown in Figure 5.12. Figures 5.12(a) - 5.12(c) show the annotated images displaying the actual target to be located and the located point as a result of convolution. My target is the point between the eyes represented by a white dot in each of these images. The point that is located through correlation between the test image and the best filter is identified by the black dot in these images. As can be seen these two dots are very

Table 5.10: Accuracy of locating point between the eyes with filters trained on images with different number of pixels between the eyes and tested on unprocessed dataset.

Training Pixel Octave	% Accuracy (within 5 pixels)	% $\frac{k}{N(=250)}$
3	0.0	0
4	0.4	0
4.58	8.8	0
5	2.8	0
5.32	2.4	0
5.58	9.2	0
5.8	38.8	1.6
6	81.2	6.0
6.17	85.2	7.6
6.32	62.4	1.6
6.46	38.4	0
6.58	29.2	0
6.7	18.8	0

Table 5.11: Accuracy of locating point between the eyes with filters trained on images with different number of pixels between the eyes and tested on unprocessed dataset.

Training Pixel Octave	% Accuracy (within 5 pixels)
Unprocessed	84
Best	95.2



(a) Annotated Image 1 (b) Annotated Image 2 (c) Annotated Image 3

(d) Correlation Output 1 (e) Correlation Output 2 (f) Correlation Output 3

Figure 5.12: Images showing results displaying the annotated images in the top row along with the correlation surfaces for the corresponding image.

close to each other in these images which implies a good accuracy. The second row of Figure 5.12 displays the correlation surfaces when the corresponding image in the top row is correlated with the best trained filter. The bright spots in Figures 5.12(d) - 5.12(f) identify the peak value and hence the located point represented as the black dot in the corresponding images in the top row.

Once a point is localized in the image, the next step is to find the coordinates of the face rectangle in that image. This requires the scale of the face be computed. The scale is obtained as a result of the process being used. I use a batch of filters and the best filter match corresponds to the scale of the face. This implies that if a filter trained with images of six octave interocular width results to be the best filter than it is more likely that the pixels between the eyes in the test image can be assumed to be six octave wide (=64 pixels). This width can then be used to obtain the width of the face rectangle which has been empirically determined to be $1.25\times$scale (interocular width). The height of the face is usually between 1.50 and 1.75 $\times$ scale.

5.4.3.5 Conclusion

A filter trained on images of larger interocular width has better chances of giving good accuracy around a larger range of interocular widths in the test images. However, there still is no single filter that can consistently give good accuracy over a wide variation in the interocular width in the test sets. Therefore, there is a need for a batch of filters. This requirement gives rise to the need of identifying a filter that yields the best results for a particular image from this bank of filters. I have used PSR to find the best filter from a batch.

5.5 Batch of Filters Versus Rescaling Images

One of the unanswered questions so far has been the choice of using a batch of filters rather than using a single filter and rescaling the images instead. Since I have discussed a batch of filters approach in detail in the previous experiments, in this section I will explain the rescaling of the images approach and the reasons for preferring the former approach over the latter.

These two approaches are presented side by side in Figure 5.13. The approach on the left shows our approach that I have been pursuing. The approach on the right is the one where I can

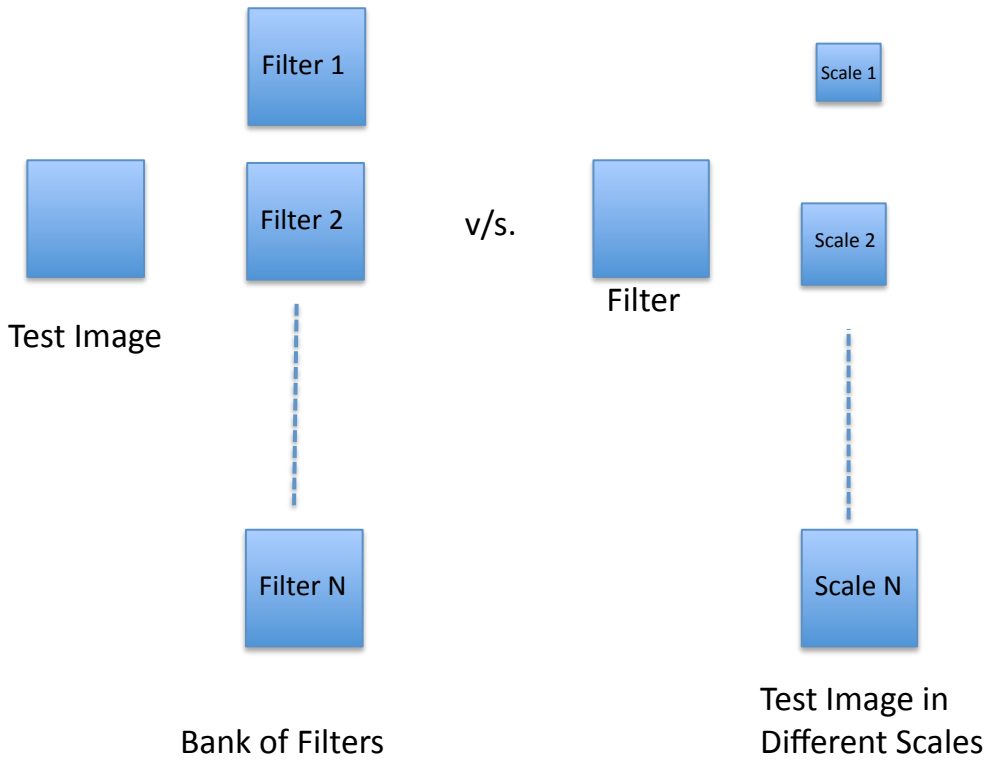


Figure 5.13: Batch of filters versus rescaling images.

train a single filter and rescale the image to locate faces of different scales. The rescaling of the images requires that whenever I want to test a different scale I need to first rescale the image in the spatial domain and then compute DFT on the rescaled image. This will increase the computation time complexity of the system. Contrary to this approach using a batch of filters requires me to compute DFT only once on the test image. It is pertinent to mention here that correlation in the Fourier domain requires that the filter and the image be of the same size. Therefore, if the image after rescaling is smaller than the filter, I will need to pad the image with zeros to make it the same size as the filter. On the other hand I will not need to pad the filters when using a batch of filters on a test image because all the filters are trained on the same size images after the transformation to customize the number of pixels between the eyes.

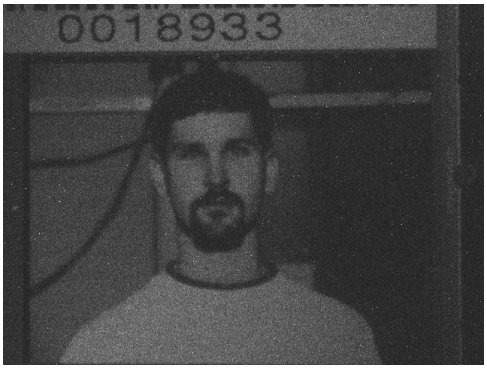
5.6 Face Detection on Hard Datasets

As previously mentioned in Chapter1, I participated in a face detection competition [PWH+11] organized in conjunction with the International Joint Conference on Biometrics (IJCB '11). Twelve participants participated in this competition out of which four algorithms were commercial and one being Viola Jones face detector. Rest of the algorithms came from universities.

The dataset for this competition was divided into 4 subsets. Some of the images were rephotographed and some were semi-synthetic. These images were taken under low light, some had atmospheric blur and were captured at the distances of 3m, 50m, 80m, and 200 m. A single representative image of each of these datasets is shown in Figure 5.14. The labels under each image in this figure represents the distance in meters from which these images or the synthetic heads were rephotographed. The reason for introducing this dataset is to present the results of using a correlation based face detector on a contemporary dataset and compare the performance with other face detectors in a very formal setting. This dataset also contained non-face images. The scales of the images within a dataset did not vary much and an approximate face width of each of these datasets was provided as part of the competition.

(a) 200m

(b) 80m



(c) Dark-3m

(d) 200m

Figure 5.14: Images from IJCB '11 Face Detection Competition [PWH+11].

5.6.1 Training and Test Datasets

The organizers picked 4 labeled training datasets belong to each of the subsets. Each training contained 50 randomly picked images. I started with training filters the provided images, however, the these images were not enough to train a correlation based filter. Therefore, I used PIE database [SBB03] because of the wide range of images present in this dataset. It has 41,638 images of 68 different people, with each person having an image under 13 different poses, 43 different illumination conditions, and with 4 different expressions. This seemed ideal to train filters with enough variation so that they could be used to test the images of the competition. Some of the filters that were trained have been shown in Figure 5.15. All the images for a particular subset have approximately the same face scale. Therefore, I did not have to train for more than the four scales corresponding to each subset. The four test sets were randomly chosen containing 200 images each and these images were randomized. I used a batch of filters approach to test each image using the trained filters, in the same manner as discussed in Section 5.4.3. The procedure to find the scale of the face was again based on the PSR value when an image is convolved with a batch of filters.

5.6.2 Results

The results of each algorithm have been evaluated on the basis of an F-score which is defined in Parris et al. [PWH$^+$11] as,

$$F(precision, recall) = \frac{2 \times precision \times recall}{precision + recall}, \tag{5.2}$$

and higher the F-score, better the performance of the algorithm. On two datasets represented by 5.14(a) and 5.14(b), my algorithm performed the worst, and on the dark dataset 5.14(c), only 3 algorithms had a lower F-score than my algorithm. Since these results were not very competitive, I am not going to talk much about them here. However, I will duplicate the results of the paper for the last dataset whose representative image is shown in Figure 5.14(d). These results are shown in Table 5.12. In this table it is quite clear that actually none of the algorithms do well. However, if I look at the performance of my approach in finding the faces in this subset, considered to be the most difficult in the competition, the F-score is higher than nine out of twelve participants.

(a) Frontal

(b) Right Profile

(c) Left Profile

Figure 5.15: Filters Trained Using PIE database.

Table 5.12: This table has been duplicated from the paper [PWH+11]. It displays contributor results for True Positives (TP), False Positives (FP), False Images(FP'), False Rejects (FR), F-Score(F). For details please refer to the paper [PWH+11].

Algorithm	TP	FP	FP'	FR	F
CBED	100	505	184	8	0.248
DEALTE FD 0.4.3	11	120	81	115	0.066
MBMCT	1	45	31	168	0.008
My Approach	27	147	147	26	0.144
PittPatt	0	0	0	200	0
RSFFD	0	1	1	199	0
SIANI	0	98	98	102	0
UCSD MPLab	5	8	8	187	0.047
Comercial A (2005)	5	6	6	189	0.047
Commercial A (2011)	0	0	0	200	0
Comemrcial B	6	156	156	38	0.033
OpenCV 2.1	80	280	152	26	0.286

CBED algorithm and the OpenCV 2.1 have a higher F-score than my correlation based approach, however, my algorithm has fewer false positives than both of them. Because of these numbers my correlation based face detector has been regarded as the most pragmatic on this subset in the competition [PWH+11].

5.7 Conclusion

This section summarizes three learnings of this chapter, such as: the transition from target localization to face detection, procedure and the need to use the peak-to-sidelobe ratio to determine the face rectangle and finally, the reason why a batch of filters is a preferred approach over a pyramid of different scale images.

I used FERET database to study the process to go from face localization to face detection. The images in the FERET database that were used have a controlled pose (primarily frontal) , and controlled background, but the faces are of different scales from image to image. There is no single filter that can be used to find the scales of all the faces in this test dataset. To accomplish this task of finding the face scales we require a batch of filters and a peak-to-sidelobe ratio metric.

Every image of the test set is convolved with this batch of filters and the filter-image convolution that gives the highest peak-to-sidelobe ratio around the peak value in the correlation surface is considered to be the best match. As such, the scale of the filter that corresponds to this best match is the scale of the face in the test image.

This approach proved very effective on FERET images. However, it was not very successful when tested on the datasets of the IJCB face detection competition in general. Notwithstanding those failures, there are images, like the ones shown in Figure 5.14(d), a subset of this competition, on which face detection using correlation filters proved to be competitive with the best in the world.

I will end this chapter by summarizing the reason for using batch of filters over image rescaling. It is important to understand that unlike batch of filters approach, image rescaling requires an extra Fourier transform every time an image requires to be rescaled. I would like to reiterate that Fast Fourier Transform has a time complexity of $O(NlogN)$. In the batch of filters approach since the size of the image remains constant during the testing process and all the filters of various scales are the same size as the image, we require the FFT of an image only once. The only operation that needs to be carried out in the Fourier domain is an element wise multiplication of an image and a filter, which is a linear operation with a time complexity of $O(N)$. Therefore, the batch of filters approach reduces the time complexity from $O(NlogN)$ to $O(N)$ when compared to the rescaling of the images approach.

Chapter 6

Face Detection in Still Images and Videos

As mentioned in Chapter 1 of this dissertation, images are all around us. These images are often captured using hand held cameras and cell phones. Very often we would like to analyze these images, whether it be for social network websites, security scenarios or for organizing family albums. This analysis requires the application of computer vision algorithms. The most important step in this analysis often involves face detection, be it for face recognition, verification or any other biometric task.

The images obtained through the use of hand held cameras and cell phones are not controlled for face scales, pose, lighting and location. All these factors make face detection challenging. In Chapter 5, I addressed one of these challenges, viz., face scale. An approach was presented to find the scale of a face in an image using a batch of filters. Since the purpose was to determine just the scale of the face, this technique was applied to the FERET database because of its controlled pose (primarily frontal) and controlled background. The same approach can be applied to find the scale of a face in other datasets as well .

My goal in this chapter is to address other challenges in face detection, such as, pose, uncontrolled lighting and location. All these variations are present in a recent dataset called the Point and Shoot Challenge (PaSC) [BPB+13]. This dataset is distinctive in that not only does it contain all the factors that make face detection a challenging task but also because all the images have been captured using hand held digital point and shoot cameras. I will begin by explaining this dataset in detail followed by the experiments to address the challenges of face detection using correlation filters and the results thereof.

6.1 Point and Shoot Challenge (PaSC) Dataset

Most of the images that are captured by people everywhere and stored are primarily captured using handheld cameras that include point and shoot and HD cameras. These images require anal-

ysis using computer vision algorithms for applications such as face recognition, however, before that can be accomplished we need to detect faces in an image.

PaSC is a dataset that has been collected to capture the challenges introduced by these popular cameras and cellphones [BPB+13]. These challenges include blurry images, uncontrolled pose, expressions, lighting and location. This dataset contains 9,376 still images of 293 people. These images have been captured to include variations in distance from the cameras, different cameras, locations and poses varying from frontal to non-frontal. This dataset also contains 2,802 videos of 265 people. All the people in the videos are a subset of the still image data set. Some images from the still dataset to display the data variation in terms of scale, pose, lighting and location are presented in Figure 6.1 and some of the frames of the videos of video dataset displaying similar characteristics as that of the still dataset are presented in Figure 6.2.

In this specific sample of images and frames it is quite evident that there are different scales of faces, complex and varied backgrounds, and lighting, and poses. In order to be able to detect faces in these images, I will first address the scale issue using the batch of filters approach. Every dataset has a different range of face sizes, therefore, I will first need to figure out the number of filters appropriate for this type of dataset to find all the scale variations. The details of an experiment to figure out an optimum number of filters and its details are described in the next section.

6.2 Optimum Number of Filters

In line with the batch of filters approach, there is a need to determine a fixed number of filters that can be trained for a test dataset. The goal is that each filter in a batch can be used for a fixed range of scales and together these filters of a batch, cover the entire scale range of the test dataset. In order to do that I ran a set of experiments on PaSC dataset by training filters between octaves 4(=2^4=16 pixels) and 7(= 2^7 =128 pixels), for specific interocular widths. All of these experiments led to similar conclusions, therefore, I am not presenting the results of all of them, except one, viz., octave 6.

I ran a controlled experiment to train a filter on a set of images, resampled to have exactly 64 pixels (octave 6) between the eyes. The resampling was done from the PaSC dataset on images

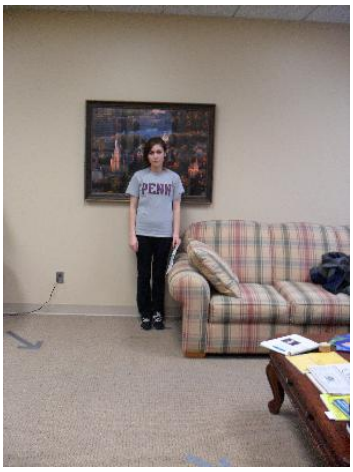
(a) Image 1

(b) Image 2

(c) Image 3

(d) Image 4

(e) Image 5

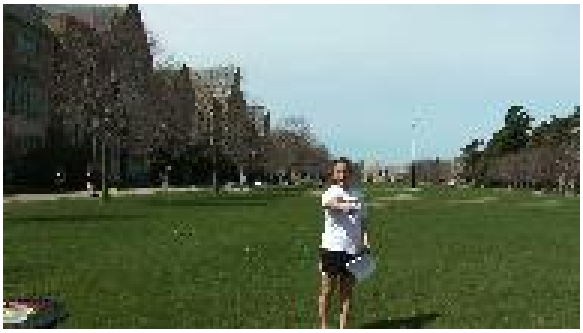


(f) Image 6

Figure 6.1: Representative images from the PaSC still dataset displaying some attributes of the dataset, like, pose, scale, lighting and location.

(a) Image 1

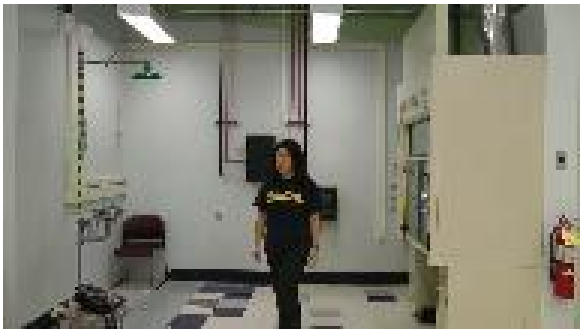
(b) Image 2



(c) Image 3

(d) Image 4

(e) Image 5



(f) Image 6

Figure 6.2: Representative frames from the PaSC Video dataset displaying some attributes of the dataset, like, pose, scale, lighting and location.

that have an interocular width greater than or equal to 64 pixels. Having a filter trained for octave 6, it was successfully used to test twenty one different test sets between octaves 5.5 and 6.5 in steps of 0.05 octaves so that the filter is tested not only on a dataset with images of octave 6 interocular width but also half an octave on either side.

All of these test sets were created in the same manner as described in Chapter 5, by sampling images for a particular test set from the images with an interocular width greater than or equal to the desired interocular width. For e.g., for the test set of octave 5.5 all the images had an interocular width of octave 5.5(= $2^{5.5}$ = 45.25 pixels). These images were created to have the same interocular width by subsampling images from the test dataset that had an interocular width greater than or equal to 5.5 octaves.

Reiterating the purpose of this experiment, the goal is to determine a range of the face scales in the test images that could be recognized using a single filter. The results are reported in the form of a figure showing the accuracy of localizing a target within 10% of the interocular width in both x and y directions. Specifically, the results of a filter trained for images of octave 6 interocular width when tested on 21 different datasets whose interocular widths range between 5.5 octaves and 6.5 octaves, are displayed in Figure 6.3. The x-axis in Figure 6.3 shows the octaves centered around octave 6 which means 2^6(=64) pixels between the eyes and y-axis shows the accuracy corresponding to each of these octaves. To explain these results, let us consider the marker 5.75 on the x-axis and its accuracy on the y-axis equal to 60%. It means, for a dataset consisting of test images having octave 5.75 (= $2^{5.75}$ = 53.81) interocular width, when a filter trained on a dataset containing images with octave 6 interocular width is used, I am able to locate the target (point between the eyes) within 10%(= 5.38 pixels for this test set) in either x or y direction from the target coordinates, in 60% of the images.

Analyzing this figure further, let us now consider a range of test octaves in this figure, specifically, a quarter octave between 5.85 and 6.10. The accuracies corresponding to these test octaves, stays between 70% and 75% which is not a big variation. This consistency of accuracy is very important to understand the range of scales we can test using a given filter. It is apparent that for an octave of 0.25 (5.85 to 6.10) around octave 6 the accuracy is fairly uniform. I ran several other

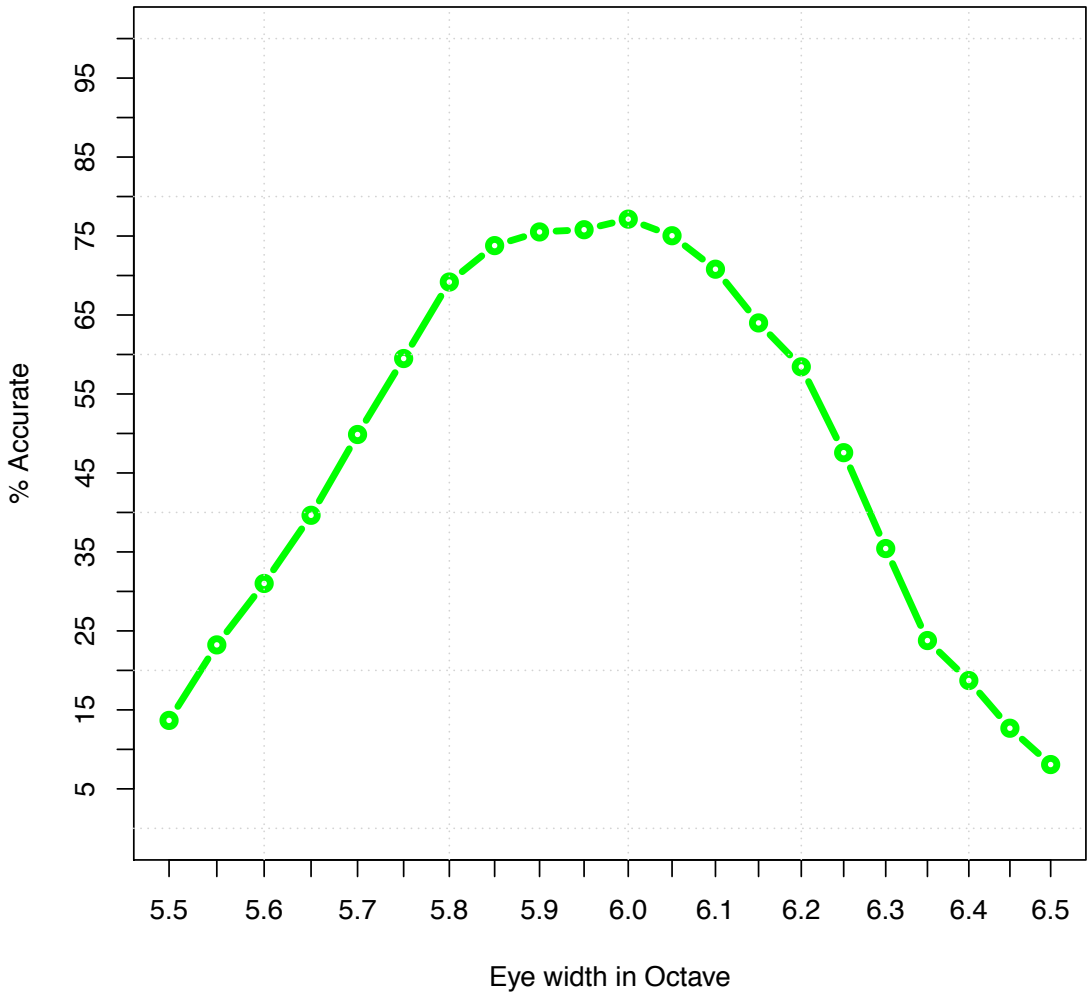


Figure 6.3: The percentage accurate is defined as the number of images which locate the face within 10% of the eye width from the target (point between the eyes) location.

experiments for filters trained on datasets of other octaves between 4 and 7 interocular width. The underlying result remains consistent across all the tests, that a filter trained for a given scale can be used to test image data sets around a quarter of an octave of the training dataset interocular width. Therefore, based on these results, I trained filters for each quarter octave between octave 4 and 7, as a result of which 13 filters were trained, corresponding to 13 scales.

6.3 Account for Pose in Face Detection

In Section 6.2, an approach was presented to find the best number of filters to train for PaSC dataset. However, in addition to faces of different scales, PaSC dataset also contains faces of different poses and needs to be addressed. Figures 6.1 and 6.2 give an idea about the variation in pose in the still and the video datasets. In this section, I will discuss an approach to address the challenge of different poses for face detection using correlation filters.

Finding the pose of the face can be very difficult owing to the various angles at which an image could be taken, particularly when it is not being controlled. The determination of the pose becomes more challenging also because, for each pose, we still have to account for different scales. Therefore, each pose is restricted by a corresponding scale of the face or vice versa. For example, for a face scale of octave 4, there could be a wide range of poses in the dataset. However, since I have already determined the range of scales required for the PaSC dataset, the pose range is restricted not just by the dataset but also by the scales. .

I ran a pilot experiment to determine the range of pose variation, and found that there are three different ranges within the PaSC training dataset: less than -12° (left side facing the camera), between -12° and +12° (in and around frontal pose) and greater than +12°(right side facing the camera). So, we have frontal or near frontal pose, left profile or between frontal and left profile pose and right profile or between right profile and near frontal pose. These three poses combined with the optimal scales that I found in the dataset (see Section 6.2), helped partition the dataset for training. In the next section I will discuss the set up for training filters for the PaSC dataset within the limitations of pose and scale variations in the training dataset.

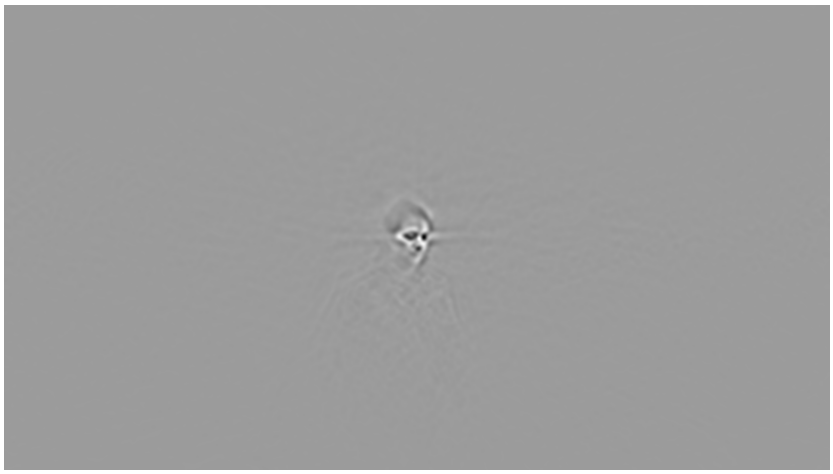
6.4 Training Filters for PaSC

As determined by the experiments of Sections 6.2 and 6.3, filters need to be trained for a scale of a quarter of an octave for three different poses. This means between octave 4 (16 pixels between the eyes) and 7 (128 pixels between the eyes), 13 filters need to be trained. In order to account for the variation in pose in the test images, at each of these 13 scales I have to train for three different poses. This all adds up to 39 different filters.

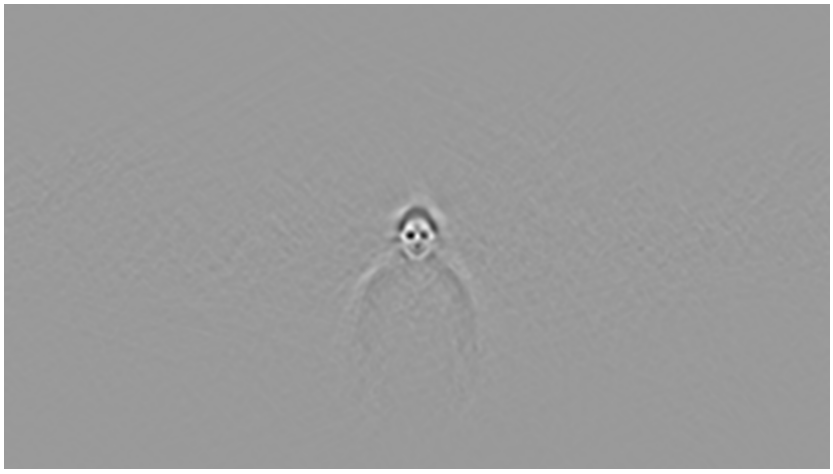
Training 39 different filters requires a large number of images. I had to use 999 videos with 255,835 frames for training to account for all poses and scales. Nine of those trained filters are displayed in Figures 6.4, 6.5 and 6.6. The three filters in each figure have the same scale but three different poses. To be more precise, in Figure 6.4, the top filter 6.4(a) has been trained on an image dataset containing images with 32 pixels between the eyes and a pose of greater than 12° or the right profile. Similarly, the filter in Figure 6.4(b) is trained on a dataset with images that have 32 pixels between the eyes and a frontal or near frontal pose. Finally, Figure 6.4(c) shows a filter trained on a dataset having images that have 32 pixels between the eyes and a pose of less than 12° or near left profile. Figures 6.5 and 6.6 can be explained in a similar fashion. The filters for other scales not shown in these figures also have similar definitions.

6.5 Testing PaSC

Our test sets consist of 9,376 still images and 38,999 video frames. The first approach to find faces in this test dataset was the same as the bank of filters approach of Section 5.4.3. However, after many experiments it started to become clear that the convolution of the filters with a test image in the Fourier domain showed very poor accuracy results. On the contrary, using spatial convolution to match just the faces with the test image proved to be a better option as is discussed in the next section.



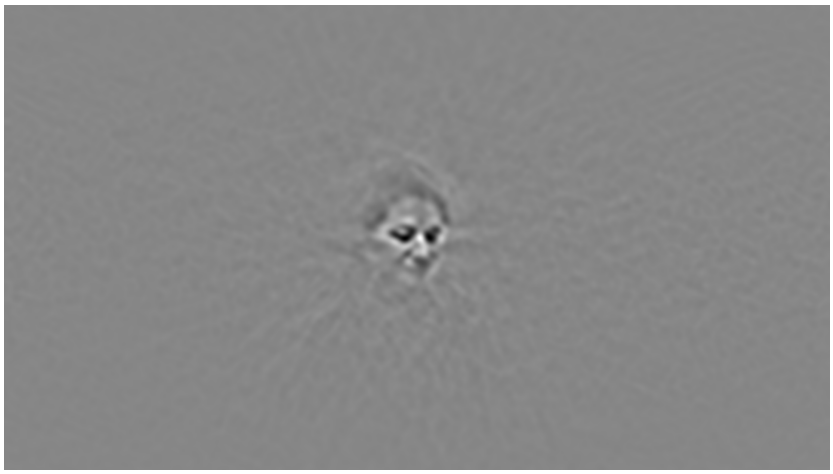
(a) pose $>12°$



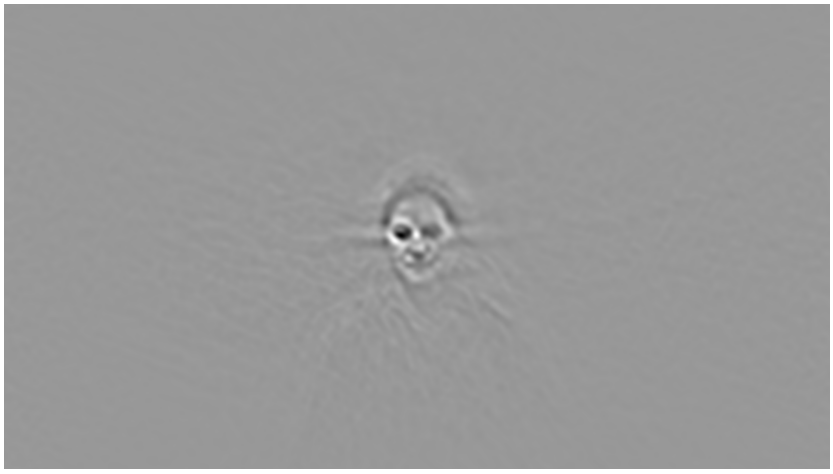
(b) pose between -12° and +12°

(c) pose $<12°$

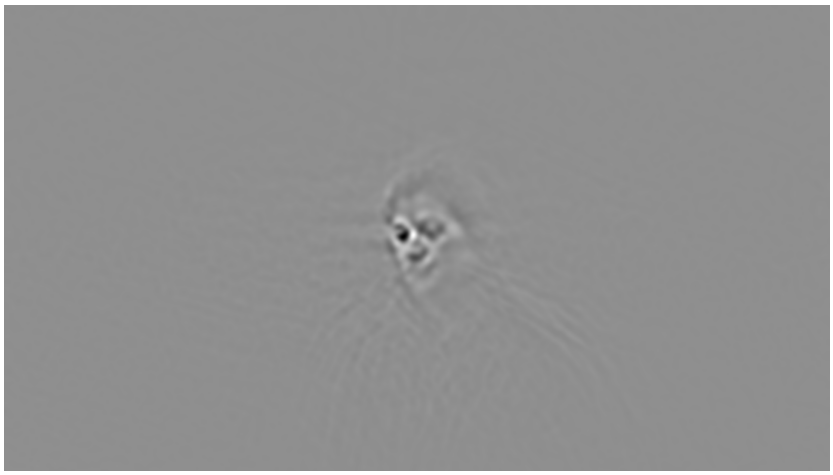
Figure 6.4: Three different poses for a scale of 32 pixels between the eyes.



(a) pose >12°

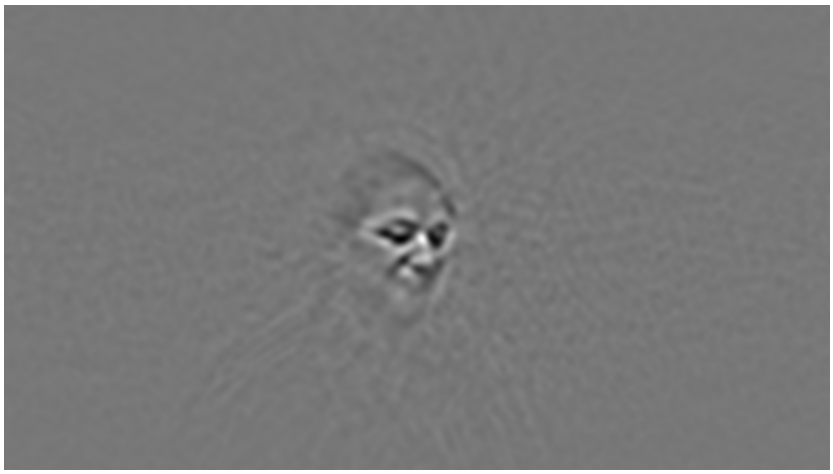


(b) pose between -12° and +12°

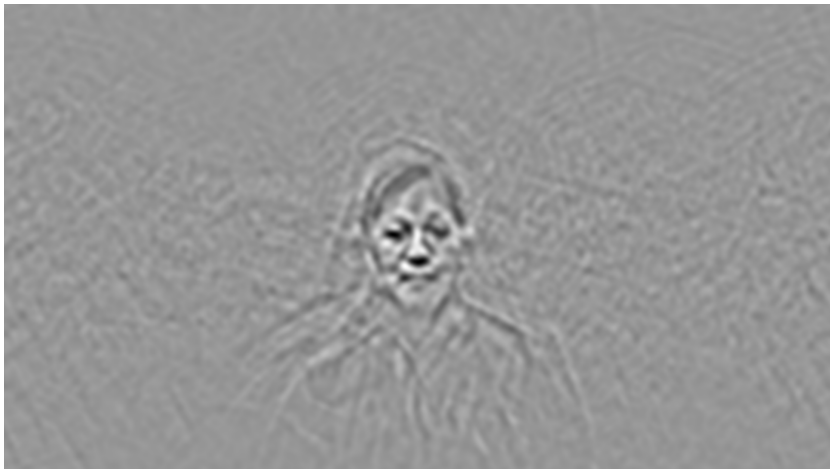


(c) pose <12°

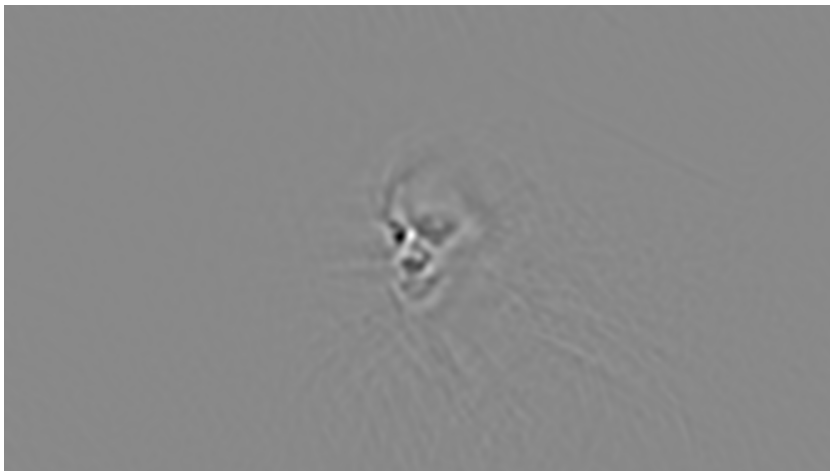
Figure 6.5: Three different poses for a scale of 64 pixels between the eyes.



(a) pose >12°



(b) pose between -12° and +12°



(c) pose <12°

Figure 6.6: Three different poses for a scale of 90 pixels between the eyes.

6.5.1 Convolution Using Spatial Correlation

In this section, I present a case for using a bank of filters approach in the spatial domain instead of the bank of filters approach in the Fourier domain because the former approach is more effective on the PaSC dataset than the latter one. Although it seems contrary to some of my earlier approaches, a set of careful experiments clearly made it an approach of choice for this dataset. One of the preprocessing steps before the convolution in the Fourier domain involves normalization of the full image. This increases the likelihood of a higher correlation value at a location other than on the face, for example, near a light source. On the contrary, normalization in the case of spatial correlation is applied for each specific location separately, corresponding to the size of the template, which is usually much smaller than the image size. This approach gives a better chance of a correct match.

In order to explain the difference between the two approaches a little more, I will now use a toy example. Consider Figure 6.7 to be an input image and Figure 6.8 to be the filter (or template). I am interested in finding this template in the input image. The first step is to normalize the

Figure 6.7: Input Image

Figure 6.8: Template Image

input test image before convolving it with the template in the Fourier domain. Secondly, since the template is not the same size as the test image, the template needs to be padded with zeros to make it the same size as the test image. The template with the padded zeros is shown in Figure 6.9. Convolution of the template and the test image results in a correlation surface with a maximum at coordinates $(0, 0)$. This correlation surface is displayed in Figure 6.10. On the contrary, spatial correlation using OpenCV template matching results in the correct identification of the peak at

Figure 6.9: Template Image of Figure 6.8 with Padded zeros

Figure 6.10: Correlation Output after convolving template Image of Figure 6.9 with Padded zeros and Figure 6.7.

coordinates $(8, 0)$. Having presented an example of how spatial filtering yields a better result, for the rest of this chapter I will follow the spatial template matching approach.

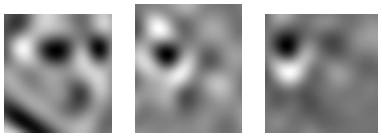
Specifically, the testing of PaSC datasets was carried out by using OpenCV template matching [Bra00]. It is basically spatial correlation of templates with the test images. In particular, I used the normalized cross correlation coefficient of OpenCV template matching. It is mathematically represented by the equation 6.1.

$$NCC(x,y) = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} I(x+i,y+j)\cdot T(i,j)}{\sqrt{\sum_{i=1}^{W}\sum_{j=1}^{H} I(x+i,y+j)^2}\cdot\sqrt{\sum_{i=1}^{W}\sum_{j=1}^{H} T(i,j)^2}}, \tag{6.1}$$

where I is a source image, T is a matching template of size $W \times H$, and (x, y) is the location on the image, I where the template is centered for matching.

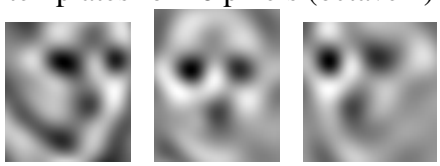
For our experiments on the PaSC dataset, the templates to match were created by cropping out the faces from the trained filters. Since there are 39 filters corresponding to 13 scales and 3 poses for each scale, there are as many face templates. Some of these templates are displayed below in Figures 6.11 through 6.15 for scales of 16 through 32 pixels(4-5 octave in quarter steps) between eyes display a pyramid of filters of different scales between these octaves.

The approach to detect faces in still images and the frames in videos is carried out using each of these filter templates, called the bank of filters, to match with a test image. Each of these templates is matched with a given test image using OpenCV template matching. The location of the peak correlation value is recorded. This location combined with the filter width and height forms a rectangle representing the detected face in the test image. A match is considered to be found between the returned face rectangle and the ground truth face rectangle if there is an overlap



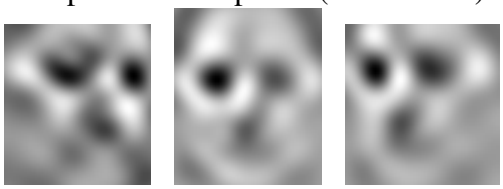
(a) >12° (b) >-12° and <+12° (c) <-12

Figure 6.11: filter templates for 16 pixels (octave 4) between the eyes.



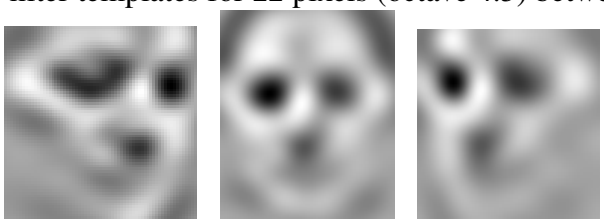
(a) >12° (b) >-12° and <+12° (c) <-12

Figure 6.12: filter templates for 19 pixels (octave 4.25) between the eyes.



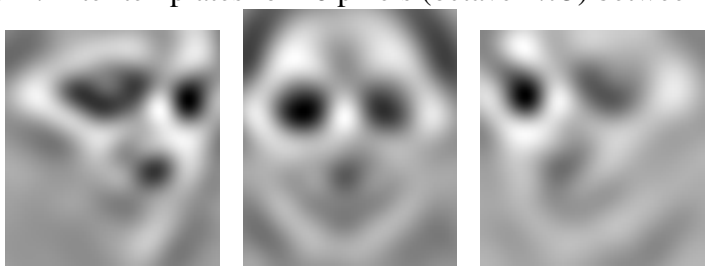
(a) >12° (b) >-12° and <+12° (c) <-12

Figure 6.13: filter templates for 22 pixels (octave 4.5) between the eyes.



(a) >12° (b) >-12° and <+12° (c) <-12

Figure 6.14: filter templates for 26 pixels (octave 4.75) between the eyes.



(a) >12° (b) >-12° and <+12° (c) <-12

Figure 6.15: Filter templates for 32 pixels (octave 5) between the eyes.

of 25% between the two rectangles. The ground truth face rectangles were obtained using the SDK 5.2.2 version of an algorithm developed by Pittsburgh Pattern Recognition (PittPatt) [BPB+13].

6.5.2 Results on PaSC Still Images

For the 9376 still images I have recorded the locations of the correlation peaks when each test image is spatially correlated with the filter templates. This peak location is the top left corner of the face rectangle whose width and height is the same as the width and height of the filter. Spatial correlation of these thirty-nine filters returns thirty-nine face rectangles. Each of these face rectangles is associated with a correlation peak. Ideally, the face rectangle associated with the highest peak is considered to be the detected face for an image. In order to determine whether this face rectangle is really a face or not, I compare it with the ground truth face rectangle. If their is an overlap of at least 25% between the two rectangles, a face is considered to be detected. However, the face rectangle with the highest correlation peak is not always the right one, and many times there is at least one of the thirty-nine filters that does find a face but the associated peak may not have the highest correlation value.

In this section, I will present two types of results: one where only the accuracy associated with the highest correlation peak filter is reported and second, the accuracy associated with different sets of filters between one and thirty nine. Figure 6.16 displays results when different numbers of filters are used. The x-axis in this figure represents the number of filters being used on the test dataset in the decreasing order of the correlation peak value and the y-axis, the percentage of images that detected the face in the location agreeing with the ground truth. There are two lines in this figure, a blue line and a green line. The blue line represents the face detection accuracy for the actual faces detected in the test images using the bank of thirty-nine templates and the green line represents the results of detecting faces when filters are randomly assigned a location in a given image.

A random face detection experiment is conducted by randomly ordering the bank of templates to represent the correlation peaks from the highest to the lowest. It is followed by randomly picking an x and y location representing the peak in an image to be associated with each of these filters. For each image I have recorded a random peak location and a random filter combination, for all

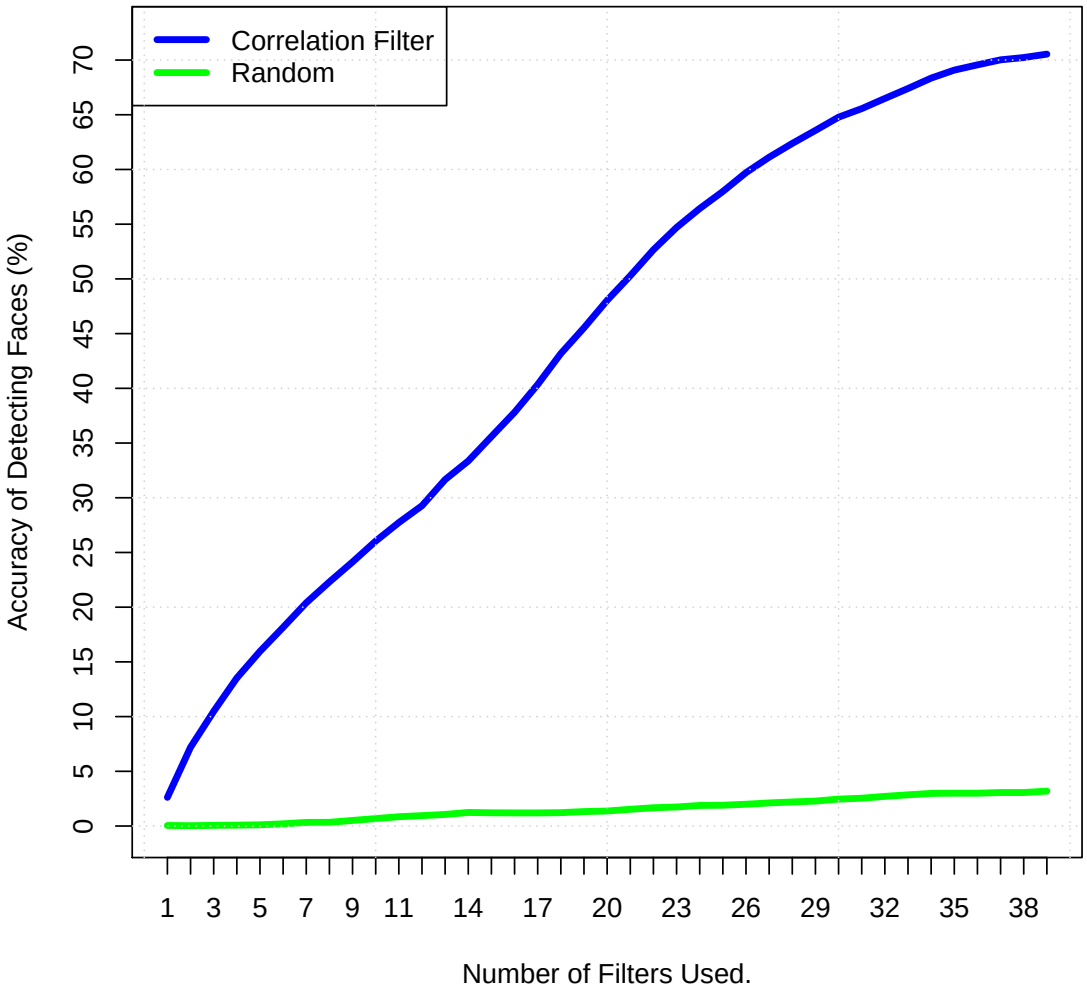


Figure 6.16: Accuracy of detecting face vs. the number of filters in still images.

the thirty-nine filters. These results together with the actual face detection results are presented in Figure 6.16.

I will describe this plot with some examples. Let me begin with an x-axis value equal to one which means I only use one filter. This filter is associated with the highest correlation value and it is able to detect a face correctly only in 2.62% of the images. It is shown on the blue line. The corresponding random experiment accuracy value shown on the green line for a randomly picked single filter is only 0.05%. This means, if I pick a random location on an image and combine it with a randomly picked filter for the face width and height, then the percentage of times a face is correctly detected in our test dataset is 0.05%. Similarly, when x is equal to 20, it means twenty filters are used corresponding to the top twenty peak correlation values and 48.04% of the time, at least one of those twenty filters correctly detects a face. This case is shown with the blue line. The corresponding random accuracy shown on the green line when twenty random filters are used is 1.37%. Finally, when the x-axis value is 39, it means all the filters are being used and the percentage of times at least one of those correctly detects the face is 70.54%, compared to 3.19% in the random experiment, shown with the green line.

6.5.2.1 Comparison of Correlation Filter Results with OpenCV Viola and Jones Face Detector

In this section, I am presenting bar plots to compare the face detection results using correlation filters, with OpenCV Viola and Jones face detector [VJ01]. These results are shown in Figure 6.17. This plot can be interpreted as follows: the x-axis represents two sets of filters and the y-axis shows the accuracy of correctly detecting a face. The x-axis label, 'Peak Correlation Filter', shows results when only a single filter is used to determine the accuracy of face detection. This filter is the one that returns the highest correlation value. For the Viola and Jones Face detector, out of the multiple face detection results that it returns, I pick the one that is associated with the largest number of neighbors. Therefore, using my correlation filter based face detector, the number of test images whose detected face rectangles will have at least 25% overlap with the ground truth face rectangle, is 2.62% which is 245 images out of 9,376. In comparison, the OpenCV Viola and Jones face detector correctly detects 76.64% i.e., 7186 out of 9376 faces.

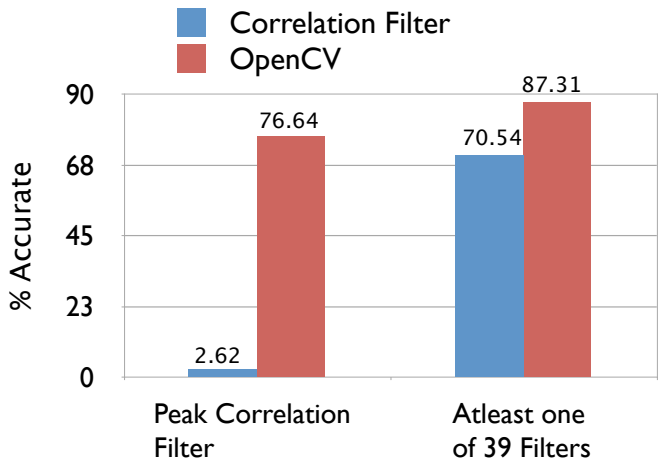


Figure 6.17: Accuracy of detecting a face in still images vs. the number of filters.

The second set of bar plots labeled as, 'Atleast one of 39 Filters', shows accuracy results when multiple filters are used to find faces and at least one of them correctly detects a face. For correlation filter based approach, I use all thirty-nine filters and if at least one of the detected face rectangles overlaps the ground truth face rectangle by 25% or more, it is considered to be a success. In the case of OpenCV, I find the overlap of all returned face rectangles with the ground truth and if at least one of them correctly detects a face, it is considered to be a successful detection. Using my approach a face was correctly detected in 70.54% i.e., 6614 out of 9376 images by at least one of the thirty-nine filters being used. In comparison, the accuracy of correctly detecting a face by at least one of the face rectangles returned by Viola and Jones face detector is 87.31%, i.e., 8187 faces out of 9376 images. Clearly, OpenCV is far better than my correlation filter based face detector on the PaSC still images dataset.

6.5.3 Results on PaSC Video Frames

My video test dataset consists of 38999 frames in 403 videos on which I test thirty-nine face templates filters. Just like for the still test dataset, for each of the template matchings between the filters and the video frames, I have a face rectangle corresponding to each of the correlation outputs. These resulting rectangles are compared with the ground truth face rectangles of the

frames and if there is a 25% overlap or more, like in the still images, I refer to it as a detected face. In addition to this filter matching, a random experiment has been carried out just like the one on the still dataset. This experiment involves randomly organizing the set of filters in a decreasing order of the correlation value followed by obtaining a random location for the correlation peak in the test frame. Each of these locations combined with the corresponding filter width and height give a face rectangle for each image. Since the number of filters is thirty-nine, I have thirty-nine locations for each image.

Again, in order to report these results, I present the accuracy of face detections corresponding to the number of filters being used in Figure 6.18. This figure also contains results of the random experiment on the video frames. The blue line represents the actual face detection experiment results and the green line shows the random experiment results.

These two lines are to be interpreted in the same manner as the ones in Figure 6.16. For example, if a single filter is used to detect faces, 14.59% of the detected face rectangles have at least 25% overlap with the ground truth face rectangle, which is 5690 out of 38999 frames. In a randomized experiment when a single random location is chosen with a randomly filter, the face detection accuracy is only 0.42%. Similarly, when all thirty-nine filters are used, the accuracy of at least one of the returned face rectangles successfully detecting a face is 84.07% which is 32786 out of 38999 frames. On the contrary if thirty-nine random locations are picked in the same dataset, the accuracy of correctly detecting a face is 6.39%.

6.5.3.1 Comparison of Correlation Filter Results with OpenCV Viola and Jones Face Detector

In order to determine how well my correlation filter based face detector performs vis-a-vis the OpenCV Viola and Jones face detector, I compared results on the video dataset, just like in the case of still images. This comparison is presented in the form of bar plots in Figure 6.19. The blue bars represent the results using my correlation based face detector. There are two cases, firstly, when a single filter corresponding to the highest correlation value is used and, secondly, when all thirty-nine filters are used for face detection. The criteria for accuracy remains the same for the two cases: if atleast one filter correctly detects a face it is counted as a successful detection.

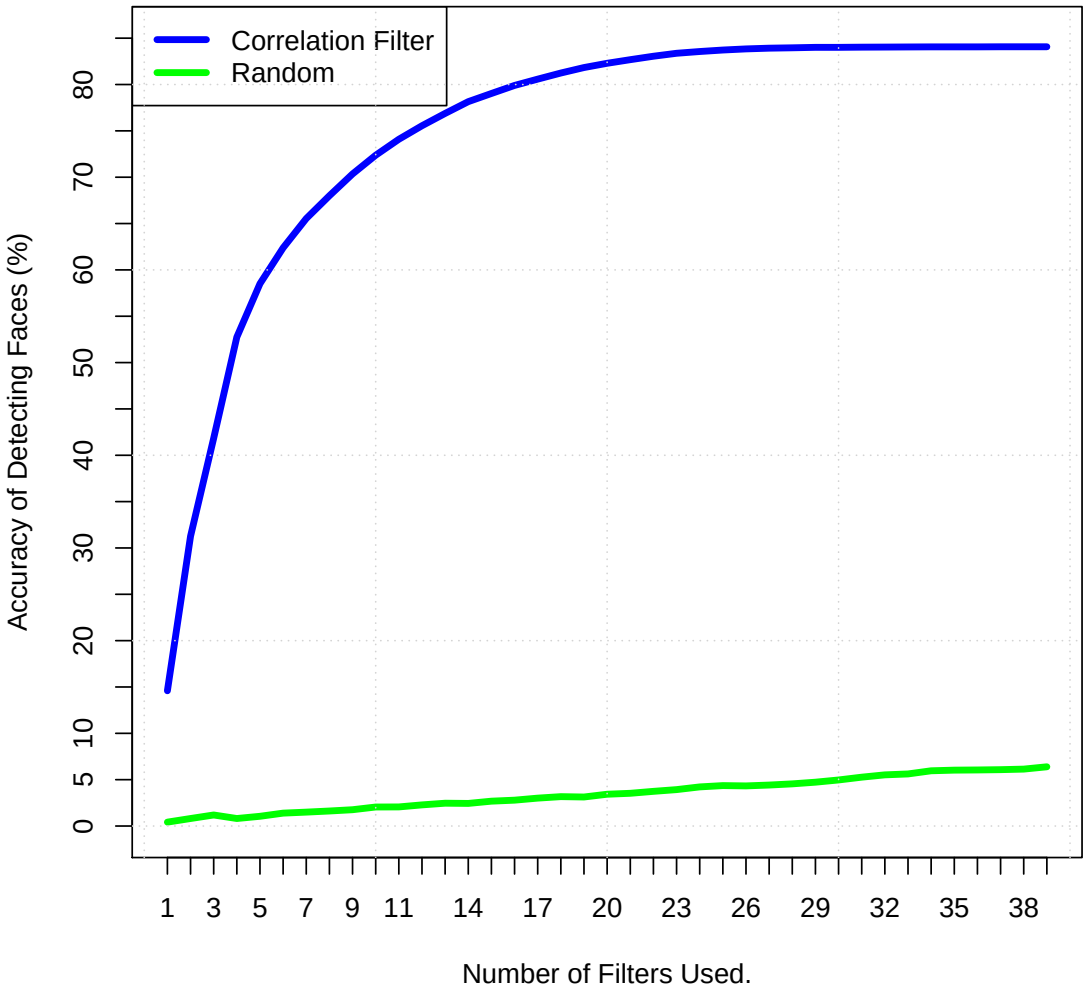


Figure 6.18: Accuracy of detecting face vs. the number of filters in video frames.

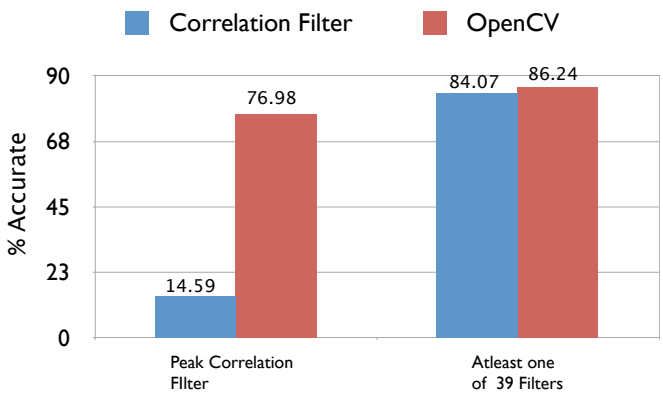


Figure 6.19: Accuracy of detecting a face in video images vs. the number of filters.

The performance for my face detector varies from 14.59% for a single filter to 84.07% when all thirty-nine filters are used.

The red bars represent the results for the OpenCV Viola and Jones face detector. There are two cases for this algorithm too. The first case is, when I consider a single face rectangle out of one or more face rectangles returned by the face detector. The single face rectangle that is chosen is the one with the highest number of neighbors associated with it. The second case for this algorithm is the one, where accuracy is measured if at least one of the returned face rectangles correctly detects a face. The accuracy for these two cases varies from 76.98% to 86.24%.

6.6 Analysis of a Case of Failure

As shown in the last section, the accuracy of finding a face in PaSC still image dataset using only a single filter, is just 2.62%. This single filter is the one that gives the highest correlation peak. In contrast, when all the thirty-nine face rectangles returned by template matching an image and the filters, are considered to determine a possible face detection, one can detect a face correctly in 70.54% of the images. Because at least one of those thirty-nine faces finds a match. This implies that in the majority of the cases the location of the highest correlation value is not well correlated with the location of the true face.

The goal of this section is to study one of the cases of a failed face detection in detail when

none of the thirty-nine filters actually found the face correctly. Such an image with the associated filters is shown in Figure 6.20. In this figure, the image labeled 'Best Match' identifies the portion

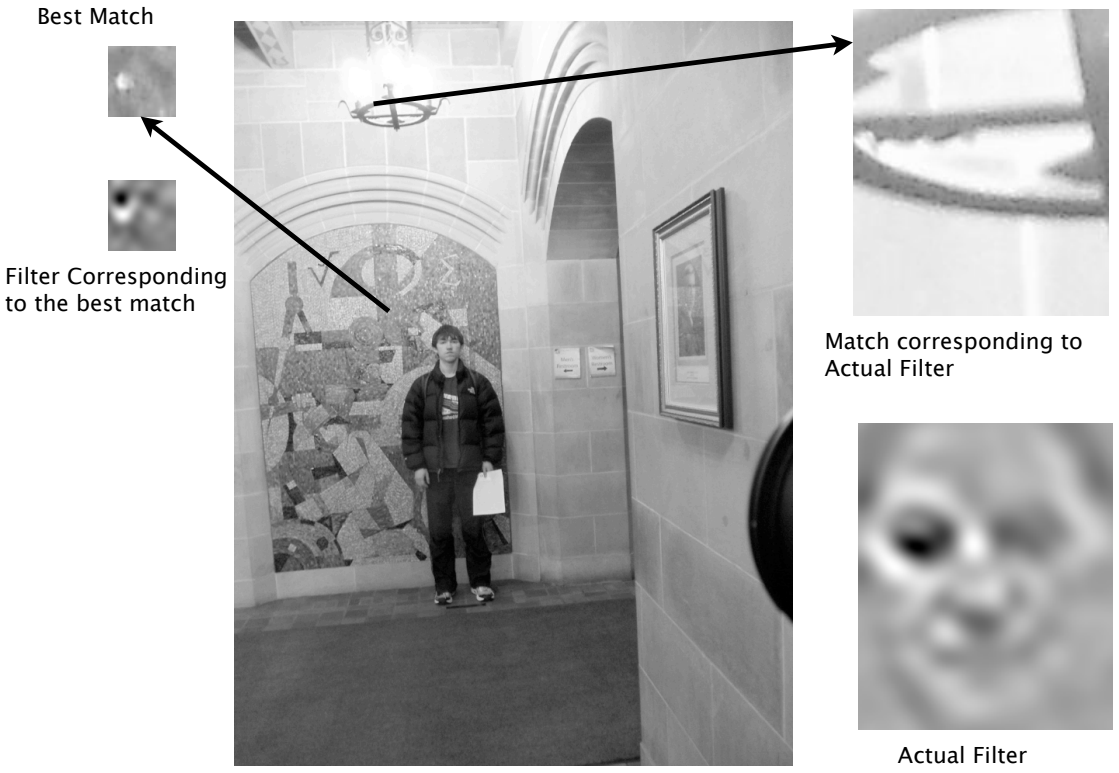


Figure 6.20: A test image showing the best detected face along with the filter and the expected filter to detect the face correctly.

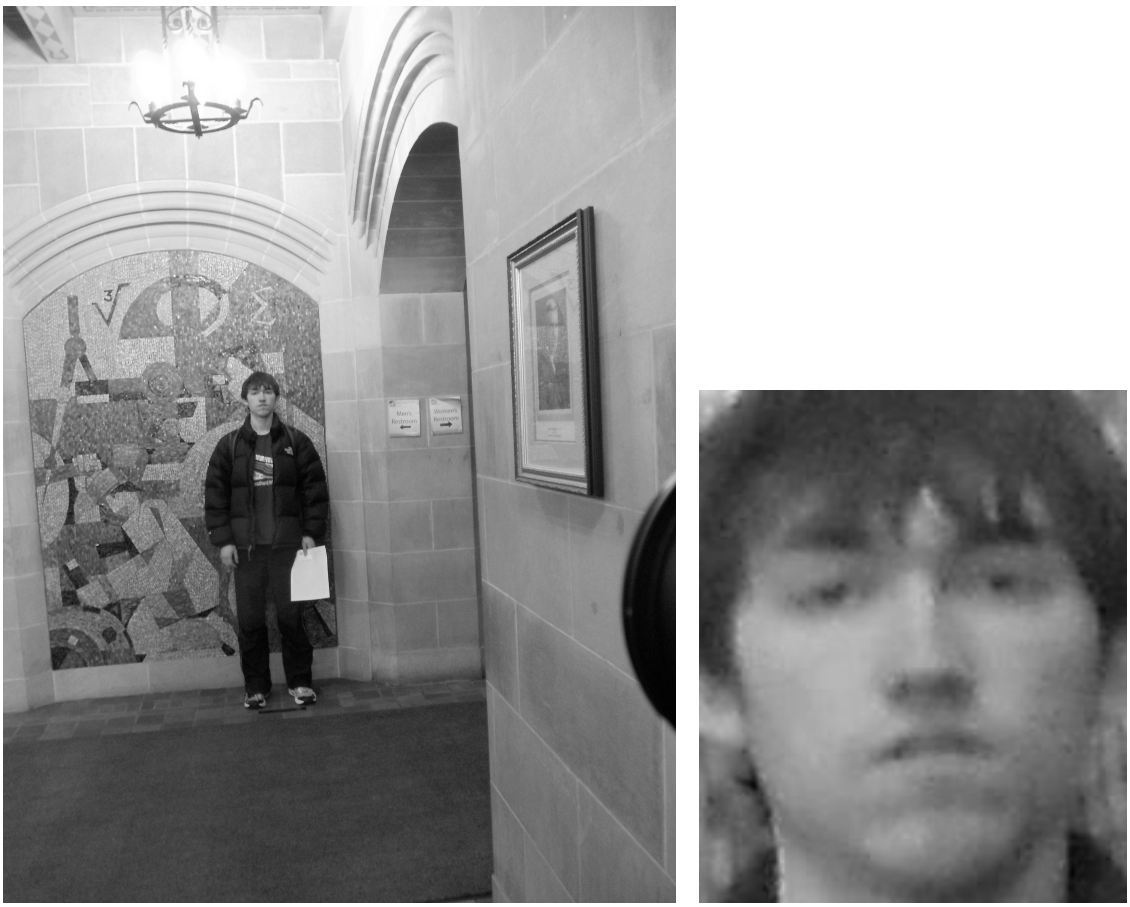
of the image corresponding to the falsely detected face. This detected face is based on the highest correlation value returned, when all the filters are spatially correlated with the test image using OpenCV. The filter that yields this highest correlation value is labeled as, 'Filter Corresponding to the best match'. It is clear from the Figure 6.20 that the face is not detected correctly.

In contrast, the template that matches the scale and pose of the face and that should have found the face correctly (but did not) has been labeled, 'Actual Filter', in the figure. It is one of the thirty-nine filters correlated with the test image. When this filter is used as a template for the test image, a totally different location of the image is identified to be a face, which is labeled, 'Match corresponding to Actual Filter' in the Figure 6.20. This is not the correct face either.

The result of spatial template matching between the test image and the two filters, labeled,

'Best Match' filter and the 'Actual Filter', result in correlation values of 0.60 and 0.29, respectively. Neither one detects the face correctly.

In order to test spatial correlation, using a perfect template, I cropped the face (see Figure 6.21(b)), from the test image 6.21(a)(resized to fit) and spatially correlate the two. The result is a perfect match. However, if this cropped face is used as an image and correlated with the 'Actual



(a) Test Image (b) Cropped Face

Figure 6.21: Test image and face cropped from it to use as a template for Spatial correlation using OpenCV.

filter' of Figure 6.20, shown separately in Figure 6.22, the correlation value is very low. It is a poor match. Therefore, even when using a seemingly perfect template, sometimes, it may still not be a good match for an image. I will discuss the reasons for this in detail in the concluding chapter.

Figure 6.22: Filter expected to detect the face in the test image.

6.7 Conclusion

In this chapter, I presented the correlation based face detection results on the PaSC still images and video frames. These datasets represent a careful combination of uncontrolled scale, pose, lighting, and location along with the sensors used to capture it, just like numerous, handheld cameras and cellphones that people around the world use to take pictures.

The results have been compared with the random results as well as OpenCV Viola and Jones face detector. It turns out that although my approach does much better than random experiment, it is not at all competitive with the Viola and Jones Face detector when a single filter is used. However, when all thirty-nine filters covering thirteen different scales and three poses for each scale, are used, the accuracy of face detection by at least one of those filters comes very close to the Viola and Jones face detections. To be precise it is 70.54% accuracy for my approach versus 87.31% accuracy using Viola and Jones face detector, on the still image dataset. On the video dataset, the accuracy is 84.07% versus 86.24%, for correlation filter based face detector and Viola and Jones face detector, respectively.

The reason for presenting and comparing the results when all filters are being used is that even though, the filter that gives the top correlation value does not always find the right face, there is at

least one filter in the set, that detects the face correctly with a high accuracy. This gives rise to two areas of research, one, that the metric, peak correlation value, for finding the right match may not be the right one, and second, correlation based face detection might not be an approach of choice for so many variations (scale, pose, background etc.,) in the dataset. In the conclusion chapter, I will delve more into the details pertaining to a metric for the best match. In the next chapter, however, I will study the impact of using a dataset with a limited variation in pose, scale, lighting and location, on face detection using correlation filters.

Chapter 7

Setting Specific Face Detection

In this research, while designing different filters and studying their impact in finding faces in still images and video frames, it started to become apparent that the more specific the filters to a particular scenario the better match they can find in the test images in that scenario. As a result, I hypothesize that if a filter is designed for a specific setting, the correlation filters can be very successful in face detection. The goal of this chapter is to test this hypothesis.

7.1 Experimental Setup

This experiment is setup to test the hypothesis presented above. I have controlled for location and as a result created a customized dataset of video frames. In this dataset the location is the same between the training and the test datasets but there is no overlap of people between the two datasets. All the videos have been shot in an indoor location. Some of the frames from six different videos are presented in Figure 7.1. These frames are a representative of this dataset displaying some of the variations in scale and slight pose, in a specific setting. The training set consists of 256 videos with 18,966 frames. The filters trained for this particular location, and two closely related poses, and two different scales are shown in Figure 7.2.

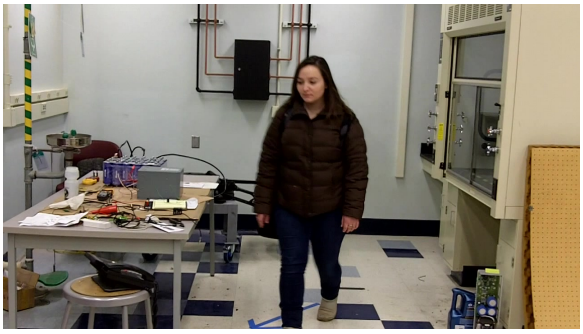
The faces in the video frames for this setting have an interocular width of 16 to 32 pixels and two closely related poses near the left profile. Therefore, I created two different datasets for training. One set consisted of images having greater than or equal to 16, but less than 24 pixels between eyes. This training set was used to train the filter shown in Figure 7.2(a). The second training set consisted of images that had an interocular width between 24 and 32 pixels. The filter trained on this dataset is presented in Figure 7.2(c).

Even though the filters are trained on the full frames to get rid of the clutter in the background, smaller face templates, shown in Figures 7.2(b) and 7.2(d), have been cropped from these trained filters to do a spatial template matching using OpenCV. This ensures that the normalization is taken

(a) Image 1

(b) Image 2

(c) Image 3



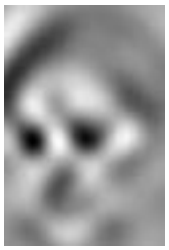
(d) Image 4

(e) Image 5

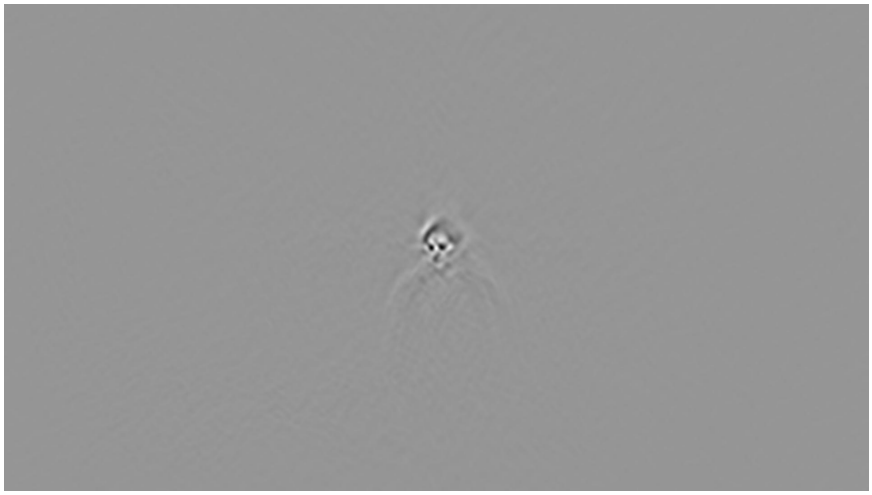
(f) Image 6

Figure 7.1: Representative frames from the custom Video dataset at a particular location.

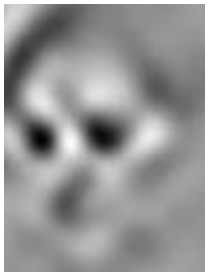
(a) Filter with scale of 16 pixels between the eyes.



(b) Template (filter) cropped from 7.2(a).



(c) Filter with a scale of 24 pixels between the eyes.



(d) Template (filter) cropped from 7.2(c).

Figure 7.2: Filters trained on 256 videos with 18966 frames for a particular location and two different scales.

at each location of an image corresponding to the size of the face template and not the entire image. This approach is similar to the one used for testing the PaSC dataset in Chapter 6.

These two face templates have been spatially correlated, using OpenCV, with the frames of a test set consisting of 73 videos, containing 1,865 frames. Each of these videos has been shot at the same location and the interocular width varies between 18 and 33 pixels. It may be reiterated that in my setup two filters have been trained on images with an interocular range of 16 to 32 pixels. If we take a look at Figure 6.3, one may notice that there is a range of test images that would give a good detection accuracy for a filter trained on images that are closer to the interocular width in the test dataset. Since the interocular widths of the images in the training and test datasets lie in the same range, the filter design choices in terms of interocular distance are appropriate.

7.2 Results

Each frame in the test dataset is matched with the face templates of Figures 7.2(b) and 7.2(d), and the coordinates of the location that returns the highest correlation value for these two matches is recorded. To choose between these two results, the coordinates of the one that has a higher correlation value is selected. This coordinate identifies the top left corner of the detected face rectangle. The width and the height of the face template determines the width and the height of the face rectangle. These face rectangle coordinates for each test image are recorded. These face rectangles are eventually used to determine the performance of the experiment.

The performance measure used in this experiment is based on an overlap of the face rectangle obtained by template matching of the face filters and the test images, as described above, and the face rectangle of the faces from the ground truth for each of these images. If there is an overlap of 25% or more between the two rectangles, the face in an image is recorded as detected. Based on this criteria for accuracy, the algorithm was able to find a face match in 1,520 frames out of 1,865 frames which equates to an 81.5% detection rate.

This result has been compared with face detection results obtained from OpenCV Viola and Jones face detector, which achieved an accuracy of 69.43% or 1,295 out of 1,865 frames. The criteria of accuracy for this face detector is the same as for the correlation filter based face detector.

These two results are displayed side-by-side in Figure 7.3. It is clear that the correlation based face detector outperforms OpenCV Viola and Jones face detector in a specific scenario and it validates the hypothesis presented in the beginning of this chapter.

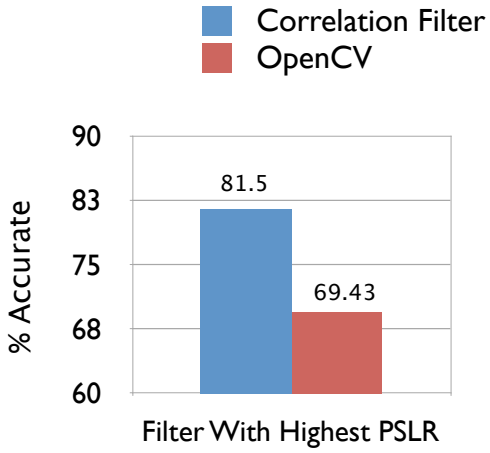


Figure 7.3: Comparison of Face detection accuracy between Correlation Filter approach and OpenCV Viola and Jones face detector.

7.3 Conclusion

In this chapter, I hypothesized that the more specific a filter, the better the accuracy in detecting faces in an image. As such, a controlled experiment was devised in which the location in both training and test datasets was restricted. The interocular widths between the training and the test datasets are also in the same range. The correlation filter based face detector was used to test the dataset and compared with the OpenCV Viola and Jones face detector. The results presented in Figure 7.3 clearly indicate that the correlation based approach is doing better for such a dataset and hence confirms the hypothesis.

This is one of the most significant face detection results obtained using correlation filters so far. A constant background setting gives an advantage to the approach because the correlation filter training appears to learn this setting and returns a filter suited for a similar background. Because of this learning, the filters ignore the background clutter during the testing process and avoid false

positives. Therefore, one of the most important application of a correlation based face detection approach is to use it in specific situations. For example, people walking into a building through a specific door where the setting remains mostly constant. This chapter is an important case study to establish the significance of face detection using correlation filters that are trained for specific scenarios and is an attempt to lead this research in that direction.

Chapter 8

The Conclusion

The purpose of this chapter is to highlight some of the more important findings of this research. I will begin with a detailed discussion of the major findings of this dissertation. Then I will summarize how these findings may be seen as useful in terms of guiding future work.

8.1 Discussion

I have presented a novel face detector based on correlation filters to detect faces in still images and video frames. Each of these images and frames contains only a single face. This algorithm is easy to train and fast. However, like any other face detector this one also has to deal with many complexities associated with face detection. Some of these complexities are associated with faces of different sizes or scales, pose, uncontrolled lighting, background and location, while some others arise out of the basic principles on which such a face detector is based.

Any correlation operation returns a peak value and hence by design it is suited to locate one face in an image associated with the highest peak. Since my goal is to work on datasets with a single face in each image, it is not a problem. My technique in general works very well for images with single faces and simpler backgrounds like the ones shown in Figure 8.1(a). It may be apparent in the correlation surface in Figure 8.1(b) resulting from the correlation of this image with a filter. The bright spot corresponding to the point between the eyes helps correctly localize a face followed by detection as discussed in this dissertation. This example displays a successful case of finding a single face required to be located for an application, such as, face recognition. Therefore, for images like these, the correlation based approach would certainly be an approach of choice.

Also, this technique can be very effective if one is interested in a single face in an image with multiple faces. This is clearly true for an image such as the one shown in Figure 8.2. If one is interested in finding just a single face in this image, even though there are more, the correlation surface peak location often corresponds to at least one true face and therefore, the algorithm is still

(a) Image (b) Correlation Surface

Figure 8.1: An image and its correlation surface showing the location of the peak between the eyes.

(a) Image (b) Correlation Surface

Figure 8.2: An image with multiple faces and its correlation surface showing the location of the peak between the eyes for a single face.

of value.

However, if there are several faces in an image and one is required to find all of those faces, can this correlation based face detector still be effective in finding all the faces? This is definitely a tricky issue in such a detector. One of the obvious approaches is to find the locations of all the peaks in a correlation surface. For this to work one needs to figure out a threshold so that any correlation value that is above it can be characterized as a face location. However, finding a threshold value is not straightforward. In some situations it could be easy to identify all the peak locations correctly and hence find the faces in an image like the one in Figure 8.3. In others, the

(a) Image (b) Correlation Surface

Figure 8.3: An image with multiple faces and its correlation surface showing the location of the peaks corresponding to the faces.

peaks are not always found in the correct locations and one often ends up with false detections and hence an incorrect threshold. This can happen not only in images with multiple faces but also with single face images. More often than not, a transition from high pixel intensity value to a low intensity value, or vice versa, generates a high peak value and hence, many times, a false face detection. This failure mode is illustrated in Figure 8.4.

This example identifies one of the biggest issues for face detection using correlation filters. Neither the use of a max value in the correlation surface nor the use of a peak-to-side-lobe ratio can prevent false peaks under these circumstances, which is key to locating faces using this technique. If there is any direction where this work needs to be taken forward it is in finding metrics to identify the peaks correctly. The impact is seen clearly when the intensities of pixels change, as in Figure 8.4. Although the face in this image is frontal, with good lighting, one would assume

(a) Image (b) Correlation Surface

Figure 8.4: An image and its correlation surface showing the effect of the change of pixel intensities between high and low values.

this face would be located quite easily, however, that is not the case. It is important to point out that there is a peak in the correlation surface that corresponds to the point between the eyes of the person in the image, but this is not the highest peak in the image. Its intensity is not even close to being as high as the string of peaks corresponding to the letters in the image. This makes it apparent, why the identification of the correct peak is very important to find the face in this image and many other images.

Although this aspect of peak measurement requires more study, it does appear like this may have something to do with the way that filters look. For example, an examination of the filter in Figure 8.5 reveals that there are dark eyes with bright pixels between them. This may cause, what I would like to call, a barcode effect. This effect becomes apparent whenever there is a change in intensity values like the ones shown in Figure 8.4. This leads to more false positives and a drop in true positives. Plus, identifying the correct peak is key to the success of a correlation based face detection. It is also important to reduce the false peaks not only because one can't find the face in an image that has only one face, but also because it makes it very difficult to select a threshold in

Figure 8.5: A Typical Correlation Filter

order to find all the faces in an image when there are more faces in them.

One of the challenges of face detection that I have discussed in detail in this dissertation is the different scales of the faces that I am trying to detect. My solution of using a batch of filters has worked well over a wide range of scales under some conditions. The same concept works for pose too. I need to have filters of different poses to get a better match. This implies, for each scale, I need to train filters that account for multiple poses. Having experimented some with it on some of the datasets in this dissertation, I can confidently conclude that the more specific a filter is to a dataset the better accuracy can be obtained using a correlation based filter for face detection.

8.2 Lessons Learned Going Forward

Prior to this dissertation, there has been a lot of study on correlation filters and their applications. However, this dissertation is the first research to extend correlation filters for face detection. The goal is not to create the best face detection approach, but to extend correlation filters to an area where they have never been applied before. More specifically, this face detector has been designed to work under very controlled scenarios, specific for a situation. It has potential value in scenarios such as, an airport security gate, an entrance to a sensitive building such as a nuclear reactor or an entrance to a train/ bus station. The main findings of this research can be summarized, as follows:

- This is the first correlation filter based face detector. I have presented scenarios where such a face detector would be successful and where it may not be an approach of choice.
- I have presented a methodology to use localization for face detection.
- A corrlation filter trained for a given scale and pose, can be used for a small range of scales and poses around the training scale and pose.
- Correlation based face detectors are most promising in deployments where the setting remains roughly constant and setting specific filters may learn to discount background distractions.

References

[AARK08] K.R. Anoop, P. Anandathirtha, K.R. Ramakrishnan, and M.S. Kankanhalli. Integrated detect-track framework for multi-view face detection in video. In *Computer Vision, Graphics Image Processing, 2008. ICVGIP '08. Sixth Indian Conference on*, pages 336 –343, dec. 2008.

[BB05] Lubomir Bourdev and Jonathan Brandt. Robust object detection via soft cascade. In *In Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2005.

[BBD10] David S. Bolme, J. Ross Beveridge, and Bruce A. Draper. Optimized correlation filters for signal processing, 06 2010.

[BBDL10] D.S. Bolme, J.R. Beveridge, B.A. Draper, and Y.M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[BCG01] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *In Proceedings of International Conference on Data Engineering*, pages 443–452, 2001.

[BDB09] David S. Bolme, Bruce A. Draper, and J. Ross Beveridge. Average of synthetic exact filters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2105–2112, August 2009.

[BLDB09] D. S. Bolme, Y. M. Lui, B. A. Draper, and J. R. Beveridge. Simple Real-Time Human Detection Using a Single Correlation FIlter. In *Twelfth International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2009.

[BPB+13] J. Ross Beveridge, P. Jonathon Phillips, David Bolme, Bruce A. Draper, Geof H. Givens, Yui Man Lui, Mohammad Nayeem Teli, Hao Zhang, Todd Scruggs, Su Cheng, Kevin Bowyer, and Pat Flynn. The Point-and- Shoot Challenge. In *Sixth IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS)*, September/October 2013.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BWS+08] S. Brubaker, Jianxin Wu, Jie Sun, Matthew Mullin, and James Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77:65–86, 2008.

[Cau82] H. J. Caulfield. Role of the horner efficiency in the optimization of spatial filters for optical pattern recognition. *Applied Optics*, 21:4391–4392, 1982.

[Cro84] Franklin C. Crow. Summed-Area Tables for Texture Mapping . *ACM SIGGRAPH Computer Graphics*, 18(3):207–212, 1984.

[CTB92] I. Craw, D. Tock, and A. Bennett. Finding face features. In *Proc. Second European Conf. Computer Vision*, pages 92–96, 1992.

[D.84] Casasent. D. Unified synthetic discriminant function computational formulation. *Applied Optics*, 23:1620–1627, 1984.

[DN96] Y. Dai and Y. Nakano. Face-texture model based on sgld and its application in face detection in a color scene. *Pattern Recognition*, 29(6):1007–1017, 1996.

[DS10] Nikolay Degtyarev and Oleg Seredin. Comparative testing of face detection algorithms. In Abderrahim Elmoataz, Olivier Lezoray, Fathallah Nouboud, Driss Mammass, and Jean Meunier, editors, *Image and Signal Processing*, volume 6134 of *Lecture Notes in Computer Science*, pages 200–209. Springer Berlin / Heidelberg, 2010.

[FE04] B. Froba and A. Ernst. Face Detection with the Modified Census Transform. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 91–96, 2004.

[FP02] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice-Hall, 2002.

[FS95] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt 95*, pages 23–37. Springer-Verlag, 1995.

[GFDRKAB05] J. Angel Gonzalez-Fraga, Victor H. Diaz-Ramirez, Vitaly Kober, and Josue Alvarez-Borrego. Improving the discrimination capability with an adaptive synthetic discriminant function filter. In Jorge S. Marques, N. Pérez de la Blanca, and Pedro Pina, editors, *Pattern Recognition and Image Analysis*, pages 83–90. Springer-Verlag Berlin Heidelberg, 2005.

[Goo96] G.-I. Goo. Correlation filter for target detection and noise and clutter rejection. In D. P. Casasent & T.-H. Chao, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 2752 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 317–332, March 1996.

[HALL05] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *In Proc. of International Conference on Computer Vision (ICCV)*, 2005.

[HAMJ02] Rein-Lien Hsu, M. Abdel-Mottaleb, and A.K. Jain. Face detection in color images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):696 –706, may 2002.

[HC80] Charles F. Hester and David Casasent. Multivariant technique for multiclass pattern recognition. *Applied Optics*, 19(11):1758–1761, 1980.

[Hor82] J. L. Horner. Light utilization in optical correlators. *Applied Optics*, 21:4511–4514, 1982.

[HRBLM07] 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.

[HSP07] Bernd Heisele, Thomas Serre, and T. Poggio. A component-based framework for face detection and identification. *International Journal of Computer Vision*, 74:167–181, 2007.

[JK08] Jun-Su Jang and Jong-Hwan Kim. Fast and robust face detection using evolutionary pruning. *Evolutionary Computation, IEEE Transactions on*, 12(5):562 –571, oct. 2008.

[JLM10] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical report, University of Massachusetts, Amherst, 2010.

[JV03] Michael J. Jones and Paul Viola. Fast multi-view face detection. Technical Report TR2003-96, Mitsubishi Electric Research Laboratories, 2003.

[Kah71] William Kahan. A survey of error analysis. In *World Computer Congress*, pages 1214–1239, 1971.

[Kin03] Andrew King. A Survey of Methods for Face Detection. Technical Report 992 550 627, University of Toronto, 2003.

[KSX07] B. V. K. V. Kumar, M. Savvides, and Chunyan Xie. Correlation Pattern Recognition for Face Recognition. *Proceedings of the IEEE*, 94(11):1963–1976, 2007.

[Kum86] B.V.K. Vijaya Kumar. Minimum-variance synthetic discriminant functions. *Journal of Optical Society of America*, 3(10):1579–1584, October 1986.

[Kum92] B.V.K. Vijaya Kumar. Tutorial survey of composite filter designs for optical correlators. *Appl. Opt.*, 31:4773–4801, 1992.

[LBP95] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Proc. 5th IEEE Int'l Conference on Computer Vision*, pages 637–644, 1995.

[Lew96] M. S. Lew. Information theoretic view-based and modular face detection. In *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pages 198–203, 1996.

[LTC95] A. Lanitis, C. J. Taylor, and T.F. Cootes. An automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401, 1995.

[LZ04] S.Z. Li and Zhenqiu Zhang. Floatboost learning and statistical face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1112 –1123, sept. 2004.

[LZZ+06] Stan Li, Long Zhu, Zhenqiu Zhang, Andrew Blake, Hong-Jiang Zhang, and Harry Shum. Statistical learning of multi-view face detection. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *European Conference on Computer Vision*, volume 2353 of *Lecture Notes in Computer Science*, pages 117–121. Springer Berlin / Heidelberg, 2006.

[MGR98] S. McKenna, S. Gong, and Y. Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern Recognition*, 31(12):1883–1892, 1998.

[MK97] Abhijit Mahalanobis and B. V. K. Vijaya Kumar. Polynomial Filters for Higher Order Correlation and Multi-input Information Fusion. In *Proc. SPIE, Euro-American Optoelectronic Information Processing Workshop*, pages 221–231, 1997.

[MKC87] Abhijit Mahalanobis, B. V. K. Vijaya Kumar, and David Casasent. Minimum average correlation energy filters. *APPLIED OPTICS*, 26(17):3633–3640, 1987.

[MKH05] T. Mita, T. Kaneko, and O. Hori. Joint haar-like features for face detection. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1619–1626, oct. 2005.

[MKS+94] Abhijit Mahalanobis, B. V. K.Vijaya Kumar, Sewoong Song, S. R. F. Sims, and J. F. Epperson. Unconstrained correlation filters. *Applied Optics*, 33(17):3751–3759, June 1994.

[MKS96] Abhijit Mahalanobis, B. V. K. Vijaya Kumar, and S. R. F. Sims. Distance Classifier Correlation Filters For Multi-class Target Recognition. *Applied Optics*, 35:3127–3133, 1996.

[MSK02] B.V.K. Vijaya Kumar M. Savvides and P. Khosla. Face verification using correlation filters. In *Third IEEE Automatic Identification Advanced Technologies*, pages 56–61, Tarrytown, NY, 2002.

[NNC07] M. Nilsson, J. Nordberg, and I. Claesson. Face detection using local smqt features and split up snow classifier. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–589 –II–592, april 2007.

[Nor63] D. O. North. An Analysis of the Factors which Determine Signal/Noise Discriminations in Pulsed Carrier Systems. In *Proc. IEEE*, volume 51, pages 1016–1027, July 1963.

[OFG97] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[PHF11] S. K. Pakazad, F. Hajati, and S. D. Farahani. A face detection framework based on selected face components. *Journal of Applied Sciences*, 11(2):247–256, 2011.

[PMRR00] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 22(10):1090–1104, 2000.

[POP98] C. Papageorgiou, M. Oren, and T. Poggio. A general frame-work for object detection. In *Proc. International Conference on Computer Vision.*, 1998.

[PSZ11] Sakrapee Paisitkriangkrai, Chunhua Shen, and Jian Zhang. Face detection with effective feature extraction. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Computer Vision – ACCV 2010*, volume 6494 of *Lecture Notes in Computer Science*, pages 460–470. Springer Berlin / Heidelberg, 2011.

[PWH+11] J. Parris, M. Wilber, B. Heflin, H. Rara, A. El-Barkouky, A. Farag, J. Movellan, M. Castrilon-Santana, J. Lorenzo-Navarro, M.N. Teli, S. Marcel, C. Atanasoaei, and T.E. Boult. Face and eye detection on hard datasets. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–10, 2011.

[RBK98] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

[Ref91] Ph. Refregier. Optimal trade-off filters for noise robustness, sharpness of the correlation peak, and horner efficiency. *OPTICS LETTERS*, 16(11):829–831, June 1991.

[RK06] R.Kerekes and B. V. K. Vijaya Kumar. Correlation Filters with Controlled Scale Response. *IEEE Trans. Image Process*, 15, July 2006.

[RKK+98] A. Rajagopalan, K. Kumar, J. Karlekar, R. Manivasakan, M. Patil, U. Desai, P. Poonacha, and S. Chaudhuri. Finding faces in photographs. In *Proc. Sixth IEEE Int'l Conf. Computer Vision*, pages 640–645, 1998.

[RMA00] D. Roth, M.Yang, and N. Ahuja. A SNoW-Based Face Detector. In *Advances in Neural Information Processing Systems*, number 12, pages 855–861, 2000.

[RS89] J. Rosen and J. Shamir. Scale-Invariant Pattern Recognition with Logarithmic Radial Harmonic Filters. *Applied Optics*, 28:240–244, 1989.

[SBB03] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(12):1615–1618, 2003.

[SJ10] R. Sznitman and B. Jedynak. Active testing for face detection and localization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(10):1914 –1920, oct. 2010.

[SK98] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 45–51, 1998.

[SK00] H. Schneiderman and T. Kanade. A statistical method for 3d object detecttion applied to faces and cars. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 746–751, 2000.

[SKK02] M. Savvides, B.V.K. Vijaya Kumar, and P. Khosla. Face verification using correlation filters. In *Proc. Of Third IEEE Automatic Identification Advanced Technologies*, pages 56–61, Tarrytown, NY, 2002.

[SP98] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(1):39–51, January 1998.

[SVK03] M. Savvides and B.V.K. Vijaya Kumar. Efficient design of advanced correlation filters for robust distortion-tolerant face recognition. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 45–52, 2003.

[SVVK03] M. Savvides, K. Venkataramani, and B.V.K. Vijaya Kumar. Incremental updating of advanced correlation filters for biometric authentication systems. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 229–232, 2003.

[TLL+10] Wen-Kwang Tsao, Anthony J.T. Lee, Ying-Ho Liu, Ting-Wei Chang, and Hsiu-Hui Lin. A data mining approach to face detection. *Pattern Recognition*, 43(3):1039 – 1049, 2010.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):754–757, 1991.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of features. In *Computer Vision and Pattern Recognition (CVPR)*, 2001.

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[VSVX02] B.V.K. VijayaKumar, M. Savvides, K. Venkataramani, and C. Xie. Spatial frequency domain image processing for biometric recognition. In *IEEE International Conference on Image Processing*, volume 1, pages 53–56, 2002.

[WAHL04] B. WU, H. Ai, C. Huang, and S. Lao. Fast Rotation Invariant MultiView Face Detection Based on Real AdaBoost. In *IEEE Automatic Face and Gesture Recognition*, 2004.

[WHY09] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *In Proc. IEEE International Conference on Computer Vision (ICCV)*, 2009.

[WQXL10] Fayu Wang, Baokun Qi, Yong Xing, and Baoan Lu. Research on face detection based on combination of haar and skin colors features. In *3rd International Congress on Image and Signal Processing (CISP)*, pages 2634–2638, October 2010.

[XLZ04] Rong Xiao, Ming-Jing Li, and Hong-Jiang Zhang. Robust multipose face detection in images. *IEEE Transactions On Circuits and Systems for Video Technology*, 14(1):31–41, January 2004.

[YC97] K. C. Yow and R. Cipolla. Feature-based human face detection. *Image and Vision Computing*, 15(9):713–735, 1997.

[YH94] G. Yang and T. S. Huang. Human face detection in complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[YKA02] M.-H. Yang, D.J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transaction on PAMI*, 24(1):34–58, 2002.

[YW96] J. Yang and A. Waibel. A real-time face tracker. In *Proc. Third Workshop Applications of Computer Vision*, pages 142–147, 1996.

[ZV08] C. Zhang and P. Viola. Multiple-Instance Pruning for Learning Efficient Cascade Detectors. In J. Platt, D. Kroller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 1681–1688, 2008.

[ZZ10a] Cha Zhang and Zhengyou Zhang. *Boosting-Based Face Detection and Adaptation*. Morgan and Claypool Publishers, 2010.

[ZZ10b] Cha Zhang and Zhang Zhengyou. A survey of recent advances in face detection. In *Technical Report, MSR-TR-2010-66*. MIcrosoft Research, Microsoft Corporation, June 2010.