DISSERTATION

THEORY AND APPLICATIONS OF OPTIMIZED CORRELATION OUTPUT FILTERS

Submitted by

David S. Bolme

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2011

Doctoral Committee:

Advisor: J. Ross Beveridge

Bruce A. Draper
Michelle Mills Strout
Michael J. Kirby

ABSTRACT

THEORY AND APPLICATIONS OF OPTIMIZED CORRELATION OUTPUT FILTERS

Correlation filters are a standard way to solve many problems in signal processing, image processing, and computer vision. This research introduces two new filter training techniques, called Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE), which have produced filters that perform well on many object detection problems. Typically, correlation filters are created by cropping templates out of training images; however, these templates fail to adequately discriminate between targets and background in difficult detection scenarios. More advanced methods such as Synthetic Discriminant Functions (SDF), Minimum Average Correlation Energy (MACE), Unconstrained Minimum Average Correlation Energy (UMACE), and Optimal Tradeoff Filters (OTF) improve performance by controlling the response of the correlation peak, but they only loosely control the effect of the filters on the rest of the image. This research introduces a new approach to correlation filter training, which considers the entire image to image mapping known as cross-correlation. ASEF and MOSSE find filters that optimally map the input training images to user specified outputs. The goal is to produce strong correlation peaks for targets while suppressing the responses to background. Results in eye localization, person detection, and visual tracking indicate that these new filters outperform other advanced correlation filter training methods and even produce better results than much more complicated non-filter algorithms.

ACKNOWLEDGEMENTS

It is a pleasure to thank those who have made this thesis possible, especially my wife and children, Christine, Jonathan, and Matthew for being supportive of my education. I would also like to thank my parents and sister, Nancy, Jerry, and Cindy, for their guidance. I am grateful for the educational and professional guidance of my adviser Ross Beveridge and others in the computer vision group at CSU, especially Bruce Draper, Yui Man Lui, and Steve O'Hara. I appreciate the assistance the other faculty, staff, and students that have helped me during this project.

David Bolme

TABLE OF CONTENTS

LIST OF ACRONYMS

ASEF Average of Synthetic Exact Filters

CAM Shift Continuously Adaptive Mean Shift

DCCF Distance Classifier Correlation Filters

DOG Difference of Gaussians

DSP Digital Signal Processors

FERET Face Recognition Technology

FFT Fast Fourier Transform

GDA Generalized Discriminant Analysis

GPU Graphics Processing Units

IVT Incremental Visual Tracking

KRR Kernel Ridge Regression

LK Lucas-Kanade

LOG Laplacian of Gaussian

MACE Minimum Average Correlation Energy

MACH Maximum Average Correlation Height

MIL Multiple Instance Learning

MOSSE Minimum Output Sum of Squared Error

MSESDF Minimum Squared Error Synthetic Discriminant Functions

MVSDF Minimum Variance Synthetic Discriminant Functions

OAB Online Ada-Boost

OCOF Optimized Correlation Output Filters

OTF Optimal Tradeoff Filters

PSR Peak-to-Sidelobe Ratio

SDF Synthetic Discriminant Functions

SIFT Scale-Invariant Feature Transform

SSE Sum of Squared Error

SURF Speeded Up Robust Features

SVM Support Vector Machine

UMACE Unconstrained Minimum Average Correlation Energy

Chapter 1

Introduction

Filtering is a basic operation used to solve many problems in computer vision and signal processing. One of the first lessons taught in textbooks and related courses is how to use filtering to perform simple signal processing or image processing tasks including noise removal, edge enhancement, derivative estimation, and object detection. Because filtering has so many uses, it plays both small and large rolls in many application areas including audio processing, video processing, and higher dimensional signal analysis such as medical tomography or multispectral image processing.

This thesis focuses on the applications of filtering to the task of detecting an object in an image or video sequence. Finding objects in images is an important and challenging computer vision problem. This problem can take many different forms but usually includes determining if an object is present in an image or where in the image an object is located. Although it is a simple concept, object detection tasks can be difficult to solve because the appearance of objects can change substantially due to simple changes in the conditions under which the object is viewed. These include changes in pose, lighting, nonrigid deformation, or natural variations within an object's class. For these reasons, designing and training algorithms that perform these tasks will remain an open problem for the foreseeable future.

Filtering is frequently used to correlate one signal with another, and correlation has found many applications in both signal and image processing because it is both simple and fast. For object detection, images are correlated with filters that are cropped examples or

templates hand selected from training images. Because correlation provides a similarity score for every pixel in the image, it can be used to detect both the presence and location of an object. The object is present where the correlation output exceeds a threshold and the local maximum provides an estimate of the location.

This technique rarely works for challenging detection problems because templates fail to adequately represent variations in appearance and poorly discriminate in the presence of a complex background. For these reasons, the majority of object detection research has focused on designing more complicated object representations and more robust classification schemes. While these techniques often improve accuracy, the improved performance comes at the expense of longer run times and complex up-front training protocols.

For this thesis, I have developed a fundamentally new way to design filters that is much better at discriminating between targets and background while retaining the speed and simplicity of correlation filters. The basic idea behind this technique is to learn filters that optimally map input images to their ideal output. By training filters in this way, they produce high responses for target objects while also learning to suppress the response to common background distractors. As is shown in [8, 9, 7], this technique is fast, easy to implement, has many advantages over alternative filter training techniques, and even outperforms complex appearance models and detectors.

While the filters do outperform many object detection techniques, they essentially perform image-to-image matching and therefore have many of the standard limitations of image to image matching techniques. Namely, producing a filter that can match across large pose changes or object deformations is extremely difficult. Therefore, these filters should work well where the targets appearance is predictable, but techniques like using multiple templates for each scale, rotation, or pose of an object are required when the objects need to be recognized under varying conditions.

1.1 Template Matching

In template matching, object appearance is captured using sample images. Typically, a template (or filter) is just an example image that has been carefully cropped and centered on the target object. In this context, the terms "template" and "filter" may be used interchangeably; however, in this dissertation the term "template" will be reserved for images that are cropped and centered examples of a target, and "filter" will be used for more advanced filters that have been trained on multiple example targets.

Template matching as an approach to object recognition also carries with it a straightforward concept of similarity based upon comparing pixels. To be specific, the similarity of one image to another is measured using a simple dot product. If we define h to be a template, and f is a vector form of an image, the similarity of those two images is computed as the dot product of the pixel values:

$$S(f,h) = f \cdot h = \sum_i f_i h_i$$

where i indexes the pixels in those images. In theory, if the two images are of the same object viewed in the same manner, then the pixel values will be similar and the resulting dot product will results in high values. On the other hand, if the two images are of different objects, the pixel values will be different and the dot product will produce small values. Therefore, objects can be detected if this dot product exceeds a threshold, in this case S_{thresh}.

$$S(f,h) > S_{thresh}$$

In object detection it is often unknown where an object will appear in an image. To find the object, the template is correlated with the image.

$$C = f \star h$$

Correlation computes that dot product for each possible alignment of the template relative to the image. The output of correlation is a new image, C, where each pixel value

in C contains the result of the dot product between a translated version of f and the template h. Therefore, each pixel of C indicates how similar that region of image f is to the template. Object detection is then performed by finding pixels of C that exceed the threshold. These local maxima are often referred to as correlation peaks, and they indicate the presence and location of the target object.

For object detection, correlation has a number of attractive properties:

- **Correlation based object detection is easy to implement.** Because correlation is commonly used in signal and image processing, correlation libraries are available in most languages and for most hardware platforms including Digital Signal Processors (DSP) and recent Graphics Processing Units (GPU). Correlation is even easy to implement from scratch, although performance will most likely be slower than that which is achieved when using specialized libraries.

- **Correlation is fast.** Because correlation is a highly regular computational operation, it can be computed very efficiently on modern processors. Furthermore, the dot product and templates used in correlation are also much simpler than many of the alternative classifier functions and appearance models that are used in many alternative object detection techniques. In addition, using smart algorithms based on the Convolution Theorem [62] or separable convolution [23] can speed up this computation even more. This results in extremely fast detection algorithms.

- **Detection is fast and simple.** After computing the correlation image, the detection step is simple to implement and fast to execute; all that is required is to scan the correlation image for one or several local maximums.

The ability of template matching to produce good peaks depends upon a number of factors including the following:

1. Objects of the same class have similar appearance.

2. The images of those objects are taken under similar conditions.

3. Objects of different classes have different appearance and therefore produce different patterns of pixels.

For images where these constraints are met, template matching will perform very well. Typically these tend to be situations where all aspects of the imaging process can be carefully controlled, such as monitoring products on an assembly line. Another example is when the two images being compared are taken of the same object, are taken at approximately the same time, and are taken from approximately the same viewpoint. Real world tasks where these conditions are met include image registration, manufactured parts inspection, and video stabilization.

This is illustrated in Figure 1.1.1. Row one of the figure shows the result of auto correlation where the image of the butterfly is correlated with itself. This results in a nice bright peak in the center of the output where the image and the template are aligned. As a result, automatically determining that a butterfly is present in this image is easy. Row two of this figure also illustrates the performance of simple template matching, which degrades rapidly as the pixel values change. Here the input image has been altered by small changes to its size and rotation. To the human eye the images are still very similar; however, the peak in the correlation output is not nearly as strong nor as compact. It is therefore more difficult to determine the presence or location of the butterfly.

My new Optimized Correlation Output Filters (OCOF) correlation filter training methods, particularly Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE), produce filters that overcome these problems. Specifically, one of the primary limitations of template matching is that there are a number of ways that images of objects can violate Condition 2. These changes can include scale, rotation, pose, illumination, or non-ridged deformation. As a result correlation with a simple template does not produce consistent peaks. To complicate matters, Condition 3 does not always hold; other classes of objects often produce similar patterns of pixels to the target objects and lead to false peaks. For these reasons, template matching is rarely used for challenging object detection scenarios.

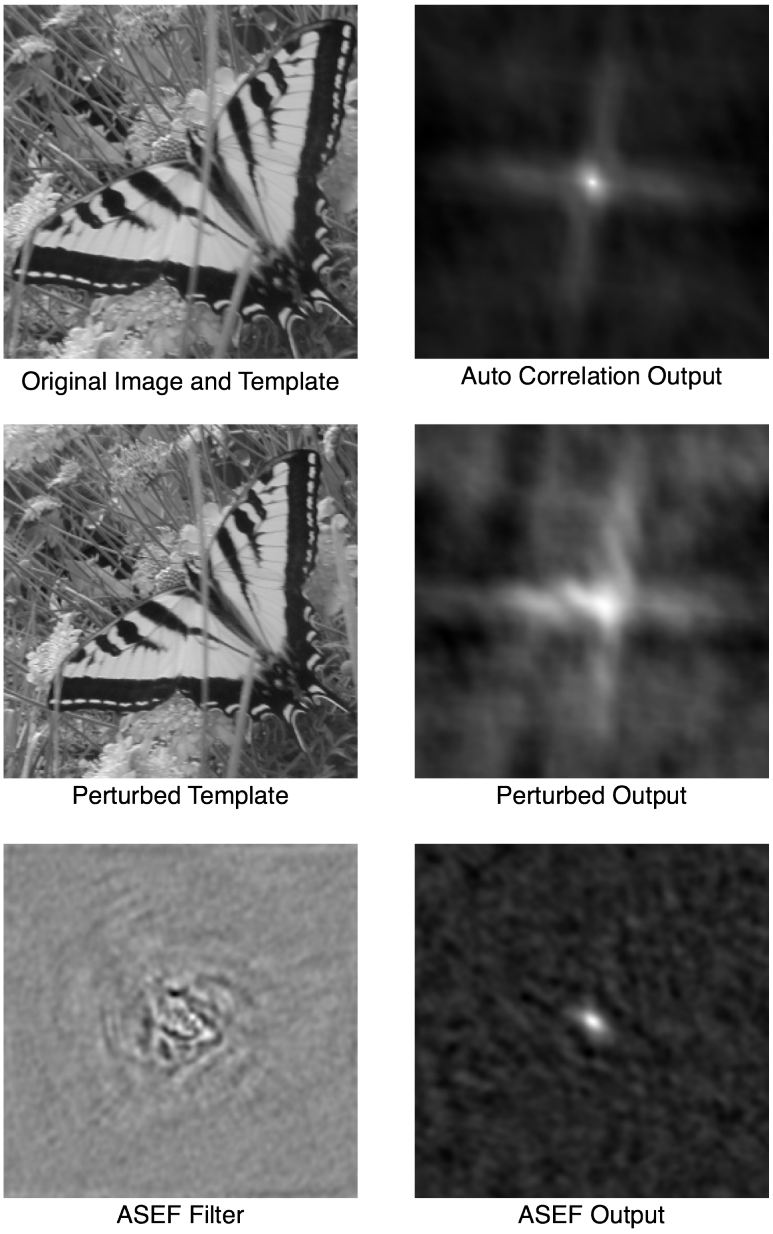

Figure 1.1.1: Detecting an object requires a filter to produce a strong peak for a target while suppressing activation of the background. Top: Auto-correlation is when the filter is identical to the original image; a good peak is produced. Middle: When the filter is perturbed by a only a small amount, detection may fail. Bottom: It is possible to find much better filters using techniques such as those that are introduced in this work.

1.2 A New Method For Producing Filters

Template matching relies on the similarity of cropped images to produce filters and this concept has led to much of the work on correlation filters over the past 30 years. The methods introduced in this thesis, referred to as OCOF, do not count on the natural similarity of template images but instead find patterns in the pixels produced by target objects that are consistent even under varying imaging conditions and also that are capable of distinguishing target objects from other possibly similar objects. The idea behind these new filters is relatively simple. The filter should produce a peak (or a value of one) where a target is present and should produce values near zero for all other locations in the image. Next, for a set of training images, it is possible to synthetically generate the desired output of the filter images with peaks in the appropriate locations. Finally, given the training images and the output images, the correlation operation will be inverted to produce an optimal filter.

OCOFs are based on a different set of conditions than those presented in the previous section. To be specific, these assumptions are as follows:

1. Objects of the target class have similar appearance and therefore produce similar patterns in images.

2. There are patterns in the pixels produced by target objects that are consistent even when that object is imaged under changing conditions.

3. There are patterns in the pixels produced by target objects that distinguish them from other objects and background.

The methods discussed here find unique patterns in that appearance. It should not be surprising that in most cases the filters learned by processes such as ASEF are recognizably associated with their target objects; however, the filters emphasize features of the target that help to distinguish it from background or other similar distractors. A filter that exploits these patterns is more tolerant of common appearance changes and can better discriminate between targets and background.

One requirement for this technique is that a large number of training images be available. These training images provide examples of the variations in appearance that the filter should tolerate. For the butterfly example in Figure 1.1.1, hundreds of training images have been generated from the single source image by artificially introducing small rotations, scales, and translations. Introducing these artificial perturbations is helpful because the filters are tolerant of small rotation and scale changes in the target. In most applications the filters also need to train on images that include lighting changes, in-depth rotation, and possibly even different species of butterflies.

As can be seen in the bottom row of Figure 1.1.1, the ASEF filter looks different from the original image. This filter was trained on many perturbations of the original image by generating the ideal output for each perturbation. The resulting filter has identified patterns that are consistent in all of those images and therefore produces good peaks under a variety of imaging conditions. The filters described in this thesis produce even better peaks than auto-correlation. For more details on how ASEF filters are trained see Chapter 3.

1.3 Summary of Research

The research conducted for this thesis investigates OCOFs and their application to three object detection problems. In general, this thesis presents the theory of OCOF filters. Results are compared to other correlation filters and more complex object detection techniques. Three application areas are investigated including eye localization, person detection, and visual tracking. This section presents a short summary of the research.

Chapter 2 provides a short tutorial level introduction to correlation and discusses the major development of correlation filters over the past twenty years. This includes filter training methods such as Minimum Average Correlation Energy (MACE) [54] and Unconstrained Minimum Average Correlation Energy (UMACE) [53]. The comparison with UMACE is particularly interesting because of the similarities between the two methods (See Table 2.2). In fact, it is shown in Chapter 3 that UMACE is just a specialization

of MOSSE. Chapter 2 also discusses many of the major research trends in the area of object detection that on on complex appearance models or powerful nonlinear classifiers.

Chapter 3 introduces the theory for producing OCOF correlation filters. This includes the basic technique and theory behind OCOFs and provides insights into why averaging over multiple training images is important and how many training images are needed to produce a good filter. This chapter also introduces the use of affine perturbations for increasing the training set size and regularization for improving the numerical stability of the filter training process.

Chapter 4 investigates the problem of eye localization on the FERET data set [60]. It shows that OCOF filters can localize the eyes to within the region of the iris 25% more often than other correlation filter training techniques when searching the entire face. The OCOF filters also improve localization by 2% over more complex eye detection algorithms while maintaining the speed benefits of the correlation based approach. In addition, this chapter presents experiments that show the effect of regularization on ASEF and MOSSE and shows that MOSSE converges to better filters with fewer training images.

Chapter 5 presents results on person detection where the task is to indicate the presence and location of people in images from the PETS 2009 data set [24]. In this case the solution based on OCOF filters are as at least 5% more accurate and three times faster than other algorithms that represent the state-of-the-art in person detection. This chapter also shows that much of the power of MOSSE filters come from their ability to train on larger images with multiple targets. This is done by comparing two versions of MOSSE, one trained on large training images and the other trained on cropped and centered templates.

Chapter 6 conducts tests on visual tracking, which examine a filter's abilities to incrementally train on small numbers of images and then track objects from frame to frame in a video sequence. This chapter introduces a new data set and evaluation protocol for comparing the ability of trackers to follow a target through many challenging video sequences. Results indicate that the filter based approach is over 10 times faster than

more complex techniques and shows a 10% increase in the number of frames tracked through complex appearance changes and occlusions.

Finally, Chapter 7 summarizes the major conclusions of this thesis. It also suggests some areas for future research including filters that operate on color images, and using the OCOF process to produce specialized edge detectors.

Chapter 2

Background

2.1 Understanding Correlation

The term correlation is used to describe many different operations in mathematics, statistics, and signal processing. Basically, correlation is a measure of similarity typically associated with an inner-product of two functions; however, depending on the context, "correlation" can refer to different mathematical operations. For example, there are many correlation measures in statistics that measure the dependence of two variables. Of these, the most popular is a normalized correlation coefficient known as Pearson's Correlation (first introduced by Galton in 1888 [14]). This text, however, adopts the definition more commonly used in mathematics and signal processing that may be better known as cross-correlation.

Correlation of a signal f with a filter h is typically denoted using the operator $\star$:

$$g = f \star h$$

This process produces a third function g, which is often considered a modified version of f. The discrete one-dimensional correlation of a function f and a filter h is typically defined as:

$$(f \star h)[x] = \sum_{dx=-\infty}^{\infty} f[dx]h[x + dx] \tag{2.1.1}$$

This work is focused on image analysis and will mostly consider the two-dimensional correlation defined as:

$$(f \star h)[x, y] = \sum_{dx=-\infty}^{\infty} \sum_{dy=-\infty}^{\infty} f[dx, dy]h[x + dx, y + dy]$$

where f and h are two-dimensional arrays representing images. The math as defined here can also be generalized to multiple dimensions or continuous functions.

Correlation is a similar operation to convolution, which can be computed as a correlation with a reversed signal. Convolution of functions f and h is defined as:

$$g = f * h$$

$$(f * h)[x] = \sum_{dx=-\infty}^{\infty} f[dx]h[x - dx]$$

Often the terms convolution and correlation are used interchangeably. If the filter is symmetric, as is the case with many image processing applications, the two operations are equivalent.

2.2 The Convolution Theorem

While this work is primarily concerned with correlation, many derivations and proofs will use the Convolution Theorem as a way to simplify the mathematical analysis and to speed up the computation of the filters. The Convolution Theorem states that the convolution of two functions in the spatial domain can be computed in the Fourier domain as the element-wise multiplication of the Fourier Transform of those two functions. To be specific, the convolution of functions f and h can be performed in the Fourier domain as follows:

$$f * h = \mathcal{F}^{-1}(\mathcal{F}(f) \odot \mathcal{F}(h)) = \mathcal{F}^{-1}(F \odot H)$$

where $\mathcal{F}$and $\mathcal{F}^{-1}$ are the Discrete Fourier and the Discrete Inverse Fourier Transforms. Here we also introduce some notation. Capital letters denote the Fourier Transform of their lower case counterparts, for example $F = \mathcal{F}(f)$. We also introduce the operator $\odot$, which will be used to explicitly denote an element-wise multiplication.

The Convolution Theorem can also be used to compute correlation. In the Fourier domain, taking the complex conjugate of one of the signals will actually reverse that signal:

$$f[t] = \mathcal{F}^{-1}(\mathcal{F}(f)^*)[-t]$$

This enables us to write correlation very simply as follows:

$$f \star h = \mathcal{F}^{-1}(F \odot H^*)$$

In the following sections and chapters many of the computations will be presented only in the Fourier domain. For example,

$$g = f \star h$$

will appear as,

$$G = F \odot H^* \tag{2.2.1}$$

It is assumed that the images or signals have already been transformed into their Fourier counterparts using the Discrete Fourier Transform.

The next section will discuss how the Convolution Theorem can be used to efficiently compute correlation, which is extremely useful when performing real-time signal processing or computer vision. As is shown in the rest of this paper, the Convolution Theorem will also be used to simplify the mathematical analysis when training advanced correlation filters. This is necessary because the spatial convolution as presented in Equation 2.1.1 defines a complex relationship between the image, filter, and output, where as in the Fourier convolution in Equation 2.2.1 there is a much simpler element-wise relationship. As we are about to explore in the next section, many of the operations upon which the techniques introduced in this thesis depend become more computationally efficient when carried out in the Fourier domain.

2.3 Fast Correlation

The Convolution Theorem is often used in signal processing because it is an efficient way to compute correlation. For the one-dimensional case, correlating a signal of length N

with a filter of length P in the spatial domain takes $O(NP)$ computations. Likewise, correlating a two-dimensional image of size $N \times M$ and with a filter of size $P \times Q$ takes $O(NMPQ)$ operations. Using the Convolution Theorem this can be sped up greatly. Using a Fast Fourier Transform (FFT), such as the Cooley and Tukey algorithm [19], the FFT can be computed in $O(N \log N)$ time. The element-wise multiplication is computed using $O(N)$ computations, and the inverse FFT is computed in $O(N \log N)$ computations. This results in a total time of $O(N \log N)$ for the one-dimensional case or $O(NM \log(NM))$ for images. Because images can easily contain thousands or millions of pixels, computing convolution via the FFT can reduce the time needed by many orders of magnitude.

In practice, using the FFT is the most efficient way to compute convolution when the size of the filter is large, but it does require some care to produce good results. Correlation when computed using the FFT is actually computing a circular cross-correlation where the image is mapped to the topology of a torus. In other words the left and right edges, and the top and bottom edges are artificially connected to each other. Because of this, a couple techniques are used to reduce the edge effects produced from the artificial connection. The first is to pad the edges of the image and filter with additional pixels in order to produce better correlation estimates up to the edge of the image. The second technique, which is used in this research, is to ignore correlation values near the edge and use a cosine window to smoothly reduce the values near the edge to zero.

On the other hand, many image processing operations can be computed using small 3×3 or 5×5 filters. For small filters it is often faster to compute correlation in the spatial domain. Furthermore, many image processing filters are separable, meaning they can be broken into two one-dimensional filters. For example, a 3×3 Sobel operator can be split into a 3×1 filter and a 1×3 filter. The correlation can then be computed by correlating with two one-dimensional filters for a further time savings. It is therefore necessary to understand under what circumstances each correlation method, FFT, Spatial, and Separable, is most appropriate.

Table 2.1: A comparison of the computational complexity of three different convolution methods.

Convolution Type	Complexity
Spatial Convolution	$O(MNPQ)$
Separable Convolution	$O(MN(P+Q))$
FFT Convolution	$O(MN\log(MN))$

Table 2.1 compares the computational complexity of each convolution method. As can be seen, the complexity actually depends on the relationship between the filter size and the log of the image size. Theoretically if the filter is small compared to the image, the spatial or separable methods will be faster. As the filter gets larger the Fourier method will dominate because it does not depend on the size of the filter. This is illustrated in Figure 2.3.1. It is also apparent that if the filter size is fixed and the image size increases, at some point the spatial methods will have an advantage. This was tested in Figure 2.3.2. In that figure the filter size was held constant at 8×8 while the image size was increased. In this case it shows that there is very little advantage to spatial convolution even using this small filter size. It is also notable that the separable convolution can be significantly faster than the full spatial convolution. Unfortunately, this only works for a special type of filter that can be separated into two one-dimensional filters. For object detection, which is the topic of this dissertation, filters will typically be large such that the FFT method will have a clear advantage.[1]

2.3.1 Correlation Kernels and Image Processing

Simple filters can be used for many tasks such as signal denoising. For example, consider Figure 2.3.3. Here a one-dimensional signal was constructed using a sinusoid with added noise. A common technique for reducing this type of noise is to correlate the signal with a smoothing filter. As shown in the red line, correlating the signal with a Gaussian filter effectively removes the noise.

[1]Performance results were run in python on a 2.4 Ghz Mac Book Pro using the accelerated scipy convolution routines and fftpack for Fourier transforms.

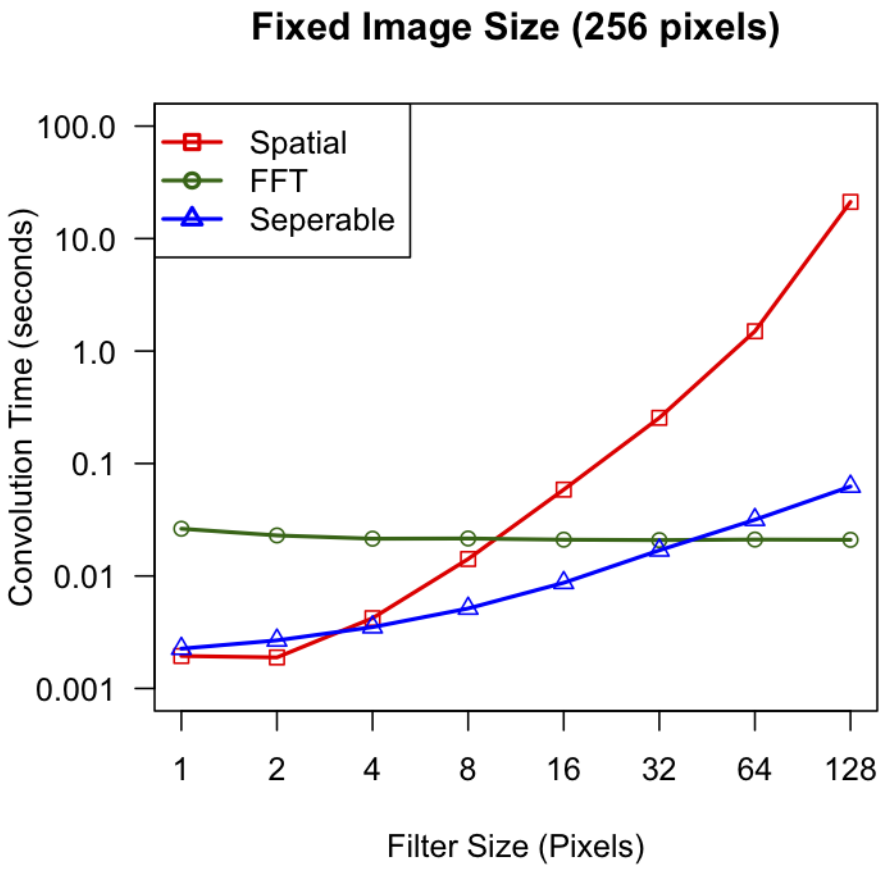

Figure 2.3.1: Correlation time for changing filter size. This shows that filter size has no effect on the time needed for FFT based convolution because small filters need to be padded to the same size as the image before correlation.

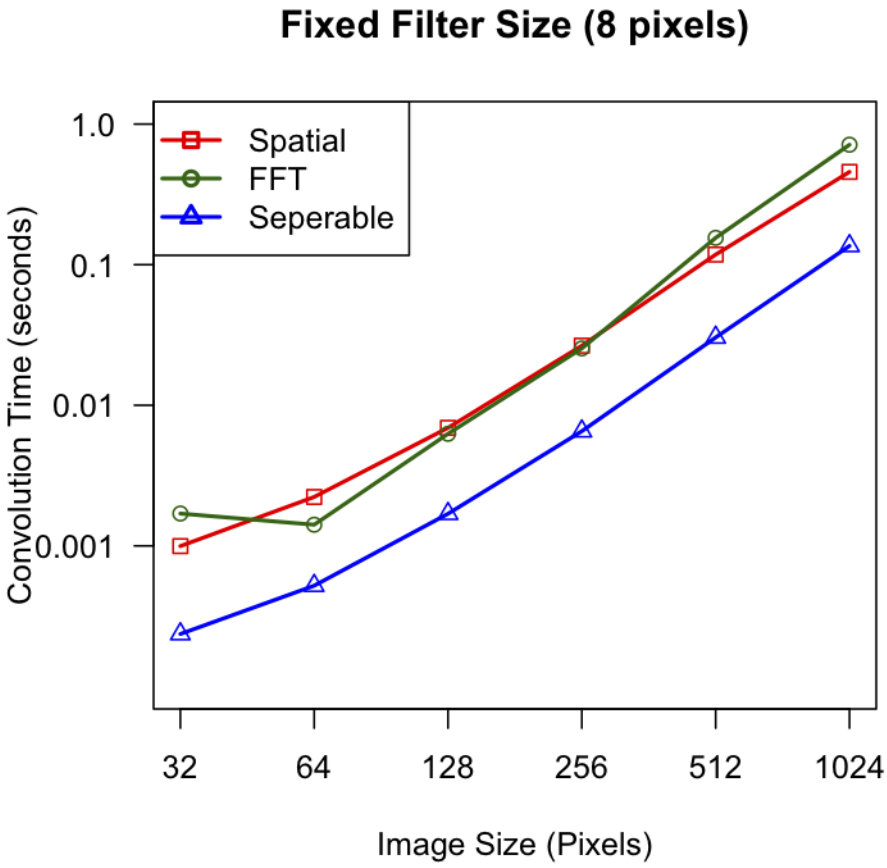

Figure 2.3.2: Convolution time for changing image size

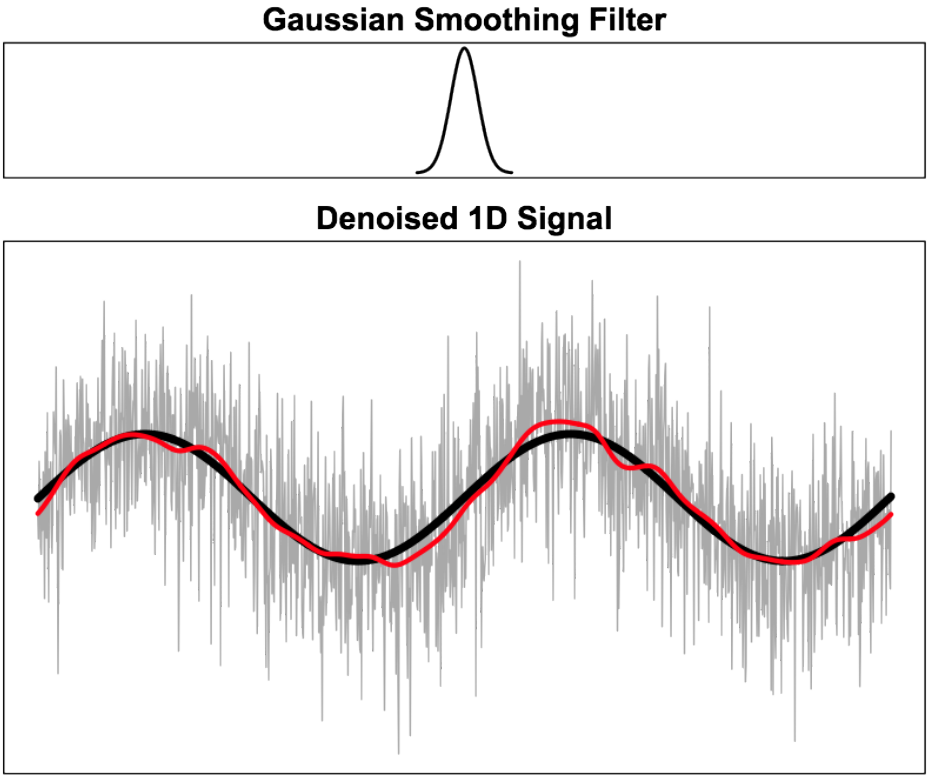

Figure 2.3.3: This demonstrates signal denoising using a smoothing filter. The black line is the original signal, the gray line is the signal with added normally distributed noise, and the red line shows the result of correlating the noisy signal with the Gaussian filter.

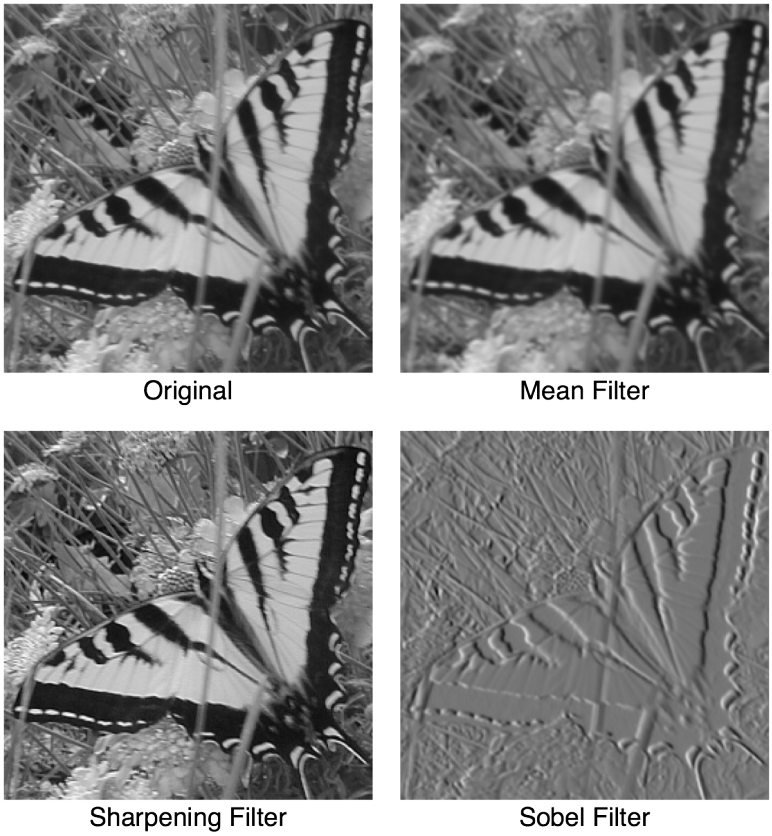

Figure 2.3.4: This shows the effect of smoothing, sharpening, and edge detection using the Sobel operator.

The same principles apply to image processing, however the images and filters are two-dimensional structures. Many tasks in image processing are accomplished using simple 3×3 filters. Figure 2.3.4 shows an image that is correlated with three common filters. The Mean Filter replaces the pixel values with the mean of the 3×3 surrounding window, which smooths the image:

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The Sharpen Filter boosts the high frequencies in an image. These frequencies often correspond to edges and the result is that the image appears sharper. This filter takes the form:

$$h = \frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 16 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

The Sobel Filter is often used to approximate the image derivative and is very useful for tasks like edge detection. In this case, the derivative is computed in the x direction and the filter takes the form:

$$h = \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

While these filters can enhance the quality of images, they are not useful for detecting objects.

2.4 Detection in One-Dimension

In signal processing, correlation is often used to compute the similarity of two signals taking into account the time-lag between those signals. Here an example is given in one-dimension but the concept is the same for two dimensional images. This is most often used to detect a short signal within a longer signal where the location of the shorter signal is not known. For example, consider sonar where a short tone is generated, it bounces off a distance object and then is picked up by a microphone near the source.

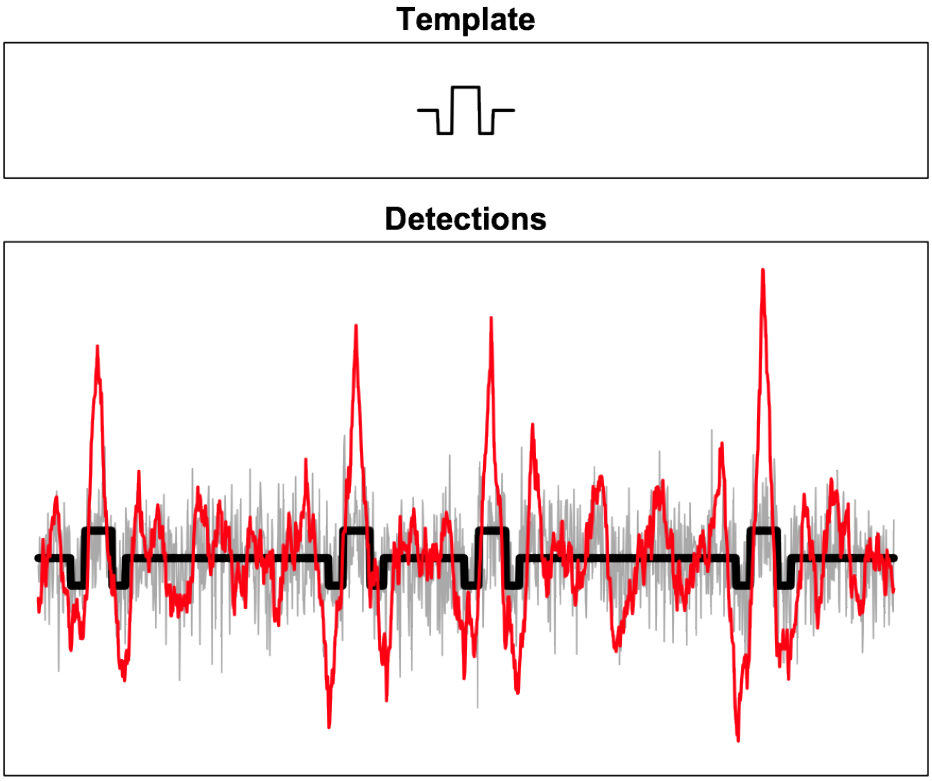

Figure 2.4.1: Using correlation to detect patterns in the presence of noise.

The distance to the object is directly proportional to the time required for the pulse to bounce off the object and return to the source. By correlating the returning signal with a sample of the original tone, it is possible to determine the presence of that pulse and the time at which that pulse returns. This is illustrated in Figure 2.4.1, which shows an example of a short pulse (top). That pulse was embedded 4 times in a one-dimensional signal (black line). White noise was added to that signal (grey line). Finally, the noisy signal was convolved with a template of the original pulse (red line). The result shows that there are four strong peaks associated with the location of the pulse in the original signal. The presence and location of a pulse can be detected by finding any correlation peak that exceeds a threshold.

2.5 Object Detection with Simple Templates

A similar technique can be used to detect an object in two dimensional images. Correlating images with a simple template is one of the easiest ways to detect objects. The filter used in this type of algorithm is usually a cropped portion of a training image. This only performs well when there is little difference between the template and target object. For this reason, templates are rarely used for object detection in cases where the object undergoes a significant appearance change.

Simple templates are successfully used in many real world applications. Some examples are video stabilization where a cropped portion of a video frame is matched to the next frame in the sequence. In this case the location of the peak indicates the shift in the location of that part of the image and can therefore be used remove that motion from the frame and stabilize the image. A similar task is estimating the distance to an object using a pair of stereo cameras. In this case, portions of the two images are matched and the difference in the locations of the object in those two images can be used to triangulate its location relative to the cameras. Simple template matching works well for these applications because there is little difference in the appearance of the object because the images that are correlated are collected at almost the same time from approximately the

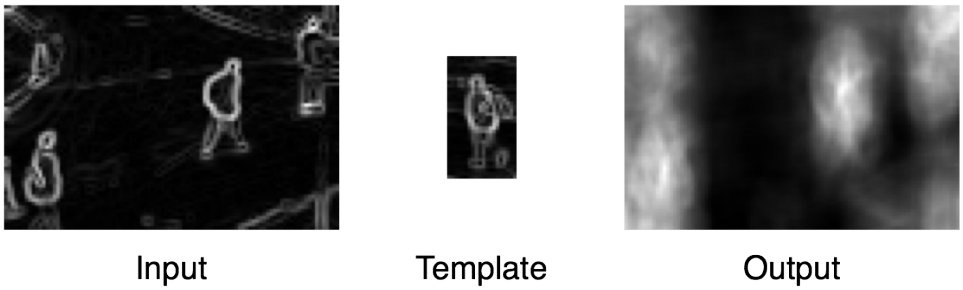

Figure 2.5.1: This is the result of correlating an image with a simple template. Responses to the the people are strong but are not compact and the output also contains some strong false peaks for non-target objects.

same view point.

An example application where simple templates fail would be person detection. In this task, a template is selected by cropping a template from a training image. The filter is then applied to another image taken later in the day. This task is much more difficult because there are many things that change about the appearance of the people in the video. First, the person used for the template is not the same person to be detected in the new image. The person can vary in height, weight, gender, and clothing. In addition, the pose of that the person, illumination, and many other factors can cause a change in appearance from the training image to the testing image. As a result, the template is often a poor representation of a person's appearance.

To be concrete, consider Figure 2.5.1. Here images of people have been preprocessed to extract edges. This removes much of the appearance variation due to clothing and pose. A simple template is cropped from a training image for use as a filter and is correlated with a testing image. The result shows soft smeared out peaks where people are present in the testing image; however, there is also a peak in the upper left corner of the output where no person is present. To reliably detect people, the detector needs to be able to distinguish the peaks produced by people from the peaks produced by background.

2.6 Overview of Advanced Correlation Filters

There are a number of good approaches to producing filters. This section will introduce those techniques and discuss the advantages and disadvantages of each method. All of the training methods discussed in this research produce a filter that can be used for object detection, but there are differences in each method that effect how the filters are trained and ultimately affect object detection accuracy. A comparison of these differences can be found in Table 2.2.

A large family of correlation filters has been developed that improves the response to a variety of input stimuli. Chronologically, Synthetic Discriminant Functions (SDF) [32] filters were introduced first; they respond well to positive training images while suppressing responses to negative training examples. Next, Minimum Variance Synthetic Discriminant Functions (MVSDF) [44] filters, Minimum Average Correlation Energy (MACE) [54] and then Optimal Tradeoff Filters (OTF) [63] were introduced. These refine aspects of the filter design to improve performance relative to noise and spatial resolution of response. All four of these methods are similar in the way that they are trained. Specifically, they all require a zero/one (target/non-target) constraint on each training image. It has been found that these hard constraints are unnecessary and can even be detrimental for producing robust correlation filters[53]. Unconstrained correlation filters such as Maximum Average Correlation Height (MACH) and Unconstrained Minimum Average Correlation Energy (UMACE) [53, 71] relax these constraints and instead favor high correlation responses on the average training image.

2.6.1 Synthetic Discriminant Functions: MVSDF, MACE, OTF

SDFs [32] encompass a family of correlation filters that are characterized by hard constraints on the output peak. This line of research has led to the popular MVSDF [44], MACE [54] and then OTF [63] correlation filters. For a set of training images $\mathbf{x}_i$ and a filter $\mathbf{h}$, the desired output of the filter is captured in a variable u_i. For positive training examples, $u_i = 1$ and for negative examples $u_i = 0$. The corresponding constraint for a

	Templates	MACE	UMACE	OCOF
Peak Shape	Uncontrolled	Sharp*	Sharp	User Defined
Easy Training	X		X	X
On-line Adaptive Training	X		X	X
Does Not Over-fit Training Data	X		X	X
Hard Constraints on Peaks		X		
Trains on Full Images				X
Considers Entire Training Output				X
Multiple Targets Per Image				X

* A variant of SDF called MSESDF[45] does provide a method to control the shape of the response around the peak but does not have the same amount of control that is demonstrated by OCOF.

Table 2.2: A comparison of correlation filter training strategies.

single training image is:

$$u_i = \mathbf{h}^\top \mathbf{x}_i \tag{2.6.1}$$

Because there are fewer constraints than pixels in the filter, there are multiple filters that will satisfy these constraints. To produce a single filter, SDFs impose additional constraints by requiring the filter to be a linear combination of the training images, while MACE requires the filter to minimize the average output energy over the training set. Under this process the filter h is defined as:

$$\mathbf{h} = \mathbf{D'}^{-1}\mathbf{X}(\mathbf{X}^\top\mathbf{D'}^{-1}\mathbf{X})^{-1}\mathbf{u} \tag{2.6.2}$$

The distinction between SDF, MVSDF, MACE, and OTF filters lies entirely in how $\mathbf{D'}$ is defined.

An MVSDF suppresses frequencies corresponding to noise. To do this we define $\mathbf{D'} = \mathbf{C}$ where $\mathbf{C}$ is a diagonal matrix such that each element of the diagonal corresponds to the power spectrum of the noise. MVSDF filters typically emphasize lower frequencies that suppress noise, but this also has the effect of producing smoother peaks that are more difficult to detect. In practice, estimating the "color" of noise is a difficult task; so in most cases white noise is assumed [71], in which case $\mathbf{C}$ becomes the identity matrix ($\mathbf{C} = \mathbf{I}$).

MACE produces sharp detectable peaks by minimizing the average correlation plane energy for the training set. For MACE $\mathbf{D'} = \mathbf{D}$ where $\mathbf{D}$ is a diagonal matrix containing the average power spectrum of the training images. MACE filters typically emphasize

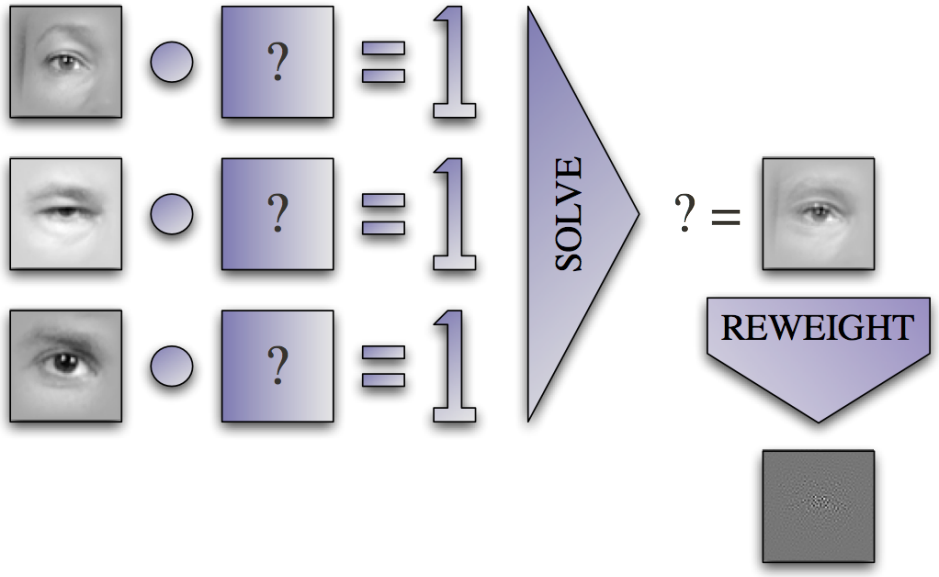

Figure 2.6.1: This figure shows how a MACE filter for detecting eyes is trained. A set of linear equations are generated, which set the height of the peak to the value 1. The filter is a solution to those equations where the frequencies have been weighted to produce sharp peaks.

high frequencies. This produces sharp peaks but also makes the filter much more sensitive to noise.

Finally, OTF finds an optimal balance between the properties of MVSDF and MACE.

$$\mathbf{D}' = D\alpha + \mathbf{C}\sqrt{1-\alpha^2} \tag{2.6.3}$$

This introduces the parameter α, which is used to tune the filter between the noise tolerance of MVSDF and the sharp, easily detected peaks of MACE. In the case that $\alpha = 0$, $\mathbf{D}'$ becomes the identity matrix and the filter is equivalent to the SDF filter. If $\alpha = 1$, $\mathbf{D}' = \mathbf{D}$ and the OTF filter is equivalent to MACE.

2.6.2 MACH and UMACE

The second type of optimal trade-off filter is an unconstrained filter called UMACE. Instead of requiring the filter to satisfy a set of hard constraints on the correlation output, UMACE only requires a high average response to the training examples. The resulting

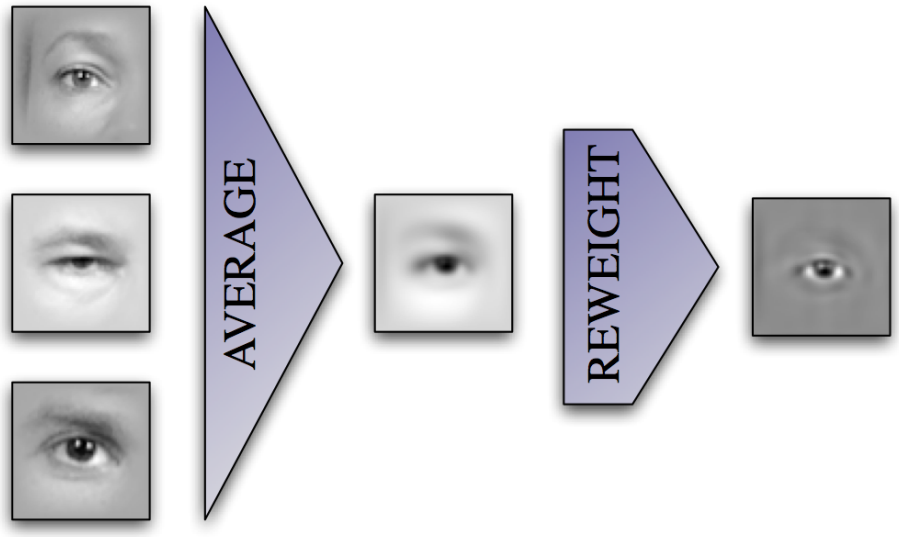

Figure 2.6.2: This figure shows how a UMACE filter for detecting eyes is trained. First a set of templates are averaged and then the frequencies are reweighted in order to produce sharp peaks.

filter is defined as:

$$\mathbf{h} = \mathbf{D'}^{-1}\mathbf{m} \tag{2.6.4}$$

where $\mathbf{m}$ is the average of the columns of $\mathbf{X}$ (or the average training image) and $\mathbf{D'}$ is the same as defined in Equation 2.6.3. Here the filter for $\alpha = 0.0$ becomes the average of the training images, and for $\alpha = 1.0$ the UMACE filter.

The MVSDF, MACE, and OTF filters mentioned above are all based on similar assumptions and have many of the same issues. Each training image is given a single "synthetic correlation value", which is the value the filter should return when the filter is centered upon the image. The result is too few constraints relative to the degrees of freedom in the filter, leading to over-fitting of the training data. While unconstrained filters such as UMACE eliminate this over-fitting, they still share many other problems with these filters; in particular the filters do not specify the response at any other location in the training image.

2.6.3 Minimum Squared Error Synthetic Discriminant Functions (MSESDF) and Distance Classifier Correlation Filters (DCCF)

Two other approaches to the design of SDF deserve mention because like Optimized Correlation Output Filters (OCOF)s they address the entire correlation surface. MSESDF [45] allows the correlation surface to have an arbitrary response shape. This type of filter has not been tested in this work because there is no canonical implementation, and it has been shown that when the desired output shape is selected to minimize the non-centered pixels this filter is equivalent to MACE. The second approach DCCF [52] produces output shapes that optimally discriminate between classes of objects. While this is useful for discriminating between objects, it has little use when accurately locating objects. Both these methods still require the training images to be centered on the target.

2.6.4 Notable Recent Research in Correlation Filters

More recent developments in correlation filters have gone beyond using simple filters. For example, in [69] a principal components analysis is run on the phase spectrum of the training images in the Fourier domain allowing the phase information to be represented as a spanning linear subspace. The primary advantage of using a subspace to represent the target instead of a single filter is that it represents a larger set of target variations. Additionally, there is only a modest increase in the computational complexity of the correlation. The result is higher peak to side-lobe ratios than MACE.

Nonlinear filters are also the subject of recent correlation filter research. It is known that many of the real-world appearance variations in natural scenes are poorly modeled by linear approximations. To account for this the field of computer vision as a whole has been investigating ways to produce nonlinear models of those changes, and so it should not be surprising that the same is going on in correlation filters. While introducing nonlinearities to the correlation filters increases the representational of power of the filters, it also increases the computational complexity. Therefore the benefits of the nonlinearities are marginalized by a reduction in speed.

One of the earliest methods for introducing nonlinearities to correlation filters is

known as polynomial filters [51]. To compute a polynomial filter, each of the pixel values in each input image is transformed using nonlinear functions and is individually convolved with different filters that collectively produce optimal responses. For example, the input image x is transformed using a set of nonlinear functions f_i and is then correlated with a set of filters h_i:

$$g = h_1 \star f_i(x) + h_2 \star f_2(x) + \cdots + h_n \star f_n(x)$$

The filters used in this equation are collectively trained to produce an optimal response g for the set of training images. It should be noted that in addition to nonlinear functions of a single image, this technique can also be used to combine aligned images from different sources. For example, f_1, f_2, and f_3 could represent the red, green, and blue channels of a color image.

As mentioned before, the increased representational power comes at the expense of speed. Specifically, each nonlinear transformation requires an additional correlation, which increases the computational complexity by a factor of n. Also the nonlinear transformation of the pixel values can add time to the computation. This additional time may not be much of a problem because each correlation can be computed hundreds or even thousands of times per second.

A different approach that is presented in [64], which has many similarities to the polynomial filter, is called Action MACH. Here the MACH filter [53] has been extended to gesture classification, resulting in two interesting developments. The first is that short video sequences are restructured into a three dimensional array so that the filters and images are actually three-dimensional structures. While most correlation filter training techniques including OCOF can be easily extended to one-dimensional, three-dimensional, and even n-dimensional signals, this is one of the few papers that demonstrates the usefulness of a three-dimensional filter. As video analysis becomes more popular and the speed of hardware improves, the use of 3D filters will become more common.

Additionally, instead of using scalar pixel values for each pixel element, those elements actually contain three values representing a motion flow field. This is a problem because

correlation typically requires only a single value per pixel. The solution is to use Clifford algebra, Clifford correlation, and the Clifford Fourier Transform to perform computations. Like the polynomial filters, Clifford algebra is an alternative mathematical technique to combine multiple channel images to produce correlation filters. Additionally, while slower than a single linear filter, this technique is still implemented using FFTs and the convolution theorem, so the computation is relatively efficient.

Using a nonlinear kernel function, or the kernel trick to compute relationships between pairs of images, has extended many linear pattern analysis techniques to nonlinear domains [73, 15, 57, 77, 5, 75, 76], and correlation filters are no exception. Because each correlation value in the correlation output plane is essentially an inner product, this computation can be replaced using a kernel function. Unfortunately, by replacing the inner product with the nonlinear kernel, the correlation cannot be computed in the Fourier domain using the Convolution Theorem, and therefore the time needed to compute even modest size correlations becomes a problem.

There are two main lines of work doing research in this area that have both come out with similar solutions. The first is discussed in [86, 43] and extends correlation filters with an arbitrary kernel function. Due to the increase in computational complexity, this research assumes that the images are pre-aligned, and therefore only the value at the expected peak needs to be computed. Computing only one correlation value is relatively fast and is therefore suitable for real-time problems; however, in doing this, the filter looses its translation insensitive properties.

The other research in this area is called correntropy MACE filters [38, 37, 39]. In this case the kernel function is limited to a Gaussian kernel, and when combined with the fast Gaussian transform, allows for a fast approximation of the full correlation output while retaining the increased representational power. In [39] it is shown that computing a generalized correlation function called correntropy directly can take over two minutes for a modestly sized input image. Approximating this computation with the the fast Gaussian transform reduces that run-time to a few seconds. With that speed increase,

this nonlinear filter can be used in some real-time applications; however, it is still much slower than linear filters where correlations can be computed hundreds of times per second.

2.7 Other Object Detection Methods

As described earlier, correlation lends itself very nicely to object detection problems because the detection algorithm is simple and fast. It is also rarely used because most researchers only try a simple template matching approach and find that the detector cannot adequately discriminate between the target and background. In fact, there is no one object detection method that works in all circumstances, and the diversity and variety of target objects has led researchers to investigate many different object detection algorithms. Often times the object detection method is tailored specifically to the target object and the imaging conditions. For example, one of the best methods for human/pedestrian detection [25] explicitly models the appearance and motion of the arms, legs, torso, and head of a person. Such a system is very closely linked to human detection and cannot be easily generalized to other object detection problems. This section will discuss some of the strategies for object detection that are most commonly proposed in the research literature.

2.7.1 Simple Features vs. Appearance Based Methods

Many of the most successful object detection strategies are based on exploiting extremely simple but also discriminating features such as motion or color. For example, it is easy to locate a white soccer ball on a green soccer field by simply discriminating between the colors of the ball and the field. This sort of detector is often easy to implement and performs well under controlled circumstances. While a color based detector may work great on test photos containing just the soccer ball and the field, it can be easily confused by a soccer player that is wearing a white jersey. In general, this particular type of detection performs poorly in gray scale images or images where the color of the target object is unpredictable, such as the color of a car.

Another commonly used and simple detection strategy exploits object motion. In this case it is assumed that the camera is stationary and the targets are moving. This type of detector often uses an image to model the background and detects objects by finding the pixel values that have changed. This method also has many limitations. If you consider a security camera overlooking a street, this method detects anything that moves, including cars, people, animals, reflections, shadows, bushes blowing in the wind, etc, and is unable to distinguish targets of interest from any other change in the image.

These simple color and motion detectors can be very useful, but they are also very predictable and are easy to confuse. They detect objects because they have some low level similarity to the target but never consider the shape or appearance of the target. Correlation and many of the more heavily researched detection algorithms consider the object's appearance. For example, a person can find a car in an image because it has the "shape of a car" and it also has wheels, headlights, doors, windows, and license plates. Likewise, a face has eyes, a nose, a mouth, and ears. Such methods are more difficult to confuse but also come with many drawbacks such as being slower and more complex.

2.7.2 Sliding Window

One common technique used for object detection is based on a sliding window approach. Basically an image is scanned using a rectangular window that slides from top to bottom, left to right, and covers multiple scales. Each window is classified as either the target object or background. Sliding window detectors use a variety of different classifiers and features to discriminate targets from background. To illustrate, consider these three well known sliding window based pedestrian detectors:

- Viola et al. [79] used a pedestrian detector based on Haar wavelet based features and a cascade classifier.
- Dalal and Triggs [22] used Histograms of Oriented Gradients as features and classification using a Support Vector Machine (SVM).
- Falzenzwald et al. [25] modeled people using a set of parts including the head,

torso, arms, and legs and used a latent SVM for classification.

In these cases, the three sliding window classifiers use different features and also different classifiers, but they all solve the same problem.

Correlation is actually a type of sliding window detector. The correlation operation actually computes a linear classification function over a window centered on every pixel in an image, and classification is performed by thresholding that output. Because correlation is such a simple operation, detection can be performed quickly even in the spatial domain. As discussed in Section 2.3, there are also a number of good algorithms that can speed up correlation further such as separable convolution or the Correlation Theorem. However, the robustness of detectors depends on the strength of the classifier and therefore most recent research focuses on more complex classification functions. They are consequently denied the computational efficiency of correlation.

As indicated by the algorithm description, the sliding window approach needs to evaluate a classification function over multiple windows. In practice this results in thousands of function evaluations per image. There are a number of ways to improve the speed of a classifier function. The first is to reduce the total number of windows that need to be evaluated. This can be done by reducing both the number of scales scanned and by increasing the step size between windows in the x and y dimensions. The other way to improve the speed of the detector is to reduce the time taken by the classification function.

One of the highest payoffs in terms of speed is to eliminate the need to search for small targets. Small targets mean small windows and therefore require more windows to cover the image. Therefore carefully selecting the minimum target size probably can change the run times of a detector by orders of magnitude. Additional time can be recovered by reducing the number of scales and by increasing the step size between windows. These optimizations will reduce the number of windows needed to scan the image and therefore speed up detection. It will also have a detrimental effect on the accuracy of the detector because it is more likely that a target might be missed because it falls between detection

windows or scales when the step sizes are large.

A second way to improve the speed of a detector is to reduce the time needed for classification. If the classification function is slow, running detection on a single image can take seconds or even minutes to complete. For this reason, much of the research in this area has concentrated on developing fast techniques that are also good at distinguishing objects from background. In general, the window size is kept as small as possible. Most modern detectors assume a window size between 20 to 30 pixels. Reducing the total number of pixels in the window can greatly improve classification speed, but this improvement in speed comes at the expense of detection accuracy. The actual classification step is performed using one of a variety of standard classification techniques including Neural Networks [67], Bayesian [72], Decision Trees [78, 80, 47], and SVMs [15, 59].

In recent years most of the research in real time object detection has focused on the Viola and Jones Cascade Classifier [78]. The cascade classifier uses two clever algorithms that make evaluating a detection window fast. The first of these is the use of Haar based features in conjunction with an integral image or summed area table [20]. The integral image is an image where every pixel value has been replaced by the sum of all the pixel values above and to the left of the current pixel. This image transformation is simple and fast. In the transformed space, computing the sums of regions of pixels takes constant time requiring 4 memory look ups and 3 addition/subtraction operations. Because the Haar features are based on these sums, computing the features used for classification is extremely fast.

The second part of the detector uses a cascade decision tree to classify the detection windows as belonging to target or background. This decision tree is structured such that at each level it either classifies the detection window as background, in which case the window is rejected and the detector moves on to the next window, or the window is accepted in which the decision tree evaluates the next node. Typically each node in this tree is trained to reject approximately 50% of all background windows while passing almost all target windows. During operation, this allows the detector to quickly reject

most of the detection windows while focusing more time distinguishing true targets from confusing background. As a result, the cascade classifier is fast while also accurately distinguishing foreground from background.

The final important component of the cascade detector (which is also used in similar sliding window approaches) is the use of Adaboost [26] during training. Because a sliding window object detector evaluates thousands of detection windows for each image it processes, the classifier needs a very low false accept rate. To achieve this the classifier must train on a large number of background windows. For training, the cascade classifier trains the first level in the cascade using randomly selected background windows. For the following levels the cascade detector is run on a set of images that do not contain target objects and false detections are used to train the next layer. This allows training to focus on mistakes, which keeps the training set small, but it also focuses training on harder background windows at each level of the cascade.

Training on hard negative examples is important for producing a good detector and the correlation filters presented in this dissertation take a different approach training on hard examples. Instead of using a technique like boosting to focus on difficult false detection mistakes, the filters simultaneously train to correctly detect the few instances of targets in an image while rejecting every possible background instance [9]. This is achieved by completely specifying the desired output of the filter for every location in the training image. Because the relationship between the training image and the output images can be simplified using the Convolution Theorem, this makes training fast. For example, in [9] training took 13 seconds while training the cascade classifier on the same data took over six hours.

2.7.3 Keypoints and Keypoint Descriptors

An alternative to the sliding window object detection approach is to use keypoints and keypoint descriptors to detect objects. Most of these detectors are based on the Scale-Invariant Feature Transform (SIFT) algorithm proposed in [49] or the faster and more robust Speeded Up Robust Features (SURF) algorithm from [6]. These algorithms detect

objects in a bottom up approach by first detecting parts of an object and then aggregating those parts into a complete object. For example, to detect a car these detectors would first find simple keypoints on the car; it would then classify descriptors of those key points as tires, doors, headlights, etc; and finally determine if a car was present based on the locations and classifications of those descriptors. While this detection method has little to do with correlation based detection, it is important to consider because it is a promising alternative to sliding window techniques.

These detectors start by first detecting keypoints. While there are a number of strategies for detecting keypoints [58], the most popular seem to be based on approximately circular image features, such as Difference of Gaussians (DOG) or Laplacian of Gaussian (LOG) filters, because they specify both the location and scale of the feature. Once a keypoint is detected, a descriptor is computed and that descriptor can be matched to a database of descriptors for many different objects. Once the key points have been labeled, the boundary of the object is determined. SIFT, for example, computes an affine transform, which maps the geometry of the training data to the geometry of the detected objects to determine the objects position. An alternative is to treat an object as an unstructured bag-of-features such as that used in [46], which efficiently converges to a subwindow that is likely to contain the target object.

In recent years SIFT has also been adapted to a number of related application areas. In image retrieval, SIFT keypoint descriptors are used like a set of "words" to describe an image [74, 36]. Using these "visual words" it is then possible to index and retrieve images using techniques similar to text retrieval [68]. Because this technique locates multiple points, it is also possible to locate the camera's position with respect to the object using triangulation. This leads to applications like the ability to determine a cellphone's location from an image [34] or markerless augmented reality [81, 2].

SIFT-like algorithms are often applied to different types of object detection problems than correlation filters and other sliding window approaches. SIFT assumes high resolution views of the objects are available so that many keypoints can be used to detect the

target. Sliding window represent targets using low resolution data. Because of this and other differences, it is difficult to directly compare SIFT-like approaches to correlation filters.

2.8 Chapter Summary

This chapter has covered a variety of topics in correlation filters and object detection. The chapter began with a simple overview of correlation and techniques for simplifying the correlation operation using the Convolution Theorem. The chapter also covered much of the relevant and more recent work in creating special filters for computer vision tasks that included SDF, MACE, OTF, and UMACE, which will be discussed further in the remaining chapters. Finally, the last sections discussed recent trends in object detection research that go beyond filter-based detection.

The techniques discussed in this chapter will be compared to the OCOFs introduced by this research. In Chapter 3 it is shown that UMACE can be implemented as a specialization of Minimum Output Sum of Squared Error (MOSSE). Results presented in the later chapters will also confirm that UMACE is probably the closest relative to OCOFs. However, Chapter 5 will show that the techniques introduced in this research are indeed producing more powerful detectors than can be achieved by UMACE.

In the rest of this dissertation, Chapter 4 will compare OCOFs to other correlation filters and object detection techniques for the problem of locating eyes in face images. Chapter 5 compares OCOFs to other filter and state-of-the-art object detectors on the problem of person detection in video. Finally, Chapter 6 compares an OCOF-based tracker to simple template correlation filters and also to two well-known visual tracking algorithms.

Chapter 3

Optimized Correlation Output Filters

For this research I have developed a new technique for training correlation filters called Optimized Correlation Output Filters (OCOF). Unlike prior training methods that recombine templates, OCOFs consider the image to image mapping that is performed during correlation and inverts this mapping to produce ideal filters. This chapter will give an overview of these training methods including exact filters, Average of Synthetic Exact Filters (ASEF), Minimum Output Sum of Squared Error (MOSSE), and generalized cost function optimization.

3.1 Exact Filters

Consider the problem of constructing a filter to detect a particular type of object. A filter is desired that produces a strong peak where the targets are present and zero values elsewhere in the correlation output. It can be constructed from a set of N training images $\{f_1, f_2, \cdots, f_N\}$ that include examples of both targets of interest and also background and other distractors. Techniques such as ASEF and MOSSE learn a filter h such that when correlated with those training images produces a peak near the center of the targets but also produces values close to zero elsewhere. Prior filter training techniques would solve this problem by cropping examples of the targets out of the training images. OCOFs take a different approach; they synthetically generate the desired output and then learn

a filter that maps the training images to those synthetic outputs.

To be specific, the first step is to create a set of outputs $\{g_1, g_2, \cdots, g_N\}$ that have peaks where the targets are located in the corresponding training images. Here it is assumed that g_i is the sum of two dimensional Gaussians centered at the targets locations (x_j, y_j) where j indexes targets in the ith training image and σ specifies the radius of the peak:

$$g_i = \sum_j e^{-\frac{(x-x_j)^2+(y-y_j)^2}{\sigma^2}} \tag{3.1.1}$$

The filter learning task then reduces to the problem of solving for a filter h that satisfies the following relation:

$$g_i = h \star f_i$$

In the spatial domain finding a good filter h is a difficult task. In the Fourier domain, the solution becomes simple. Because correlation in the Fourier domain is an element-wise multiplication, solving this problems leads to element-wise divisions. Keeping with our notation where capital letters indicate images in the Fourier domain, $\odot$ indicates an element-wise multiplication, and $*$ indicates the complex conjugate, the previous equation becomes :

$$G_i = H^* \odot F_i$$

When there is only one training image, solving this equation for the filter is trivial and results in an element-wise division:

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

We call H_i^* an exact filter because it exactly transforms f_i to g_i. We also index this filter by i, which indicates this particular exact filter is associated with the ith training image. This is inspired by deconvolution and inverse filters [62], which use element-wise divisions in the Fourier domain to accomplish tasks like debluring images. Using simple manipulations we can show that H_i^* can be computed as:

$$H_i^* = \frac{G \odot F_i^*}{F_i \odot F_i^*} \tag{3.1.2}$$

This equation has some nice properties. First, this is the standard way to implement the complex division. The denominator $F_i \odot F_i^*$ will only result in real positive values. The complex division is therefore simple because inverting the real valued denominator is a well defined operation. Second, the numerator and denominators have nice interpretations: the numerator is the correlation of g_i with f_i and the denominator is the energy spectrum of f_i. Finally, in Section 3.4, this formulation will be used to add a simple regularization term, which improves the stability of the training process.

3.2 Average of Synthetic Exact Filters (ASEF)

One problem with exact filters is that they perform poorly when tested on images that are different than the training images. This is because they leverage idiosyncrasies in the their associated training images and therefore do not generalize well to larger problems. To produce more general filters, the average of multiple exact filters is computed. When averaged, features of the exact filters that are consistent across the training images are emphasized while features of the exact filters that are inconsistent cancel out.

Figure 3.2.1 shows an overview of the ASEF training process for an eye detection filter. As can be seen, the exact filters for each training image look like static and will probably not generalize to other images. Averaging thousands of exact filters results in an ASEF Filter, which has the appearance of an eye and generalizes well.

ASEF averages a set of weak filters and therefore can be thought of in terms of aggregation theory [13] or bagging. Exact filters are basically weak classifiers that perform perfectly on their associated training images. Aggregation theory suggests that accuracy can be greatly improved by simply averaging the output of a large number of weak classifiers. Because filtering is a linear operation, instead of averaging the correlation outputs, the filters themselves can be averaged to produce a single correlation filter, which saves an enormous amount of computation when applying the filter to new imagery. This

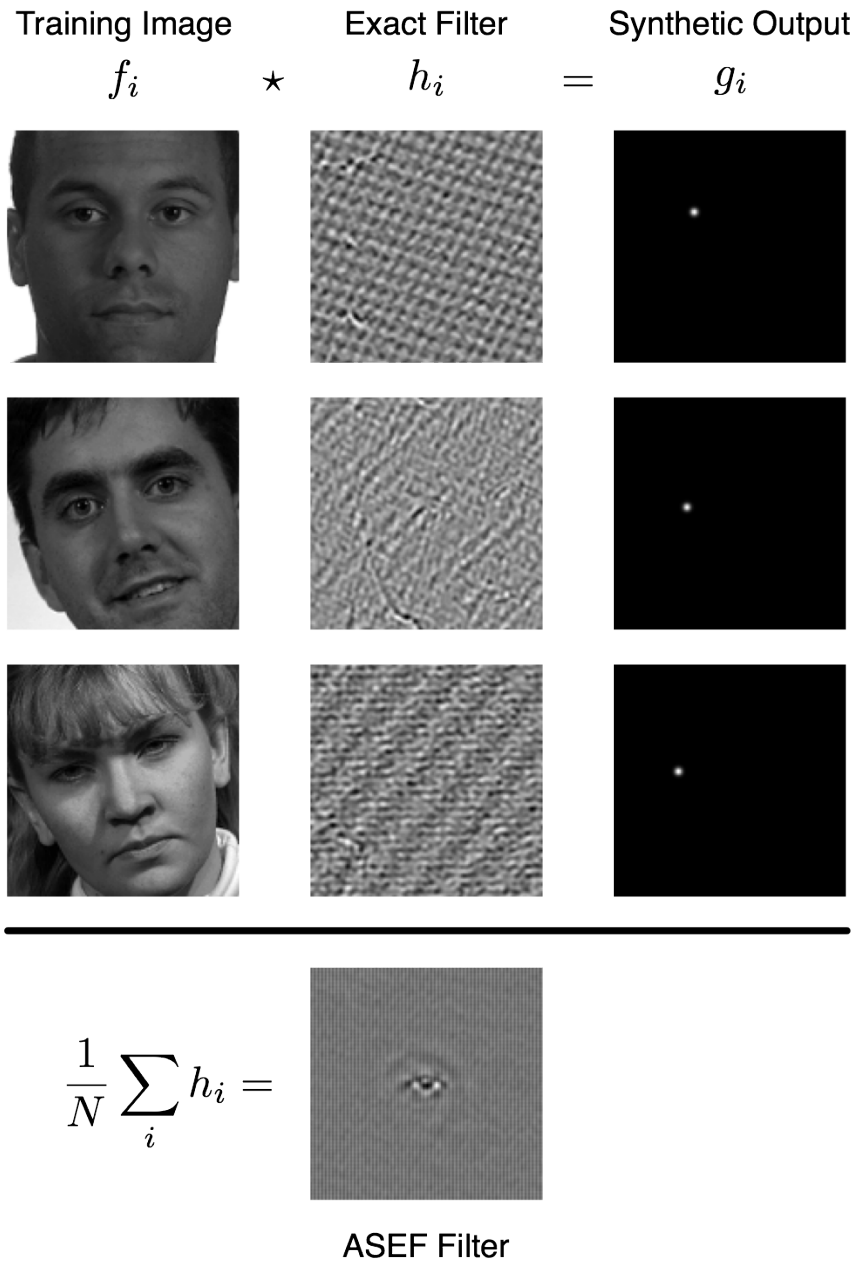

Figure 3.2.1: This figure illustrates the ASEF training process. Here we are training the filter to detect the left eye in the image. It is clearly shown that the exact filters are specific to each training image and do not have the appearance of an eye. The bottom row shows the ASEF filter, which is computed by 7500 exact filters. MOSSE training is similar but does not compute intermediate exact filters. *From [8].*

average can be computed in either the Fourier or the spatial domain.

$$H_\mu^* = \frac{1}{N}\sum_{i=1}^{N} H_i^*$$
$$h_\mu = \frac{1}{N}\sum_{i=1}^{N} h_i$$

where H_μ or h_μ are ASEF filters. When averaged in the spatial domain, the exact filters can be cropped, which allows ASEF filters to be trained on images of different sizes.

ASEF requires thousands of training images to converge to a good filter, but often that many images are not available. To increase the size of the training set, the images can be randomly perturbed by scaling, rotating, and translating by small amounts. This also exposes the filters to variations that are encountered in real world conditions and therefore results in better performance.

3.3 Minimizing the Output Sum of Squared Error (MOSSE)

MOSSE is an algorithm for producing ASEF-like filters but only requires a small number of training images. Because of this, training is fast and can be done online and in real-time (see Chapter 6 for details). The input training data for MOSSE is identical to ASEF. It requires a set of training images f_i and training outputs g_i. MOSSE then finds a filter H that minimizes the sum of squared error between the *actual* output of the correlation and the *desired* output of the correlation. This minimization problem takes the form:

$$H^* = \min_{H^*} \sum_i |F_i \odot H^* - G_i|^2 \tag{3.3.1}$$

The idea of minimizing Sum of Squared Error (SSE) over the output is not new. In fact, the optimization problem in Equation 3.3.1 is almost identical to optimization problems presented in [45] and [53]. The difference is that in those works it was assumed that the target is always carefully centered in f_i and that the output (g_i) contained one centered peak and was identical for the entire training set. In contrast, customizing every g_i is a fundamental idea behind ASEF and MOSSE. As a result, the target is not always

centered, and the peak in g_i moves to follow the targets in f_i. It is also more general in the sense that g_i can have any shape. For example, in Chapter 5 and [9] f_i contains multiple targets and g_i has multiple corresponding peaks.

Because correlation in the Fourier domain is an element-wise multiplication, each element of the filter H can be optimized independently. The optimization problem can therefore be transformed from a multivariate optimization problem to a problem that optimizes each element of H independently.

$$H_{\omega\nu} = \min_{H_{\omega\nu}} \sum_i |F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu}|^2$$

where ω and ν index frequencies.

This function is real valued, positive, and convex so it will have only a single optima. Normally to find the optima of a function, the stable points are found by setting the derivative equal to zero and then solving for the variable of interest. Finding the stable point for this function is different because it is a real valued function of a complex variable and care needs to be taken to solve this problem correctly. To summarize, it involves rewriting the function in terms of both $H_{\omega\nu}$ and $H^*_{\omega\nu}$. Then, the partial W.R.T. $H^*_{\omega\nu}$ is set equal to zero, while treating H as an independent variable. [56]

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i |F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu}|^2$$

It can be shown that any $H_{\omega\nu}$ that satisfies this equation is a stable point. A short tutorial on this technique can be found in [56]. Transforming this equation leads to:

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i (F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu})(F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu})^*$$

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i [(F_{i\omega\nu} H^*_{\omega\nu})(F_{i\omega\nu} H^*_{\omega\nu})^* - (F_{i\omega\nu} H^*_{\omega\nu}) G^*_{i\omega\nu} - G_{i\omega\nu}(F_{i\omega\nu} H^*_{\omega\nu})^* + G_{i\omega\nu} G^*_{i\omega\nu}]$$

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i F_{i\omega\nu} F^*_{i\omega\nu} H_{\omega\nu} H^*_{\omega\nu} - F_{i\omega\nu} G^*_{i\omega\nu} H^*_{\omega\nu} - F^*_{i\omega\nu} G_{i\omega\nu} H_{\omega\nu} + G_{i\omega\nu} G^*_{i\omega\nu}$$

The next step is to compute the partial derivative. The last two terms in the previous equation drop out because we are treating $H_{\omega\nu}$ as an independent variable:

$$0 = \sum_i [F_{i\omega\nu} F^*_{i\omega\nu} H_{\omega\nu} - F_{i\omega\nu} G^*_{i\omega\nu}]$$

We can then distribute the summation and solve for $H_{\omega\nu}$.

$$H_{\omega\nu} = \frac{\sum_i F_{i\omega\nu} G^*_{i\omega\nu}}{\sum_i F_{i\omega\nu} F^*_{i\omega\nu}}$$

Finally, by performing some simple manipulations and solving for the filter H^*, the MOSSE filter can be written in the original array notation as:

$$H^* = \frac{\sum_i G_i \odot F^*_i}{\sum_i F_i \odot F^*_i} \tag{3.3.2}$$

From Equation 3.3.2, we can easily show that Unconstrained Minimum Average Correlation Energy (UMACE) is a specialization of MOSSE. UMACE is defined as $H^* = D^{-1} m^*$ where m is a vector containing the Fast Fourier Transform (FFT) of the average centered cropped training images, and D is a diagonal matrix containing the average energy spectrum of the training images[70]. Because D^{-1} is a diagonal matrix, it is essentially an element-wise division. When rewritten in the current notation, UMACE takes the form:

$$H^* = \frac{\sum_i F^*_i}{\sum_i F_i \odot F^*_i}$$

However, UMACE requires that the target is centered in F_i. Recentering can be performed using correlation. If we define g_i to be a Kronecker delta (with a peak of one at the target center and zero elsewhere) this will essentially recenter the target and compute an UMACE filter. The difference between this implementation of UMACE and the traditional implementation is that here the training image is cropped and then translated while the traditional implementation translates and then crops. While this demonstrates a mathematical connection between UMACE and MOSSE, it will be shown in Section 5.3 that much of the power of MOSSE comes from its ability to train on images with multiple uncentered targets.

3.4 Stability and Regularization

Regularization is a simple way to improve the stability of the filter training process. The process used to compute exact filters, e.g. ASEF and MOSSE, is an instance of what is more generally referred to as an inverse problem. Numerically, the number of constraints on an exact filter is equal to the number of unknowns. As a result in most cases there is an exact filter that maps an input image exactly to its desired output. One interesting case is when a frequency in the Fourier Transform of the training image contains zero energy. Because the closed form solution to computing the exact filter requires an element-wise division by the energy spectrum (see Equation 3.1.2) this operation is a divide by zero and has no solution. In other words, division by the training image is undefined for that frequency.

With natural images this is usually not a problem. The Fourier Transform of these images rarely contains frequencies that are zero. There are a few cases that may come up in practice. For example, when images are completely dark or over saturated. This can occur from illumination conditions or as a byproduct of image preprocessing steps. A more common problem is when frequencies are very small due to some sort of preprocessing or filtering. In exact filters, frequencies are weighted inversely proportional to their energy. This results in some frequencies being overly represented in the resulting exact filters. These cases indicate that the inverse problem is ill-conditioned.

Regularization is often used to improve stability for inverse problems that are ill-conditioned. One interpretation is that it encourages solutions that are consistent with Occam's razor: simpler is better. In machine learning, regularization has many benefits including reduced over-fitting and noise tolerance and is therefore found in many successful machine learning algorithms such as: Support Vector Machine (SVM)s [15], Generalized Discriminant Analysis (GDA) [5], and Kernel Ridge Regression (KRR) [33].

Adding regularization to Equation 3.1.2 is easy. The denominator in Equation 3.1.2 is actually the energy spectrum of the training image: $F \odot F^*$. A nice property of this term (each element is multiplied by its complex conjugate) is that each element in the

denominator is both real and positive. Regularizing is performed by simply adding a small value to every element: $F \odot F^* + \epsilon$ where ϵ is the regularization parameter. With regularization, the equation for an exact filter becomes:

$$H_i^* = \frac{G_i \odot F_i^*}{F_i \odot F_i^* + \epsilon}$$

Exact filters computed using this method are much more stable. The impact on ASEF is that the filters can be trained on fewer images with little risk of any one exact filter dominating the averaging process. The resulting ASEF filter is computed as:

$$H_i^* = \frac{1}{N} \sum_i \frac{G_i \odot F_i^*}{F_i \odot F_i^* + \epsilon}$$

The denominator for MOSSE filters can also be regularized in an identical way. MOSSE is less likely to have any one frequency be particularly small because the denominator is summed over the set of all training images and because all elements in that sum are real positive numbers. This makes MOSSE naturally more stable. Even so, regularization is still useful and helps to further stabilize the MOSSE training process. With regularization, the MOSSE filter becomes:

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^* + \epsilon}$$

This modification is related to Minimum Variance Synthetic Discriminant Functions (MVSDF), Minimum Average Correlation Energy (MACE), and Optimal Tradeoff Filters (OTF) as discussed in Section 2.6.1. Regularization is similar to a process used to create OTFs [63] that are similar but more stable than MACE [54] or UMACE filters. In MACE peak sharpness is achieved by multiplying with the inverse diagonal matrix $\mathbf{D}$ where the diagonal elements are the average energy spectrum of the training images.

OTFs find an optimal balance between peak sharpness from MACE and robustness to noise from MVSDF. The matrix $\mathbf{D}$ is replaced with:

$$\alpha\mathbf{D} + \beta\mathbf{C}$$

where $\mathbf{C}$ contains the energy spectrum of the noise and α and β control the optimal weighting (see Equation 2.6.3). In practice the energy spectrum of the noise is often difficult to estimate and so white noise is used. The results are therefore very similar to the regularization presented in this chapter. Specifically regularization for MACE and UMACE can be achieved by replacing D with:

$$\mathbf{D} + \epsilon\mathbf{I}$$

where ϵ is equivalent to the regularization parameter presented earlier and $\mathbf{I}$ is the identity matrix, which represents white noise.

3.5 Cost Function Minimizing Filters and Gradient Descent

One criticism of ASEF filters is that they do not optimize a well defined figure of merit in a manner similar to other filter training techniques. For example, a MACE filter imposes a hard constraint on the filter for each training tile and then minimizes the average correlation energy for the rest of the output. This section will show how to produce a filter that optimizes any differentiable cost function computed over the output. This new technique allows additional flexibility in how filters are trained but also introduces additional complexity. Many of the benefits of ASEF are retained, such as the ability to control the entire convolution output and to train on multiple targets per training image.

In this more general approach, training starts by defining a cost function C that is a function of the filter output and ground truth for the training image. It then uses the derivative of the cost function with respect to the filter to quickly converge to a local optimum using standard nonlinear gradient descent optimization algorithms.

Here the correlation output for the current training image is referred to as g':

$$g'_i = \mathcal{F}^{-1}(F_i \odot H')$$

This example will minimize the same sum of squared error function optimized by MOSSE

between the filter output and the desired output:

$$C(g_i', g_i) = \sum_i |g_i - g_i'|^2$$

The next step is to compute the gradient of the cost function with respect to the filter output g'. In this case the gradient is simply:

$$\nabla_{g_i'} C(g_i', g_i) = 2(g_i - g_i')$$

By applying the chain rule and the convolution theorem, the gradient of the cost function with respect to the filter h' can be quickly computed.

$$\nabla_{h'} C(g', g_i) = \mathcal{F}(F_i \odot \mathcal{F}^{-1}(\nabla_{g'} C(g', g_i)))$$

Finally, a nonlinear gradient descent algorithm is used to solve for the true optimum. Often these optimization algorithms require the cost and gradient for the entire training set so the average cost can be computed:

$$\begin{aligned} C(h') &= \frac{1}{N} \sum_{i=1}^{N} C(g', g_i) \\ \nabla_{h'} C(h') &= \frac{1}{N} \sum_{i=1}^{N} \nabla_{h'} C(g', g_i) \end{aligned}$$

This new technique is certainly more complex than ASEF or MOSSE and therefore requires more care during training. For example, the approach has many of the same problems as any other gradient descent algorithm. First, if the cost function is not convex then the process used to search for the optimal filter can get stuck in local optima. It is therefore important to carefully choose good cost functions and carefully select the starting point for the search. Second, optimization often encounters regions that are ill conditioned and may require optimization techniques such as nonlinear conjugate gradient descent. Finally, some cost functions have a tendency to over-fit the training set, and therefore, use of a validation set to detect over-fitting is recommended.

Chapter 4

Correlation Filters for Facial Feature Localization

This chapter looks at the performance of Optimized Correlation Output Filters (OCOF) on the problem of eye localization [8]. The results include a detailed comparison among many different kinds of filters, including Average of Synthetic Exact Filters (ASEF), Synthetic Discriminant Functions (SDF), Minimum Average Correlation Energy (MACE), Optimal Tradeoff Filters (OTF), and Unconstrained Minimum Average Correlation Energy (UMACE) filters. In addition, these methods are compared to some popular techniques for detecting eyes such as Gabor Jets[85] and a Haar based Cascade Classifier [80]. This chapter also looks at Minimum Output Sum of Squared Error (MOSSE) filters and will conduct a careful comparison between MOSSE and ASEF. It also discusses training correlation filters and will compare ASEF and MOSSE to filters like MACE.

First, Section 4.1 discusses the data sets and measures of performance used in this chapter. Section 4.2 shows that ASEF and MOSSE produce filters that are very similar in both appearance and performance when trained on many images; however, MOSSE converges to a good filter much faster and therefore should perform better when trained on fewer images. Like [8] Section 4.3 shows that ASEF and MOSSE perform well as eye detectors and perform better than other commonly used eye detection techniques when the search for the eyes is restricted to just the eye regions. Finally Section 4.4 shows that ASEF and MOSSE perform much better than other filters when tested on the more

difficult problem where the entire face is searched for the eyes.

4.1 FERET Data Set and Measuring Performance

This chapter follows the experimental protocols described in [8]. To test the abilities of the filters to locate eyes, experiments were run on the Face Recognition Technology (FERET) face database [60]. This database contains 3,368 images of 1204 people with manually annotated eye coordinates. For these experiments this data set was split into training, validation, and testing partitions. First, the data set was split into two partitions of 602 people and 1699 images each. The first partition became the testing set. The second partition was then split randomly into a partition of 1024 images that are used for training and a partition of 675 images used for validation. For each experiment, the training set is used to train the eye detection algorithms and the validation set is used to determine the best tuning of that algorithm. Once a particular training and tuning are selected, those algorithms are evaluated using the testing set.

When initially testing eye detection, the OpenCV face detector was used to produce a bounding box describing the face location. Because most faces in the data set are frontal and upright, the face detector would often place the eye coordinates in approximately the same locations relative to the face detection rectangle. In real life this is not always the case and the face may be rotated, translated, or scaled within the detection rectangle making the eye coordinates unpredictable. To simulate these more difficult eye detection problems, affine transformations were used to artificially perturb the image and therefore make the eye coordinates unpredictable. The face images were first transformed such that the eyes were located at pixels $(32.0, 40.0)$ and $(96.0, 40.0)$ in a 128×128 image. That image was then perturbed by rotating by up to $\pm\pi/16$, scaling by up to 1.0 ± 0.1, and translates by up to ± 4.0 pixels in each direction. To produce a larger training set, each of the $1,024$ training images were randomly perturbed 8 times to produce a set of $8,192$ images available for training.

For the correlation based approaches, each image was normalized before training or

testing. First, the log is taken of each pixel value, which has the effect of normalizing some of the effects of lighting: bright areas are reduced, and dark areas from shadows are enhanced. Next the pixel values were normalized such that they have a mean value of zero and a standard deviation of one, which reduces the effect of an overall change in illumination intensity and insures that all images have the same dynamic range.

Correlating images in the Fourier domain computes what is called a circular correlation where the images are mapped to the topology of a Taurus. In other words, the top row of the image is adjacent to the bottom row of the image, and the left column is adjacent to the right column. This can produce an artificial edge at the boundary of the image that can cause problems during correlation. To reduce this effect, a cosine window is applied to the image that smoothly reduces the values near the edges of the window to zero. Because the image was already normalized to have a mean of zero, this effectively clamps the edges to the mean pixel value of the image.

To determine the eye coordinates using the correlation based methods, each image is correlated with a filter. That filter should produce a maximum response at the location of the eye. The correlation output can be scanned to find the maximum output and that location is used as the eye coordinate. Two types of experiments were conducted. The easier task that is presented in Section 4.3 only searches for a maximum within a circle of radius 20 pixels around the expected location of the eye. This region contains the eye in all training, validation, and testing images and therefore eliminates possible false matches elsewhere in the image.

For the harder task discussed in Section 4.4, the filters search the entire face image. In that task the filters need to discriminate between the target eye and other features on the face such as the other eye, eye brows, nose, mouth, etc.

To evaluate the accuracy of the detection, the experimental protocol described in [82] was adopted. The accuracy of the location is measured using the distance of the detection from the manually located eye normalized by the inner-ocular distance:

$$D = \frac{||P_l - M_l||}{||M_l - M_r||} \tag{4.1.1}$$

where D is the normalized distance, P_l is the predicted eye location from the algorithm and M_l and M_r are the manually selected left and right eye coordinates. A localization is considered to be correct when $D < 0.10$, which corresponds to a target roughly the size of the iris. This criteria is used to select the best tuning of each algorithm and also during the final evaluation.

4.2 Understanding Training Set Size and Regularization

One challenge with SDFs such as MACE is that they tend to over-fit the training data. In other words as the filters are trained on more images, they perform worse on the validation set. This is because the filters are required to produce a peak height of exactly one for every image in the training set. In order to satisfy those constraints, the filters trained on many images do not generalize well to a larger data set. This can be seen in Figure 4.2.1 where SDF, MACE, and OTF filters become unrecognizable as more training images are added because the filters need to learn idiosyncrasies in the training data in order to produce the desired output value for every training example.

UMACE, ASEF, and MOSSE filters do not exhibit this behavior. Because these filters do not have hard constraints to satisfy, they can converge to more general filters as more data is added. UMACE is based on averaging the training templates and therefore learns features that appear in most training examples and also tends to repress idiosyncrasies.

ASEF and MOSSE are completely constrained in the case where the training set has only one image. This is the special case of an exact filter that learns the idiosyncrasies of that training image in order to exactly produce the desired output. An ASEF filter trained on a small number of images is similar to an over-fit SDF filter that performs well for the training set but poorly on images in the validation set. However, as the number of training images increases, the ASEF filter becomes much more stable and the filter generalizes well to a larger data set.

One issue with ASEF is that the filters need a large number of training images to achieve their best performance. This is illustrated in Figures 4.2.2 and 4.2.3 where the

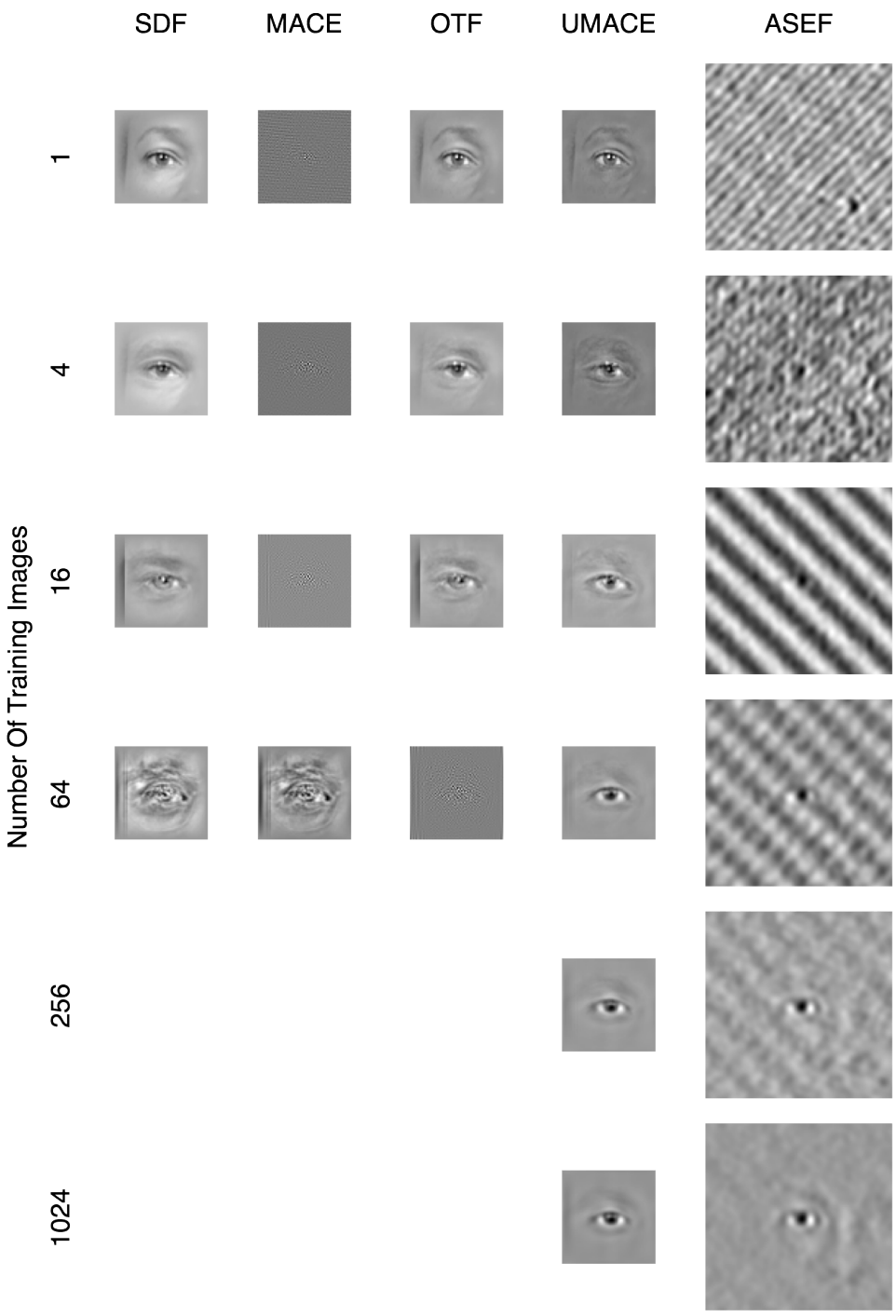

Figure 4.2.1: The images above illustrate how filters with hard constraints can over-fit the data while filters based on averaging such as UMACE and ASEF do not.

accuracy of ASEF does not level off until it has been trained on at least 256 images. As discussed in Section 3.4, this is because frequencies with small energies are weighted heavily in the final filter, and therefore, the exact filters often look more like static than the object that they are trying to detect. The result is that thousands of training images may be required to produce a good ASEF filter. There are two solutions to this problem. The first is to perform regularization on the exact filters, which encourages simpler filters and generalizes better. The second solution is to use a MOSSE filter that has already been shown to generalize better when trained on a small number of images [7].

To better understand this issue, a small set of experiments was conducted that compared ASEF and MOSSE. Filters for eye detection were trained for both methods using different numbers of training images and different values for the regularization parameter. The correct localization rate on the validation set is then reported for each configuration of the algorithm. Figure 4.2.2 summarizes these results. As discussed before, ASEF without regularization takes a large number of training images to converge to a good result. For MOSSE, or when regularization is used with ASEF, the filters converge to good solutions with fewer training images.

Figure 4.2.3 examines the effect of changing the value of the regularization parameter. It should be noted that just a small amount of regularization can have a big effect on the performance of these filters when trained on a small number of images. If the regularization parameter is too large, it can also have detrimental effects on performance. The figure shows that when the filters are trained on a large number of images, increasing the regularization parameter by too much can cause the performance to drop. These results are consistent with those reported in [7].

Finally, Figure 4.2.4 looks at the actual filters produced by each method, with and without regularization ($\epsilon = 1.0$). It is shown that the MOSSE, MOSSE with regularization, or ASEF with regularization converge to good filters much faster than the original ASEF.

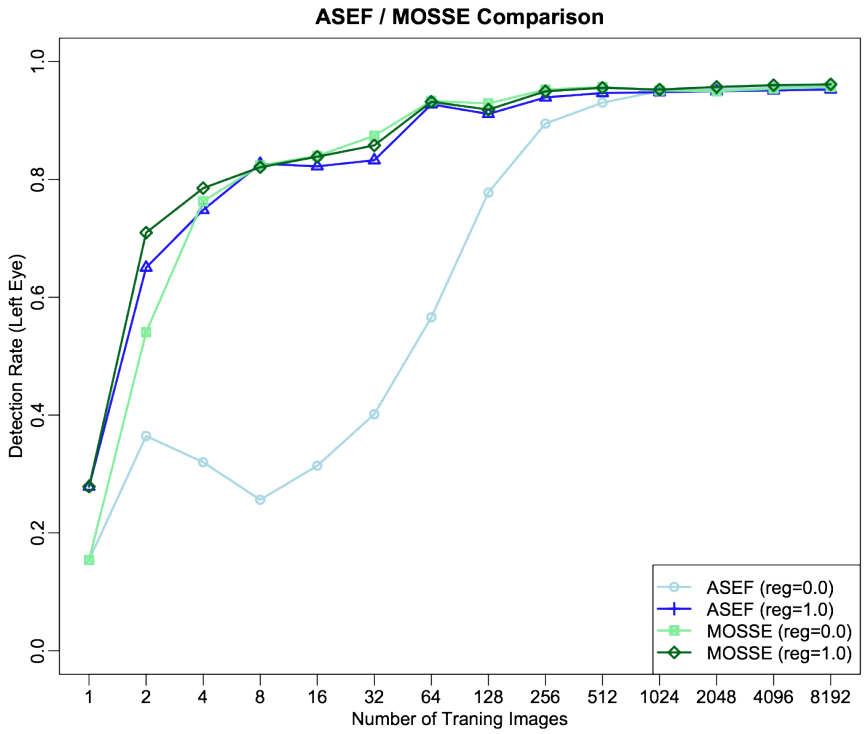

Figure 4.2.2: This figure shows that when trained on lots of images ASEF and MOSSE with and without regularization perform well. When trained on a few images, regularization matters for ASEF but does not matter for MOSSE.

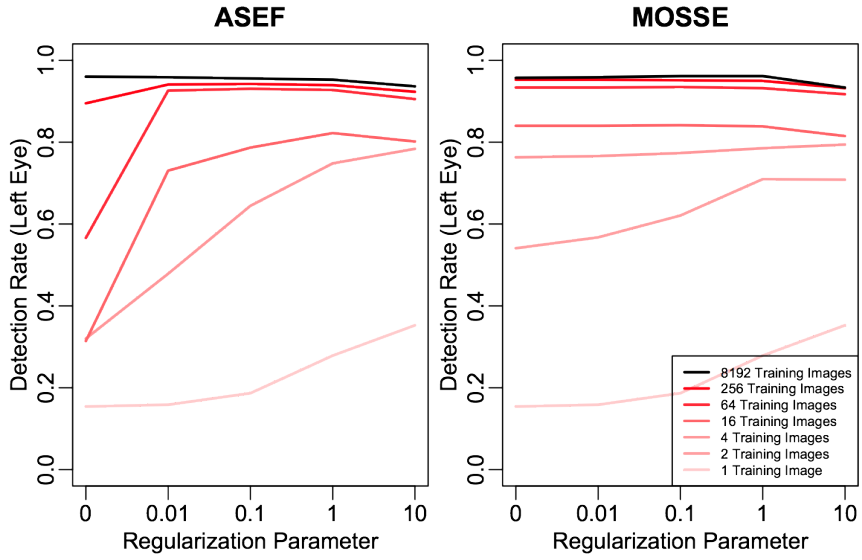

Figure 4.2.3: This figure compares the effect of regularization on the performance of ASEF and MOSSE filters. MOSSE is shown to be more stable when trained on fewer images.

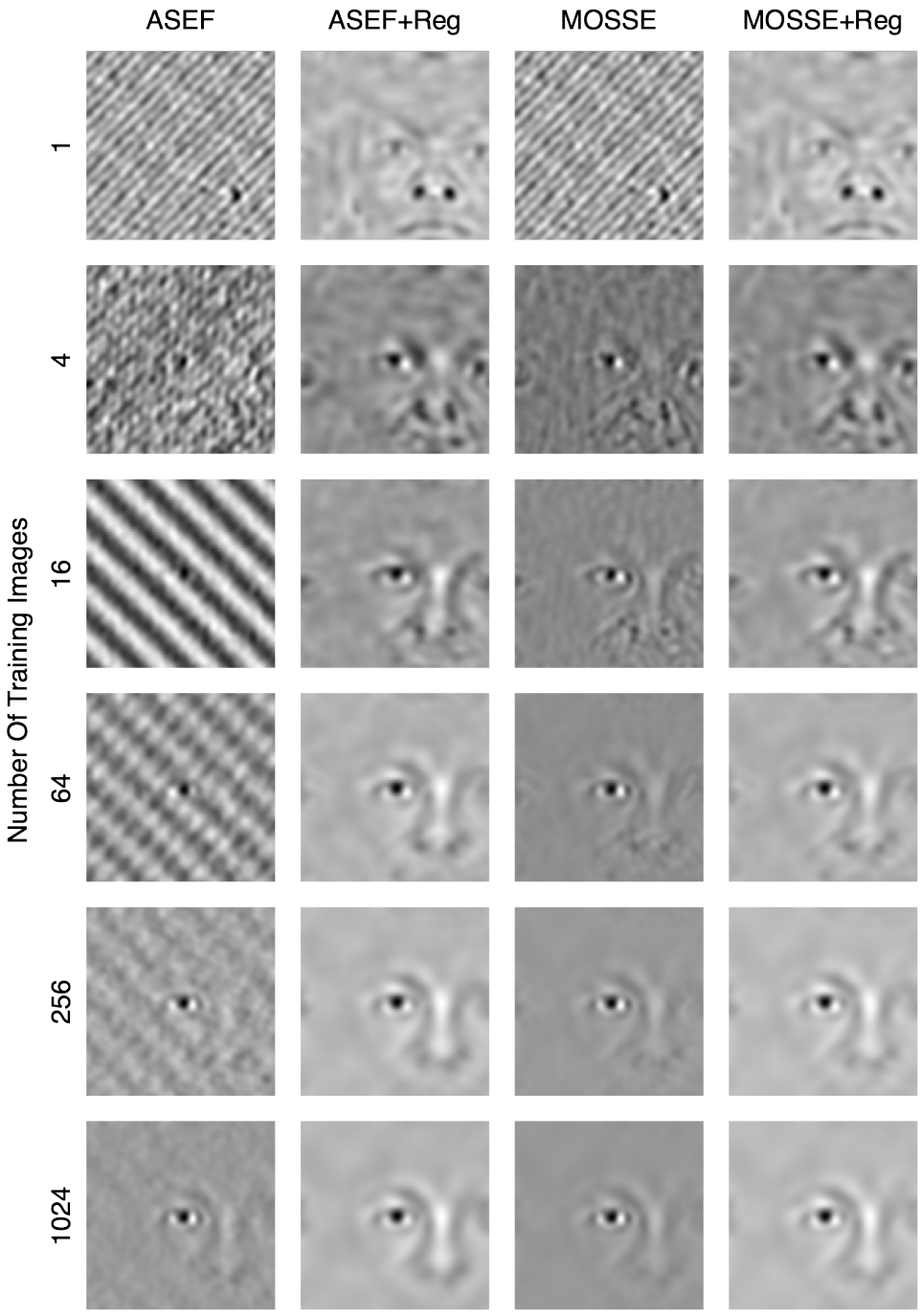

Figure 4.2.4: This is a visual depiction of the filters that compares the effects of regularization with the number of training images.

4.3 Eye Localization Restricted to Eye Regions

In this section we compare the eye localization ability of the ASEF and MOSSE filters, other correlation filter techniques, and commonly used eye localization techniques. In these experiments, the algorithms only search a small region of radius 20 pixels around the expected location of the eye for the maximum correlation response. Because the correct eye is always near the expected location, this can be used to reduce the number of false matches and makes the eye localization problem much easier. For each of these two regions the location of the maximum value is returned, therefore producing one location for each eye.

Here the filters are also compared to two traditional eye localization techniques. The first uses the Cascade detector trained in [16] to detect eyes. In initial tests it was found that this detector produced on average 4.5 eye detections for each eye. To eliminate these extra detections, only the detection closest to the expected location was used. If there were no detections within 20 pixels of the expected location, then the expected location was returned as the eye coordinate. This guarantees that the Cascade detector produces one and only one location estimate for each eye coordinate.

The other traditional method is based on the Gabor jet method described in [85]. Gabor jets are basically a local frequency space representation of an image. To produce a jet the image is convolved with a set of 80 Gabor filters. A jet is a feature vector containing all 80 correlation values associated with a particular location in the face image. By analyzing the phase information in the jet, it is possible to estimate location error of that jet with respect to a training jet. That location error can then be used to quickly converge to the location where the new jet optimally matches the training jet. In this work 256 training jets were used for comparison. By comparing to the training jet with the most similarity, the algorithm in theory can accurately locate facial features with a variety of expression and lighting conditions. When determining the location of the eye, the Gabor jet method performs a gradient based search that seeks out a local maximum in jet similarity. The search is started at the expected location of the eye and therefore

also only searches a small region of the face near the expected eye location.

To determine the best configuration for the filters, an experiment was performed. The SDF, MACE, and OTF filters were trained using different numbers of training images in the range of 1 to 24 and values of α ranging from 0.0 to 1.0 at a spacing of 0.1. (These parameters are described in Section 2.6.1). The results of this experiment are shown in Figure 4.3.1 where the x-axis shows the number of filters included in the training set, and the y-axis shows the number of eyes correctly located in the validation set. It was found that the OTF with the best configuration used an $\alpha = 0.4$ and two training images. The OTF also performed considerably better than SDF ($\alpha = 0.0$) and MACE ($\alpha = 1.0$). A similar experiment was performed for UMACE filters; however, UMACE improves as more images are added to the training set, thus 8192 were used for the final configuration. For UMACE it was found that $\alpha = 0.9$ produced the best results.

ASEF and MOSSE also tend to improve as more training data is added, so 8192 images were used in the final configurations of those algorithms. In addition the parameter σ was tested at values of 1.0, 2.0, 3.0, 4.0, 5.0, and 6.0, and $\sigma = 4.0$ was found to produce good results and is used for comparison for both this section and the next.

Figure 4.3.2 compares the performances of both filter and non filter techniques on the testing set. The x-axis is the threshold used to determine if an eye was correctly detected where $D = 0.05$ is a target approximately the size of the pupil and $D = 0.10$ is a target about the size of the iris. The y-axis shows the correct localization rate at that threshold. The unconstrained filters based on UMACE and the over constrained filters based on ASEF and MOSSE all performed well at the eye localization task. They clearly have an advantage over SDF, MACE, and OTF, which exhibit lower performance. Because UMACE, ASEF, and MOSSE can train on all 8192 training images, more information about what is important about the appearance of the eyes becomes part of the filter.

Training SDF, MACE, and OTF filters is much more difficult because those filters have a tendency to over-fit the training data. In this case we were not able to train a filter that showed performance near that demonstrated using techniques such as ASEF

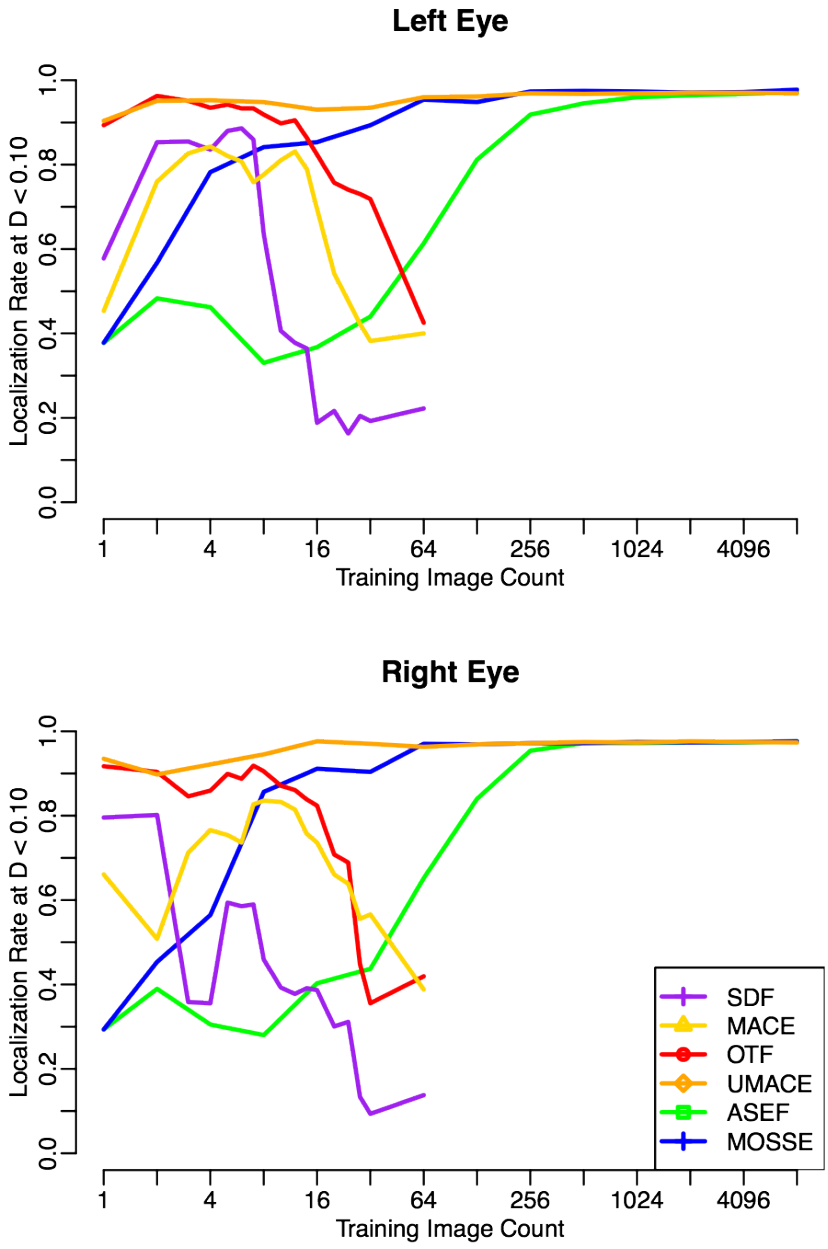

Figure 4.3.1: The effect of training set size on the performance of different filters when the search is restricted to a 20 pixel region around the expected eye location.

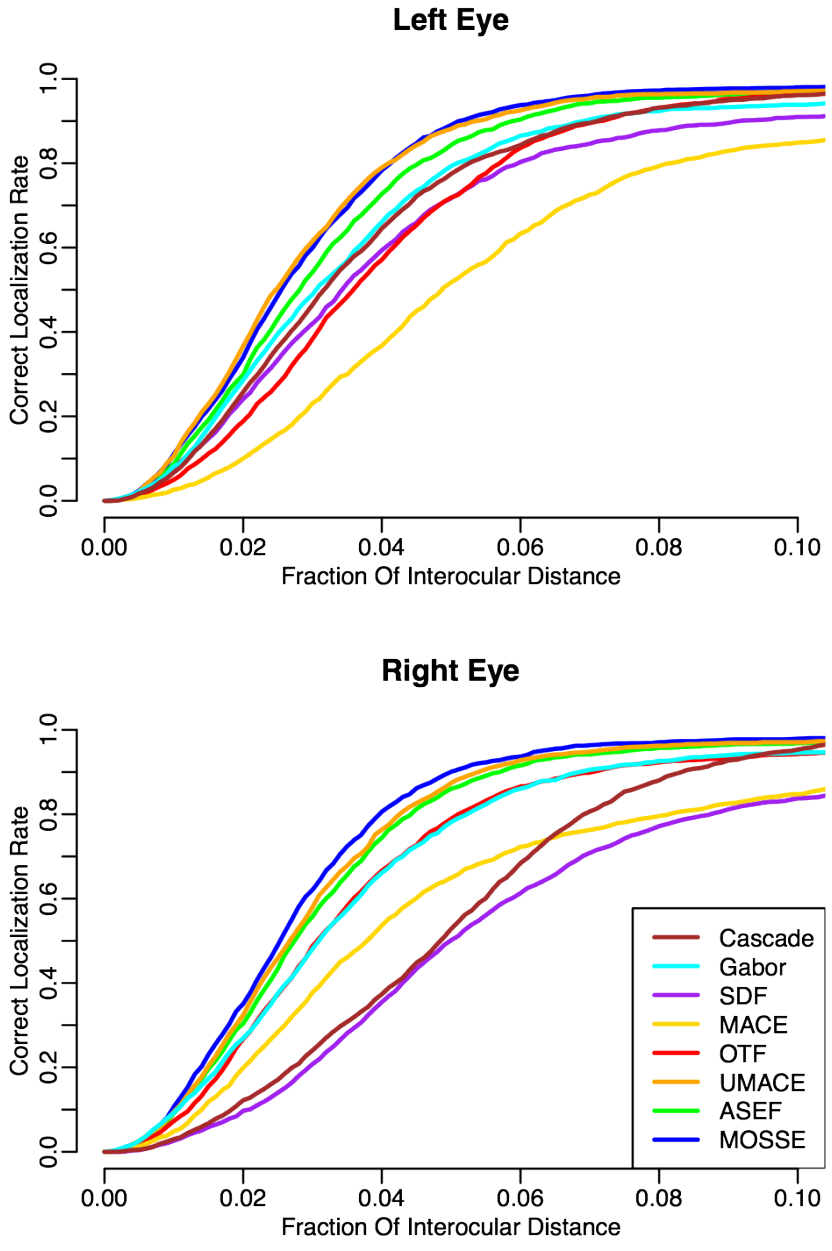

Figure 4.3.2: The performance on the testing set when the search for the eye is restricted to a region less than 20 pixels from the expected location.

or MOSSE. Because SDF, MACE, and OTF filters only can use a limited number of training images before the training process over-fits the training data. Therefore, the training images are very important and must be selected carefully.

4.4 Eye Localization Over the Entire Face

In this experiment the filters are tested on a more difficult problem where the filters will search the entire face for the location of the eye. This test is much more difficult because the filter needs to tell the difference between the target feature of interest, which is the eye, versus other features on the face such as the nose and mouth. In addition the filters must discriminate between the left eye and the right eye, which is extremely challenging because the eyes are so visually similar.

This new experiment uses similar procedures to those stated in the previous section. Like in that section, the images were split into the same training, validation, and testing sets. In addition, similar validation experiments were conducted to tune the filters to this new task. The results of this experiment are shown in Figure 4.4.1.

In many ways the filters performed similarly in this validation experiment to what they did in the previous section. For example, the SDF, MACE, and OTF filters still had a tendency to over-fit the training data. Also, the UMACE, ASEF, and MOSSE filters improved as additional training data was added. One key difference is that the ASEF and MOSSE filters still performed well on this task, while the performance of the SDF, MACE, OTF, and UMACE filters dropped considerably. Those particular filters were having a difficult time distinguishing the target eye from other features of the face, especially the non-target eye. The ASEF and MOSSE filters made very few mistakes of this type.

The reason for this is the method used to train the filters. The ASEF and MOSSE filters have access to the entire face image during training and the eyes can be located anywhere within the training image. Because training uses the entire face, the filters learn to both respond to the target of interest and also to suppress the response to other

facial features. The larger training images also allow the filters to learn about the context in which the eyes are found. Because faces have a predictable structure, the filters can learn that the eyes are located below eyebrows and next to noses. Because of this the nose is clearly visible in the filters trained using ASEF and MOSSE but is absent in the filters trained using SDF, MACE, OTF or UMACE. (See Figures 4.2.1 and 4.2.4.)

The SDF, MACE, OTF, and UMACE filters have a disadvantage. Training images for those filters are cropped such that the target, in this case the eye, is in the center of the image. In doing so, much of the surrounding face is removed. For this reason these filters cannot learn about the appearance of other face facial features and have much less context to draw from in terms of locating the target eye.

Figure 4.4.2 shows the performance of OTF, UMACE, ASEF, and MOSSE on the testing set. The OTF and UMACE filters still had difficulty discriminating between the correct eye and other facial features, while the ASEF and MOSSE filters were capable of accurately locating the eyes 96% of the time to within 10% of the interocular distance. ASEF and MOSSE show an increase of the accuracy of 20% over OTF and UMACE.

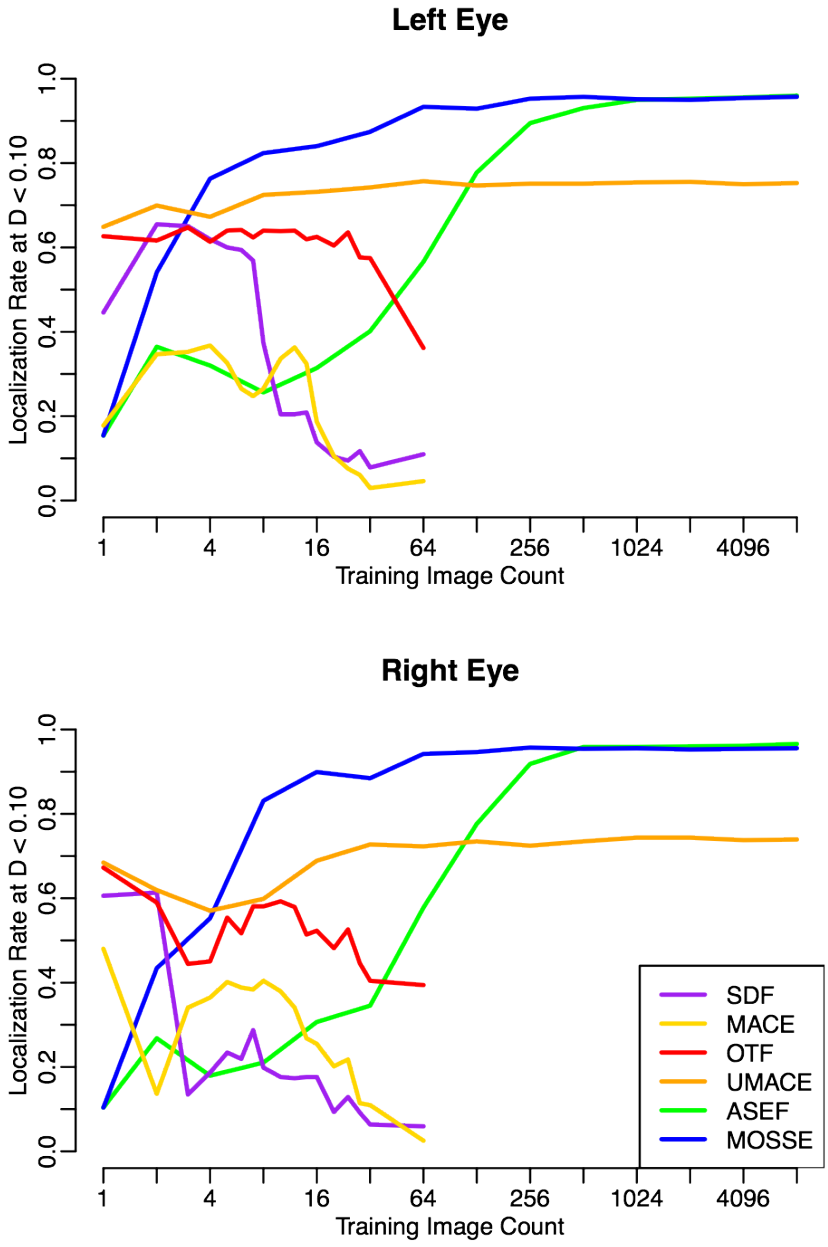

Figure 4.4.1: The effect of training set size on the performance of different filters when the search spans the entire face.

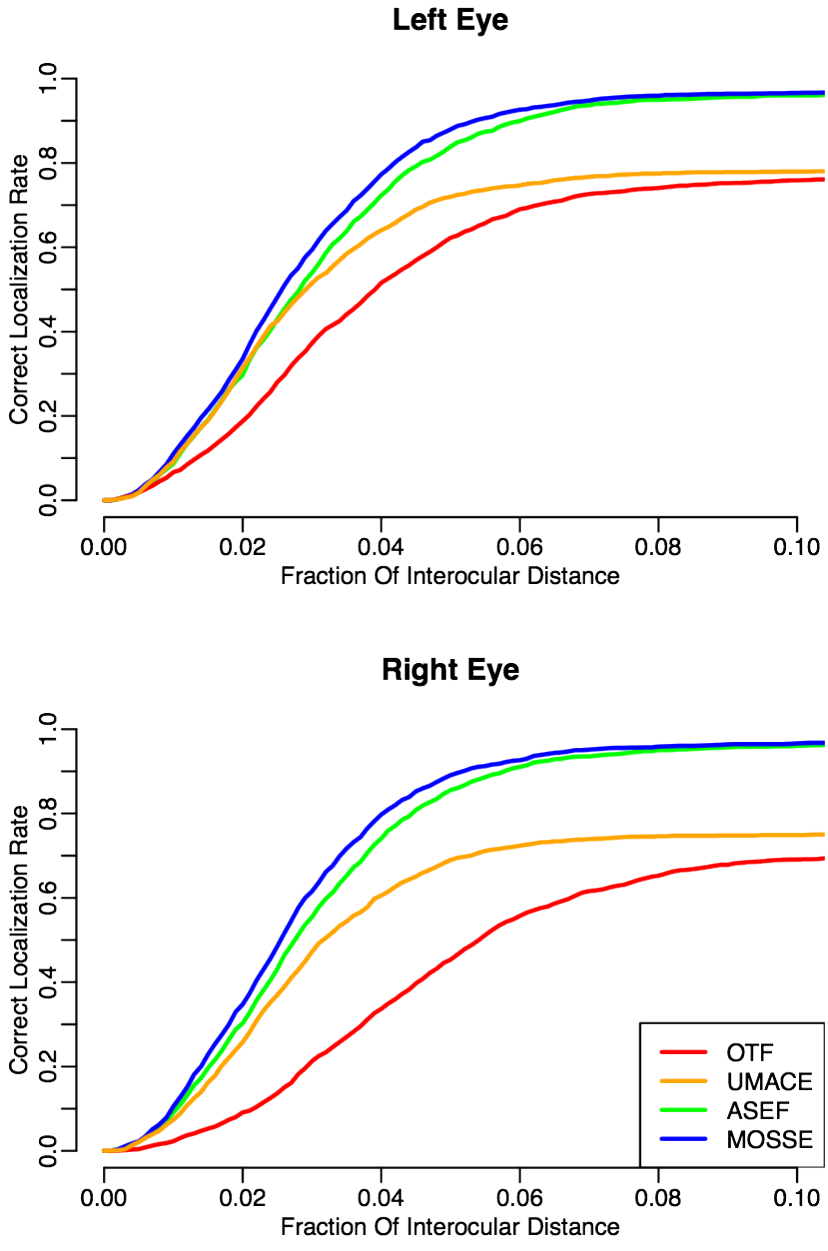

Figure 4.4.2: The performance on the testing set when the search for the eye spans the entire space.

Chapter 5

Filters for Person Detection

Object detection is one of the best known and most challenging problems in computer vision. Basically, object detection determines if an object is in an image and where it is located. Depending on the type of object and the complexity of the scene, the difficulty of object detection can range from an easy problem to a challenging one. Many of the challenges come from the ways that objects change appearance. For example, if you are trying to detect a specific type of chair, an image of that chair may look very different from the front than it does from the side. Additionally, different objects within a class can have different appearances; for example, an office chair may appear very different from a kitchen chair.

This chapter looks at the problem of detecting people in surveillance video. What makes this problem challenging is that there are many different people in this video and they are all wearing different clothing. Additionally, the detector must recognize these people from all different angles. Also, as the people walk, they are also changing shape, and their legs and arms move, which complicates their appearance. In order to accurately detect these people, a correlation filter will need to be produced that finds consistencies in the appearance of these people that can account for all these variations.

The previous chapter looked at an eye localization problem, where the eye was known to be in the image. With localization the algorithm simply had to find the maximum in the correlation output. For detection the correlation peaks for targets need to exceed a threshold while the correlation output for background must be below that threshold.

This difference makes detection a much more challenging problem than localization.

This chapter will also highlight some differences in the way Optimized Correlation Output Filters (OCOF)s are trained from prior methods. The standard way to create a correlation filter would be to crop many small images of people where there is one person in the center of each training image. Using Minimum Output Sum of Squared Error (MOSSE) and Average of Synthetic Exact Filters (ASEF), filters are produced from much larger training images that contain multiple people. Furthermore, Section 5.3 shows that this difference in how OCOFs are trained accounts for much of the improvement in filter performance.

Experiments are run using videos from the PETS 2009 data set, and the results are compared to two other filters and detectors. The results are compared to other correlation filters such as Unconstrained Minimum Average Correlation Energy (UMACE), which will show that the OCOFs perform better at this detection problem. In addition, the results are compared to two other state-of-the-art detection algorithms. The first is a Viola and Jones Cascade Classifier [78, 80, 79] that was trained to detect people on this data set. The second is a Parts-Based Classifier [25], which represents the state-of-the-art in person detection technology.

5.1 Image Preprocessing

The object detectors discussed in this chapter find people in the video frames based on appearance. Many systems that produce object detections in video do so by either learning a background model and doing background subtraction or by detecting frame to frame motion to determine object boundaries. Here we only consider appearance based detectors that do not require background models or frame to frame motion. Each image in the video is processed individually with the goal of detecting the people in the frame while producing only a small number of false detections. An example of this is shown in Figure 5.1.1 for the ASEF detector.

For each frame in the video, correlation-based object detection will be run in four

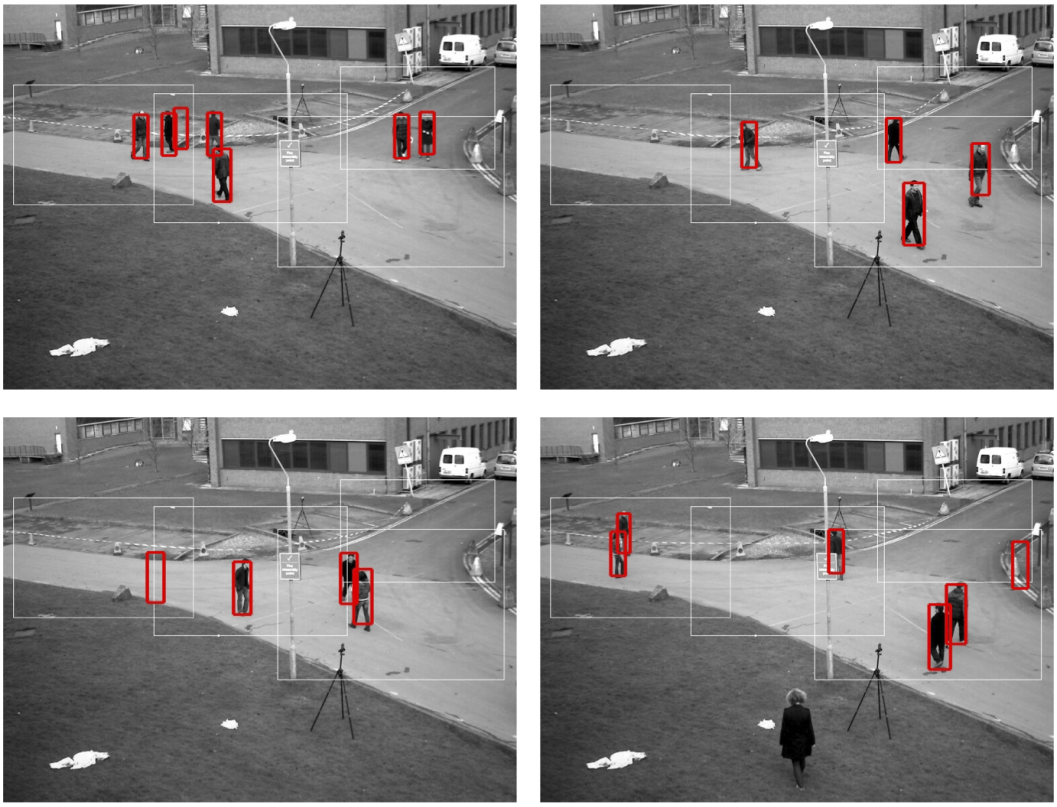

Figure 5.1.1: Some example frames from the ASEF-based detector on the PETS Data set.

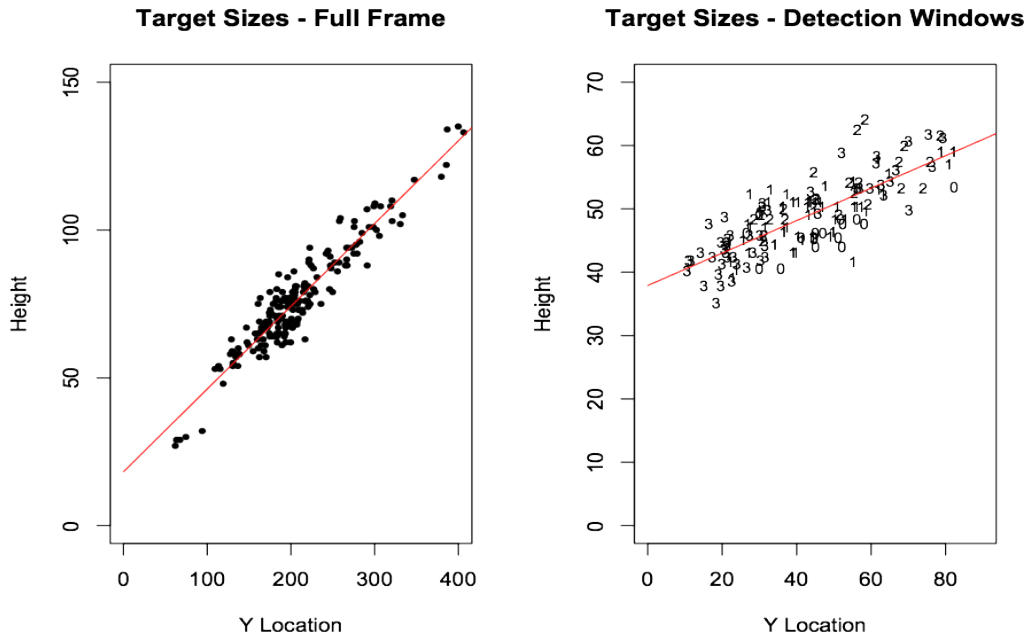

Figure 5.1.2: Detection Rectangles

subwindows; see Figure 5.1.2. One reason for splitting the image into these regions is to account for the scale change in the person's size. Because the camera is elevated, the person's size is related to his y pixel location using a linear expression. As the person gets farther away from the camera or higher up in the image, his size gets smaller. As he gets closer to the camera or lower down in the image, he gets larger. By measuring the distance from the top of a person's head to the bottom of the person's shoes for multiple people in multiple frames, the size of a person can be easily predicted. A linear regression was fit that related the y location to the person height:

$$h = 18.252 + 0.280y$$

This fit is shown as the line in the full frame plot in Figure 5.1.2. Rescaling the detection rectangles to 192×128 reduces the variability in a person's size so that a single correlation filter can be used for detection. Otherwise, to perform detection across multiple scales, many correlations would have to be performed using filters at different scales.

Another reason for splitting the frame into these regions is to allow each region to be easily correlated with the filter. Because we are using the Fourier transform to compute correlation, each region has to be exactly the same size. Each region is cropped and then rescaled to the size of 192×128, which is the same size as the filter. At this size, training and correlation can be performed quickly in the Fourier domain.

Before correlation, each region is preprocessed. Figure 5.1.3 shows an image containing three people. One difficulty with detecting these people is that these people are wearing different colored clothes. One person is wearing light colored clothes while the other two are wearing dark colored clothes. Because correlation filters are looking for patterns in the gray-scale version of this image, they have particular trouble with clothes. Specifically it is difficult to produce a filter that can simultaneously respond to a light person on a darker background and a dark person on a lighter background.

One solution to this problem is to produce two filters: One filter for a light person on a darker background and the other filter for a dark person on a lighter background. To detect these people two correlation operations would then have to be performed. A

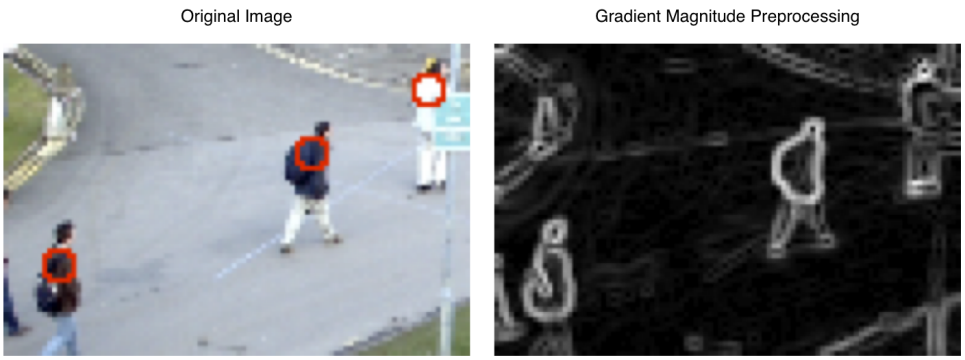

Figure 5.1.3: Gradient Magnitude Preprocessing

better solution is to preprocess the detection region such that the light people and dark people appear similar. In this case the gradient magnitude of the image is computed, which essentially performs edge detection. In the preprocessed image, people can then be detected based on their boundaries, and thus people with light colored clothes and dark colored clothes have similar appearances.

5.2 Training Filters for Detection

The ASEF and MOSSE filters were trained on 32 randomly chosen frames from the training sequence at time code 14-03. Each frame was split into the four regions shown in Figure 5.1.2, and each of those regions was perturbed four times to produce 16 training images for each of the 32 frames. The people in each frame were manually annotated by clicking a point in the middle of each person's torso. These annotations were used to generate the training outputs by placing a Gaussian peak at the location of each annotation.

In addition UMACE and Minimum Average Correlation Energy (MACE) filters were also generated for this training set. Instead of using the full detection region, those filters require that the training images use a small tile with a single person in the center of each one. For that reason, 64×64 templates were produced centered on each person. UMACE filters were trained on all the people in the training frames, while Synthetic Discriminant Functions (SDF) based filters only used 20 training tiles in order to mitigate over-fitting.

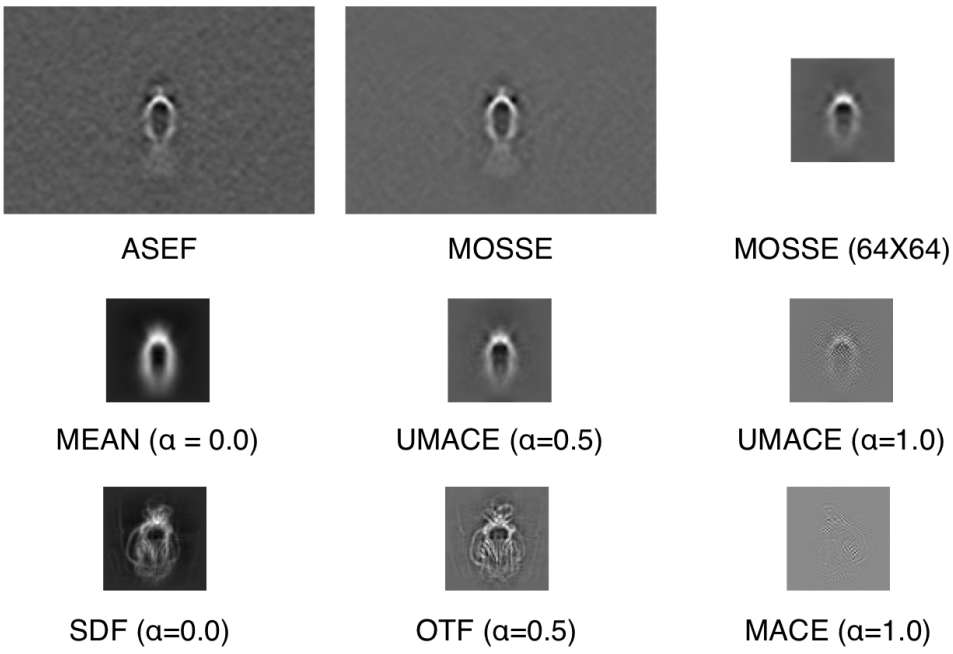

Figure 5.2.1: Filters Trained Using Different Techniques

In order to determine the source of the performance improvement of MOSSE, a version of the MOSSE filter was also trained on the same tiles that were used to train UMACE. As discussed in section 3.3, a UMACE filter is just a special case of MOSSE. By training MOSSE on the smaller 64 x 64 tiles, the filter is approximately halfway between a UMACE filter and the MOSSE filter trained on the larger image. Its results can therefore be used to determine whether or not the detection accuracy of MOSSE comes from training on larger training images or having a Gaussian shaped output.

The filters tested are shown in Figure 5.2.1. The first row shows ASEF, MOSSE, and the version of MOSSE trained on the 64×64 tiles. The second row shows three versions of UMACE filters using different settings for α. An α of zero is simply the mean of the training templates. The final row shows three versions of SDF filters where $\alpha = 0$ is equivalent to a standard SDF, $\alpha = 1.0$ is a standard MACE filter, and $\alpha = 0.5$ results in an Optimal Tradeoff Filters (OTF). All the correlation filters run at approximately the same speed, which is about 25 frames per second.

These filters are also compared to two non-filter-based methods that are chosen to

represent state-of-the-art performance. The first is a Viola and Jones Cascade Detector from OpenCV [84, 80, 79]. This detector used the same preprocessing as the filters and was trained on the same data set. This is one of the fastest non-filter based detectors, which on this data set runs at seven frames per second.

The second detector is based on the work of Felzenszwalb et al. and represents the current state-of-the-art in terms of accuracy [25]. It uses a Parts-Based model and a Support Vector Machine (SVM) classifier to detect people. The training used for this detector came prepackaged with the Matlab source code. While this detector offers good accuracy, it is also much slower than the filter-based method and processes about one frame every 2 seconds.

5.3 Comparing Detection Methods

To evaluate the detectors, they were run on sequence S2L1_T1234 of the PETS 2009 data set. The resulting detections were compared to ground truth from 50 manually annotated frames. These results are shown in Figure 5.3.2, which shows the detection rate for each method at a false detection rate of one per frame.

It can be seen that ASEF and MOSSE filters produced the best detection rates followed closely by the state-of-the-art Part-Based method of Felzenszwalb [25]. None of the other correlation filters came close to the detection rate of ASEF or MOSSE. The next best filter is UMACE with an alpha of 0.5.

Figure 5.3.3 presents the same results as a precision recall curve similar to those proposed in [1]. These curves show the fraction of correctly detected people on the y-axis compared to the fraction of false detections on the x-axis. ASEF and MOSSE produce nearly identical curves; however MOSSE evaluates slightly better. This is consistent with the results in Chapter 4 and further suggests that MOSSE is the preferred solution due to slightly better results and its stability when trained on small numbers of images. These results also show that ASEF and MOSSE have similar correct detection rates to the Parts-Based method at low false detection rates.

It is interesting to observe the drop in the detection rates for MOSSE filters trained on the 64×64 tiles. The results show that it is slightly worse than UMACE but still better than the other filters. Because we know that the MOSSE filters trained using the 64×64 tiles are in many ways similar to UMACE, it is therefore not surprising that when trained on the same data both methods result in similar performance.

What is more interesting is that this indicates that much of the power of ASEF and MOSSE filters comes from their ability to train on larger images with multiple uncentered targets. Training in this way provides not only more information about the targets and the context in which the targets are located but also more information about the background. ASEF and MOSSE can therefore produce filters that better discriminate targets from background. In addition, the larger training images better represent the images that are used as input during testing than the artificially cropped and centered templates that the other filters use for training.

Figure 5.3.4 compares the rate at which the detector can process frames. The computation time required for all of the filter-based detectors should be identical at about 25 frames per second.[1] The Cascade Detector is representative of the state-of-the-art in real-time object detection and runs at about 7 frames per second.[2] While this is much slower than the filters, it is still fast enough to be run on live video. The Parts-Based detector represents the state-of-the-art in accuracy but is poorly suited for real time detection. The processing that takes place for the Parts-Based detector is significantly more complex and runs much slower at under one frame per second. While the Parts-Based detector was run in Matlab[3] and could be improved by conversion to C, it is probably

---

[1]Run in Python on a 2.4 Ghz Mac Book Pro using the scipy library and fftpack for computationally intense processing.

[2]Run in python on a 2.4 Ghz Mac Book Pro. The cascade detector was provided by the OpenCV library.

[3]Matlab version 7.6 HP-XW4600-Core2 Duo 2.3Ghz

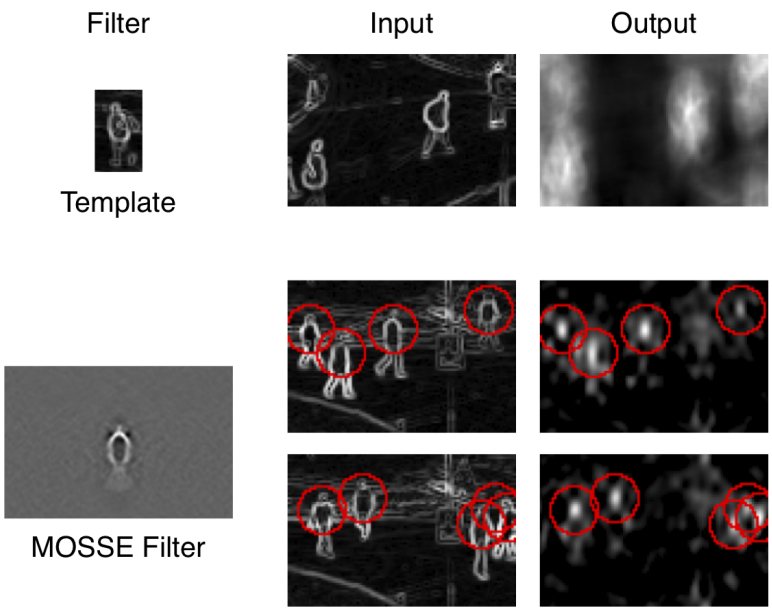

Figure 5.3.1: Visual Comparison of a Template to a MOSSE Filter

still too slow for most real time systems.

5.4 Chapter Summary

This chapter examined OCOF filters in an object detection context. The accuracy was compared to other correlation filters and also to two well known person detection algorithms. The results showed that OCOF filters performed much better than other correlation filter training techniques including MACE and UMACE filters. These results also indicate that much of the power of OCOF filters comes from their ability to train on larger training images that contain multiple examples of targets. The OCOF detectors also evaluated slightly better than the Viola and Jones Cascade Detector [78, 80] and the Parts-Based detector of Felzenszwalb [25], which are representative of the state-of-the-art in person detection techniques. In addition, it is shown that the filter-based approaches are much faster, which make them suitable for real-time video processing.

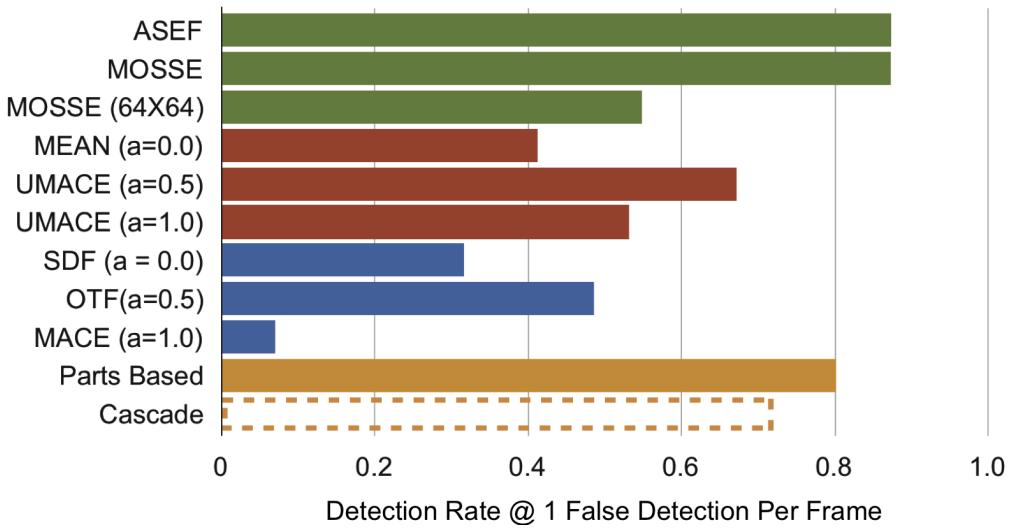

Figure 5.3.2: All filter comparison S1L2_T1234

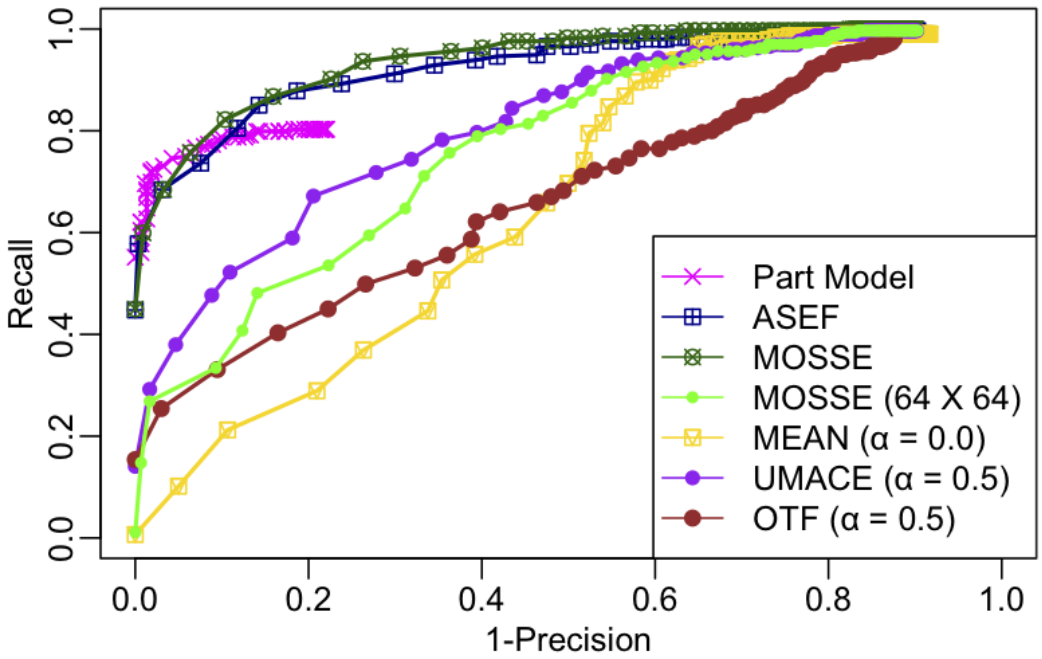

Figure 5.3.3: Precision Recall on Selected Filters. (S1L2_T1234)

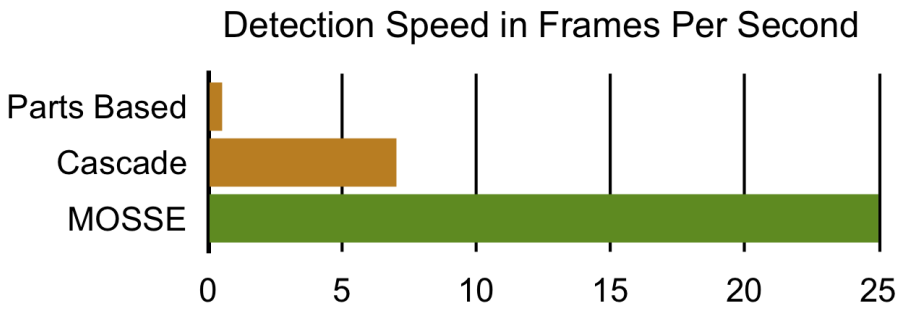

Figure 5.3.4: This plot shows that filter-based detection is much faster than the more complicated person detection techniques.

Chapter 6

Filters for Visual Object Tracking

This chapter extends the work of [7], which applies Minimum Output Sum of Squared Error (MOSSE) filters to the problem of visual tracking. When a target is located in a video, it is often useful to track that target through additional frames because each frame provides additional information about the target. In visual tracking the target appearance changes very little from frame to frame, which makes it a much easier problem than detection. Because of this, tracking uses fewer computational resources than performing detection on each frame.

Correlation filters are ideal for object tracking. As discussed before, even simple templates work well for tracking when the target's appearance changes slowly from one frame to the next. Therefore, templates can in most cases find the same targets in following frames. So far it has been demonstrated that MOSSE filters improve upon simple templates in object detection under much harder scenarios. It will be shown here that by using a MOSSE filter instead of a simple template, tracking becomes very robust. In addition, the tracker has other nice features like the ability to adapt to changing target appearance and the ability to detect and recover from occlusion.

Visual tracking has been a known problem for many years. Many of the earliest solutions used simple techniques like templates matching and gradient descent search to quickly track objects in video. One example, which is still used in many real systems today, is the Lucas-Kanade (LK) tracker [50] that is based on template matching paired with gradient descent to find a locally optimal match to the target in a previous frame. Its

popularity is due to its speed; however, it is also known to fail quite easily and typically cannot track targets for extended lengths of time.

Other older trackers use simple features such color or motion to identify a "blob" in an image and then track it from frame to frame. Continuously Adaptive Mean Shift (CAM Shift) [12, 11] for example tracks an object using its color histogram. This works well when the target and background occupy different regions of color space, for example a white soccer ball on a green soccer field. It also runs into trouble when the target and background are similar colors or when applied to gray scale videos. Like the LK tracker, Mean Shift tracking still works well for many applications and is still the subject of recent research [17, 18].

Another simple method relies on the motion of a target and assumes that the background is not changing. The object is initially detected by subtracting a video frame from a background image or model. When subtracted, background pixels become close to zero and moving objects cause non-zero values in the resulting image [27, 21, 61, 41, 42, 55]. Objects are then tracked using techniques like Kalman filters [83, 40] to follow those objects through multiple frames. While this technique is fast, tracking is easily confused if the targets stops or if two moving targets intersect. Motion-based tracking has found many uses in video surveillance and is a key component to most video analytic systems [28, 31].

In recent years, many robust tracking strategies have been introduced that can track objects through complex appearance changes. One issue with the older and simpler tracking techniques is that they use simple heuristics or gradient descent to determine the new location of a target. This is an issue because those techniques can often converge to a local minimum that is not the true location of the target. The study of much recent research in visual tracking is a stochastic search method called condensation or particle filters [35]. These algorithms evaluate many random image windows near the expected location of the target, and the one that evaluates best is used as the new location. While this is a relatively inefficient way to search for the best location, it is

much more efficient than an exhaustive search and can also simultaneously evaluate and track multiple hypothesized locations.

One of the most popular particle filter-based trackers is called Incremental Visual Tracking (IVT) [48, 65], that pairs a particle filter search strategy with an incremental principal components analysis to model the targets appearance. This technique was immediately popular because it demonstrated tracking in some pretty complex scenes and was able to estimate the rotation and scale of the target in addition to its x and y locations.

The remainder of this chapter will discuss the adaptation of MOSSE filters for adaptive real-time visual tracking and will compare it to three modern tracking algorithms. The first is IVT that represents one of the turning points in how objects are visually tracked. The second is called Online Ada-Boost (OAB) [29, 30] that adopts many elements of the popular Viola and Jones Detector [80] in an online and adaptive algorithm for visual tracking. The other tracker is called Multiple Instance Learning (MIL) Track [3] that has many similarities to OAB but uses online MIL to model target appearance. Source code for these trackers was obtained from [66] and [4]. The trackers will be evaluated on a new data set of 72 videos that is designed to test their ability to follow a target through many common tracking difficulties and provides a framework for statistical hypothesis testing.

6.1 MOSSE Filter Based Object Tracking

This section describes the MOSSE filter visual tracking algorithm. An illustration is found in Figure 6.1.1 and additional details can be found in [7]. Visual tracking is initialized on the first frame of a video sequence. Given that frame and a bounding rectangle, the tracker generates a filter representing the appearance of the target. For additional frames in the sequence, the tracker convolves the frames with the filter and then locates the target by finding the peak in the correlation output. The tracker then uses that location and the image data in the new frame to update the correlation filter.

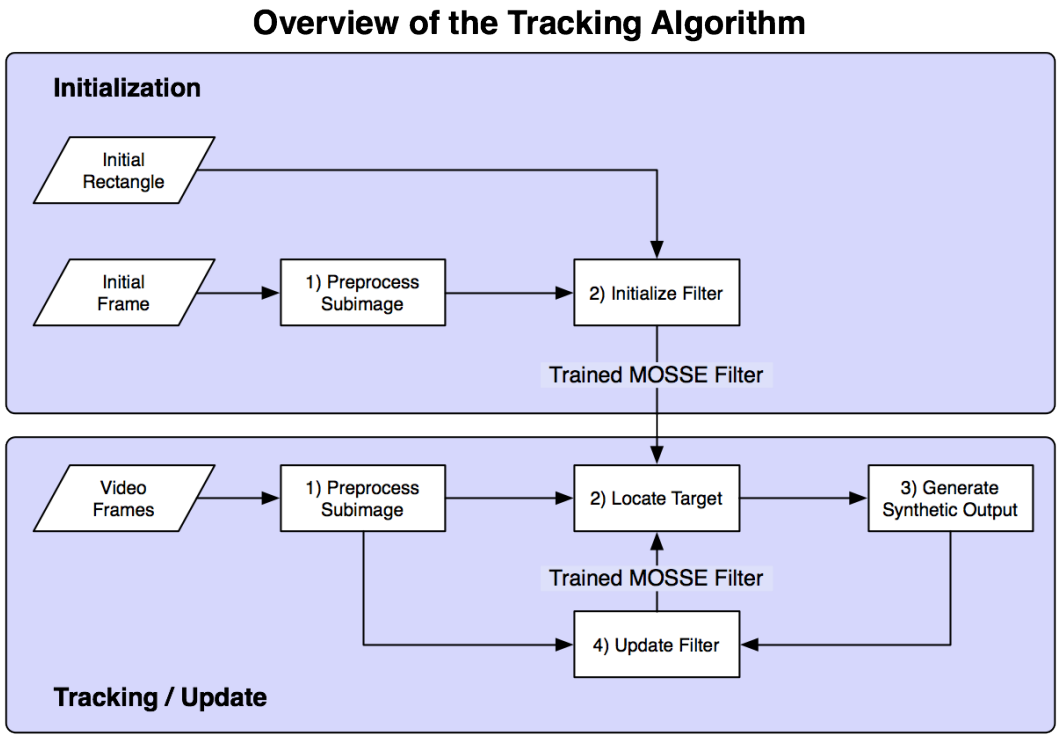

Figure 6.1.1: This provides an overview of the tracking algorithm including both initialization on the first frame and then tracking on additional frames.

The filter is therefore constantly adapting to the new appearance of the target, which allows it to accurately follow the target through long video sequences where the target appearance is constantly changing.

6.1.1 Preprocessing

Tracking begins by pre-processing the frame. First, the tracking subimage corresponding to the tracking rectangle is cropped from the frame and rescaled to 64×64. The pixel values then need to be normalized in order to reduce lighting effects and other issues related to contrast. The filters will naturally select high contrast edges as the strongest features to track. Therefore, transforming the pixel values using a log function is used to enhance low contrast regions and provides more edges for the tracker to exploit.

Next the pixel values are normalized to have a mean of zero and the norm of one. This normalization is standard practice as it reduces the effects of changing illumination

and makes sure that each frame has the same influence on the trained filter. Finally, because correlations in the Fourier domain are actually performing a circular correlation, a cosine window is applied to the subimage. This eliminates any edge effects and also focuses the tracker on features near the center of the target.

6.1.2 Initialization

If this is the first frame in the video sequence, an initial correlation filter needs to be generated. For this purpose a standard MOSSE filter is trained by using nine affine perturbations of the subimage. Because the tracker needs to be trained in between frames of a real-time video sequence only a small number of images can be used for this initial training. For this reason, a MOSSE filter is used because it converges faster than Average of Synthetic Exact Filters (ASEF) under such conditions. In addition, regularization is also used to ensure fast convergence.

For MOSSE, the training images (f_i) come from the preprocessed and affine perturbed subimages that were extracted from the first frame of the sequence. The training outputs (g_i) are synthetically generated correlation outputs that contain a single Gaussian peak centered on the target object. A standard MOSSE filter can then be created using the equation:

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^* + \epsilon} \tag{6.1.1}$$

See Section 3.4 for details.

6.1.3 Tracking and Filter Update

For additional frames the correlation filter is then used to track. When the new frame is given to a tracker, the first step is to correlate the filter with the new preprocessed subimage extracted from the location of the tracking rectangle:

$$G = H^* \odot F$$

This results in a peak that indicates the new location of the target and is used to update the location of the tracking rectangle.

The location of the target in the new frame is then used to update the filter. Basically, given the new location, a new synthetic output is generated with the location centered on the target. This provides a new input and output training pair that can be used to update the filter, using the equations:

$$H_i^* = \frac{C_i}{D_i} \tag{6.1.2}$$

$$C_i = \eta(G_i \odot F_i^*) + (1-\eta)C_{i-1} \tag{6.1.3}$$

$$D_i = \eta(F_i \odot F_i^* + \epsilon) + (1-\eta)D_{i-1} \tag{6.1.4}$$

where η is a learning rate in the range $[0, 1]$. This new equation is a straightforward modifications of Equation 6.1.1, which allows the filter to continually adapt to the changing conditions of the scene while placing the more influence on recent frames. Learning rates are typically in the range [0.01,0.15]. Slower learning rates reduced the likelihood that the tracker will drift off target or adapt to follow a non-target object, but it also limits the tracker's ability to adapt to quick appearance changes. Faster learning rates allow the filter to adapt quickly but the tracker is also more likely to adapt to track objects in the background.

6.2 Track Quality: Occlusion Detection and Recovery

One of the main advantages of using MOSSE for tracking is that a simple measure of correlation peak quality called the Peak-to-Sidelobe Ratio (PSR) is used to determine that the tracker is working properly. The PSR is a standard metric used to evaluate the "sharpness" of a correlation peak and is the ratio of the peak height to the standard deviation of the non peak value or side-lobe. In this case, the side-lobe is the entire correlation output excluding an 11×11 pixel region surrounding the peak. Given the

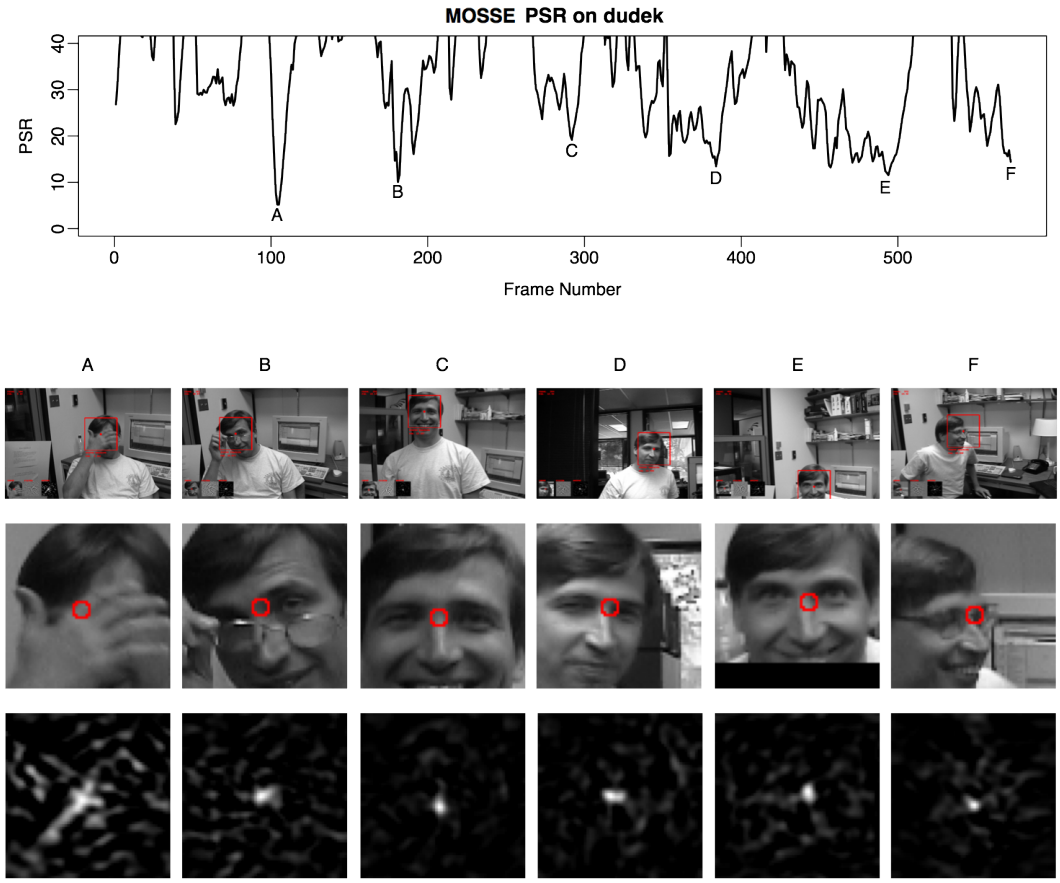

Figure 6.2.1: Upper: the plot shows that the PSR on the Dudek sequence is closely tied to track quality. Drops in the PSR correspond directly to the most difficult frames. Lower: This shows the raw frames, tracking input and correlation output for the 6 frames labeled in the plot. The correlation output shows the resulting peaks, where "A" shows that the peak has almost disappeared during an occlusion. Figure adapted from [7].

correlation output g the PSR is computed as:

$$\text{psr}(g) = \frac{g_{\max} - \mu_{\texttt{sl}}}{\sigma_{\texttt{sl}}}$$

where g_{max} is the value of the peak, μ_{sl} is the mean value, and σ_{sl} is the standard deviation of the values in the side-lobe. The PSR is therefore a dimensionless ratio that measures the peak value relative to the "standard deviation" of the side-lobe.

An example of this is shown in Figure 6.2.1, which shows the PSR for each frame of the Dudek sequence. There are a few interesting things to notice in this plot. First, most frames in the video produce a high PSR often exceeding a value of 12.0. This indicates

high confidence that the tracker is operating correctly. There are six frames where the PSR is especially low that have been labeled "A", "B", "C", etc. These locations correspond to what human judgement would say are some of the most challenging sections of the video. The raw frames, input images, and correlation outputs are shown below the plot. It can be seen that even for some of the more difficult frames (B, E, and F) the correlation peak is still strong.

For frame A the face is mostly occluded and the PSR drops to 5.19, which is a very low value. It can be seen in the correlation output that the peak is difficult to distinguish relative to other values in the side-lobe. By analyzing many videos, it was found that the tracker rarely fails for PSRs larger than 9.0. Likewise, PSRs below 7.0 often correspond to instances where the object has been occluded.

One of the most successful applications of the PSR is in occlusion detection and recovery. A threshold is set between 7.0 and 9.0 that is used to detect occlusion. When the PSR drops below this value, the filter update is stopped. This keeps the filter from being corrupted by non-target data while the target is not visible. In some cases the target reemerges from the occlusion and the PSR returns to values above the failure threshold. In these cases there is high confidence that the target is again visible and tracking can be resumed. While occlusion recovery is one benefit of this measure of track quality, it could also be useful in the context of larger tracking systems as a way to detect and handle tracking failure.

6.3 Evaluation: Tracking Through Challenges

A typical way to evaluate visual tracking algorithms is to run the algorithm on a small number of videos that contain many challenges. The value of the tracking method is demonstrated by showing that the algorithm can track the target through the entire video and those results are often compared to a few competing methods. These demonstrations have a number of problems including: there is no agreed upon protocol, results are evaluated qualitatively instead of quantitatively, and if the tracker fails early in the video

there is no chance for redemption for the remainder of the video. In addition there is no agreed upon set of videos and therefore results are often self censored by not publicising videos in which the tracker performed poorly. The bottom line is that many tracking papers make a case that a new tracking algorithm is an improvement on the state-of-the-art based on only one or two tracking failures.

Another problem with current methods is that in the long videos most of the frames clearly show the target object and the appearance of that object changes relatively slowly. For these frames, tracking is relatively easy. An evaluation should focus on the most difficult sections of these videos where distinctions between algorithms will stand out. In these difficult sections, the trackers have the most difficulty and either successfully follow the target through the challenging frames or loose track of the target and fail.

In [7], the MOSSE filter was evaluated with similar methods and additional videos have been posted on YouTube: http://www.youtube.com/user/bolme2008. Although these results showed the tracker performing well, this chapter will set up a more formal test of tracking ability and use that test to compare MOSSE to other tracking algorithms.

The philosophy behind this evaluation is to curate a large collection of short video sequences that are focused on the most difficult tracking challenges. The videos span a variety of difficulties from moderate to extremely challenging, where the goal was to include enough variety of "tracking challenges" and "target types" that creating an algorithm that could track 100% of the frames would be impossible with current technology. The trackers will be evaluated based on how many of these challenges can be successfully overcome. A collection of 72 video clips were extracted that each contain 90 to 150 frames. They represent many of the most challenging sections of the demonstration videos that have been posted on the web. These videos come from the IVT and MIL Track websites that contain tracking sequences for a variety of object types, the VIVID data set which contains videos of vehicles, and other publicly available videos collected from websites like YouTube.

Each video contains one or more "tracking challenges" that include: scale changes,

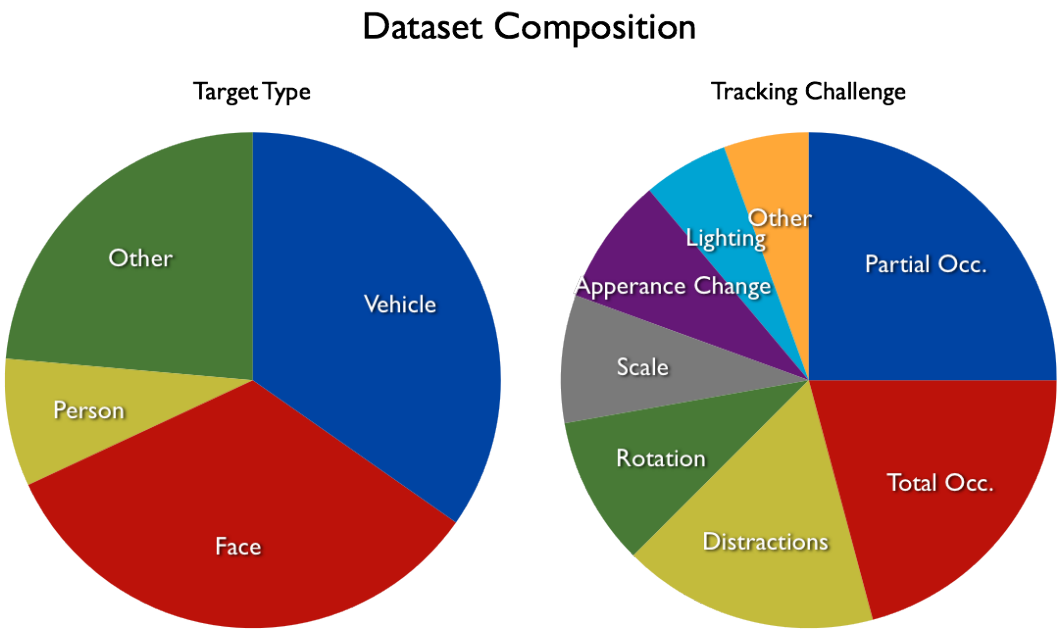

Figure 6.3.1: These charts illustrate the composition of the tracking data set by both target type and tracking challenge.

rotation changes, lighting changes, miscellaneous appearance changes, distracting objects, partial occlusion, and total occlusion. In addition, the videos contain different target types that include faces, vehicles, people, toys, and a few other random objects. The data set composition is shown in Figure 6.3.1.

6.3.1 Evaluating Successful Tracking

The trackers are evaluated on each frame of the videos to determine if an object is being successfully tracked. Ground truth for this evaluation was created by manually annotating the target center point in each frame of the videos using an image annotation program for the Apple iPad. Target velocity can also be estimated by finding the difference in location in the current frame from the previous frame. This provides a target location and velocity (both measured in pixels) for each frame of the video.

In some frames it is difficult to determine the target location, and so those frames were excluded from the evaluations. These cases are frames where the target is more than 50% occluded, subject to out-of-plane rotations of more than 45 degrees, and frames containing

significant motion blur where the target center cannot be determined. The trackers will not be penalized for not following a partially occluded object; however, when the target emerges from the occlusion, it is expected to restart tracking.

A few simple heuristics are used to determine if a target is being tracked in each individual frame. First, the target center point produced by the tracker is expected to be within a $s \times s$ pixel box centered on the ground truth location. The size of this box is a function of the size of the target $w \times h$ and the estimated 95% confidence interval on the accuracy of the manual locations (± 5.0 pixels).

$$s = \frac{\max(w, h)}{2} + 2(5.0)$$

The later is necessary because when the target size is small, errors in the manually selected coordinates become significant. Some tracking algorithms such as MIL Track and OAB tend to drift back and forth across the target while still maintaining a track. This box provides a reasonably sized region that will allow the trackers to drift without counting against the tracker as a failure. The velocity measured by the tracker also has to be close to the velocity measured from ground truth. Therefore the difference in the velocities needs to be below the threshold:

$$||v_{target} - v_{truth}|| < 0.02(\max(w, h)) + 5.0$$

In most cases, the trackers to a better job of determining the targets velocity than the ground truth, so the tolerance that is built into the previous equations is to correct for inconsistencies in the manually selected coordinates.

Once the individual frames have been labeled as success "S" or failure "F", the entire sequence of frames needs to be evaluated for each track. Examples of this can be found in Figure 6.3.2. It is not uncommon for some individual frames to be incorrectly evaluated. For example, some successful frames evaluate as failures due to incorrect velocity measurements in ground truth data, or some failed frames evaluate as successes when the motion of the target, camera, and track all just happen to coincide. These errors however tend to be brief.

To robustly detect a "Point of Failure" for a track, the sequences of "S" and "F" are considered in 10 frame windows. (At this point frames missing ground truth because of occlusion or other issues have already been dropped.) If the first frame in that window evaluates as a failure and if over 70% of the frames in that window are failures, then the track is considered to have failed at that frame. This provides some tolerance should the manual annotation disagree with the tracking output for just a small number of frames. Likewise if a track has failed and the 10 frame window starts with a success and contains 70% success, the track is considered to have recovered and nullifies any previous failures. It should be noted that failures and then recoveries are rare and occurred only four times for the MOSSE tracker with occlusion detection and fewer times for the other trackers.

6.3.2 Comparing Algorithms

When evaluating algorithms, it is often beneficial to provide one score that characterizes an algorithm's performance on a particular test. In this case the fraction of total frames tracked is used for this purpose (See Figure 6.3.2 for details). This score provides a single number that can be used to compare the performance of the algorithms. The results of this evaluation are shown in Figure 6.3.3 where a score of 0.0 indicates the tracker failed on the entire data set, and a score of 1.0 indicates perfect performance.

Six different tracking algorithms are evaluated here. Three of those algorithms represent recently published work in the area of visual tracking. The other three algorithms are different variants of the MOSSE based tracker:

- Incremental Visual Tracking (IVT) - Uses a particle filter to model the targets position and an incremental principal components analysis to model appearance.[65]
- Online Adaboost (OAB) - Uses an exhaustive search within a given radius to determine target motion and an OAB classifier for target appearance.[29]
- Multiple Instance Learning (MIL Track) - Uses an exhaustive search within a given radius to determine target motion and a MIL classifier for target appearance.[3]

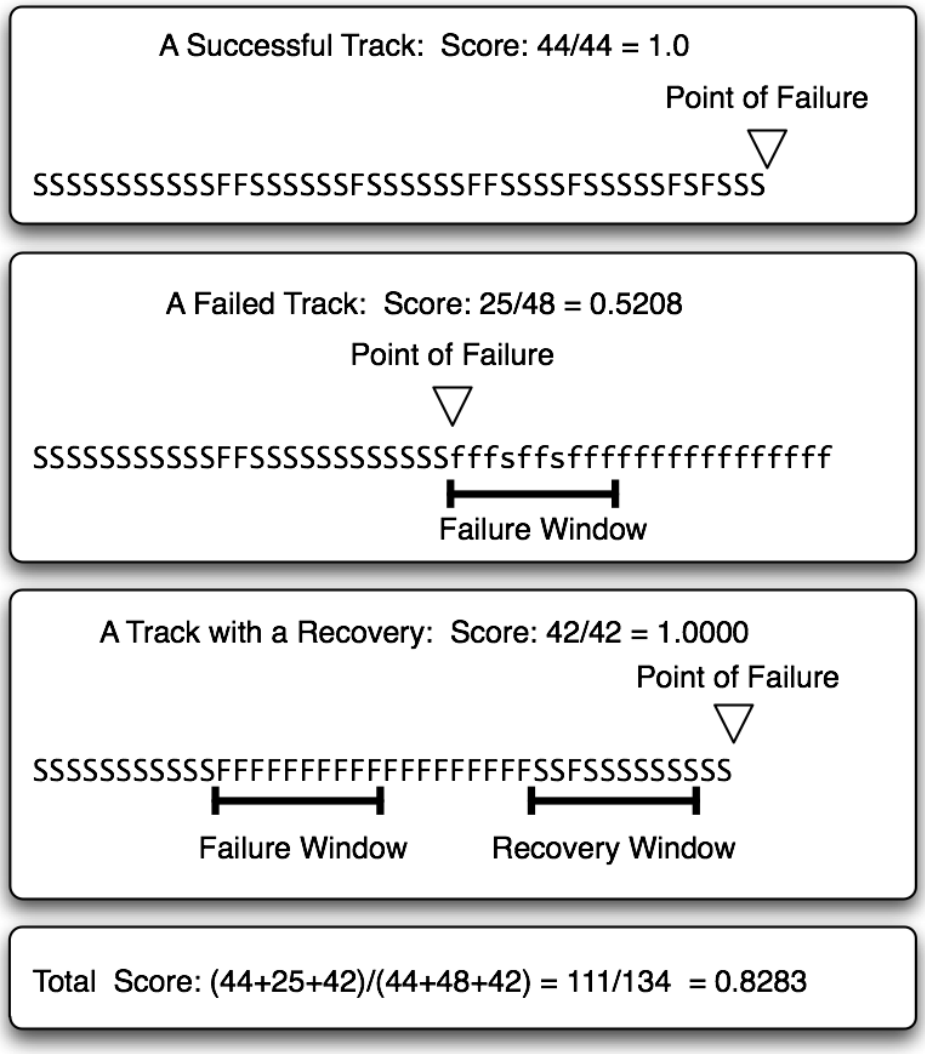

Figure 6.3.2: This is an example of how track scores are computed. Each S and F indicate the evaluation of Success or Failure on a single frame in the video sequences. (Frames that have missing ground truth due to issues like occlusion have already been removed.) A 10 frame window is used to robustly detect failures or recoveries. If the first frame in the window is a F and 7 of 10 frames in that window are failures, the start of that window is the point of failure. The opposite is used to detect recoveries that nullify any prior failures. The track scores are computed based on the "Point of Failure" relative to the total number of frames evaluated. Therefore any "single frame failures" before the "Point of Failure" are counted as successes (upper case letters), and any "single frame successes" after the point of failure are counted as failures (lower case letters).

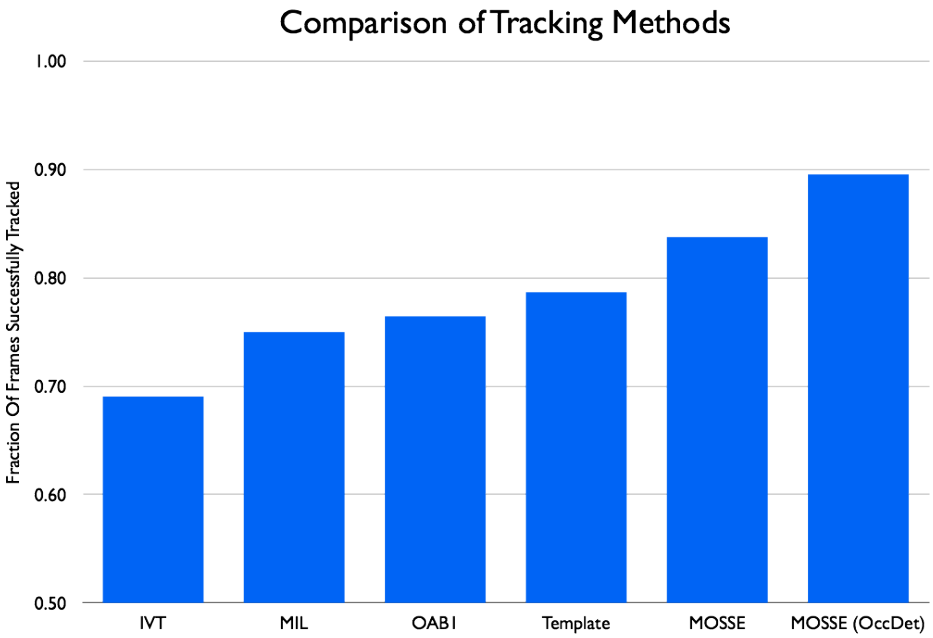

Figure 6.3.3: This compares the scores of the tracking algorithms evaluated.

- Template MOSSE - Uses a variant of the MOSSE tracker that has been configured to approximate a correlation with a simple template that is a weighted average of previous frames. It is intended to be a straw man for filter based methods. Occlusion detection is not an option.[7]

- MOSSE - Uses a standard MOSSE tracker with occlusion detection disabled. The tracker will therefore always select the highest value in the correlation output as the new track center and it will always perform a filter update.[7]

- MOSSE (Occ Detection) - Uses a variant of the MOSSE tracker with occlusion detection enabled. This tracker will detect when the target is no longer visible and stop the online update. If the target reappears in the tracking window, the filter may detect its presence and resume tracking.[7]

Figure 6.3.3 compares the total number of frames tracked by each algorithm. There are a number of surprising things in these results. First IVT, MIL Track, and OAB are intended to represent the state-of-the-art in visual tracking. None of those algorithms

have occlusion detection so it was expected that MOSSE with no occlusion detection would have similar performance. This is not the case; MOSSE actually performs much better.

What is more surprising is that the template approach, which was intended to be a straw man for the filter base methods, actually produced a higher score than all three of the recently published and much more complex trackers. The fact that the template-based tracking is doing so well leads to a couple of important questions:

Why is such a simple technique performing as well as the state-of-the-art in visual tracking? The template-based tracker is actually based on the same code base as the MOSSE tracker. The only difference is that instead of computing a standard MOSSE filter for the online update it has been altered to approximate a simple template. The MOSSE tracker represents many man months of research and development, has been run on hours of video, and has been carefully tuned to be an excellent general purpose tracking algorithm. It is therefore reasonable to assume that even when the MOSSE filter is swapped with a much less sophisticated simple template, the tracker will still perform well.

Why aren't the newer and more sophisticated trackers performing better on this data set? There are probably a number of reasons. The videos selected for this data set are intentionally chosen to be difficult. While the goal was to span a broad range of target types and tracking challenges so that no single tracking algorithm would have an advantage, it is quite possible that the selection was subconsciously influenced to prefer videos for which correlation might perform well. Some evidence for this is in the large proportion of videos that contain total occlusions for which only the MOSSE tracker has a solution. In addition, because I am not an expert in the other algorithms, it is possible that they may not have been well tuned for this particular data set.[1] It is

[1]For IVT results are reported for the Dudek tuning. Other tunings were tried but did not have a significant effect on tracking ability. For MIL Track and OAB the code was not modified. Two variants of OAB (OAB1, OAB5) were tested. Results are reported for OAB1 which performed much better on

also possible that the more sophisticated algorithms have been adapted to perform well on a specific subdomain of tracking problems and therefore are not well suited to the variety of challenges presented by this data set [10].

To shed more light on why the template based approach performed better on this test, MIL Track was compared directly to the Template tracker. Because in the majority of the videos both trackers either succeeded at tracking all frames or failed at the same place, the analysis counted the video sequences where the two trackers differ. The template-based approach tracked the target longer on 18 videos and MIL Track performed better on 11 videos with 42 ties.

Figure 6.3.4 shows the results of this analysis partitioned by video source, target type, and tracking challenge. The primary conclusion is that the template-based approach performed much better on vehicle tracking scenarios while MIL Track performed better on face tracking scenarios. While it was expected that MIL Track would have a strong advantage on sequences for its own website, this was found not to be the case. From viewing the output of IVT and OAB, it would appear that those methods also had considerable difficulty on the vehicle tracking scenarios from the VIVID data set.

6.3.3 Speed Comparison

In addition to a tracker's ability to follow an object through difficult parts of videos, there are other aspects of the tracker that could be evaluated. One of the most important is the tracker's speed. An advantage of the filter-based approach is that correlation can be computed efficiently using the Convolution Theorem. This makes the filter-based approach very fast. In contrast, the appearance models and search strategies for the other trackers are more complicated and therefore much slower. Evaluating the speed is also important because tracking should reduce the need to run detection on the entire frame.

---

this test.

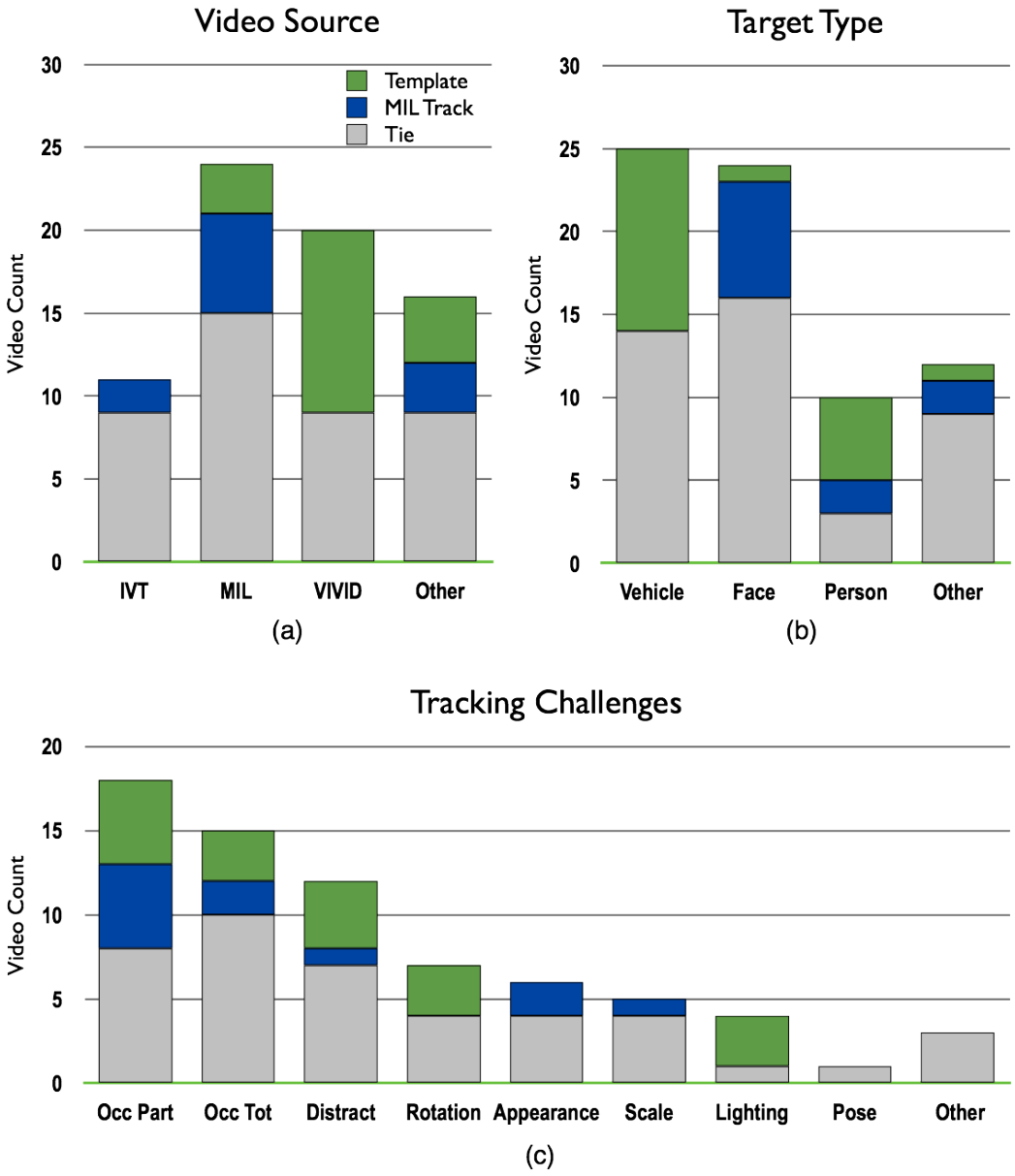

Figure 6.3.4: These are stacked bar charts comparing the Template Tracker to MIL Track based on (a) video source, (b) target type, and (c) tracking challenge. Green indicates the number of video sequences where Templates outperformed MIL Track and blue indicates the number of sequences where MIL Track outperformed Templates.

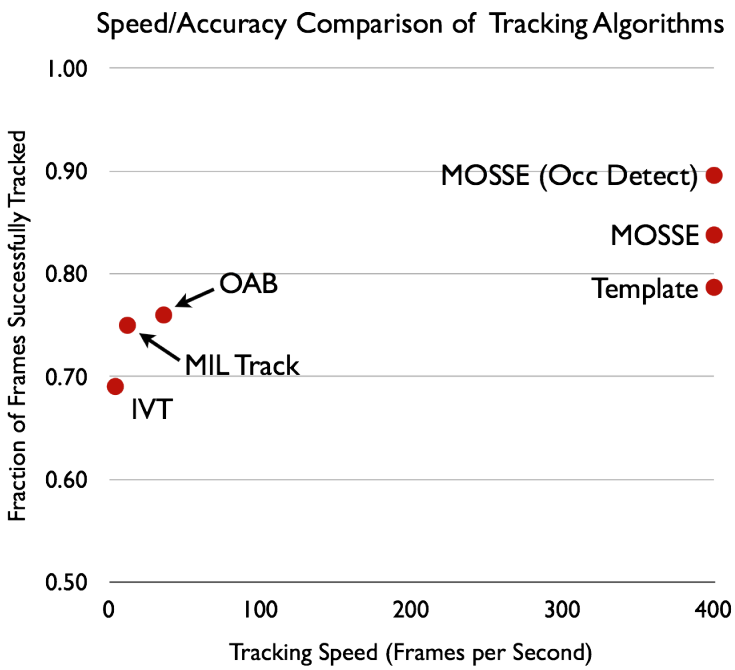

Figure 6.3.5: This plot shows the accuracy and speed of the tracking algorithms.

Figure 6.3.5 compares both the accuracy and the speed of the six trackers. Speeds were benchmarked running on a single core of a Mac OS 2.66Ghz Intel Core i7 Mac Book Pro with 2 cores and 4GB of RAM.[2] These results indicate that the more complex trackers may have had issues keeping up with the 30 frames per second of real-time video. IVT was run in Octave and processed four frames per second, MIL Track was benchmarked at 12 frames per second, and OAB was benchmarked at 36 frames per second. While it may be possible to speed up IVT by converting it to C or MIL Track by using multiple cores, it would seem that keeping up with the 30fps that is delivered by most modern cameras could be a problem.

Because correlations can be computed so efficiently, frame rates are not an issue for

[2]MOSSE was run in Python 2.6 compiled using gcc 4.2.1 and using fftpack library for FFT computations. IVT was run using a standard Octave installation from Mac Ports. MIL Track and OAB were run using Windows 7 under parallels and were compiled using the Microsoft Visual Studio 2010 and Intel IPP ver. 7.0.

the correlation base trackers. In these tests the correlation trackers benchmark at about 400 frames per second for the Python based implementation. Results in [7] indicate that converting select parts of this algorithm to C also yield a further 2.5 times speed up. Not only is this fast enough to keep up with frame rates in real-time videos, but it also provides plenty of head room for additional processing.

6.3.4 Hypothesis Testing

One of the final goals of this evaluation is to provide a data set of sufficient size to allow for statistical hypothesis testing. This data set provides 72 videos, and the evaluation has been structured such that each video provides a single data point that can be used to conduct a pairwise test between algorithms. For example, given two algorithms (A and B) each video can be categorized as follows: A tracked longer than B, A and B completed the same number of frames, or B tracked longer than A. Given counts for each of these outcomes, it is possible to use a binomial or sign test to determine the equivalence of the two algorithms.

Much like in McNemar's test or the sign test, outcomes where the two algorithms fail at the same point in the videos provides no information to distinguish one algorithm from another. For this reason, cases where A and B fail within 5 frames of each other are eliminated from the test. That leaves two possible outcomes (A $>$ B or B $<$ A) where one algorithm has outperformed the other. The counts for these cases will follow a binomial distribution, and if the algorithms are equivalent, the probability of one versus the other should be 0.5. Binomial statistics can then be used to compute a P-value and determine whether or not to reject the null hypothesis.

For these experiments we are interested in two particular tests:

- Test 1: Does the MOSSE filter show an improvement over simple templates and therefore advance the state-of-the-art in correlation filter tracking?
- Test 2: Does the MOSSE filter show an improvement over MIL Track that represents the current state-of-the-art for visual tracking?

Test 1: MOSSE vs. Template		Test 2: MOSSE vs. MIL Track	
MOSSE > Template	22	MOSSE > MIL Track	28
MOSSE < Template	1	MOSSE < MIL Track	6
P-Value (Two-Sided)	5.722e-06	P-Value (Two-Sided)	0.0002
Null Hypothesis	Rejected	Null Hypothesis	Rejected

Table 6.1: Statistical Hypothesis Tests

The results of the exact binomial tests are shown in Table 6.1. In both cases, the null hypotheses were rejected with P-values indicating that the differences in performance were highly significant. Further analysis of the 6 cases where MIL Track out performed MOSSE showed that 5 of 6 were on face tracking scenarios, and 3 of those videos came from demo videos from the MIL Track website. Likewise 18 of the 28 successes for MOSSE were from vehicle tracking scenarios on which MOSSE is known to perform particularly well. This is evidence that the trackers could be specializing in particular subdomains of the tracking problem. More investigation is needed.

6.4 Chapter Summary

This chapter has investigated the application of Optimized Correlation Output Filters (OCOF), in particular MOSSE, to the problem of visually tracking objects in video. It has described the current state of tracking research and how correlation-based tracking compares to other approaches. The MOSSE tracking algorithm was described in detail including the use of the Peak-to-Sidelobe Ratio to detect and handle occlusion. Finally, it discussed a new evaluation methodology and data set for visual trackers that showed that the MOSSE tracker out performed three state-of-the-art trackers in both accuracy and speed.

Chapter 7

Conclusions

In this research I have introduced a new way to train correlation filters called Optimized Correlation Output Filters (OCOF), which include Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE) (Chapter 3). These new techniques have a number of advantages over existing filters. The protocol for training is more flexible than for prior techniques. Other methods such as Minimum Average Correlation Energy (MACE) and Unconstrained Minimum Average Correlation Energy (UMACE) use cropped and center templates of the target during training. OCOFs train on entire images that may contain multiple targets, and those targets no longer need to be centered in the image. This is accomplished by providing OCOF training methods with a pair of training images. One is the training input that should be representative of an image used for testing. The other image is the desired correlation output that contains peaks where the targets are located in the associated training image. OCOF techniques such as MOSSE can then produce filters that optimally map those inputs to the corresponding outputs.

This technique represents a fundamental departure in the way correlation filters are trained. By providing an OCOF with larger training images that include both targets and background, the filters can exploit more elements in those images than are available in small cropped and centered templates. This allows the filters to learn more about the context in which the targets are located and also how to better discriminate between the targets and the background. It was demonstrated in Section 5.3 that much of the power

of techniques like MOSSE comes from exploiting these larger training images.

7.1 Summary of Experiments

This research has looked at three application domains for OCOF filters: eye localization, person detection, and visual tracking. Each of these application areas have different challenges, but they are all different aspects of the larger object detection problem. In EYE LOCALIZATION (Chapter 4) it is assumed that an eye is in the image and the problem is finding the location of that eye with high precision. In PERSON DETECTION (Chapter 5) the challenge is detecting people whose appearance may change dramatically due to clothing or pose while being able to reject other objects in the background. In VISUAL TRACKING (Chapter 6) the challenge is efficiently following the target through video sequences while being able to adapt to many appearance changes such as rotation, scale, deformation, occlusion, etc.

For each of these experiments OCOFs were compared to other filter training techniques, and also to state-of-the-art algorithms in those particular application areas. In all cases, OCOF filters had higher accuracy than the other filter training techniques, and they also matched or exceeded the accuracy of the state-of-the-art algorithms. The advantage of OCOF filters also comes from their speed. In most cases they run many times faster than the more complex state-of-the-art algorithms including the Viola and Jones Cascade Detector. By using an OCOF, it is possible to get state-of-the-art accuracy with much improved run times. Therefore, these new filters should open new possibilities in time critical application areas such as real-time video analysis.

7.2 Future Work

There are a few avenues of research that are open and worth pursuing. The first is the Cost Function Minimizing Filters. In Section 3.5 a method was defined for optimizing nonlinear cost functions. However, this technique was never evaluated in any of the experiments in this dissertation. The reason is the cost function technique is much slower

to train, and we have not yet found a cost function or application area for which that filter outperforms ASEF or MOSSE. Nonetheless, the ability to specify the cost functions provides increased flexibility for the filter and in some configurations may result in better performance.

Prior correlation techniques such as MACE and its extensions have found most of their applications in target identification instead of target detection. One prominent example of this is biometrics where correlation filters have been used to produce some of the best face recognition and iris recognition algorithms. I have conducted a few simple tests in object identification that show promising results, and this line of research is worth pursuing. However, OCOF based filters will probably not show the same improvement in identification as in detection.

Other research that is worth pursuing is the design of filters that are sensitive to color or can be applied to multi-channel data. Color provides more information about an object's appearance and can be used to easily solve many object detection tests. Therefore, correlation filters that include color may offer a large improvement in detection performance. The most promising techniques for combining multi-channel data are polynomial filters or Clifford correlation. It is possible that these techniques could also extend OCOF to nonlinear filtering.

One ability of OCOF that goes beyond what is possible with MACE or UMACE is the ability to specify output images where the correlation surface is more complicated than simply the superposition of one or a few peaks. For example, the filters can be trained to respond to edges. This may be extremely valuable for problems like image segmentation where the filters can be trained to respond to certain edges in an image while ignoring others. Improvements in techniques to segment images may have a big impact in areas such as medical imaging.

REFERENCES

[1] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[2] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 73–82. IEEE Computer Society, 2009.

[3] B. Babenko, Ming-Hsuan Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *CVPR*, 2009.

[4] Boris Babenko. Tracking with online multiple instance learning (miltrack). WWW Page: http://vision.ucsd.edu/ bbabenko/project_miltrack.shtml, 2010.

[5] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10):2385–2404, 2000.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006.

[7] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition*, San Francisco, California, June 2010.

[8] David S. Bolme, Bruce A. Draper, and J. Ross Beveridge. Average of synthetic exact filters. In *Computer Vision and Pattern Recoginition*, Miami Beach, Florida, June 2009.

[9] David S. Bolme, Yui Man Lui, Bruce A. Draper, and J. Ross Beveridge. Simple real-time human detection using a single correlation filter. In *Winter Workshop on Performance Evaluation of Tracking and Surveillance*, December 2009.

[10] Kevin Bowyer and P. Jonathon Phillips. *Emperical Evaluation Techniques in Computer Vision*, chapter 1, pages 1–11. IEEE Computer Society Press, 1998.

[11] G.R. Bradski. Computer vision face tracking as a component of a perceptual user interface. In *Workshop on Applications of Computer Vision*, pages 214–219, October 1998.

[12] G.R. Bradski. Computer vision face tracking for use in a perceptual user interface. Technical report, Microcomputer Research Lab, Intel Corporation, Santa Clara, CA,, 1998.

[13] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[14] MG Bulmer. *Francis Galton: pioneer of heredity and biometry*. Johns Hopkins Univ Pr, 2003.

[15] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[16] M. Castrillon Santana, J. Lorenzo Navarro, O. Déniz Suárez, and A. Falcón Martel. Multiple face detection at different resolutions for perceptual user interfaces. In *2nd Iberian Conference on Pattern Recognition and Image Analysis*, Estoril, Portugal, June 2005.

[17] R. Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recoginition*, 2003.

[18] R.T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1631–1643, 2005.

[19] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[20] F.C. Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH Computer Graphics*, 18(3):212, 1984.

[21] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1337–1342, 2003.

[22] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, 2005.

[23] Steve Eddins. Steve on image processing: Separable convolution. Online: http://blogs.mathworks.com/steve/2006/10/04/separable-convolution/, 2006.

[24] A. Ellis, A. Shahrokni, and J. Ferryman. Overall evaluation of the pets2009 results. In *PETS Workshop*, 2009.

[25] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *Computer Vision and Pattern Recognition, Anchorage, USA, June*, 2008.

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

[27] L.M. Fuentes and S.A. Velastin. Foreground segmentation using luminance contrast. In *Proceedings of the WSES/IEEE Conference on Signal XXXX and Image Processing*, volume 240, pages 241–243, 2001.

[28] Valerie Gouaillier and Aude-Emmanuelle Fleurant. Intelligent video surveillance: Promises and challenges. Technical report, Centre de recherche informatique de Montréal and Technopôle Defence and Security, April 2009.

[29] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. BMVC*, volume 1, pages 47–56. Citeseer, 2006.

[30] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. *Computer Vision–ECCV 2008*, pages 234–247, 2008.

[31] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, S. Pankanti, A. Senior, C.F. Shu, et al. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine*, 22(2):38–51, 2005.

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

[33] A.E. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, pages 80–86, 2000.

[34] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2599–2606, 2009.

[35] M. Isard and A. Blake. Condensation—conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[36] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010.

[37] K.H. Jeong, S. Han, and J.C. Principe. The fast correntropy MACE filter. In *Proceedings of the International Conference on Acoustics, Speech, Signal Processing (ICASSP)*, 2007.

[38] K.H. Jeong, P.P. Pokharel, J.W. Xu, S. Han, and J.C. Principe. Kernel based synthetic discriminant function for object recognition. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5. IEEE, 2006.

[39] Kyu-Hwa Jeong, Weifeng Liu, Seungju Han, Erion Hasanbelliu, and Jose C. Principe. The correntropy mace filter. *Pattern Recogn.*, 42:871–885, May 2009.

[40] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, (82):35–45, 1960.

[41] K. Kim, T.H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-time imaging*, 11(3):172–185, 2005.

[42] K. Kim, D. Harwood, and L. Davis. Background updating for visual surveillance. *Advances in Visual Computing*, pages 337–346, 2005.

[43] B. Kumar, M. Savvides, and C. Xie. Correlation pattern recognition for face recognition. *Proceedings of the IEEE*, 94(11):1963–1976, 2006.

[44] B.V.K. Kumar. Minimum-variance synthetic discriminant functions. *Journal of the Optical Society of America. A, Optics and image science*, 3(10):1579–1584, 1986.

[45] B.V.K. Kumar, A. Mahalanobis, S. Song, S.R.F. Sims, and J.F. Epperson. Minimum squared error synthetic discriminant functions. *Optical Engineering*, 31:915, 1992.

[46] C.H. Lampert, M.B. Blaschko, T. Hofmann, and S. Zurich. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *Proc. of CVPR*, 2008.

[47] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. *Pattern Recognition: 25th Dagm Symposium, Magdeburg, Germany, September 10-12, 2003: Proceedings*, 2003.

[48] J. Lim, D. Ross, R.S. Lin, and M.H. Yang. Incremental learning for visual tracking. *Advances in neural information processing systems*, 17:793–800, 2005.

[49] David G. Lowe. Object recognition from local scale-invariant features. *iccv*, 02:1150, 1999.

[50] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, page 3. Citeseer, 1981.

[51] A. Mahalanobis and B. V. K. Vijaya Kuma. Polynomial filters for higher-order correlation and multi-input information fusion. In *Optoelectronic Information Processing Workshop,*, pages 221–231, 1997.

[52] A. Mahalanobis, B. V. K. Vijaya Kumar, and S. R. F. Sims. Distance-classifier correlation filters for multiclass target recognition. *Appl. Opt.*, 35(17):3127–3133, 1996.

[53] A. Mahalanobis, BVK Vijaya Kumar, S. Song, SRF Sims, and JF Epperson. Unconstrained correlation filters. *Applied Optics*, 33(17):3751–3759, 1994.

[54] Abhijit Mahalanobis, B. V. K. Vijaya Kumar, and David Casasent. Minimum average correlation energy filters. *Appl. Opt.*, 26(17):3633, 1987.

[55] NJB McFarlane and CP Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3):187–193, 1995.

[56] David G. Messerschmitt. Stationary points of a real-valued function of a complex variable. Technical Report UCB/EECS-2006-93, EECS Department, University of California, Berkeley, Jun 2006.

[57] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and KR Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[58] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. *Computer Vision–ECCV 2006*, pages 490–503, 2006.

[59] E. Osuna, R. Freund, F. Girosi, et al. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[60] P.J. Phillips, H.J. Moon, S.A. Rizvi, and P.J. Rauss. The FERET Evaluation Methodology for Face-Recognition Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, October 2000.

[61] A. Prati, I. Mikic, M.M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):918–923, 2003.

[62] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C - 13.1 Convolution and Deconvolution Using the FFT*, pages 538–545. Cambridge University Press, Cambridge, 2 edition, 1988.

[63] P. Refregier. Optimal trade-off filters for noise robustness, sharpness of the correlation peak, and Horner efficiency. *Optics Letters*, 16:829–832, June 1991.

[64] M.D. Rodriguez, J. Ahmed, and M. Shah. Action MACH a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[65] D.A. Ross, J. Lim, R.S. Lin, and M.H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008.

[66] David Ross. Incremental learning for visual tracking. WWW Page: http://www.cs.toronto.edu/ dross/ivt/, 2009.

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

[68] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[69] M. Savvides, B. Kumar, and PK Khosla. Corefaces-robust shift invariant pca based correlation filter for illumination tolerant face recognition. In *Computer Vision and Pattern Recognition*, volume 2, 2004.

[70] M. Savvides, B.V.K.V. Kumar, and P. Khosla. Face verification using correlation filters. 2002.

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

[72] H. Schneiderman and T. Kanade. A Statistical Method for 3D Object Detection Applied to Faces and Cars. In *IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, volume 1. IEEE Computer Society; 1999, 2000.

[73] B. Scholkopf, A. Smola, and K.R. Muller. Kernel principal component analysis. *Artificial Neural Networks—ICANN'97*, pages 583–588, 1997.

[74] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Ninth IEEE international conference on computer vision, 2003. Proceedings*, pages 1470–1477, 2003.

[75] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[76] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large Scale Multiple Kernel Learning. *The Journal of Machine Learning Research*, 7:1531–1565, 2006.

[77] JAK Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[78] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *CVPR*, pages 511–518, 2001.

[79] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63(2):153–161, 2005.

[80] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[81] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008. ISMAR 2008*, pages 125–134, 2008.

[82] Peng Wang, Matthew B. Green, Qiang Ji, and James Wayman. Automatic eye detection and its validation. In *Computer Vision and Pattern Recognition - Workshops*, 2005.

[83] G. Welch and G. Bishop. An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.

[84] Willow Garage. Opencv libarary, April 2009. `http://opencv.willowgarage.com`.

[85] Laurenz Wiskott, Jean-Marc Fellous, Norbert Kruger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. *Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.

[86] Chunyan Xie, M. Savvides, and B.V.K.V. Kumar. Redundant class-dependence feature analysis based on correlation filters using frgc2.0 data. *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*, 3:153–153, June 2005.