# Maximum Margin Correlation Filters

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Andres F. Rodriguez-Perez

B.S., Electrical Engineering, Brigham Young University

M.S., Electrical Engineering, Brigham Young University

Carnegie Mellon University

Pittsburgh, Pennsylvania

August, 2012

# ABSTRACT

Support vector machines (SVMs) and correlation filters (CFs) are popular for automatic target recognition (ATR) and other computer vision tasks. SVMs are designed to maximize the separation between two classes in some feature space. SVMs are popular for *classification* (determining the class-label of a target) and generalize well for targets not in the training set, but SVMs are not specifically designed for *localization* (finding where a target is). When using SVMs, regions of interest (ROIs) are usually first extracted using some other detector before SVMs are applied. CFs accurately localize the target of interest in a large scene but their classification performance may not be as good (compared to SVMs) for targets not in the training set. In this thesis we introduced new linear CFs by combining the criteria used in state-of-the-art CFs to improve performance. Using our improved linear CF designs, we present a new type of classifier called the Maximum Margin Correlation Filters (MMCFs), which combine the generalization capabilities of SVMs and the shift-invariance of correlation filters (CFs), i.e., MMCF is also invariant to shifts between the training images and the query image, thereby avoiding the need for image registration and detection before SVM-based classification. We extend our work to quadratic correlation filters (QCFs) and present the Quadratic MMCF (QMMCF) and show its relation to the second order polynomial Kernel SVM (referred to as Quadratic SVM (QSVM) in this thesis). We extend the capabilities of CFs and therefore of MMCFs and QMMCFs to include vector features (as opposed to scalar features, i.e., gray-scaled pixels) in order to use features such as the Histogram of Oriented Gradients (HOG). We test the efficacy of the our designs on real data and show improvement over linear and quadratic CFs and linear and quadratic SVMs.

## Acknowledgments

I would like to thank my advisor, Vijayakumar Bhagavatula, for his continual support, example, and willingness to advise. Even though he had many other responsibilities and students to advise, he was fully committed to my research. I would like to thank the other members of my committee Marios Savvides, Abhijit Mahalanobis and Kevin Priddy for taking the time to make my thesis better. I would like to thank the Air Force Research Laboratory for providing the main funding for this work. I would like to thank my friends Naresh Boddetti, Ramu Bhagavatula, John Smereka, and Kathy Brigham, for the many hours they spent discussing the topics presented in this work. I would like to thank Clark Taylor for teaching me how to do research and preparing me to start my PhD. I would like to thank my mother and father for their example of diligence and excellence. I would like to thank my sons David and Tomas for brightening my days and for their patience on the many days when their dad could not play with them and had to work long nights. Finally, I would like to thank my best friend and sweetheart Mary-Kathryn for her tremendous support, inspiration, and the love she has always given me.

# Table Of Contents

# List Of Figures

# LIST OF TABLES

# List Of Abbreviations

| | |
|---|---|
| ACE | average correlation energy |
| ASEF | Average of Synthetic Exact Filter |
| ASM | average similarity measure |
| ATR | automatic target recognition |
| ATR-ADID | ATR Algorithm Development Image Database |
| CCF | generalized Constrained Correlation Filter |
| CF | correlation filter |
| CHF | circular harmonic function |
| CPCF | Constrained PCF |
| DCCF | Distance Classifier Correlation Filter |
| DFT | discrete Fourier transform |
| EASEF | Extended-ASEF |
| ECPSDF | Equal Correlation Peak Synthetic Discriminant Function |
| EEMACH | Eigen-EMACH |

| | |
|---|---|
| EMACH | Extended MACH |
| EMOSSE | Extended-MOSSE |
| EOTSDF | Extended-OTSDF |
| FIR | finite impulse response |
| FKQCF | Fukunaga-Koontz QCF |
| FOM | figure of merit |
| FR | Fisher ratio |
| GMACH | Generalized MACH |
| HOG | Histogram of Oriented Gradients |
| IDFT | inverse DFT |
| iMACE | inequality MACE |
| iOTSDF | inequality OTSDF |
| KF | Kalman filter |
| LDA | linear discriminant analysis |
| MACE | Minimum Average Correlation Energy |
| MACE-MRH | MACE Mellin Radial Harmonic |
| MACH | Maximum Average Correlation Height |
| mCCF | modified Constrained Correlation Filter |
| MF | Matched filter |
| MFCF | Multi-Frame Correlation Filter |

| | |
|---|---|
| MICE | Minimum Correlation Energy |
| MINACE | Minimum Noise and Correlation Energy |
| MMCF | Maximum Margin Correlation Filter |
| MMSE | minimum mean squared error |
| MOSSE | Minimum Output Sum of Squared Error |
| MRH | Mellin radial harmonics |
| MRHF | Mellin Radial Harmonic Function |
| MSE | mean squared error |
| MSESDF | Minimum Squared Error Synthetic Discriminant Function |
| MVSDF | Minimum Variance Synthetic Discriminant Function |
| ONV | output noise variance |
| OTCHF | Optimal Tradeoff CHF |
| OTMACH | Optimal Tradeoff MACH |
| OTSDF | Optimal Tradeoff Synthetic Discriminant Function |
| PCE | peak-to-correlation energy |
| PCF | Polynomial Correlation Filter |
| PDCCF | Polynomial DCCF |
| PSD | power spectral density |
| PSR | peak-to-sidelobe ratio |
| QCF | Quadratic Correlation Filter |

| | |
|---|---|
| QMACE | Quadratic MACE |
| QMMCF | Quadratic Maximum Margin Correlation Filter |
| QMMCF | Quadratic Maximum Margin Correlation Filter |
| QSVM | Quadratic SVM |
| ROI | region of interest |
| RQ | Rayleigh quotient |
| RQQCF | Rayleigh Quotient QCF |
| SAR | synthetic aperture radar |
| SDF | synthetic discriminant function |
| SPREF | Spatio-temporal Regularity Flow |
| SSQSDF | Subspace Quadratic Synthetic Discriminant Function |
| SVM | support vector machine |
| TQCF | Transformed QCF |
| UCF | generalized Unconstrained Correlation Filter |
| UMACE | Unconstrained MACE |
| UMSESDF | Unconstrained MSESDF |
| UOTSDF | Unconstrained OTSDF |

CHAPTER 1

# INTRODUCTION

Automatic target recognition (ATR) is important in numerous applications including reconnaissance systems, smart weapons, and unmanned vehicles. ATR is composed of correct *localization* (finding where a target is) and correct *classification* (determining the class-label of a target). Thus, correct *recognition* implies correct localization *and* correct classification. ATR is a difficult problem because targets from the same class can vary greatly in appearance due to variations in illumination, viewpoints, and non-rigid deformations. The problem may be further compounded when computational capacity is limited.

Two different types of algorithms that are used in ATR applications are support vector machines (SVMs) and correlation filters (CFs). SVMs are designed to maximize the separation between two classes in some feature space. SVMs are popular for classification and generalize well for targets not in the training set, but SVMs are not specifically designed for localization. When using SVMs, regions of interest (ROIs) are usually first extracted using some other detector (e.g., face detector when doing face recognition) before SVMs are applied. CFs accurately localize the target of interest in a large scene but their classification performance may not be as good (compared to SVMs) for targets not in the training set.

In this introductory chapter we provide a high-level overview of CFs and SVMs. We summarize the contributions of this thesis, which include explaining the sometimes conflicting relationship between localization and generalization, and introducing new linear and quadratic classifiers that bridge CFs and SVMs to achieve improved localization and classification. We also describe the datasets we used in our experiments, we introduce the notation that is used throughout the thesis, and finally we present the organization of the rest of this thesis.

## 1.1 Background

### 1.1.1 Correlation filters

We offer a high-level overview of CFs in this section and give more details in Chapter 2. CFs have been investigated for object recognition (note that in this thesis, we use the terms *object recognition*, *target recognition*, and *ATR* interchangeably). Attractive properties of CFs such as shift-invariance, noise robustness, graceful degradation, and distortion tolerance can be useful in a variety of pattern recognition applications including face localization [12], pedestrian localization [13], object localization and tracking [11], biometric recognition [64, 67, 79], and vehicle recognition [60]. In this approach, a carefully designed template (called a *filter* in the frequency domain) $h(m, n)$ is cross-correlated with the query image $x(m, n)$ to produce the output $g(m, n)$. This operation is efficiently carried out in the frequency domain,

$$G(u, v) = X(u, v)H^*(u, v), \tag{1.1}$$

where $*$ denotes the conjugate, and $G(u, v)$, $X(u, v)$ and $H(u, v)$ are the 2-D discrete Fourier transforms (DFTs) of the correlation output, the query image, and the template, respectively. When the query image is an authentic match (also called true-class or Class-1) $g(m, n)$ should exhibit a sharp peak at the center of the target's location, and when the query image is an impostor (also called false-class or Class-2) $g(m, n)$ should not have any significant peak. The higher the peak the higher the probability that the query image is from the authentic class, and the location of the peak indicates the location of the object. Thus CFs offer the ability to simultaneously localize and classify objects. Fig. 1.1 shows the desired correlation output to an input image correlated with the CF template.

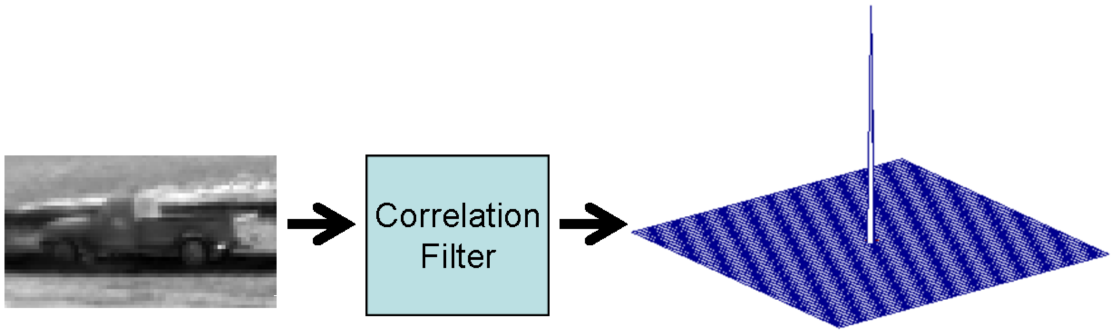


Figure 1.1: The desired output (right) to an input image (left) correlated with the CF template. CFs can be designed to output a sharp peak when the input is the desired target.

### 1.1.2 Support vector machines

We offer a high-level overview of SVMs in this section and give more details in Section 4.1.1. SVMs have been used in vision tasks such as face localization [51] and pedestrian localization [21]. Given $N$ training vectors $\mathbf{x}_i \in \mathbb{R}^d$ (the elements of these vectors can be the raw pixel values or features computed from the selected region) and class labels $l_i \in \{-1, 1\}\ \forall i \in \{1, \cdots, N\}$, the SVM approach finds the hyperplane that maximizes the *soft* margin (i.e., L-2 norm distance) between two classes by solving

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \frac{1}{2}\mathbf{h}^T\mathbf{h} + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & l_i(\mathbf{h}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \end{aligned} \tag{1.2}$$

where superscript $T$ denotes the transpose, $\mathbf{h}$ and $b$ represent the hyperplane ($\mathbf{h}$ is orthogonal to the hyperplane and $b$ is the bias), $C > 0$ is a tradeoff parameter, and $\xi_i \geq 0$ is a penalty term corresponding to misclassified training samples. The term *soft* margin refers to the inclusion of this penalty term that allows some training vectors to be misclassified to account for outliers. When the penalty terms $\xi_i = 0$ for all $i$, the margin is referred to as a *hard* margin, and all training vectors should be correctly classified, i.e., $l_i(\mathbf{h}^T\mathbf{x}_i + b) \geq 1$ for all $i$. Correctly classifying all training vectors is possible only for linearly separable data. If the data is not linearly separable, penalty terms are required. The solution to Eq. 1.2 is a linear combination of the training vectors, i.e.,

$$\mathbf{h} = \sum_{i=1}^{N} a_i \mathbf{x}_i, \tag{1.3}$$

where the coefficients $a_i$ are non-zero only for the support vectors.

Assuming that the features used are pixels values (i.e., images with $d$ pixels lexicographically rearranged into column vectors), one can use the resulting solution $\mathbf{h}$ for simultaneous object localization and classification by cross-correlating $\mathbf{h}$ (or more correctly the 2-D template represented by it) with the query image. However, since the template is not optimized to produce sharp correlation peaks, the resulting correlation plane (i.e., the 2-D correlation output) would exhibit very broad peaks which may result in poor object localization.

There are some previous approaches that are aimed at achieving shift-invariant classification. Most of these approaches cross-correlate the template with the query image and are sometimes known as sliding window algorithms. Scholkopf et al. [70] proposed a method to achieve shift-invariance by training an SVM on centered images, generating shifted images of the support vectors and re-training the SVM. Decoste et al. [22] described different algorithms for training shift-invariant SVMs, and Chapelle et al. [18] proposed algorithms to incorporate shift-invariance in non-linear SVMs. These methods include shift-invariance constraints explicitly as inequalities, i.e.,

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\sum_{j=1}^{d}\xi_i^j \\ s.t. \quad & l_i(\mathbf{w}^T\mathbf{x}_i^j + b) \geq 1 - \xi_i^j, \end{aligned} \tag{1.4}$$

where $\mathbf{x}_i^j$ is the image $\mathbf{x}_i$ shifted by $j$ pixels. For images, $j$ refers to shifts in both the $x$- and $y$-directions. In these methods, the number of constraints gets multiplied by the number of shifts making the complexity of these methods prohibitive for large number of shifts. In fact, the approaches in [18, 22, 70] make the classifier invariant to just 1 or 2 pixel shifts in the images, and hence precise object localization is still very challenging. Thornton et al. [78] proposed what they called SVM Correlation Filter. They simply treat shifted versions of the true-class training images as virtual false-class training images, which does not scale well with the number of training images, i.e., the number of training images gets multiplied by the number of shifts making the complexity of these methods prohibitive for large number of shifts. Moreover, they do not deal with actual false-class training images as well as their shifted versions which, if included, could further increase the complexity of the problem making the optimization problem intractable.

## 1.2 HIGH-LEVEL APPROACH AND CONTRIBUTIONS

In this thesis we investigate the relationship between localization and classification. We introduce a general form of CFs that includes the various criteria normally used, and describe how to achieve an optimal tradeoff among these criteria. We combine the design principles of SVMs and our generalized CFs which results in a new type of classifier called the Maximum Margin Correlation Filter (MMCF). The MMCF classifier is less susceptible to over-fitting than traditional CFs while providing geometric shift-invariance to SVM classifiers. That is, MMCF not only correctly classifies the query but also determines the relative shift between the query and the training samples efficiently. MMCF design leads to a more distinguishable peak in the correlation plane. Sharper correlation peaks lead to better object localization because the values in the vicinity of a sharp peak will be much smaller than the peak making the peak location accurate. We extend the linear MMCF to quadratic MMCF, and introduce different features that we can used in all these algorithms.

The main contributions of this thesis are as follows:

- Improved existing CFs. We present two generalized linear CFs that encompass all the state-of-the-art linear CFs. We present a method to enhanced the performance of quadratic CFs both in computational speed and performance, and show experimental results. We present a design that improves recognition performance when using CFs in a sequence of images, i.e., in video, by using the correlation output with a tracker. We provide experimental results showing that our algorithms are more efficient and yield improved performance on our dataset.

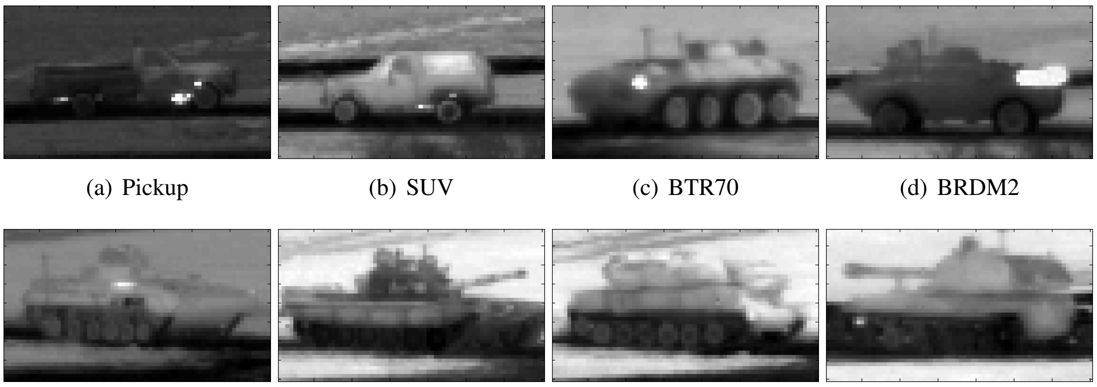
- Designed a new algorithm that combines the capabilities of SVMs and our generalized CFs to yield the Maximum Margin Correlation Filter (MMCF). We investigate the relationship between localization and classification, and show that the criteria used to achieved localization and generalization are conflicting, i.e., one is optimized at the expense of the other, and we show how to achieve an optimal tradeoff between these criteria. The MMCF classifier is less susceptible to over-fitting than traditional CFs while providing geometric shift-invariance to SVM classifiers. We show that MMCF is an extension to both SVM and CFs by deriving MMCF first starting with an SVM and then with a CF, thus we bridge these two important research fields. We provide experimental results showing that MMCF usually outperforms linear CFs and always outperforms SVMs on our dataset.

- Designed the Quadratic MMCF (QMMCF) by extending the maximum margin principles to quadratic CFs (QCFs). QMMCF is better able to exploit the higher-order statistics of the data resulting in superior performance. This improved performance comes at the cost of added computation during the testing. We show the relationship between QCFs and Quadratic SVMs (QSVMs) (also known as second order Kernel SVMs). We provide experimental results showing that QMMCF outperforms QCFs and QSVMs on our dataset, as well as MMCFs.

- Extended the capabilities of CFs and therefore of MMCF to include vector features. CFs usually used scalar features, i.e., gray-scaled pixels. We adapt our algorithms to include non-scalar features, e.g., features such as Histogram of Oriented Gradients (HOG) which have recently gained much popularity. We provide experimental results showing that MMCF and QMMCF maintain superior performance over other state-of-the-art linear and quadratic algorithms, respectively, when using HOG features on our dataset.

## 1.3 TEST DATASETS

We provide experimental results using real videos. In this section, we present a brief overview of our dataset and leave the details to Chapter 7. We consider target recognition on a set of infrared videos where the vehicle's class-label and location are unknown. We use the recently approved for


(a) Pickup (b) SUV (c) BTR70 (d) BRDM2

(e) BMP2 (f) T72 (g) ZSU23-4 (h) 2S3

Figure 1.2: Example of the different classes of targets.

public release *ATR Algorithm Development Image Database* (ATR-ADID) [17] produced by the Military Sensing Information Analysis Center. The database contains infrared videos of 512×640 pixels/frame from eight military vehicles (one vehicle in each video), shown in Fig. 1.2, taken at multiple ranges during day and night time at 30 Hz. The vehicles were driven at about 10 mph making a circle of diameter of about 100 meters, therefore exhibiting full 360° of azimuth rotation. Each video is 60 seconds long, (i.e., 1800 frames) allowing the vehicle to complete at least one full circle. We used videos from each vehicle collected during day time at a range of 1000 meters and compared our results to the ground truth data provided in the database.

## 1.4 Notation

Throughout this thesis, we use the following notational conventions:

- lower case non bold letters, e.g., $x_i(m,n)$, denote spatial domain 2-D arrays (also called planes or images)
    - we loosely use $x_i(m,n)$ to represent both image $i$ and the $(m,n)$ pixel value of image $i$
- upper case non bold letters, e.g., $X_i(u,v)$, denote frequency domain 2-D arrays
- lower case bold letters denote vectors (lexicographically arranged) of 2-D frequency domain

arrays, e.g, $\mathbf{x}_i$ denotes $X_i(u, v)$

  - we loosely refer to $\mathbf{x}_i$ as the $i$th image when we mean the vectorized 2-D DFT representation of the $i$th image
  - we loosely refer to $\mathbf{g}$ as the correlation plane when we mean the vectorized 2-D DFT representation of the correlation plane
  - we loosely refer to $\mathbf{h}$ as the template (or filter) when we mean the vectorized 2-D DFT representation of the template

- lower case bold letters with an inverted hat denote vectors of 2-D spatial domain arrays, e.g, $\check{\mathbf{x}}_i$ denotes $x_i(m, n)$
- upper case bold letters denote matrices, e.g., $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$
  - some matrices are diagonal matrices representations of vectors, e.g., $\mathbf{X}_i = \text{diag}(\mathbf{x}_i)$
- the $\otimes$ symbol denotes the 2-D correlation operator of either two 2-D arrays, e.g., $g_i(m, n) = x_i(m, n) \otimes h(m, n)$, or of the implied 2-D arrays represented by their vector versions, e.g., $\mathbf{g} = \mathbf{x}_i \otimes \mathbf{h}$
- the $\odot$ symbol denotes the Hadamard product, e.g., $\mathbf{x}_i \odot \mathbf{h}$
- the $\oslash$ symbol denotes the Hadamard divide, e.g., $\mathbf{g}_i \oslash \mathbf{x}_i$
- the overbar symbol denotes the mean of a set, e.g., $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$
- the overdot symbol denotes the desired output, e.g., $\dot{\mathbf{g}}_i$ denotes the desired $\mathbf{g}_i$
- the superscript $T$ symbol denotes the transpose, e.g., $\mathbf{h}^T$
- the superscript asterisk symbol denotes the conjugate, e.g., $\mathbf{X}_i^*$
- the superscript dagger symbol denotes the conjugate transpose, e.g., $\mathbf{h}^\dagger$
- the partial derivative of a scalar with respect to a vector denotes the gradient, e.g., $\frac{\partial \mathcal{L}}{\partial \mathbf{h}}$

## 1.5 Organization

The rest of this thesis is organized as follows. Chapter 2 provides an extensive review of CF theory and applications. Chapter 3 contains our contributions to CFs; we introduce new CFs and offer improvements to some existing CFs. Chapter 4 introduces the MMCF. We show the relationship between SVMs and CFs and how MMCF is a generalization of both. Chapter 5 introduces the Quadratic MMCF as an extension to the linear MMCF, and we show how it relates to QSVMs. Chapter 6 shows how vector features, in particular HOG features, can be used in our model. Chapter 7 describes the datasets used in the experiments, the pre- and post-processing techniques applied, and the experimental results. The thesis concludes in Chapter 8 with a summary of the main findings and contributions of our work and some possible directions for future research.

Chapter 2

# Correlation filters

In this chapter we review advances in correlation filter (CF) design during the past three decades. In the CF approach, a carefully designed template (loosely called a *filter*) is cross-correlated with a query image to produce a correlation output also known as correlation plane. This output is then searched for the highest peak or some other relevant metric such as the peak-to-correlation energy (PCE) or peak-to-sidelobe ratio (PSR) (explained in Section 7.2.5) and compared to a threshold to determine whether the query image is from the true-class (also known as the authentic class) or from the false-class (also known as the impostor class). Among the many advantages that CFs offer is built-in shift invariance; that is, when an input image is translated by a certain amount, the correlation output is translated by that same amount.

The most basic CF is the Matched filter (MF). Although the MF is optimal for detecting a known image in the presence of additive white noise, its recognition performance decreases significantly when the image has small distortions (e.g., from rigid body motion, lighting conditions, background, etc). Thus, it requires one MF for every possible distortion, making their use costly and impractical.

In 1980, Hester and Casasent introduced the design of one CF from multiple training images called the Equal Correlation Peak Synthetic Discriminant Function (ECPSDF) filter [28]. This filter (and most CFs) depends on a set of training images that captures the expected distortions (in testing) and builds a template from these images. Although the ECPSDF filter recognition performance is usually inadequate for images not in the training set, it opened the way for future CFs. The Minimum Variance Synthetic Discriminant Function (MVSDF) filter [82] was designed to minimize the output noise variance in the correlation plane, but also suffers from poor recognition performance. The Minimum Average Correlation Energy (MACE) filter [41] was a significant advance in CF designs. This filter is designed to reduce the energy of the correlation output resulting in a sharp peak at the location of the target facilitating target recognition. The Optimal Tradeoff Synthetic Discriminant Function (OTSDF) filter [55] and the Minimum Noise and Correlation Energy (MINACE) filter [54] are extensions of the MACE filter to achieve robustness to additive noise. The Minimum Squared Error Synthetic Discriminant Function (MSESDF) filter [88] is a general form of the MACE filter that allows the user to specify the desired correlation output. Each of these filters were constrained to have a certain value for the inner product between each training image and the filter. This inner product is referred to as the value at the origin (for centered images) or loosely referred to as the correlation peak in the correlation plane. For example the correlation peak can be constrained to be $1$ for the true-class images and $0$ for false-class images so that in testing the filter response has a high value (near 1) for true-class images and a low value (near 0) for false-class images. Note that referring to this inner product as the correlation peak is a slight abuse in terminology. The correlation peak is known as the highest value in the correlation plane, and this inner product, in theory, is not necessarily the highest value. However, in practice the correlation peak is usually the highest value for training images.

Another advance in CF designs was removing the correlation peak constraints. Removing these peak constraints increases the solution space and improves the chances of finding a filter with better recognition performance. The first of these filters are the Maximum Average Correlation Height (MACH) filter [45] (recently a 3-D MACH filter [63] was used for action classification), the Unconstrained MACE (UMACE) filter [45], and the Unconstrained MSESDF (UMSESDF) filter [45]. More recent designs include the Average of Synthetic Exact Filter (ASEF) [12] and the Minimum Output Sum of Squared Error (MOSSE) filter [13]. These unconstrained filters are unconstrained forms of the mentioned constrained CFs (meaning that they have the same objective functions but do not have the hard correlation peak constraints) with the addition that MACH filter also minimizes a measure of the scatter of the correlation outputs. Another linear CF is the Distance Classifier Correlation Filter (DCCF) [43] which transforms the training images so that their classes become more compact and more separated from each other.

A different type of CF design is used when the filter requires invariance to in-plane rotations or invariance to scale changes. Circular Harmonic Function (CHF) filters are used for in-plane rotations [29, 92] and Mellin Radial Harmonic Function (MRHF) filters are used for scale changes [46, 76]. More recent CF designs that are built on these methods are the Optimal Tradeoff CHF (OTCHF) filters [89] and the MACE Mellin Radial Harmonic (MACE-MRH) filters [31].

Another type of CFs is non-linear CFs. Typically they exhibit superior recognition performance but require more computation. The Polynomial Correlation Filters (PCFs) [40] design uses a set of linear filters applied to point nonlinear versions of the input, and their outputs are added to produce a single output. The Quadratic Correlation Filter (QCF) [36] determines and uses a quadratic non-linearity to maximize the separation between two classes. The rest of this chapter gives more details on these CFs.

## 2.1 Matched filter and efficient application of correlation filters

The most basic CF is the Matched filter (MF) whose origins are in detecting signals in received radar returns. It is well known that the MF maximizes the Signal-to-Noise Ratio (SNR) for additive white noise [87]. In radar the MF output can be used to estimate the relative time shift between transmitted and received signal, and from that estimate the distance to the target.

In practice, MFs do not work well for target recognition. These filters perform poorly when a target is distorted. Therefore, too many MFs would be required to account for all the different distortions (e.g., from rigid body motion, illumination, and/or background changes). However, MFs serve as a good theoretical foundation to explain how linear CFs works and their shift-invariance property.

We describe how to apply a MF through an example. This same technique is used to apply many CFs. Suppose that we want to know if and where an image $h(m,n)$ of a given object (e.g., tank, helicopter, etc.) is found in the larger image $x(m,n)$. In this example the image $h(m,n)$ of the desired object is the CF template and $x(m,n)$ is the test image. In most other CFs, $h(m,n)$ is a template designed from multiple images. A naive technique is to use a shifting window approach, i.e., comparing $h(m,n)$ with every block (referred hereafter as *test chip*) within $x(m,n)$ that is of size equal to $h(m,n)$. For every test chip, we compute the sum of all the elements in the Hadamard (i.e., point-wise) product between $h(m,n)$ and the test chip. When this sum is above some pre-specified threshold that test chip is labeled as containing the target of interest. The shifting window operation, referred to as correlation is expressed as

$$\begin{aligned} g(m,n) &= \sum_{k,l} x(m+k,n+l)h(k,l) \\ &= x(m,n) \otimes h(m,n) \\ &= x(m,n) \star h(-m,-n), \end{aligned} \tag{2.1}$$

where $\otimes$ and $\star$ denotes the 2-D correlation and convolution operator, respectively, and $g(m,n)$ is called the correlation output. Note that correlation is equivalent to convolution with the spatial-reversed (i.e., flipped) template.

The correlation operator can be efficiently computed using DFTs as

$$g(m,n) = \mathcal{F}^{-1}\left\{\mathcal{F}\{x(m,n)\}\mathcal{F}^{*}\{h(m,n)\}\right\}, \tag{2.2}$$

where $\mathcal{F}\{\cdot\}$ and $\mathcal{F}^{-1}\{\cdot\}$ denote the 2-D DFT and inverse DFT (IDFT) operators, respectively, and the superscript $*$ symbol denotes the conjugate. Recall from DFT theory that $x$ and $h$ require zero-padding in order to avoid circular correlation.

### 2.1.1 Derivation

The goal is to find if $\mathbf{x}$ is a good match for the desired pattern $\mathbf{h}$. A simple method is to find the squared error $e$, i.e.,

$$\begin{aligned} e &= |\mathbf{h}-\mathbf{x}|^2 \\ &= (\mathbf{h}-\mathbf{x})^\dagger(\mathbf{h}-\mathbf{x}) \\ &= \mathbf{h}^\dagger\mathbf{h}+\mathbf{x}^\dagger\mathbf{x}-2\mathbf{h}^\dagger\mathbf{x}. \end{aligned} \tag{2.3}$$

Assuming that the energy of both $\mathbf{x}$ and $\mathbf{h}$ has been normalized, i.e., $\mathbf{h}^\dagger\mathbf{h} = \mathbf{x}^\dagger\mathbf{x} = 1$, then minimizing the error $e$ is equivalent to maximizing the correlation term $\mathbf{h}^\dagger\mathbf{x}$. This inner product is maximized when the vectors are in the same direction. Since the vectors are unit norm then $-1 \leq \mathbf{h}^\dagger\mathbf{x} \leq 1$ is maximum when $\mathbf{h}$ and $\mathbf{x}$ are parallel ($\mathbf{h}^\dagger\mathbf{x} = 1$), i.e., when they are the same.

## 2.2 Equal Correlation Peak Synthetic Discriminant Function (ECPSDF) filter

The ECPSDF filter was introduced in 1980 [28], and it is also known as the conventional-SDF. This filter (like all composite CFs) is designed using a set of training images that attempts to capture the distortions expected to show up in testing. The filter is designed to be a linear combination of training images where the combination weights are chosen so that the peak filter response to the training images is a pre-specified value (usually 1 for the true-class and 0 for the false-class). When the filter is applied to a test image (not in the training set) the correlation plane tends to have a higher value (close to the pre-specified value of 1) at the location of the object of interest.

In practice this filter is not used as it gives peaks with very large side lobes which makes recognition more challenging because the center of the peak is not as distinguishable, and it is easily confused with the peaks and side-lobes from false-class images. This is due to the filter not being designed to capture the high frequency components of the training images, and, thus, easily confuses similar targets.

### 2.2.1 Derivation

The constrained peak value is $\mathbf{h}^\dagger \mathbf{x}_i = u_i$, where $\mathbf{x}_i$ represents the $i$th image, $\mathbf{h}$ represents the filter, and $u_i$ is the pre-specified peak filter response (usually $u_i = 1 \ \forall i$). Another constraint in the ECPSDF filter design is that the template $\mathbf{h}$ is a linear combination of the training images, i.e., $\mathbf{h} = \sum_{i=1}^{N} a_i \mathbf{x}_i$. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, then the constraints are

$$\mathbf{X}^\dagger \mathbf{h} \quad = \quad \mathbf{u}, \tag{2.4}$$

and

$$\mathbf{h} = \mathbf{X}\mathbf{a}, \tag{2.5}$$

where $\mathbf{u} = [u_1, \ldots, u_N]^T$, and $\mathbf{a} = [a_1, \ldots, a_N]^T$. Substituting Eq. 2.5 into Eq. 2.4 gives

$$\mathbf{X}^\dagger \mathbf{X}\mathbf{a} = \mathbf{u}. \tag{2.6}$$

The weights can be computed from Eq. 2.6 as follows,

$$\mathbf{a} = (\mathbf{X}^\dagger \mathbf{X})^{-1}\mathbf{u}. \tag{2.7}$$

Substituting the weights in Eq. 2.7 into Eq. 2.5 gives the filter equation:

$$\mathbf{h} = \mathbf{X}(\mathbf{X}^\dagger \mathbf{X})^{-1}\mathbf{u}. \tag{2.8}$$

## 2.3 Minimum Variance Synthetic Discriminant Function (MVSDF) filter

The MVSDF filter was introduced in 1986 [82]. This filter is designed to minimize the correlation output noise variance (ONV) when the input training images are corrupted by additive zero-mean noise, while simultaneously satisfying the correlation output constraints, i.e., $\mathbf{h}^\dagger \mathbf{x}_i = u_i$.

In practice this filter is not commonly used for the same reasons as the ECPSDF filter. In addition, the filter requires the knowledge of the covariance of the additive noise. If the noise is assumed to be white noise then the MVSDF filter equals the ECPSDF filter.

### 2.3.1 Derivation

In the spatial domain, the desired peak value is $\check{\mathbf{h}}^T\check{\mathbf{x}}_i = c_i$, where $\check{\mathbf{x}}_i$ represents the $i$th image, $\check{\mathbf{h}}$ represents the template, and $c_i$ is the pre-specified filter response. The images are corrupted with noise so the actual correlation peak is $\check{\mathbf{h}}^T(\check{\mathbf{x}}_i + \check{\mathbf{n}}) = c_i + c_{\check{\mathbf{n}}}$, where $c_{\check{\mathbf{n}}} = \check{\mathbf{h}}^T\check{\mathbf{n}}$ and $\check{\mathbf{n}}$ is modeled as a zero-mean wide sense stationary random process. The ONV is defined as

$$\begin{aligned} var(u_{\check{\mathbf{n}}}) &= E\{(\check{\mathbf{h}}^T\check{\mathbf{n}})^2\} \\ &= \check{\mathbf{h}}^T E\{\check{\mathbf{n}}\check{\mathbf{n}}^T\}\check{\mathbf{h}} \\ &= \check{\mathbf{h}}^T\mathbf{C}\check{\mathbf{h}}, \end{aligned} \tag{2.9}$$

where $\mathbf{C} = E\{\check{\mathbf{n}}\check{\mathbf{n}}^T\}$ is the $d \times d$ covariance matrix of the noise, where $d$ is the dimension of the template.

$var(c_{\check{\mathbf{n}}})$ can be expressed in the frequency domain as follows. The variance is the value at the origin of the autocorrelation function, and therefore $var(c_{\check{\mathbf{n}}})$ can be computed by summing over all frequencies of $S_c$, the power spectral density (PSD) of $c_{\check{\mathbf{n}}}$. Further, $S_c$ can be computed as the PSD of $\check{\mathbf{n}}$ times the square of the magnitude of the filter frequency response. Thus, the ONV can be represented in vector notation as follows,

$$\begin{aligned} var(c_{\mathbf{n}}) &= \sum_{k=0}^{d-1} S_c[k] \\ &= \sum_{k=0}^{d-1} \mathbf{p}[k]|\mathbf{h}[k]|^2 \\ &= \mathbf{h}^\dagger\mathbf{P}\mathbf{h}, \end{aligned} \tag{2.10}$$

where $d$ is the dimension of $\mathbf{h}$, $\mathbf{p}$ represents the PSD of the input noise $\check{\mathbf{n}}$, and diagonal matrix $\mathbf{P}$ contains $\mathbf{p}$ along its diagonal. Using Parseval's theorem [49],

$$c_i = \check{\mathbf{h}}^T\check{\mathbf{x}}_i = \frac{1}{d}\mathbf{h}^\dagger\mathbf{x}_i = \frac{1}{d}u_i, \tag{2.11}$$

where $u_i = dc_i$. The quadratic $\mathbf{h}^\dagger\mathbf{P}\mathbf{h}$ is minimized subject to the linear constraints $\mathbf{X}^\dagger\mathbf{h} = \mathbf{u}$ when

(using the Lagrange multipliers method explained in Appendix A)

$$\mathbf{h} = \mathbf{P}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{P}^{-1}\mathbf{X})^{-1}\mathbf{u}. \tag{2.12}$$

When the additive noise is white noise, i.e., $\mathbf{P} = \alpha\mathbf{I}$ for $\alpha \neq 0$, the MVSDF filter equals the ECPSDF filter. This means that the ECPSDF filter minimizes the ONV under additive white noise.

## 2.4 Minimum Average Correlation Energy (MACE) filter

The MACE filter was introduced in 1987 [41]. This is the first filter designed to control the shape of the correlation plane $g_i(m, n)$ and not just the peak value $\mathbf{h}^{\dagger}\mathbf{x}_i$. The correlation output shape is controlled by minimizing the average correlation energy (ACE) of the correlation plane from to the training images while simultaneously satisfying the correlation outputs $\mathbf{h}^{\dagger}\mathbf{x}_i$ to yield a pre-specified value $u_i$ for all $i$.

The MACE filter facilitates recognition by producing very sharp delta-function-like peaks with minimum sidelobes for desired class training images and no such sharp peaks for false-class images. However the recognition performance significantly decreases for non-training intra-class images [16, 41]. In practice many images of interest have strong low frequency components. Since the MACE filter effectively whitens the spectrum (on the average), it enhances high frequency components. It is therefore very sensitive to distortions, i.e., to images outside of the training set, as well as to additive (high frequency) noise. In practice, although this filter has been successfully used in pattern recognition applications [68], variations of this filter are more robust (e.g., the OTSDF [55], GMACE [16], MSESDF [88], and MINACE [54] filters discussed below).

### 2.4.1 Derivation

The hard constraints are given by $\mathbf{h}^{\dagger}\mathbf{x}_i = u_i$, where $\mathbf{x}_i$ represents the $i$th image, $\mathbf{h}$ represents the filter, and $u_i$ is the pre-specified peak filter response. The correlation plane in response to image $\mathbf{x}_i$ is represented by $\mathbf{g}_i = \mathbf{X}_i^{*}\mathbf{h}$, where diagonal matrix $\mathbf{X}_i$ contains $\mathbf{x}_i$ along its diagonal. The energy of the correlation output $\breve{\mathbf{g}}_i$ is $E_i = \breve{\mathbf{g}}_i^T\breve{\mathbf{g}}_i = \frac{1}{d}\mathbf{g}_i^{\dagger}\mathbf{g}_i$. Since all the $E_i$ ($i = 1, \ldots, N$) cannot be simultaneously minimized subject to $\mathbf{X}^\dagger\mathbf{h} = \mathbf{u}$, the ACE is minimized instead. The ACE can be expressed as

$$\begin{aligned} E_{avg} &= \frac{1}{N}\sum_{i=1}^{N} E_i \\ &= \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{g}_i^\dagger \mathbf{g}_i \\ &= \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{h}^\dagger \mathbf{X}_i \mathbf{X}_i^* \mathbf{h} \\ &= \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{h}^\dagger \mathbf{D}_i \mathbf{h} \\ &= \mathbf{h}^\dagger \left( \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{D}_i \right) \mathbf{h} \\ &= \mathbf{h}^\dagger \mathbf{D} \mathbf{h}, \end{aligned} \tag{2.13}$$

where diagonal matrix $\mathbf{D}_i = \mathbf{X}_i\mathbf{X}_i^*$ contains the power spectrum of $\mathbf{x}_i$ along its diagonal, diagonal matrix $\mathbf{D} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{D}_i$ contains the average power spectral density of the training images along its diagonal, and the $\frac{1}{d}$ scalar is to account for the fact that inner products in the space domain are scaled by $\frac{1}{d}$ in the frequency domain, where $d$ is the dimension of the training vectors $\mathbf{x}_i$,

The quadratic $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ is minimized subject to the linear constraints $\mathbf{X}^\dagger\mathbf{h} = \mathbf{u}$ when (using the Lagrange multipliers method explained in Appendix A)

$$\mathbf{h} = \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{u}. \tag{2.14}$$

Premultiplying by the inverse of the average power spectral density $\mathbf{D}^{-1}$ is equivalent to whitening the average spectrum of the training images, resulting in sharper peaks (see Section 7.2.4 for a detailed explanation). In practice, the $d \times d$ matrix $\mathbf{D}$ is not constructed. Instead large matrix multiplication can be avoided taking advantage of $\mathbf{D}$'s diagonality.

One measurement of quality of the filter is the correlation output variance for the training images in the presence of additive zero-mean noise. Assuming zero-mean white noise with variance $\sigma^2$ then

$$\begin{aligned} var(\check{\mathbf{h}}^\dagger \check{\mathbf{n}}) &= E\{(\check{\mathbf{h}}^\dagger \check{\mathbf{n}})^2\} \\ &= E\{\check{\mathbf{h}}^\dagger \check{\mathbf{n}} \check{\mathbf{n}}^\dagger \check{\mathbf{h}}\} \\ &= \check{\mathbf{h}}^\dagger E\{\check{\mathbf{n}} \check{\mathbf{n}}^\dagger\} \check{\mathbf{h}} \\ &= \sigma^2 \check{\mathbf{h}}^\dagger \check{\mathbf{h}} \\ &= \sigma^2 E_{\check{\mathbf{h}}}, \end{aligned} \tag{2.15}$$

where $E_{\check{\mathbf{h}}}$ is the energy of the filter. Note that a filter with higher energy produces higher variance in the filter output.

A modification of the MACE filter suggested in the original paper is as follows: Correlation planes will have different energy values. In order to decrease the variation in the energy of these planes, average energy is modified as $\bar{E}_{avg} = \frac{1}{N}\sum_{i=1}^{N} a_i E_i$. Start with $a_i = 1\ \forall i$, compute $\mathbf{h}$, and alter $a_i$ according to $a_i = [E_i/E_{max}]^R$, where $E_{max} = max_i(E_i)$, and $R$ is a constant which determines the rate of convergence. This suboptimal filter will increase the ACE but reduce the variation in the energy of the planes.

### 2.4.2 Extensions

Savvides and Kumar [67] showed that the MACE filter can be efficiently trained online without inverting a matrix at each iteration, thus reducing the computational requirement. The design is able to train online and adapt to varying data streams caused by changes in illuminations, backgrounds, and/or different views (e.g., due to rotation, scale, pose, non-rigid deformation, etc). The application investigated was biometric authentication systems.

Boddeti, Su, and Kumar [10] used a modified form of the MACE filter for biometric encryption. The traditional MACE filter produces a sharp peak at the correlation output of centered images. The modified MACE filter produces multiple peaks at different locations by adjusting the phase of the images in the frequency domain. This filter is used for face verification and the peak locations are

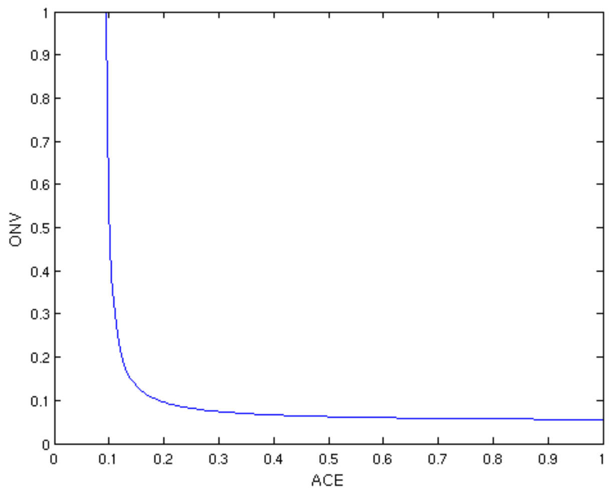


Figure 2.1: Tradeoff between the output noise variance (ONV) and the average correlation energy (ACE). Note that, although not shown, the values extend beyond 1.

used to encode a secret key of the authorized user.

## 2.5 Optimal Tradeoff Synthetic Discriminant Function (OTSDF) filter

The OTSDF filter was introduced in 1990 [55, 56]. The ACE is denoted by $E_1 = \mathbf{h}^\dagger \mathbf{D} \mathbf{h}$ and the ONV is denoted by $E_2 = \mathbf{h}^\dagger \mathbf{P} \mathbf{h}$. Minimizing $E_2$ typically leads to high-frequency emphasizing filters whereas minimizing $E_2$ typically leads to low-frequency emphasizing filters. Thus, minimizing one criterion significantly deteriorates the performance from the point of view of the other criterion. An optimal filter is defined such that for a given value of $E_1$, $E_2$ is minimized. Typically, slightly increasing the value of $E_1$ from its minimum, greatly improves $E_2$, and vice-versa as shown in Fig. 2.1.

In practice this filter is widely used. Since sharp peaks are usually more desirable than noise robustness, the filter is usually designed to perform more like the MACE filter than the MVSDF filter.

### 2.5.1 Derivation

The goal is to minimize $E_1$ subject to a specified value of $E_2$ and to the linear constraints $\mathbf{X}^\dagger\mathbf{h} = \mathbf{u}$ (this method could be generalized for more than two criteria). Lagrange multipliers are used to obtain the following functional

$$\begin{aligned}\mathcal{L}_\beta(\mathbf{h}, \beta, \Delta) &= E_1 + \beta E_2 - 2\Delta^\dagger(\mathbf{X}^\dagger\mathbf{h} - \mathbf{u}) \\ &= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} + \beta\mathbf{h}^\dagger\mathbf{P}\mathbf{h} - 2\Delta^\dagger(\mathbf{X}^\dagger\mathbf{h} - \mathbf{u}) \\ &= \mathbf{h}^\dagger\mathbf{T}\mathbf{h} - 2\Delta^\dagger(\mathbf{X}^\dagger\mathbf{h} - \mathbf{u}), \qquad (2.16)\end{aligned}$$

where $0 \leq \beta \leq \infty$ is a scalar Lagrange multiplier, $\Delta \neq 0$ is a vector of nonzero Lagrange multipliers, and $\mathbf{T} = \mathbf{D} + \beta\mathbf{P}$. Note that negative values for $\beta$ are not considered because $E_2 \geq 0$. The values of $\beta$ lie between $\beta = 0$ when only $E_1$ is optimized and $\beta = \infty$ when only $E_2$ is optimized. The solution that minimizes one criterion subject to a specific value for the other can be shown (see Appendix A, Eq. A.2) to be

$$\mathbf{h} = \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}. \qquad (2.17)$$

In order to replace use a bounded scalar, $\beta$ can be replaced with $\beta = \frac{1}{\lambda}(1-\lambda)$, where $0 \leq \lambda \leq 1$, i.e.,

$$\begin{aligned}\mathbf{T} &= \mathbf{D} + \beta\mathbf{P} \\ &= \mathbf{D} + \frac{1}{\lambda}(1-\lambda)\mathbf{P} \\ &= \frac{1}{\lambda}(\lambda\mathbf{D} + (1-\lambda)\mathbf{P}) \\ &\propto \lambda\mathbf{D} + (1-\lambda)\mathbf{P}, \qquad (2.18)\end{aligned}$$

where we define $\frac{0}{0} = 1$, and the $\frac{1}{\lambda}$ can be ignored because it does not affect the filter, i.e.,

$$\begin{aligned} \mathbf{h} &= \left(\frac{1}{\lambda}\mathbf{T}\right)^{-1} \mathbf{X} \left(\mathbf{X}^{\dagger} \left(\frac{1}{\lambda}\mathbf{T}\right)^{-1} \mathbf{X}\right)^{-1} \mathbf{u} \\ &= \frac{\lambda}{\lambda}\mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u} \\ &= \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}. \end{aligned} \tag{2.19}$$

For $\lambda = 0$ the filter becomes the MVSDF filter and for $\lambda = 1$ it becomes the MACE filter. In many experiments $\lambda$ takes on a value close to 1 to ensure a sharp peak but also some noise robustness.

## 2.6 Minimum Noise and Correlation Energy (MINACE) filter

The MINACE filter was introduced in 1992 [54]. The authors first developed a filter similar to the MACE filter called the Minimum Correlation Energy (MICE) filter. Although the MACE filter guarantees the least average correlation plane energy, it provides little control over the variability of the correlation plane energies and may result in a biased treatment of a particular training image (e.g., a training images whose correlation energy is significantly different from the mean correlation energy of the remaining training images). While the MACE filter averages the spectra of the training images, MICE uses a tight envelope of the training image spectra. Although the authors claim the MICE reduces variability of the correlation plane energies, they show no proof to support this claim.

The MINACE filter adds a flat frequency spectrum (the spectrum of white noise) into the MICE envelope. The MINACE filter is similar to the OTSDF filter providing a tradeoff between the peak sharpness and noise robustness. However, when the noise is additive noise, OTSDF filter is designed to outperform (i.e., reduce the ONV given an ACE) any other tradeoff filter (the MINACE filter included).

In practice MINACE has similar recognition performance to the OTSDF filter. Some publications claim a slight superior recognition performance over the OTSDF filter [54, 71] and others inferior recognition performance [26, 33] (the experiments cited for inferior recognition performance assumed additive white noise). Because sensor noise and clutter are not strictly additive noise we cannot show theoretically which filter has superior performance.

### 2.6.1 Derivation

The goal of MICE is to minimize a tight upper bound $E_{max}$ for all the energies in the training images, i.e., find $E_{max}$ so that

$$E_{max} \geq E_i, \tag{2.20}$$

where

$$E_i \;\; = \;\; \mathbf{h}^\dagger \mathbf{D}_i \mathbf{h}, \tag{2.21}$$

where diagonal matrix $\mathbf{D}_i$ contains the power spectrum of $\mathbf{x}_i$ along its diagonal. The next step is to find an $E_{max}$ of the form (the MINACE authors chose the quadratic assumption probably to emulate the ACE form)

$$E_{max} = \mathbf{h}^\dagger \mathbf{T} \mathbf{h} \tag{2.22}$$

that satisfies Eq. 2.20. One solution is to find a $\mathbf{T}(k)$ where

$$\mathbf{T}(k) \geq \mathbf{D}_i(k) \tag{2.23}$$

for all $i$ and for all frequencies $k$ (i.e., for all the diagonal elements of $\mathbf{T}$ and $\mathbf{D}_i$).

One possible solution is

$$\mathbf{T}_{SUM} = \sum_{i=1}^{N} \mathbf{D}_i \tag{2.24}$$

which is a scaled version of the $\mathbf{T}$ used in the MACE filter. $\mathbf{T}_{SUM}$, however, may result in a biased treatment of a particular training image. The authors claim that using

$$\mathbf{T}(k) = \max[\mathbf{D}_1(k), \cdots, \mathbf{D}_N(k)] \tag{2.25}$$

for each $k$ diagonal element in the matrices provides a tighter bound. However, the authors do not point out that $\mathbf{T}$ may also result in a biased treatment of a particular training image.

The quadratic $\mathbf{h}^\dagger \mathbf{T} \mathbf{h}$ is minimized subject to the linear constraints $\mathbf{X}^\dagger \mathbf{h} = \mathbf{u}$ when (using the

Lagrange multipliers method explained in Appendix A)

$$\mathbf{h} = \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}. \tag{2.26}$$

The MICE filter is computed substituting the $\mathbf{T}$ in Eq. 2.25 into Eq. 2.26. The MICE filter suffers from the same drawbacks as the MACE filter in that it is not designed to be robust to additive noise. In order to not emphasize the high frequency components of noise, the power spectrum of the noise $\mathbf{P}$ is added to the MICE envelope, i.e.,

$$\mathbf{T} = \max[\mathbf{D}_1(k), \cdots, \mathbf{D}_N(k), \beta\mathbf{P}(k)], \tag{2.27}$$

where $0 \leq \beta < \infty$ is used as a tradeoff between peak sharpness and noise robustness. When the power spectrum of the (usually high) frequency components fall below $\beta\mathbf{P}(k)$ for some frequency $k$, $\beta\mathbf{P}(k)$ is used for frequency $k$ in order to not overemphasize the high frequency components when computing the inverse of $\mathbf{T}$. Substituting the $\mathbf{T}$ in Eq. 2.27 into Eq. 2.26 results in the MINACE filter.

## 2.7 Gaussian MACE (GMACE) filter

The GMACE filter was introduced in 1991 [16]. To improve the intraclass recognition performance of the MACE filter, an additional constraint is added requiring the filter's correlations planes to approximate a specified shape function. This is achieved by minimizing the sum of squared errors between the filter's outputs and the desired shape of the training images, while simultaneously satisfying the correlation output constraints, $\mathbf{h}^{\dagger}\mathbf{x}_i = u_i$. Unlike the MACE filter, the GMACE filter is not designed to minimize the ACE. However if the desired correlation output is a delta function, then the MACE and GMACE filters are equivalent. The shape the authors chose is a Gaussian-function-like shape that allows control of the peak's sharpness by adjusting the width of the Gaussian. The sharper the peak, the more accurate the localization at the expense of a decrease in recognition performance for non-training images. The filter becomes more tolerant to distortion and noise as the width of the Gaussian increases. In practice, a Gaussian with a very small width (almost a delta-function-like) as the specified correlation shape is usually used.

### 2.7.1 Derivation

The mean squared error (MSE) between the correlation output $\mathbf{g}$ and the desired correlation output $\dot{\mathbf{g}}$ of the training images is given as follows,

$$\begin{aligned}
\text{MSE} &= \frac{1}{Nd}\sum_{i=1}^{N} |\mathbf{g}_i - \dot{\mathbf{g}}|^2 \\
&= \frac{1}{Nd}\sum_{i=1}^{N} \left(\mathbf{g}_i^\dagger \mathbf{g}_i - 2\mathbf{g}_i^\dagger \dot{\mathbf{g}} + \dot{\mathbf{g}}^\dagger \dot{\mathbf{g}}\right) \\
&= \frac{1}{Nd}\sum_{i=1}^{N} \left(\mathbf{h}^\dagger \mathbf{X}_i \mathbf{X}_i^* \mathbf{h} - 2\mathbf{h}^\dagger \mathbf{X}_i \dot{\mathbf{g}} + \dot{\mathbf{g}}^\dagger \dot{\mathbf{g}}\right) \\
&= \mathbf{h}^\dagger \left(\frac{1}{Nd}\sum_{i=1}^{N} \mathbf{X}_i \mathbf{X}_i^*\right) \mathbf{h} - 2\mathbf{h}^\dagger \left(\frac{1}{Nd}\sum_{i=1}^{N} \mathbf{X}_i \dot{\mathbf{g}}\right) + \frac{1}{d}\dot{\mathbf{g}}^\dagger \dot{\mathbf{g}} \\
&= \mathbf{h}^\dagger \mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger \mathbf{p} + E_f, \qquad (2.28)
\end{aligned}$$

where diagonal matrix $\mathbf{X}_i$ contains $\mathbf{x}_i$ along its diagonal, $\mathbf{g}_i = \mathbf{X}_i^* \mathbf{h}$ represents the correlation output for the $i$th image, $\mathbf{D} = \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{X}_i \mathbf{X}_i^*$ is the average power spectral density of the $N$ training images, $\dot{\mathbf{g}}$ represents the desired Gaussian-function-like correlation output, $E_f = \frac{1}{d}\dot{\mathbf{g}}^\dagger \dot{\mathbf{g}}$, and

$$\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{X}_i \dot{\mathbf{g}} = \frac{1}{d}\bar{\mathbf{X}}\dot{\mathbf{g}}, \qquad (2.29)$$

where diagonal $\bar{\mathbf{X}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_i$ contains the average of the 2-D DFT of the images along its diagonal.

The expression $\mathbf{h}^\dagger \mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger \mathbf{p} + E_f$ is minimized subject to the linear constraints $\mathbf{X}^\dagger \mathbf{h} = \mathbf{u}$ when (using the Lagrange multipliers method explained in Appendix A)

$$\begin{aligned}
\mathbf{h} &= \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^\dagger \mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{u} + \left[\mathbf{I} - \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^\dagger \mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{X}^\dagger\right]\mathbf{D}^{-1}\mathbf{p} \\
&= \mathbf{h}_{MACE} + \mathbf{h}_{shape}, \qquad (2.30)
\end{aligned}$$

where $\mathbf{h}_{MACE}$ ensures sharp peaks and $\mathbf{h}_{shape}$ serves to control the shape of the correlation plane.

### 2.7.2 Extensions

The Minimum Squared Error Synthetic Discriminant Function (MSESDF) filter introduced in 1992 [88] presents a small extension to the GMACE filter. The derivation includes using different desired shape functions as the correlation output for each of the training images, i.e., $= \frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \dot{\mathbf{g}}_i|^2$ (MSESDF) instead of $\frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \dot{\mathbf{g}}|^2$ (GMACE). The only difference in the derivation is that $\mathbf{p}$ is replaced by $\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i$ allowing each desired correlation output to be unique.

## 2.8 Unconstrained correlation filters: MACH and UMACE and UMSESDF filters

The first unconstrained CFs introduced in 1994 [45] were the Maximum Average Correlation Height (MACH) filter, the Generalized MACH (GMACH) filter, the Unconstrained MACE (UMACE) filter, and the unconstrained MSESDF (UMSESDF) filter. Previous SDF filters were constrained to produce an inner product of $\mathbf{h}^{\dagger}\mathbf{x}_i = u_i$ for the training images, but such hard constraints are not necessarily satisfied by the non-training images. Removing these constraints increases the solution space and may improve the chances of finding a filter with better recognition performance. The draw-back with these filters is their high dependency on the average of the training images $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ which, in some cases, may be too blob-like. This issue is addressed below in this section's "Extensions".

The MACH filter has been investigated for many applications [38, 42, 48, 63, 91]. The MACH filter is designed to minimize the average (dis-)similarity measure (ASM), i.e., the scatter of the correlation planes, and simultaneously minimize the ACE and maximize the average correlation peak intensity ($|\bar{\mathbf{g}}|^2 = |\mathbf{h}^{\dagger}\bar{\mathbf{x}}|^2$). The Generalized MACH filter is designed as a MACH filter with the additional criterion to minimize the variance of the correlation peaks.

The UMACE and UMSESDF filters are variations of the MACH filter that ignore the ASM or the ACE, respectively. Numerical results show [45] that MACH filter outperforms these other filters in recognition performance.

Savvides and Kumar [67] showed that the UMACE filter can be efficiently trained online and adapt to varying data streams caused by changes in illuminations, backgrounds, and/or different views (e.g., due to rotation, scale, pose, non-rigid deformation, etc).

### 2.8.1 Derivation

The first step is to compute the ASM between an ideal desired correlation output represented by $\dot{\mathbf{g}}$ and the actual correlation outputs represented by $\mathbf{g}_i$. The ideal desired correlation output that minimizes the distortion of the correlation outputs with respect to $\dot{\mathbf{g}}$ measured as the MSE

$$e = \frac{1}{Nd}\sum_{i=1}^{N} |\mathbf{g}_i - \dot{\mathbf{g}}|^2 \tag{2.31}$$

is found by taking the gradient of $e$ with respect to $\dot{\mathbf{g}}$, setting it equal to zero, and solving for $\dot{\mathbf{g}}$. This gives

$$\dot{\mathbf{g}}_{OPT} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}_i = \bar{\mathbf{g}}. \tag{2.32}$$

Substituting Eq. 2.32 into Eq. 2.31 gives the ASM,

$$\begin{aligned} \text{ASM} &= \frac{1}{Nd}\sum_{i=1}^{N} |\mathbf{g}_i - \bar{\mathbf{g}}|^2 \\ &= \frac{1}{Nd}\sum_{i=1}^{N} |\mathbf{X}_i^*\mathbf{h} - \bar{\mathbf{X}}^*\mathbf{h}|^2 \\ &= \mathbf{h}^\dagger \left( \frac{1}{Nd}\sum_{i=1}^{N} (\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^* \right) \mathbf{h} \\ &= \mathbf{h}^\dagger \mathbf{S}\mathbf{h}, \end{aligned} \tag{2.33}$$

where diagonal matrix $\mathbf{X}_i$ contains $\mathbf{x}_i$ along its diagonal, and diagonal matrix $\mathbf{S} = \frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^*$ represents a measure of the similarity (or more correctly, dissimilarity) of the training images to the true-class mean. The ACE is the previously derived quadratic (see Eq. 2.13)

$$\text{ACE} = \mathbf{h}^\dagger \mathbf{D}\mathbf{h}, \tag{2.34}$$

where diagonal matrix $\mathbf{D} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*$ contains the average power spectrum of the training images along its diagonal.

The average peak intensity may be expressed as

$$|\bar{u}|^2 = |\mathbf{h}^\dagger\bar{\mathbf{x}}|^2 = \mathbf{h}^\dagger\bar{\mathbf{x}}\bar{\mathbf{x}}^\dagger\mathbf{h}. \tag{2.35}$$

The filter $\mathbf{h}$ that simultaneously maximizes the average peak intensity $|\bar{u}|^2$ and minimizes both ASM and ACE is obtained using the following Rayleigh quotient (RQ),

$$J(\mathbf{h}) = \frac{\mathbf{h}^\dagger\bar{\mathbf{x}}\bar{\mathbf{x}}^\dagger\mathbf{h}}{\mathbf{h}^\dagger\mathbf{S}\mathbf{h} + \mathbf{h}^\dagger\mathbf{D}\mathbf{h}} \tag{2.36}$$

which is maximized (see Appendix B) when

$$\mathbf{h}_{MACH} = (\mathbf{D} + \mathbf{S})^{-1}\bar{\mathbf{x}}. \tag{2.37}$$

The GMACH filter is obtained by minimizing the following variance of the correlation peaks,

$$\begin{aligned}\sigma^2_{\mathbf{g}_0} &= \frac{1}{Nd}\sum_{i=1}^{N}\left|\mathbf{h}^\dagger\mathbf{x}_i - \mathbf{h}^\dagger\bar{\mathbf{x}}\right|^2 \\ &= \mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\dagger\right)\mathbf{h} \\ &= \mathbf{h}^\dagger\mathbf{V}\mathbf{h}, \end{aligned} \tag{2.38}$$

where $\mathbf{V} = \frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\dagger$. To minimize this variance and the ASM and ACE, Eq. 2.36 is rewritten as

$$J(\mathbf{h}) = \frac{\mathbf{h}^\dagger\bar{\mathbf{x}}\bar{\mathbf{x}}^\dagger\mathbf{h}}{\mathbf{h}^\dagger\left(\delta\mathbf{V} + \mathbf{S} + \mathbf{D}\right)\mathbf{h}}, \tag{2.39}$$

where $\delta$ is used to control the emphasis on $\mathbf{V}$. The solution to Eq. 2.39 is

$$\mathbf{h}_{GMACH} = (\delta\mathbf{V} + \mathbf{D} + \mathbf{S})^{-1}\bar{\mathbf{x}}. \tag{2.40}$$

Although the sum of these matrices is a non-diagonal matrix, $\mathbf{h}_{GMACH}$ can be efficiently computed [48] requiring the inversion of an $N \times N$ matrix (instead of $d \times d$), where the number of training images $N$ is usually much less than the number of pixels $d$.

The UMACE filter is obtained by ignoring $\mathbf{S}$ and $\mathbf{V}$, i.e.,

$$\mathbf{h}_{UMACE} = \mathbf{D}^{-1}\bar{\mathbf{x}}. \tag{2.41}$$

This looks like the MACE filter, i.e.,

$$\begin{aligned} \mathbf{h}_{MACE} &= \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{u} \\ &= \mathbf{D}^{-1}\mathbf{X}\mathbf{a} \\ &= \mathbf{D}^{-1}\hat{\mathbf{x}}, \end{aligned} \tag{2.42}$$

where $\hat{\mathbf{x}}$ represents a weighted average of the training images and $\mathbf{a} = (\mathbf{X}^{\dagger}\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{u}$ is the weight vector necessary to satisfy the hard constraints on the training images. It is interesting to note that by choosing $\mathbf{u}$ carefully $\mathbf{h}_{MACE}$ can equal $\mathbf{h}_{UMACE}$.

The UMSESDF filter is obtained by ignoring $\mathbf{D}$ and $\mathbf{V}$, i.e.,

$$\mathbf{h}_{UMSEDSF} = \mathbf{S}^{-1}\bar{\mathbf{x}}. \tag{2.43}$$

However $\mathbf{h}_{UMSESDF}$ is not used in practice because it does not produce sharp peaks.

The authors also noted that by normalizing the amplitude of the Fourier transform of the training images, the $\mathbf{D}$ matrix is simplified to

$$\mathbf{D} = \frac{1}{dN}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^{*} = \frac{1}{d}\mathbf{I}, \tag{2.44}$$

where $\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^{*} = \mathbf{I}$ which can improve computational efficiency.

### 2.8.2 Extensions

Savvides, Venkataramani, and Kumar [65] used what they coined the Unconstrained OTSDF (UOTSDF) filter as an extension to UMACE filter. The UMACE filter's $\mathbf{D}$ in Eq. 2.41 is replaced by the OTSDF filter's $\mathbf{T}$ in Eq. 2.18, i.e.,

$$\mathbf{h}_{UMACE} = \mathbf{T}^{-1}\bar{\mathbf{x}}. \tag{2.45}$$

Our experimental results (see Ch. 7) show that UOTSDF outperforms UMACE.

Alkanhal, Kumar, and Mahalanobis [4] proposed the Extended MACH (EMACH) filter to address the high dependence on the mean training image. They claim that the response of the MACH filter (and other SDF filters) to a training image follows too closely the response of the mean training image, and that the mean training image is not always a good representation of the true-class. In addition, the MACH filter gives a biased treatment to the low-frequency components represented by the mean training images (the high frequency components in the training images where much of the discriminatory information is found can be blurred when computing the mean). The EMACH filter modifies the objective function in Eq. 2.36 to

$$J(\mathbf{h}) = \frac{\mathbf{h}^\dagger \mathbf{E} \mathbf{h}}{\mathbf{h}^\dagger \mathbf{S} \mathbf{h} + \mathbf{h}^\dagger \mathbf{D} \mathbf{h}}, \tag{2.46}$$

where

$$\mathbf{S} \;=\; \frac{1}{Nd} \sum_{i=1}^{N} \left( \mathbf{X}_i - (1-\beta)\bar{\mathbf{X}} \right) \left( \mathbf{X}_i - (1-\beta)\bar{\mathbf{X}} \right)^*, \tag{2.47}$$

and

$$\mathbf{E} \;=\; \frac{1}{Nd} \sum_{i=1}^{N} (\mathbf{x}_i - \beta \bar{\mathbf{x}})(\mathbf{x}_i - \beta \bar{\mathbf{x}})^\dagger, \tag{2.48}$$

where $\beta$ is a value between 0 and 1 ($\beta = 0$ is the MACH filter). This is maximized (see Appendix B) when $\mathbf{h}$ is the eigenvector corresponding to the largest eigenvalue of $(\mathbf{D} + \mathbf{S})^{-1}\mathbf{E}$.

Kumar and Alkanhal [83] proposed the Eigen-EMACH (EEMACH) filter. This follows the same design as the EMACH filter but instead approximates the matrix $\mathbf{S}$ by its dominant eigenvectors. This causes the filter to be less specific on the training images and generalize better to true-class images outside the training set.

A comparative study by Kerekes and Kumar [33] show that MACH outperformed the EEMACH and EMACH filters using gray-scaled images. Another study by Van Nevel and Mahalanobis [48] showed that the GMACH filter outperformed the MACH and EMACH filters using LADAR data.

A natural extension to the MACH filter is the Optimal Tradeoff MACH (OTMACH) filter presented by Kumar, Carlson, and Mahalanobis [84] using the ideas from the OTSDF filter (see Section 2.5). Note that in their paper they referred to that filter as the OTSDF filter but others (e.g., [7]) call it the OTMACH filter to avoid confusion with the the OTSDF filter introduced in Section 2.5. In this design they minimize ASM (or modify ASM–see Eq. 2.47), ONV, and ACE. The solution is similar to Eq. 2.37,

$$\mathbf{h} = \mathbf{T}^{-1}\bar{\mathbf{x}}, \tag{2.49}$$

where $\mathbf{T} = \alpha\mathbf{P} + \beta\mathbf{D} + \gamma\mathbf{S}$ is a linear combination of the ONV, ACE, and ASM. Banerjee, Chandra, and Datta [7] proposed using a neural network to find suitable parameters $\alpha, \beta, \gamma$ for face recognition.

A comparative study by Singh and Kumar [75] showed that the OTMACH filter outperformed the MACH, EMACH, DCCF (discussed in Section 2.14), and Polynomial DCCF (discussed in Section 2.15) filters, and that the EMACH filter outperformed the MACH filter on synthetic aperture radar (SAR) data.

## 2.9 Action MACH filter

The Action MACH filter was introduced in 2008 [63]. Instead of using a set of 2-D training images, the Action MACH filter uses a set of 3-D training videos. The 3-D DFT of each video is computed, vectorized, and used for training in the same exact way that vectorized images are used to train the MACH filter.

The Action MACH filter can be designed using both scalar and vector features with some modifications to the DFT computations when using feature vectors. The types of features used in training must also be used for testing. For scalar features the temporal derivative of each pixel is computed. Vector features used the Spatio-temporal Regularity Flow (SPREF) [2] features. Each pixel is replaced by a 3-D feature vector that represents the direction along which the intensity changes the least. In addition, the Clifford Fourier transform [23] is used in order to compute Fourier transforms of the videos with vector features.

### 2.9.1 Derivation

The derivation is the same as the MACH filter (see Section 2.8) using 3-D (instead of 2-D) DFTs for the training and testing videos.

## 2.10 Average of Synthetic Exact Filter (ASEF)

The ASEF was introduced in 2009 [12]. This unconstrained filter is designed by building one filter $\mathbf{h}_i$ for each training image and then taking the average of all the filters to obtain $\mathbf{h}$. Averaging the filters results in a filter that avoids over-fitting to the training set. Each filter $\mathbf{h}_i$ exactly maps a training image to a desired correlation plane $\dot{\mathbf{g}}_i$. Similar to the GMACE filter, the authors chose a Gaussian-function-like shape as the desired correlation plane and used the Gaussian's width to tradeoff between sharp peaks for easy localization (small width), and broad peaks for distortion (large width). In their design the images do not need to be centered as long as the Gaussian's center is at the target's centered location. In fact, it is possible to have multiple targets in one training image [13] with Gaussians-function-like shapes centered at each of the target locations.

The filter has been successfully used for ocular localization (the eye and surrounding regions) [12] and for pedestrian localization [13]. One disadvantage of ASEF is the large number of training images required (in the order of hundreds) for good recognition performance. This disadvantage is addressed by the MOSSE filter discussed in the next section.

### 2.10.1 Derivation

Given the 2-D DFT of a training image $X_i(k,l)$ and a desired 2-D DFT of a correlation plane $\dot{G}_i(k,l)$, the exact filter is

$$H_i(k,l) = \frac{\dot{G}_i(k,l)}{X_i^*(k,l)}, \tag{2.50}$$

and the ASEF is

$$H(k,l) = \frac{1}{N}\sum_{i=1}^{N} H_i(k,l). \tag{2.51}$$

If ASEF is trained on a small number of images, the filter can become unstable when some frequencies of $X_i$ are close to zero.

## 2.11 Minimum Output Sum of Squared Error (MOSSE) filter

The MOSSE filter was introduced in 2010 [11]. This filter is designed to minimize the MSE between the desired correlation plane and the actual correlation plane. The difference between the MOSSE and MSESDF filter is that no hard constraints are placed on the correlation peak in the MOSSE filter (recall that the MSESDF filter required $\mathbf{X}^\dagger\mathbf{h} = \mathbf{u}$), and that the targets do not need to be centered in the training set (this second difference is insignificant when viewed in the frequency domain). The simplicity of the MOSSE filter allows the filter to adapt in real time to changes due to rigid-body motion, deformation, and/or lighting. The filter adapts by weighting new images more, with weights for older images decaying exponentially over time. The application investigated was adaptively recognizing faces and other objects-of-interest as the images go through different changes in illuminations and poses. It was reported that the filter processed frames at a rate of 669 frames per second using a 2.4 GHz Core 2 Duo CPU.

### 2.11.1 Derivation

The MSE between the correlation output $\mathbf{g}$ and the desired correlation output $\dot{\mathbf{g}}_i$ of the training images is given by

$$\begin{aligned}
\text{MSE} &= \frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \dot{\mathbf{g}}_i|^2 \\
&= \frac{1}{Nd}\sum_{i=1}^{N}\left(\mathbf{g}_i^\dagger\mathbf{g}_i - 2\mathbf{g}_i^\dagger\dot{\mathbf{g}} + \dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i\right) \\
&= \frac{1}{Nd}\sum_{i=1}^{N}\left(\mathbf{h}^\dagger\mathbf{X}_i\mathbf{X}_i^*\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{X}_i\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i\right) \\
&= \mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*\right)\mathbf{h} - 2\mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i\right) + \frac{1}{Nd}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + E_f, \qquad (2.52)
\end{aligned}$$

where diagonal matrix $\mathbf{X}_i$ contains the entries of the $\mathbf{x}_i$ along its diagonal, $\mathbf{D} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*$, $\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i$, $E_f = \frac{1}{Nd}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^{\dagger}\dot{\mathbf{g}}_i$, and $\dot{\mathbf{g}}$ represents the desired correlation output. The $\mathbf{h}$ that minimizes the MSE is found by taking its gradient and setting it equal to zero, i.e.,

$$2\mathbf{D}\mathbf{h} - 2\mathbf{p} = \mathbf{0}, \tag{2.53}$$

and solving for $\mathbf{h}$ gives,

$$\begin{aligned}\mathbf{h} &= \mathbf{D}^{-1}\mathbf{p} \\ &= \left(\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*\right)^{-1}\left(\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i\right),\end{aligned} \tag{2.54}$$

which can be expressed as

$$H(k,l) = \frac{\sum_{i=1}^{N} X_i(k,l)\dot{G}_i^*(k,l)}{\sum_{i=1}^{N} X_i(k,l)X_i^*(k,l)}. \tag{2.55}$$

This filter can be modified so that it can be adapted to new training images. For a new frame $X_i(k,l)$ the filter is updated as follows,

$$H_i(k,l) = \frac{A_i(k,l)}{B_i(k,l)}, \tag{2.56}$$

where

$$A_i(k,l) = \eta(X_i(k,l)\dot{G}_i^*(k,l)) + (1-\eta)A_{i-1}(k,l), \tag{2.57}$$

and

$$B_i(k,l) = \eta\left(X_i(k,l)X_i^*(k,l)\right) + (1-\eta)B_{i-1}(k,l), \tag{2.58}$$

where $\eta$ is the learning rate (the MOSSE authors used $\eta = 0.125$).

## 2.12 Optimal Tradeoff Circular Harmonic Function (OTCHF) filter

The OTCHF was introduced in 2000 [89]. Circular harmonic functions (CHFs) were used in 1982 [29] to design in-plane rotation invariant filters giving a constant output as the input is rotated. However, that design uses only one harmonic and, therefore, ignores a lot of the discriminatory pattern information. In 1985, Schils and Sweeney [69] showed that the ECPSDF (see Section 2.2) can be used to produce rotationally-invariant filters using multiple in-plane rotations as training images and taking the limit of the interval between each rotation to zero (i.e., having an infinite number of images). In 1986, Kumar [90] introduced a filter design where the correlation output can be varied in a specified manner with input rotation. The OTCHF filter improves upon this design and includes the ACE, ONV, and ASM criteria discussed previously. The result is a filter whose correlation planes have sharp peaks at the object's locations for a specific range of rotations (e.g., between $45^\circ$ and $90^\circ$) and low values elsewhere.

### 2.12.1 Derivation

Let $X(\rho,\phi)$, $H(\rho,\phi)$, and $G(\rho,\phi)$ be the frequency domain polar representation of $X(u,v)$, $H(u,v)$, and $G(u,v)$, respectively, where $\rho = (u^2+v^2)^{\frac{1}{2}}$ corresponds to the magnitude and $\phi = \arctan(\frac{v}{u})$ corresponds to the angle between $u$ and $v$, respectively. The correlation value at the origin (the correlation peak) is

$$\begin{aligned} g_0 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(u,v)dudv \\ &= \int_{0}^{\infty}\int_{0}^{2\pi} G(\rho,\phi)\rho d\phi d\rho \\ &= \int_{0}^{\infty}\int_{0}^{2\pi} X(\rho,\phi)H^*(\rho,\phi)\rho d\phi d\rho \end{aligned} \tag{2.59}$$

(note that in Cartesian coordinates the integration is over $dudv$ and in polar coordinates is over $\rho d\phi d\rho$).

Periodic signals can be expressed using a Fourier series expansion. Since $X(\rho,\phi)$, $H(\rho,\phi)$, and $G(\rho,\phi)$ are periodic in $\phi$ with period $2\pi$ they can be expressed as

$$X(\rho,\phi) = \sum_{k=-\infty}^{\infty} X_k(\rho)e^{jk\phi}, \tag{2.60}$$

and

$$X_k(\rho) = \frac{1}{2\pi}\int_0^{2\pi} X(\rho,\phi)e^{-jk\phi}d\phi, \tag{2.61}$$

where $X_k(\rho)$ is the $k$th CHF of $X(u,v)$, and similarly for $H(\rho,\phi)$ and $G(\rho,\phi)$. Substituting these

Fourier series expansions back into Eq. 2.59 gives

$$\begin{aligned} g_0 &= \int_0^\infty \int_0^{2\pi} \sum_{k=-\infty}^{\infty} X_k(\rho) e^{jk\phi} \sum_{l=-\infty}^{\infty} H_l^*(\rho) e^{-jl\phi} \rho d\phi d\rho \\ &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \int_0^{2\pi} e^{j(k-l)\phi} d\phi \int_0^\infty X_k(\rho) H_l^*(\rho) \rho d\rho. \end{aligned} \tag{2.62}$$

Noting that

$$\int_0^{2\pi} e^{j(k-l)\phi} d\phi = \begin{cases} 2\pi & k = l \\ 0 & k \neq l \end{cases}, \tag{2.63}$$

Eq. 2.62 can be rewritten as follows,

$$\begin{aligned} g_0 &= \sum_{k=-\infty}^{\infty} 2\pi \int_0^\infty X_k(\rho) H_k^*(\rho) \rho d\rho \\ &= \sum_{k=-\infty}^{\infty} C_k, \end{aligned} \tag{2.64}$$

where

$$C_k = 2\pi \int_0^\infty X_k(\rho) H_k^*(\rho) \rho d\rho \tag{2.65}$$

is loosely referred to as the $k$th CHF weight.

A rotation in the input Cartesian image is equivalent to a circular shift in the polar image along the $\phi$ axis which is equivalent to a phase change in the frequency polar domain (note that Eq. 2.60 changes to $X(\rho, \phi + \theta) = \sum_{k=-\infty}^{\infty} X_k(\rho) e^{jk(\phi+\theta)}$). The correlation peak value of an input image rotated by $\theta$ is therefore

$$\begin{aligned} g_\theta &= \sum_{k=-\infty}^{\infty} 2\pi \int_0^\infty \left( X_k(\rho) e^{jk\theta} \right) H_k^*(\rho) \rho d\rho \\ &= \sum_{k=-\infty}^{\infty} C_k e^{jk\theta}. \end{aligned} \tag{2.66}$$

In order to have a constant correlation peak for all rotations, i.e., $g_\theta = c \ \forall \theta$, only one CHF weight can be used in Eq. 2.66. This is because using more than one CHF weight includes oscillatory terms from $e^{jk\theta}$. However, if only one CHF weight is nonzero then all but one of the terms of the Fourier expansion are ignored which leads to poor discrimination. One solution is to find coefficients $C_k$ that approximate a desired $g_\theta$, e.g., a high value between $45^\circ$ and $90^\circ$ and low values everywhere else. This is equivalent to the finite impulse response (FIR) filter design problem. There are excellent FIR designs methods presented elsewhere [49] that can be used to find the coefficients $C_k$.

Once the coefficients $C_k$ are computed, Eq. 2.65 is used to solve for the CHFs $H_k(\rho)$. There are infinite solutions, and therefore other constraints such as ONV, ACE and ASM can be included to find a unique solution.

The ONV is obtained by computing the variance at the correlation peak due to additive noise. Assuming the noise to be isotropic (i.e., $P_n(\rho, \phi) = P_n(\rho)$),

$$\begin{aligned}
\text{ONV} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P_n(u,v)|H(u,v)|^2 du dv \\
&= \int_0^{\infty}\int_0^{2\pi} P_n(\rho,\phi)|H(\rho,\phi)|^2 \rho d\phi d\rho \\
&= \int_0^{\infty}\int_0^{2\pi} P_n(\rho)|H(\rho,\phi)|^2 \rho d\phi d\rho \\
&= \int_0^{\infty}\int_0^{2\pi} P_n(\rho)\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} H_k(\rho)H_l^*(\rho)e^{j(k-l)\phi}\rho d\phi d\rho \\
&= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\int_0^{\infty} P_n(\rho)H_k(\rho)H_l^*(\rho)\rho d\rho\left[\int_0^{2\pi} e^{j(k-l)\phi}d\phi\right] \\
&= 2\pi\sum_{k=-\infty}^{\infty}\int_0^{\infty} P_n(\rho)|H_k(\rho)|^2\rho d\rho. \qquad (2.67)
\end{aligned}$$

The ACE is obtained by averaging the correlation energies for each angle $\theta$, i.e.,

$$\begin{aligned}
\text{ACE} &= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi}|G(\rho,\phi+\theta)|^2\rho d\phi d\rho d\theta \\
&= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi}|X(\rho,\phi+\theta)|^2|H(\rho,\phi)|^2\rho d\phi d\rho d\theta \\
&= \frac{1}{2\pi}\iint_0^{2\pi}\int_0^{\infty}\left[\sum_{k,l}X_k(\rho)X_l^*(\rho)e^{j(k-l)(\phi+\theta)}\right]\left[\sum_{m,n}H_m(\rho)H_n^*(\rho)e^{j(m-n)\phi}\right]\rho d\rho d\phi d\theta \\
&= \sum_{k,l,m,n}\int_0^{\infty}X_k(\rho)X_l^*(\rho)H_m(\rho)H_n^*(\rho)\rho d\rho\int_0^{2\pi}e^{j(k-l+m-n)\phi}d\phi\left[\frac{1}{2\pi}\int_0^{2\pi}e^{j(k-l)\theta}d\theta\right] \\
&= \sum_{k=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\int_0^{\infty}|X_k(\rho)|^2H_m(\rho)H_n^*(\rho)\rho d\rho\left[\int_0^{2\pi}e^{j(m-n)\phi}d\phi\right] \\
&= 2\pi\sum_{k=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\int_0^{\infty}|X_k(\rho)|^2|H_m(\rho)|^2\rho d\rho \\
&= 2\pi\sum_{m=-\infty}^{\infty}\int_0^{\infty}P_X(\rho)|H_m(\rho)|^2\rho d\rho, \qquad (2.68)
\end{aligned}$$

where

$$P_X(\rho)=\sum_{k=-\infty}^{\infty}|X_k(\rho)|^2 \qquad (2.69)$$

is the sum of the power spectra of the CHF $X_k(\rho)$.

The ASM measures the dissimilarities in the correlations planes for all possible angles $\theta$, i.e.,

$$\begin{aligned}
\text{ASM} &= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi}|G(\rho,\phi+\theta)-\bar{G}(\rho,\phi)|^2\rho d\phi d\rho d\theta \\
&= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi}|X(\rho,\phi+\theta)H(\rho,\phi)-\bar{X}(\rho,\phi)H(\rho,\phi)|^2\rho d\phi d\rho d\theta \\
&= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi}|X(\rho,\phi+\theta)-\bar{X}(\rho,\phi)|^2|H(\rho,\phi)|^2\rho d\phi d\rho d\theta, \qquad (2.70)
\end{aligned}$$

where

$$\begin{aligned}\bar{X}(\rho,\phi) &= \frac{1}{2\pi}\int_0^{2\pi} X(\rho,\phi+\psi)d\psi \\ &= \frac{1}{2\pi}\int_0^{2\pi}\sum_{z=-\infty}^{\infty} X_z(\rho)e^{jz(\phi+\psi)}d\psi \\ &= \sum_{z=-\infty}^{\infty} X_z(\rho)e^{jz\phi}\left[\frac{1}{2\pi}\int_0^{2\pi} e^{jz\psi}d\psi\right] \\ &= X_0(\rho). \end{aligned} \tag{2.71}$$

Substituting Eq. 2.71 into Eq. 2.70 gives

$$\begin{aligned}\text{ASM} &= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi} |X(\rho,\phi+\theta)-X_0(\rho)|^2|H(\rho,\phi)|^2\rho d\phi d\rho d\theta \\ &= \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\int_0^{2\pi} \sum_{k,l\neq 0} X_k(\rho)X_l^*(\rho)e^{j(k-l)(\phi+\theta)}\sum_{m,n} H_m(\rho)H_n^*(\rho)e^{j(m-n)\phi}\rho d\phi d\rho d\theta \\ &= 2\pi\sum_{k\neq 0}\sum_{m=-\infty}^{\infty}\int_0^{\infty} |X_k(\rho)|^2|H_m(\rho)|^2\rho d\rho \\ &= 2\pi\sum_{m=-\infty}^{\infty}\int_0^{\infty} P_{ASM}(\rho)|H_m(\rho)|^2\rho d\rho, \end{aligned} \tag{2.72}$$

where

$$P_{ASM}(\rho) = \sum_{k\neq 0} |X_k(\rho)|^2. \tag{2.73}$$

To minimize the ASM, ACE, and ONV, Refregier [55] showed that an optimal tradeoff among quadratic criteria can be obtained by minimizing a weighted sum of the criteria. The filter is obtained minimizing the following figure of (de-)merit (FOM) subject to Eq. 2.65,

$$\begin{aligned}\text{FOM} &= \text{ACE} + \gamma\text{ASM} + \beta\text{ONV} \\ &= 2\pi\sum_{m=-\infty}^{\infty}\int_0^{\infty} P_{FOM}(\rho)|H_m(\rho)|^2\rho d\rho, \end{aligned} \tag{2.74}$$

where

$$P_{FOM}(\rho) = P_X(\rho) + \gamma P_{ASM}(\rho) + \beta P_n(\rho) \tag{2.75}$$

and $\gamma, \beta \geq 0$. It can be shown that minimizing this quadratic criterion subject to the linear constraints yields

$$H_k(\rho) = \lambda_k^* \frac{X_k(\rho)}{P_{FOM}(\rho)}, \tag{2.76}$$

where

$$\lambda_k = \frac{C_k}{\int_0^{2\pi} \frac{|X_k(\rho)|^2}{P_{FOM}(\rho)} \rho d\rho}. \tag{2.77}$$

## 2.13 MACE-Mellin Radial Harmonic (MACE-MRH) filter

The MACE-MRH filter was introduced in 2006 [31]. The Mellin transform is invariant to scale changes in the spatial domain (this is similar to the Fourier transform magnitude being invariant to shifts in the spatial domain) and is commonly used in pattern recognition applications; however, it requires the images to be centered. Mellin radial harmonics (MRH) were first used in 1988 [46] to design shift-invariant filters that were also invariant to scale. However, that design only uses one harmonic and therefore ignores much of the discriminatory pattern information. The MACE-MRH filter only provides scale invariance for a limited range of scales but is able to use more harmonics, therefore improving discrimination. In addition it applies the ACE criterion discussed previously to have sharp peaks at the object's location in the correlation plane.

### 2.13.1 Derivation

Let $X(\rho, \phi)$, $H(\rho, \phi)$, and $G(\rho, \phi)$ be the polar representations of $X(u, v)$, $H(u, v)$, and $G(u, v)$, respectively, where $\rho = (u^2 + v^2)^{\frac{1}{2}}$ corresponds to the magnitude and $\phi = \arctan(\frac{v}{u})$ corresponds to the angle between $u$ and $v$, respectively. The MRH expansion of a signal $X(\rho, \phi)$ is [46]

$$X(\rho, \phi) \quad = \quad \sum_{k=-\infty}^{\infty} X_k(\phi) \rho^{j2\pi k - 1}, \tag{2.78}$$

and

$$X_k(\phi) = L^{-1} \int_{r_0}^{R} X(\rho, \phi) \rho^{-j2\pi k - 1} \rho d\rho, \tag{2.79}$$

where

$$L = \ln R - \ln r_0 \tag{2.80}$$

is a positive integer value (this is to satisfy the orthogonality required among the MRH components [46]), and the limits of integration are chosen to expand the radial bandwidth of the pattern, e.g., $r_0$ is a positive number close to zero and $R$ a number greater than the radial bandwidth $\rho_{\max}$.

The correlation peak using the MRH expansion is (see Eq. 2.59) given as follows,

$$\begin{aligned}
g_0 &= \int_0^\infty \int_0^{2\pi} X(\rho,\phi)H^*(\rho,\phi)\rho d\phi d\rho \\
&\approx \int_{r_0}^R \int_0^{2\pi} X(\rho,\phi)H^*(\rho,\phi)\rho d\phi d\rho \\
&= \int_{r_0}^R \int_0^{2\pi} \left[\sum_{k=-\infty}^{\infty} X_k(\phi)\rho^{j2\pi k-1}\right]\left[\sum_{l=-\infty}^{\infty} H_l^*(\phi)\rho^{-j2\pi l-1}\right]\rho d\phi d\rho \\
&= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} \int_0^{2\pi} X_k(\phi)H_l^*(\phi)d\phi \left[\int_{r_0}^R \rho^{j2\pi(k-l)-2}\rho d\rho\right] \\
&= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} \int_0^{2\pi} X_k(\phi)H_l^*(\phi)d\phi \left[L\delta(k-l)\right] \\
&= L\sum_{k=-\infty}^{\infty} \int_0^{2\pi} X_k(\phi)H_k^*(\phi)d\phi \\
&= L\sum_{k=-\infty}^{\infty} C_k, \qquad (2.81)
\end{aligned}$$

where

$$C_k = \int_0^{2\pi} X_k(\phi)H_k^*(\phi)d\phi. \qquad (2.82)$$

The radial integral $\int_{r_0}^R \rho^{j2\pi(k-l)-2}\rho d\rho = L\delta(k-l)$ because of the orthogonality of MRH components. This can be shown mathematically as follows. Note that

$$\int_{r_0}^R \rho^{j2\pi(k-l)-2}\rho d\rho = \int_{r_0}^{r_0 e^L} \rho^{j2\pi(k-l)-1}d\rho \qquad (2.83)$$

($R = r_0 e^L$ comes from Eq. 2.80). When $k = l$

$$\begin{aligned}
\int_{r_0}^{r_0 e^L} \rho^{j2\pi(k-l)-1} d\rho &= \int_{r_0}^{r_0 e^L} \rho^{-1} d\rho \\
&= \ln \rho \Big|_{\rho=r_0}^{r_0 e^L} \\
&= \ln\left(r_0 e^L\right) - \ln r_0 \\
&= \ln r_0 + \ln\left(e^L\right) - \ln r_0 \\
&= L, \qquad (2.84)
\end{aligned}$$

and when $k \neq l$

$$\begin{aligned}
\int_{r_0}^{r_0 e^L} \rho^{j2\pi(k-l)-1} d\rho &= \frac{1}{j2\pi(k-l)} \rho^{j2\pi(k-l)} \Big|_{\rho=r_0}^{r_0 e^L} \\
&= \frac{1}{j2\pi(k-l)} \left[\left(r_0 e^L\right)^{j2\pi(k-l)} - (r_0)^{j2\pi(k-l)}\right] \\
&= \frac{1}{j2\pi(k-l)} \left[e^{j2\pi(k-l)L} (r_0)^{j2\pi(k-l)} - (r_0)^{j2\pi(k-l)}\right] \\
&= 0 \qquad (2.85)
\end{aligned}$$

noting that $(k-l)L$ is an integer and $e^{j2\pi\tau} = 1$ for any integer $\tau$.

When an input image is scaled by $\beta > 0$, i.e., $\hat{x}(m,n) = x(\beta m, \beta n)$ the Fourier transform is given by $\hat{X}(u,v) = \frac{1}{\beta^2} X\left(\frac{u}{\beta}, \frac{v}{\beta}\right)$ and the polar 2-D FT is given by

$$\hat{X}(\rho, \phi) = \frac{1}{\beta^2} X\left(\frac{\rho}{\beta}, \phi\right) \qquad (2.86)$$

with MRHs

$$\begin{aligned}
\hat{X}_k(\phi) &= L^{-1} \int_{r_0}^{R} \hat{X}(\rho, \phi) \rho^{-j2\pi k - 1} \rho d\rho \\
&= L^{-1} \int_{r_0}^{R} \frac{1}{\beta^2} X\left(\frac{\rho}{\beta}, \phi\right) \rho^{-j2\pi k - 1} \rho d\rho \\
&= L^{-1} \int_{\frac{r_0}{\beta}}^{\frac{R}{\beta}} \frac{1}{\beta^2} X(\tilde{\rho}, \phi) (\beta\tilde{\rho})^{-j2\pi k - 1} (\beta\tilde{\rho}) \beta d\tilde{\rho} \\
&= \beta^{-j2\pi k - 1} L^{-1} \int_{\frac{r_0}{\beta}}^{\frac{R}{\beta}} X(\tilde{\rho}, \phi) \tilde{\rho}^{-j2\pi k - 1} \tilde{\rho} d\tilde{\rho} \\
&\approx \beta^{-1} \beta^{-j2\pi k} X_k(\phi), \qquad (2.87)
\end{aligned}$$

where $\tilde{\rho} = \frac{\rho}{\beta}$. If $\beta$ is a large value, $R$ is chosen to be large enough so that $\frac{R}{\beta}$ still encompasses the radial bandwidth of $X(\rho, \phi)$ and the approximation is valid. The correlation peak using this scaled image is (see Eq. 2.81)

$$\begin{aligned} g(\beta) &= \beta^{-1} L \sum_{k=-\infty}^{\infty} C_k \beta^{-j2\pi k} \\ &= \beta^{-1} L \sum_{k=-\infty}^{\infty} C_k e^{\ln\left(\beta^{-j2\pi k}\right)} \\ &= \beta^{-1} L \sum_{k=-\infty}^{\infty} C_k e^{-j2\pi k \ln \beta}. \end{aligned} \tag{2.88}$$

Selecting only one MRH weight gives

$$\tilde{g}(\beta) = \beta^{-1} L C_M e^{-j2\pi M \ln \beta}. \tag{2.89}$$

Then, a scale change results in only an additional phase factor in the correlation output. The relative intensity distribution (e.g., using PCE or PSR–see Section 7.2.5) remains unaffected. Therefore the filter is invariant to shifts and scales. However, if only one MRH weight is nonzero then all but one of the terms of the Fourier expansion are ignored which leads to poor discrimination. The novelty of MACE-MRH is that it provides a tradeoff between discrimination and scale-control.

If a desired scale response $g_\beta$ is specified for different $\beta$ values then the problem becomes finding the $C_k$ that produces such a response. This can be accomplished by taking advantage of finite impulse response (FIR) design methods. For this purpose, an invertible logarithmic transformation $\mathcal{L}$ is defined as follows,

$$g_{\mathcal{L}}(\beta) = \mathcal{L}\{g(\beta)\} = \frac{1}{L} e^{\frac{L\beta}{2\pi}} g\left(e^{\frac{L\beta}{2\pi}}\right). \tag{2.90}$$

Applying it to Eq. 2.88 yields

$$\begin{aligned} g_{\mathcal{L}}(\beta) &= \frac{1}{L} e^{\frac{L\beta}{2\pi}} \left(e^{\frac{L\beta}{2\pi}}\right)^{-1} L \sum_{z=-\infty}^{\infty} C_z e^{-j2\pi z \ln\left(e^{\frac{L\beta}{2\pi}}\right)} \\ &= \sum_{z=-\infty}^{\infty} C_z e^{-j\beta(zL)} \\ &= \sum_{k=-\infty}^{\infty} C_k e^{-j\beta k} \\ &\approx \sum_{k=-K}^{K} C_k e^{-j\beta k}, \end{aligned} \tag{2.91}$$

where $k = zL$. Eq. 2.91 is equivalent to the FIR filter design problem. There are excellent FIR designs methods presented elsewhere [49] that can be used to find the coefficients $C_k$.

Once the coefficients $C_k$ are computed, Eq. 2.82 can be used to solve for the MRHs $H_k(\phi)$. There are infinite solutions, and therefore other constraints such as ONV, ACE and ASM can be included to find a unique solution.

The MACE-MRH filter uses the ACE criterion as follows. Let $P_X(\phi)$ represent the average of $X(\rho, \phi)$ along the radial axis (from $r_0$ to $R$). Then the ACE can be computed as

$$\begin{aligned} \text{ACE} &= \int_{r_0}^{R} \int_{0}^{2\pi} P_X(\phi) |H(\rho, \phi)|^2 \rho d\phi d\rho \\ &= \int_{0}^{2\pi} P_X(\phi) \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} H_k(\phi) H_l^*(\phi) \int_{r_0}^{R} \rho^{j2\pi(k-l)-2} \rho d\rho d\phi \\ &= L \sum_{k=-\infty}^{\infty} \int_{0}^{2\pi} P_X(\phi) |H_k(\phi)|^2 d\phi. \end{aligned} \tag{2.92}$$

The MRH $H_k(\phi)$ is obtained by solving

$$\begin{aligned} \min_{H_k(\phi)} \quad & \int_{0}^{2\pi} (P_X(\phi) + \alpha) |H_k(\phi)|^2 d\phi \\ s.t. \quad & \int_{0}^{2\pi} X_k(\phi) H_k^*(\phi) d\phi = C_k. \end{aligned} \tag{2.93}$$

The regularization parameter $\alpha$ is added in case $P_X(\phi) = 0$ for some $\phi$. This is the equivalent to including the ONV criterion and assuming that the power spectrum of the noise along the radial axis is $P_n(\phi) = \alpha \; \forall \phi$. The solution to Eq. 2.93 is

$$H_k(\phi) = \lambda_k^* \frac{X_k(\phi)}{P_X(\phi) + \alpha}, \tag{2.94}$$

where

$$\lambda_k = \frac{C_k}{\int_0^{2\pi} \frac{|X_k(\phi)|^2}{P_X(\phi)+\alpha} d\phi}. \tag{2.95}$$

## 2.14 Distance Classifier Correlation Filter (DCCF)

The DCCF for two-class classification was introduced in 1993 [43] and extended to multiclass in 1996 [44]. The DCCF transforms the images for each class so that transformed images of each class are compact and separated from the other classes. In other words, the goal is to maximize the distance between transformed class means and minimize the transformed in-class scatter. The formulation is similar to linear discriminant analysis (LDA) when using the frequency domain of the images; the difference being in how the in-class scatter is computed. LDA minimizes the scatter of the projected (a single value) in-class images while DCCF minimizes the scatter of the transformed (an entire plane) in-class images.

This filter does not appear to be widely used. There are two problems with DCCF. First, unlike other CFs, DCCF is not designed to provide a sharp peak at the target's location. This reduces the shift-invariance advantage of CFs. In their experiments the authors used centered images so no localization (i.e., shift-invariance) performance is given. Second, the metric used during training is not the metric used for testing. That is, DCCF is trained to maximize the squared difference of the projected class means (or correlation peak values), but in testing the entire correlation plane is used. The authors claim that by making the in-class correlation planes similar, then the in-class peak values should be similar to each other. However, overall consistency in the correlation planes does not necessarily mean consistency in the correlation peak values. The DCCF may be better used as a post-localization stage to classify a target between a group of already localized targets.

### 2.14.1 Derivation

The goal is to maximize the separation between two classes. Under transform $\mathbf{h}$, the squared difference between two class means is

$$\begin{aligned} A(\mathbf{h}) &= \frac{1}{d}|\mathbf{h}^\dagger \bar{\mathbf{x}}_1 - \mathbf{h}^\dagger \bar{\mathbf{x}}_2|^2 \\ &= \frac{1}{d}\mathbf{h}^\dagger(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^\dagger \mathbf{h} \\ &= \mathbf{h}^\dagger \mathbf{S}_A \mathbf{h}, \end{aligned} \tag{2.96}$$

where $\bar{\mathbf{x}}_c$ represents the mean image for Class $c$, and $\mathbf{S}_A = \frac{1}{d}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^\dagger$ (note that $\mathbf{S}_A$ is *not* a diagonal matrix). For $C$ classes, where $C > 2$, $A(\mathbf{h})$ is the sum of the squared differences between the transformed mean of each class and the transformed global mean, i.e.,

$$\begin{aligned} A(\mathbf{h}) &= \frac{1}{d}\sum_{c=1}^{C}|\mathbf{h}^\dagger \bar{\mathbf{x}}_c - \mathbf{h}^\dagger \bar{\mathbf{x}}|^2 \\ &= \frac{1}{d}\mathbf{h}^\dagger\left(\sum_{c=1}^{C}(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^\dagger\right)\mathbf{h} \\ &= \mathbf{h}^\dagger \mathbf{S}_A \mathbf{h}, \end{aligned} \tag{2.97}$$

where $\mathbf{S}_A = \frac{1}{d}\sum_{c=1}^{C}(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^\dagger$, and $\bar{\mathbf{x}}$ represents the global mean.

In addition, the in-class scatter is minimized. For this purpose, the in-class scatter can be represented by adding the ASM (see Eq. 2.33) over all classes, i.e.,

$$\begin{aligned} B(\mathbf{h}) &= \sum_{c=1}^{C}\sum_{i=1}^{N}\frac{1}{d}\mathbf{h}^\dagger(\mathbf{X}_{ic} - \bar{\mathbf{X}}_c)(\mathbf{X}_{ic} - \bar{\mathbf{X}}_c)^*\mathbf{h} \\ &= \frac{1}{d}\mathbf{h}^\dagger\left(\sum_{c=1}^{C}\sum_{i=1}^{N}(\mathbf{X}_{ic} - \bar{\mathbf{X}}_c)(\mathbf{X}_{ic} - \bar{\mathbf{X}}_c)^*\right)\mathbf{h} \\ &= \mathbf{h}^\dagger \mathbf{S}_B \mathbf{h}, \end{aligned} \tag{2.98}$$

where diagonal matrix $\mathbf{S}_B = \frac{1}{d}\sum_{c=1}^{C}\sum_{i=1}^{N}(\mathbf{X}_{ic} - \bar{\mathbf{X}}_c)(\mathbf{X}_{ic} - \bar{\mathbf{X}}_c)^*$, diagonal matrix $\mathbf{X}_{ic}$ contains the 2-D DFT of the $i$th image of Class $c$ along its diagonal, and diagonal matrix $\bar{\mathbf{X}}_c$ contains $\bar{\mathbf{x}}_c$

along its diagonal. For comparison, in LDA under transform $\mathbf{h}$ the in-class scatter is

$$\begin{aligned}\sigma_c^2 &= \frac{1}{d}\sum_{i=1}^{N}|\mathbf{h}^\dagger\mathbf{x}_{ic} - \mathbf{h}^\dagger\bar{\mathbf{x}}_c|^2 \\ &= \mathbf{h}^\dagger\left(\frac{1}{d}\sum_{i=1}^{N}(\mathbf{x}_{ic}-\bar{\mathbf{x}}_c)(\mathbf{x}_{ic}-\bar{\mathbf{x}}_c)^\dagger\right)\mathbf{h}, \qquad (2.99)\end{aligned}$$

where $\mathbf{x}_{ic}$ represents the $i$th image in Class $c$. The total scatter in LDA can be represented as the sum of each in-class variance, i.e.,

$$\begin{aligned}C(\mathbf{h}) &= \sum_{c=1}^{C}\sigma_c^2 \\ &= \mathbf{h}^\dagger\left(\frac{1}{d}\sum_{c=1}^{C}\sum_{i=1}^{N}(\mathbf{x}_{ic}-\bar{\mathbf{x}}_c)(\mathbf{x}_{ic}-\bar{\mathbf{x}}_c)^\dagger\right)\mathbf{h} \\ &= \mathbf{h}^\dagger\mathbf{S}_C\mathbf{h}, \qquad (2.100)\end{aligned}$$

where $\mathbf{S}_C = \frac{1}{d}\sum_{c=1}^{C}\sum_{i=1}^{N}(\mathbf{x}_{ic}-\bar{\mathbf{x}}_c)(\mathbf{x}_{ic}-\bar{\mathbf{x}}_c)^\dagger$ (note that $\mathbf{S}_C$ is *not* a diagonal matrix). It is worth noting that the diagonal elements of $\mathbf{S}_C$ are the same as the diagonal elements of $\mathbf{S}_B$. Also if the number of dimensions $d$ is greater than $N - c$ (this is usually the case in most CF problems) then $\mathbf{S}_C$ is singular, whereas $\mathbf{S}_B$ is non-singular.

In order to maximize $A(\mathbf{h})$ and minimize $B(\mathbf{h})$ simultaneously, the ratio

$$J(\mathbf{h}) = \frac{A(\mathbf{h})}{B(\mathbf{h})} = \frac{\mathbf{h}^\dagger\mathbf{S}_A\mathbf{h}}{\mathbf{h}^\dagger\mathbf{S}_B\mathbf{h}} \qquad (2.101)$$

is maximized with respect to $\mathbf{h}$. The optimum solution (see Appendix B) is the dominant eigenvector of $\mathbf{S}_B^{-1}\mathbf{S}_A$.

To test a test chip $\mathbf{z}$, the squared difference between the transformed image and the transform of each class mean is computed, i.e.,

$$\begin{aligned}
l(\mathbf{z}) &= \min_c |\mathbf{H}^*\mathbf{z} - \mathbf{H}^*\bar{\mathbf{x}}_c|^2 \\
&= \min_c \left(\bar{\mathbf{x}}_c^\dagger \mathbf{H}\mathbf{H}^*\bar{\mathbf{x}}_c - 2\mathbf{z}^\dagger \mathbf{H}\mathbf{H}^*\bar{\mathbf{x}}_c + \mathbf{z}^\dagger \mathbf{H}\mathbf{H}^*\mathbf{z}\right) \\
&= \min_c \left(\bar{\mathbf{x}}_c^\dagger \mathbf{H}\mathbf{H}^*\bar{\mathbf{x}}_c - 2\mathbf{z}^\dagger \mathbf{H}\mathbf{H}^*\bar{\mathbf{x}}_c\right) \\
&= \min_c \left(b_c - \mathbf{z}^\dagger \mathbf{h}_c\right) \\
&= \max_c (\mathbf{z}^\dagger \mathbf{h}_c - b_c)
\end{aligned} \tag{2.102}$$

where $l(\mathbf{z})$ represents the class-label of image $\mathbf{z}$, diagonal matrix $\mathbf{H}$ contains $\mathbf{h}$ along its diagonal, $b_c = \bar{\mathbf{x}}_c^\dagger \mathbf{H}\mathbf{H}^*\bar{\mathbf{x}}_c$, and $\mathbf{h}_c = 2\mathbf{H}\mathbf{H}^*\bar{\mathbf{x}}_c$. To test an image $\mathbf{z}$ larger than the training image, the spatial domain representation of $\mathbf{z}$ and $\mathbf{h}_c$ is cross-correlated, the scalar $b_c$ is subtracted from the entire correlation plane (i.e., from each value in the plane), and then the maximum value for all the classes $c$ is computed.

### 2.14.2 Extensions

Anwaar-ul-Haq et al. [5] extended the DCCF to 3-D for action recognition. They show superior performance over the Action MACH filter.

## 2.15 Polynomial Correlation Filter (PCF)

PCFs were introduced in 1997 [40]. The PCF output is a nonlinear function of the input that can enhance recognition performance. The application of this filter can easily be extended to data fusion from multiple sensors.

### 2.15.1 Derivation

The PCF output can be expressed as

$$g(m,n) = \sum_{p=1}^{P} h_p(m,n) \otimes x^p(m,n), \tag{2.103}$$

where $x^p(m,n)$ represents the elements of image $x(m,n)$ raised to the $p$th power. This is expressed in the frequency domain as

$$\mathbf{g} = \sum_{p=1}^{P} \mathbf{X}^{p*}\mathbf{h}_p, \tag{2.104}$$

where diagonal matrix $\mathbf{X}^p$ contains $\mathbf{x}^p$ along its diagonal. The goal is to find corresponding filters $\mathbf{h}_p$ that maximize some objective, e.g., to maximize the average peak value at the origin and minimize some other metric (e.g., ASM, ONV, ACE). The authors used ASM in their paper, i.e., maximize

$$\begin{aligned} J(\mathbf{h}) &= \frac{|\mathbf{h}^\dagger \hat{\mathbf{x}}|^2}{\mathbf{h}^\dagger \mathbf{S}\mathbf{h}} \\ &= \frac{\left|\sum_{p=1}^{P} \mathbf{h}_p^\dagger \bar{\mathbf{x}}^p\right|^2}{\sum_{p=1}^{P}\sum_{q=1}^{P} \mathbf{h}_p^\dagger \mathbf{S}_{pq}\mathbf{h}_q}, \end{aligned} \tag{2.105}$$

where

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_P \end{bmatrix}, \hat{\mathbf{x}} = \begin{bmatrix} \bar{\mathbf{x}}^1 \\ \vdots \\ \bar{\mathbf{x}}^P \end{bmatrix} \tag{2.106}$$

is a vector form by concatenating a series of vectors that represent the mean image raised to various powers,

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \cdots & \mathbf{S}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{S}_{P1} & \cdots & \mathbf{S}_{PP} \end{bmatrix} \tag{2.107}$$

is a block matrix of diagonal matrices, and $\mathbf{S}_{pq} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_i^p - \bar{\mathbf{X}}^p)(\mathbf{X}_i^q - \bar{\mathbf{X}}^q)^*$. The solution that maximizes the Rayleigh Quotient $J(\mathbf{h})$ is (see Appendix B)

$$\mathbf{h} = \mathbf{S}^{-1}\hat{\mathbf{x}}. \tag{2.108}$$

This method is not restricted to power nonlinearities. It can use any point nonlinear function $f_p(x(m,n))$, i.e.,

$$g(m,n) = \sum_{p=1}^{P} h_p(m,n) \otimes f_p(x(m,n)). \tag{2.109}$$

The solution is the same as Eq. 2.108 with

$$\hat{\mathbf{x}} = \begin{bmatrix} f_1(\bar{\mathbf{x}}) \\ \vdots \\ f_P(\bar{\mathbf{x}}) \end{bmatrix}, \tag{2.110}$$

where $f_p(\bar{\mathbf{x}})$ is $f_p(\bar{x}(m,n))$ arranged as a vector, and

$$\mathbf{S}_{pq} = \frac{1}{N}\sum_{i=1}^{N}\left(f_p(\mathbf{X}_i) - f_p(\bar{\mathbf{X}})\right)\left(f_q(\mathbf{X}_i) - f_q(\bar{\mathbf{X}})\right)^*. \tag{2.111}$$

This extension can be used for sensor fusion by treating the data processed from different sensors as a nonlinear transformation.

### 2.15.2 Extensions

Alkanhal and Kumar [3] combined ideas from PCF and DCCF to form the Polynomial DCCF (PDCCF). It is a DCCF design but uses a series of images transformed by a function as shown in Eq. 2.109. They present an extensive analysis of PDCCF, but because they used centered images, their experiments fail to show any localization (shift-invariance) performance. A comparative study by Singh and Kumar [75] showed that PDCCF outperformed DCCF in almost all cases in their synthetic aperture radar (SAR) data experiments.

Al-Mashouq, Kumar, and Alkanhal [1] presented the Constrained PCF (CPCF) that constrains the values at the peak. In addition an analysis is presented suggesting that using the power functions in Eq. 2.103 helps recognition performance for low values of $p$. As $p$ increases pass a given value the contributions to the total peak from the $p$th order filter decreases and the correlation peak variance due to clutter increases. They suggest that the highest order should be restricted to $p = 4$.

## 2.16 QUADRATIC CORRELATION FILTER (QCF)

QCFs were introduced in 2004 [37]. A disadvantage of linear CFs is that several CFs are required to handle the wide variety of target appearances. When these CFs are applied to a test image, their outputs are compared to select a winner. A QCF requires several linear CFs as well but has the advantage that these CFs are designed to work together to produce a single correlation output. In addition, quadratic classifiers are able to exploit the higher-order statistics of the data, potentially leading to superior recognition performance. In one set of experiments using gray-scale images, QCFs were shown to outperform other CFs [33].

### 2.16.1 Derivation

QCF maximizes a metric of separation between the overall outputs for two classes of targets. To express this separation metric, let $\check{\mathbf{x}}^{(c)}$ (all the notation in this derivation is in the spatial domain) be a vectorized image from Class $c$ where $c \in \{1, 2\}$ with $d$ elements and let $\mathbf{Q}$ be a $d \times d$ matrix. The goals is to make $y = \left(\check{\mathbf{x}}^{(c)}\right)^T \mathbf{Q}\check{\mathbf{x}}^{(c)}$ large and positive when $c = 1$ (i.e., when the target is from Class 1) and large and negative when $c = 2$ (i.e., when the target is from Class 2). Let

$$\phi_c = E\left\{\left(\check{\mathbf{x}}^{(c)}\right)^T \mathbf{Q}\check{\mathbf{x}}^{(c)}\right\} \tag{2.112}$$

indicate the mean QCF output for all training images from Class $c$. In the most basic QCF design, $\mathbf{Q}$ is computed such that

$$J(\mathbf{Q}) = |\phi_1 - \phi_2| \tag{2.113}$$

is a large value. In other words, a large between-class separation is desired.

To enable a correlation-type QCF implementation architecture, $\mathbf{Q}$ is assumed to be of the form

$$\mathbf{Q} = \sum_{i=1}^{N_1} \check{\mathbf{f}}_i \check{\mathbf{f}}_i^T - \sum_{i=1}^{N_2} \check{\mathbf{b}}_i \check{\mathbf{b}}_i^T. \tag{2.114}$$

It can be shown [37] that $J(\mathbf{Q})$ is maximized for a given $N_1$ and $N_2$ when $\check{\mathbf{f}}_1, \cdots, \check{\mathbf{f}}_{N_1}$ and $\check{\mathbf{b}}_1, \cdots, \check{\mathbf{b}}_{N_2}$ are the eigenvectors corresponding to the $N_1$ largest positive and the $N_2$ largest negative eigenvalues, respectively, of

$$\mathbf{R} = \mathbf{R}_1 - \mathbf{R}_2, \tag{2.115}$$

where

$$\mathbf{R}_c = E\left\{\check{\mathbf{x}}_i^{(c)}\left(\check{\mathbf{x}}_i^{(c)}\right)^T\right\} \tag{2.116}$$

is the correlation matrix for all the training images in Class $c$.

To show how to apply the QCF to a test image $x(m,n)$, first let test chip $\check{\mathbf{z}}$ denote a vectorized subregion (equal in size to a training image) of $x(m,n)$ with QCF output

$$\begin{aligned} y &= \check{\mathbf{z}}^T\mathbf{Q}\check{\mathbf{z}} \\ &= \check{\mathbf{z}}^T\left(\sum_{i=1}^{N_1}\check{\mathbf{f}}_i\check{\mathbf{f}}_i^T\right)\check{\mathbf{z}} - \check{\mathbf{z}}^T\left(\sum_{i=1}^{N_2}\check{\mathbf{b}}_i\check{\mathbf{b}}_i^T\right)\check{\mathbf{z}} \\ &= \check{\mathbf{z}}^T\mathbf{F}\mathbf{F}^T\check{\mathbf{z}} - \check{\mathbf{z}}^T\mathbf{B}\mathbf{B}^T\check{\mathbf{z}} \\ &= \check{\mathbf{v}}^T\check{\mathbf{v}} - \check{\mathbf{w}}^T\check{\mathbf{w}}, \end{aligned} \tag{2.117}$$

where $\mathbf{F} = [\check{\mathbf{f}}_1,\ldots,\check{\mathbf{f}}_{N_1}]$, $\mathbf{B} = [\check{\mathbf{b}}_1,\ldots,\check{\mathbf{b}}_{N_2}]$, $\check{\mathbf{v}} = \mathbf{F}^T\check{\mathbf{z}}$, and $\check{\mathbf{w}} = \mathbf{B}^T\check{\mathbf{z}}$. The value of $v_i$ (the $i$th element of $\check{\mathbf{v}}$) can be obtained for all locations within the image $x(m,n)$ by means of 2-D correlation, i.e.,

$$v_i(m,n) = x(m,n) \otimes f_i(m,n), \tag{2.118}$$

where the eigenfilter $f_i(m,n)$ is the $\check{\mathbf{f}}_i$ eigenvector reshaped as 2-D filter. A similar derivation can be shown for $w_i$ (the $i$th element of $\check{\mathbf{w}}$). Fig. 2.2 shows the architecture to apply the QCF to image $x(m,n)$. The QCF output $g(m,n)$ is

$$\begin{aligned} g(m,n) &= \sum_{i=1}^{N_1} v_i^2(m,n) - \sum_{i=1}^{N_2} w_i^2(m,n) \\ &= \sum_{i=1}^{N_1}(x(m,n)\otimes f_i(m,n))^2 - \sum_{i=1}^{N_2}|(x(m,n)\otimes b_i(m,n))^2, \end{aligned} \tag{2.119}$$

which can be efficiently implemented in the frequency domain as

$$g(m,n) = \sum_{i=1}^{N_1}\left(\mathcal{F}^{-1}\left\{\mathcal{F}\{x(m,n)\}\mathcal{F}^*\{f_i(m,n)\}\right\}\right)^2 - \sum_{i=1}^{N_2}\left(\mathcal{F}^{-1}\left\{\mathcal{F}\{x(m,n)\}\mathcal{F}^*\{b_i(m,n)\}\right\}\right)^2, \tag{2.120}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the DFT and IDFT, respectively. This requires $2(N_1+N_2)+1$ 2-D DFTs (combining bot the DFTs and IDFTs). Note that when the QCF is applied to multiple images

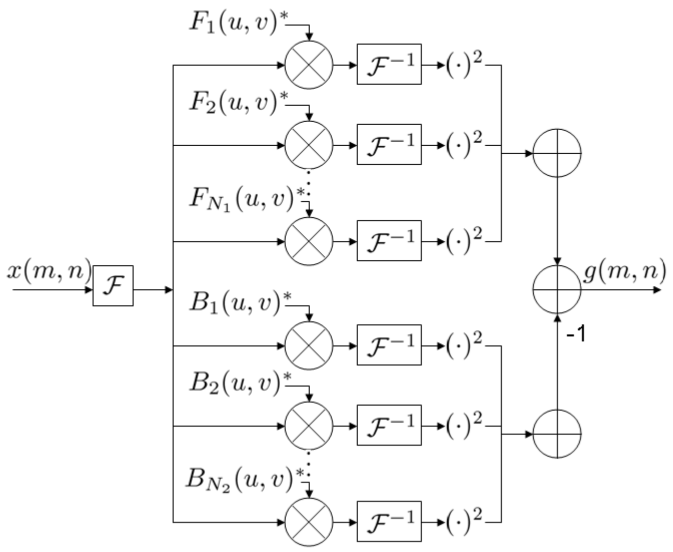


Figure 2.2: Efficient architecture to apply the QCF to image $x(m,n)$.

(e.g., in a video), the 2-D DFTs of the eigenvectors only need to be computed once. Therefore in video processing, QCF requires $N_1 + N_2 + 1$ 2-D DFTs per video frame.

In QCF design the number of eigenfilters affects the recognition performance. As the number of eigenfilters used increases, we observed that in the testing set recognition performances first increases and then decreases. In our experiments we use $N_1 = 7$ and $N_2 = 1$. More training images results in better recognition performance. Some CFs that use the average of the training images for their designs (e.g., the MACH filter [84]) limit the number of training images to prevent the average from becoming a blurry image. However, QCF uses the average of the QCF output (see Eq. 2.112), and therefore over-fitting is not a problem. For example, Mahalanobis et al. [36] used over 20,000 training images (over 5,000 true-class images and over 15,000 false-class images). In our experiments, we use a small number of training images and show that QCF still achieves good recognition performance.

### 2.16.2 Extensions

Mahalanobis et al. [36, 37] proposed the Rayleigh Quotient QCF (RQQCF). This filter is sometimes referred to as the Fukunaga-Koontz QCF (FKQCF) because it can be derived using the Fukunaga-Koontz transform [27]. In addition to maximizing the class separation, it attempts to reduce the class-scatter by equalizing the average output from Class 1 with the average negative output of Class 2, i.e., $E\left\{\left(\check{\mathbf{x}}^{(1)}\right)^T \mathbf{Q}\check{\mathbf{x}}^{(1)}\right\} = -E\left\{\left(\check{\mathbf{x}}^{(2)}\right)^T \mathbf{Q}\check{\mathbf{x}}^{(2)}\right\}$ (see Eq. 2.112). However, note that there is no theoretical justification that this will reduce the class-scatter, and in fact it may reduce performance by emphasizing this criterion over maximum class separation. This objective can be accomplished by rewriting the objective function in Eq. 2.113 as

$$J(\mathbf{Q}) = \frac{|\phi_1 - \phi_2|}{\phi_1 + \phi_2}. \tag{2.121}$$

The solution is the same as Eq. 2.114 with $\check{\mathbf{f}}_1, \cdots, \check{\mathbf{f}}_{N_1}$ and $\check{\mathbf{b}}_1, \cdots, \check{\mathbf{b}}_{N_2}$ as the eigenvectors corresponding to the $N_1$ largest positive and $N_2$ largest negative eigenvalues, respectively, of $(\mathbf{R}_1 + \mathbf{R}_2)^{-1}(\mathbf{R}_1 - \mathbf{R}_2)$ (see Eq. 2.116). Sims and Mahalanobis [73] showed that RQQCF outperforms the MACH filter when tested with infrared imagery.

Muise et al. [47] proposed the Subspace Quadratic Synthetic Discriminant Function (SSQSDF). The SSQSDF matrix assumes the form

$$\mathbf{Q} = \sum_{i=1}^{N} \beta_i \check{\mathbf{x}}_i \check{\mathbf{x}}_i^T. \tag{2.122}$$

The outputs $\check{\mathbf{x}}_i^T \mathbf{Q} \check{\mathbf{x}}_i$ are constrained to be $1$ for true-class training images and $0$ for false-class training images. This helps to reduce the correlation peak variance (a challenge in the RQQCF design). Experiments show similar recognition performance between SSQSDF and QCF [33, 73].

## 2.17 Multi-Frame Correlation Filter (MFCF)

MFCF was introduced in 2009 [34]. The MFCF combines information from the current correlation output with previous correlation outputs to enhance recognition in a Bayesian framework. Each correlation value is mapped to a probability value, i.e., the probability that there is a true-class target centered at the location given the correlation value at that location. The array of probability values is convolved with a circular Gaussian motion model to get a prior probability array to use for the next mapping. Using a circular Gaussian motion model assumes that the target can move in any direction (e.g., Brownian motion). It works well for situations where the targets move by only a few pixels from frame to frame.

### 2.17.1 Derivation

The MFCF is based on a Bayesian framework where the filter's output is mapped into a probability array,

$$P\left(g_i(m,n)|T_i(m,n)\right), \tag{2.123}$$

where $g_i(m,n)$ represents the correlation output in response to the $i$th image, and $T_i(m,n)$ denotes the event that a target is centered at pixel $(m,n)$ and $\hat{T}_i(m,n)$ denotes the event that there is *no* target centered at $(m,n)$. Using training examples, the distributions of the likelihoods $P(g_i(m,n)|T_i(m,n))$ and $P(g_i(m,n)|\hat{T}_i(m,n))$ can be estimated. The posterior probability is computed as follows,

$$P(T_i(m,n)|g_i(m,n)) = \frac{P(g_i(m,n)|T_i(m,n))P(T_i(m,n))}{P(g_i(m,n)|T_i(m,n))P(T_i(m,n)) + P(g_i(m,n)|\hat{T}_i(m,n))P(\hat{T}_i(m,n))} \tag{2.124}$$

where $P(T_i(m,n)|g_i(m,n))$ represents the probability that a target is centered at location $(m,n)$ given a correlation value at that location at frame $i$, $P(T_i(m,n))$ represents the prior probability that a target is centered at location $(m,n)$, and $P(\hat{T}(m,n)) = 1 - P(T(m,n))$.

Next, a fixed 2-D Gaussian $\tau(m,n) \sim \mathcal{N}(0,\sigma^2)$ is correlated with the posterior to estimate the prior probability of the object at any location in the next frame,

$$P(T_{i+1}(m,n)) = P(T_i(m,n)|g_i(m,n)) \otimes \tau(m,n). \tag{2.125}$$

This is a simple motion model and it implies that the target may move in any direction with equal probability, with decreasing probability of moving by larger amounts. The standard deviation of $\tau(m,n)$ is proportional to the maximum velocity (in pixels/frame) of the target.

As successive input frames become available, the probability array is updated using Bayes theorem to reflect the probability based on both the estimates and the observed data. It was shown [34] that this Bayesian approach improves the recognition performance of conventional CFs by suppressing false alarms and improving the ability to recognize targets in the presence of clutter.

In addition to computing the correlation plane (and possibly the PSR plane), the MFCF requires two additional 2-D DFTs for the required convolution with the 2-D circular Gaussian. Note that other motion models can be used instead of Gaussian.

### 2.17.2 Extensions

Mahalanobis, Stanfill, and Chen [39] used MMCF with two motion models (instead of one), one for stationary and another for non-stationary objects. They tested their models with walking humans and showed that MMCF outperformed single-frame correlation filtering even when using a tracker.

## 2.18 SUMMARY

In this chapter we presented the advances in CF design over the past three decades. We presented different linear CFs that are constrained to have a certain peak value for training images, and discussed different linear unconstrained filters that do not have these constraints and have a larger solution space to find a filter that may have better recognition performance. We presented CFs that in addition to shift-invariance have rotation-control or scale-control. We presented non-linear filters that may have better recognition performance at the expense of additional computation. We concluded by presenting a recent CF design used to enhance performance in a sequence of images, i.e., in video.

CHAPTER 3

# CORRELATION FILTER IMPROVEMENTS

In the previous chapter we reviewed different CF designs. In this chapter we present two generalized linear CFs that encompass all of the state-of-the-art linear CFs discussed in Chapter 2. We provide some experimental results comparing our generalized linear CFs to those already discussed. In addition, we present a method to enhance the performance of quadratic CFs (QCFs) both in computational speed and performance, and show experimental results. We conclude by presenting a design that improves recognition performance when using CFs in a sequence of images, e.g., in video.

Before proceeding, we recall from Section 1.4 that lower case bold letters denote vectors (lexicographically arranged) of 2-D frequency domain arrays, and that we loosely refer to $\mathbf{x}_i$ or $\mathbf{g}_i$ as the $i$th image or the $i$th correlation output, respectively, when we mean the vectorized 2-D DFT representation of the $i$th image or the $i$th correlation output, respectively. Also note that by Parseval's theorem, the inner products in the frequency domain are scaled by $\frac{1}{d}$ [50].

## 3.1 GENERALIZED UNCONSTRAINED AND CONSTRAINED CORRELATION FILTERS

Using the criteria that are generally used in state-of-the-art linear CFs, we can design a generalized Unconstrained Correlation Filter (UCF) and a generalized Constrained Correlation Filter

(CCF), and show how linear composite CFs are subsets of these filters. The criteria used in linear CFs are summarized as follows.

1. The mean square error (MSE) is the average squared difference between the actual correlation plane $\mathbf{g}_i$ and the desired correlation plane $\dot{\mathbf{g}}_i$, i.e.,

$$\begin{aligned}\text{MSE} &= \frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \dot{\mathbf{g}}_i|^2 \\ &= \frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{g}_i^\dagger\mathbf{g}_i - 2\mathbf{g}_i^\dagger\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i) \\ &= \frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{h}^\dagger\mathbf{X}_i\mathbf{X}_i^*\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{X}_i\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i) \\ &= \mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*\right)\mathbf{h} - 2\mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i\right) + \frac{1}{Nd}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i \\ &= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + E_f, \end{aligned} \tag{3.1}$$

where $\mathbf{g}_i = \mathbf{X}_i^*\mathbf{h}$ represents the correlation output for the $n$th image, $\mathbf{D} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*$ is the average power spectral density of the $N$ training images, $E_f = \frac{1}{Nd}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i$ is the average energy of the desired correlation planes, and $\mathbf{h}^\dagger\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{g}_i^\dagger\dot{\mathbf{g}}_i$ is a measurement of the similarity of the actual correlation planes to the desired correlation planes where

$$\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i. \tag{3.2}$$

The average correlation energy (ACE) criterion $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ (see Eq. 2.13) is a special case of the MSE criterion when $\dot{\mathbf{g}}_i = \mathbf{0}$. Minimizing the ACE usually produces correlation planes with values near zero (except at the location of the target due to the constraints) making the peak (i.e., the location of the target) more distinguishable.

2. The average similarity measure (ASM) is a measure of the scatter (similar to the variance but for an entire plane) in the correlation planes, i.e., $\frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \bar{\mathbf{g}}|^2$, where $\bar{\mathbf{g}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i$ represents the average of the correlation planes. In other words, ASM measures how dissimilar the correlation planes are with respect to the mean of the correlation planes. Note the

difference between ASM and ACE. Minimizing the ASM makes the correlation planes more similar. Minimizing the ACE will minimize the energy in the correlation (on the average), but there could be significant differences between the responses of each training image. The ASM can be written as

$$\begin{aligned}
\text{ASM} &= \frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \bar{\mathbf{g}}|^2 \\
&= \frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{g}_i^\dagger\mathbf{g}_i - 2\mathbf{g}_i^\dagger\bar{\mathbf{g}} + \bar{\mathbf{g}}^\dagger\bar{\mathbf{g}}) \\
&= \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{g}_i^\dagger\mathbf{g}_i - \frac{2}{d}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^\dagger\right)\bar{\mathbf{g}} + \frac{1}{d}\bar{\mathbf{g}}^\dagger\bar{\mathbf{g}} \\
&= \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{g}_i^\dagger\mathbf{g}_i - \frac{1}{d}\bar{\mathbf{g}}^\dagger\bar{\mathbf{g}} \\
&= \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{h}^\dagger\mathbf{X}_i\mathbf{X}_i^*\mathbf{h} - \frac{1}{d}\mathbf{h}^\dagger\bar{\mathbf{X}}\bar{\mathbf{X}}^*\mathbf{h}, \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - \mathbf{h}^\dagger\mathbf{M}\mathbf{h}, \qquad (3.3)
\end{aligned}$$

where the diagonal matrix $\bar{\mathbf{X}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i$ contains $\bar{\mathbf{x}}$ along the diagonal, and the diagonal matrix $\mathbf{M} = \frac{1}{d}\bar{\mathbf{X}}\bar{\mathbf{X}}^*$ contains the power spectral density of the $\bar{\mathbf{x}}$ along the diagonal.

3. The output noise variance (ONV) is a measure of the variance of the correlation plane pixels caused by additive noise in the images. In the space domain this is defined as

$$\begin{aligned}
\text{ONV} &= \sigma^2 \\
&= Var\{\check{\mathbf{h}}^T(\check{\mathbf{x}}_i^{(n\downarrow)} + \check{\boldsymbol{\eta}})\} \\
&= Var\{\check{\mathbf{h}}^T\check{\boldsymbol{\eta}}\} \\
&= Var\{\check{\mathbf{c}}_{\boldsymbol{\eta}}\}, \qquad (3.4)
\end{aligned}$$

where $\check{\mathbf{x}}_i^{(n\downarrow)}$ represents the vector $\check{\mathbf{x}}_i$ shifted by $n$ pixels, $Var\{\check{\mathbf{h}}^T\check{\mathbf{x}}_i^{(n\downarrow)}\} = 0$ for all shifts $n$, $\check{\boldsymbol{\eta}}$ is a zero-mean wide sense stationary random process. That is, the variance at each pixel caused by additive noise is the same. Note that the ONV is the value at the origin of the autocorrelation function of $\check{\mathbf{c}}_{\boldsymbol{\eta}}$ which can be obtained by summing over all frequencies of $\mathbf{P}_{\mathbf{c}_{\check{\boldsymbol{\eta}}}}$, the power spectral density of $\mathbf{c}_{\check{\boldsymbol{\eta}}}$, i.e.,

$$\begin{aligned} \text{ONV} &= \sigma^2 \\ &= \sum_{k=0}^{d-1} \mathbf{P}_{\mathbf{c}_{\check{\boldsymbol{\eta}}}}[k] \\ &= \mathbf{h}^\dagger \mathbf{P} \mathbf{h}, \end{aligned} \tag{3.5}$$

where $\mathbf{P}$ is the power spectral density of the input noise $\check{\boldsymbol{\eta}}$ and $d$ is the dimension of $\mathbf{h}$. In many scenarios, unit-variance white noise is assumed, i.e., $\mathbf{P} = \frac{1}{d}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

### 3.1.1 Generalized Unconstrained Correlation Filter (UCF)

Based on these criteria, we present the multi-objective generalized Unconstrained Correlation Filter (UCF)

$$\min_{\mathbf{h}} \qquad (\text{MSE, ASM, ONV}) \tag{3.6}$$

where MSE, ONV, ASM $\geq 0$. Refregier [55] (see also [15]) showed that multi-quadratic criteria are optimized (i.e., the solution yields the best for one criterion for a fixed value of the others) as follows. Lagrange multipliers are used to obtain the following functional,

$$\begin{aligned} \mathcal{L}(\mathbf{h}, \gamma, \beta) &= \text{MSE} + \gamma\text{ASM} + \beta\text{ONV} \\ &= \mathbf{h}^\dagger \mathbf{D} \mathbf{h} - 2\mathbf{h}^\dagger \mathbf{p} + E_f + \gamma(\mathbf{h}^\dagger \mathbf{D} \mathbf{h} - \mathbf{h}^\dagger \mathbf{M} \mathbf{h}) + \beta \mathbf{h}^\dagger \mathbf{P} \mathbf{h} \\ &= \mathbf{h}^\dagger((1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P})\mathbf{h} - 2\mathbf{h}^\dagger \mathbf{p} + E_f \\ &= \mathbf{h}^\dagger \mathbf{T} \mathbf{h} - 2\mathbf{h}^\dagger \mathbf{p} + E_f, \end{aligned} \tag{3.7}$$

where $0 \leq \gamma, \beta < \infty$ are scalar Lagrange multipliers, and $\mathbf{T} = (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}$. Note that when $\gamma = 0$ only the MSE and ONV criteria are optimized, when $\beta = 0$ only the MSE and ASM criteria are optimized, and when $\gamma = \beta = 0$ only the MSE criterion is optimized.

Taking the gradient of Eq. 3.7 and setting it to zero gives,

$$\frac{\partial \mathcal{L}(\mathbf{h},\gamma,\beta)}{\partial \mathbf{h}} = 2\mathbf{T}\mathbf{h} - 2\mathbf{p} = \mathbf{0}, \tag{3.8}$$

and solving for $\mathbf{h}$ gives

$$\mathbf{h} \;=\; \mathbf{T}^{-1}\mathbf{p}. \tag{3.9}$$

In order to use bounded scalars, $\gamma$ and $\beta$ can be replaced with $\gamma = \frac{1}{\psi}(1-\psi)$ and $\beta = \frac{1}{\lambda}(1-\lambda)$, respectively, where $0 < \psi, \lambda \leq 1$, i.e.,

$$\begin{aligned}\mathbf{T} &= (1+\frac{1}{\psi}(1-\psi))\mathbf{D} - \frac{1}{\psi}(1-\psi)\mathbf{M} + \frac{1}{\lambda}(1-\lambda)\mathbf{P} \\ &= \frac{1}{\lambda\psi}\left(\lambda\mathbf{D} - \lambda(1-\psi)\mathbf{M} + (1-\lambda)\psi\mathbf{P}\right) \\ &\propto \lambda\mathbf{D} - \lambda(1-\psi)\mathbf{M} + (1-\lambda)\psi\mathbf{P}, \end{aligned} \tag{3.10}$$

where the scale factor $\frac{1}{\lambda\psi}$ can be ignored because all the correlation outputs are scaled accordingly and therefore it does not affect the filter's performance. In other words, the filters $\mathbf{h}$ and $\lambda\psi\mathbf{h}$ have the same recognition performance when they are applied to a test image.

#### *3.1.1.1 Efficient computation*

UCFs can be efficiently trained online as follows. After the $n$th image the filter is computed as follows,

$$\mathbf{h}_n = \frac{1}{n}\mathbf{p}'_n \oslash \left(\frac{1}{n}(1+\gamma)\mathbf{d}_n + \frac{1}{n^2}\gamma\mathbf{s}_n \odot \mathbf{s}^*_n + \beta\right), \tag{3.11}$$

where $\odot$ represents the Hadamard product, $\oslash$ represents the Hadamard division, additive unit-variance white noise for the ONV is assumed, i.e., $\mathbf{P} = \frac{1}{d}\mathbf{I}$, and

$$\mathbf{p}'_n \;=\; \mathbf{x}_n \odot \dot{\mathbf{g}}_n + \mathbf{p}'_{n-1}, \tag{3.12}$$

$$\mathbf{d}_n \;=\; \mathbf{x}_n \odot \mathbf{x}^*_n + \mathbf{d}_{n-1}, \tag{3.13}$$

$$\mathbf{s}_n \;=\; \mathbf{x}_n + \mathbf{s}_{n-1}, \tag{3.14}$$

where the filter is initialized with the values $\mathbf{p}'_0 = \mathbf{d}_0 = \mathbf{s}_0 = 0$. It can be shown that after $N$ iterations, the same filter as in Eq. 3.9 is obtained, i.e., after $N$ iterations,

$$\begin{aligned}
\mathbf{h}_N &= \frac{1}{N}\mathbf{p}'_n \oslash \left(\frac{1}{N}(1+\gamma)\mathbf{d}_n + \frac{1}{N^2}\gamma\mathbf{s}_n \odot \mathbf{s}^*_n + \beta\right) \\
&= \frac{1}{N}\left(\mathbf{x}_n \odot \dot{\mathbf{g}}_n + \mathbf{p}'_{n-1}\right) \oslash \cdots \\
&\quad \left(\frac{1}{N}(1+\gamma)\left(\mathbf{x}_n \odot \mathbf{x}^*_n + \mathbf{d}_{n-1}\right) + \frac{1}{N^2}\gamma\left(\mathbf{x}_n + \mathbf{s}_{n-1}\right) \odot \left(\mathbf{x}_n + \mathbf{s}_{n-1}\right)^* + \beta\right) \\
&= \frac{1}{N}\left(\sum_{n=1}^{N}\mathbf{x}_n \odot \dot{\mathbf{g}}_n\right) \oslash \cdots \\
&\quad \left(\frac{1}{N}(1+\gamma)\left(\sum_{n=1}^{N}\mathbf{x}_n \odot \mathbf{x}^*_n\right) + \frac{1}{N^2}\gamma\left(\sum_{n=1}^{N}\mathbf{x}_n\right) \odot \left(\sum_{n=1}^{N}\mathbf{x}_n\right)^* + \beta\right) \\
&= \left(\frac{1}{Nd}\sum_{n=1}^{N}\mathbf{x}_n \odot \dot{\mathbf{g}}_n\right) \oslash \left((1+\gamma)\left(\frac{1}{Nd}\sum_{n=1}^{N}\mathbf{x}_n \odot \mathbf{x}^*_n\right) + \gamma\frac{1}{d}\bar{\mathbf{x}} \odot \bar{\mathbf{x}}^* + \frac{1}{d}\beta\right) \\
&= \left((1+\gamma)\left(\frac{1}{Nd}\sum_{n=1}^{N}\mathbf{X}_n\mathbf{X}^*_n\right) + \gamma\frac{1}{d}\bar{\mathbf{X}}\bar{\mathbf{X}}^* + \beta\frac{1}{d}\mathbf{I}\right)^{-1}\left(\frac{1}{Nd}\sum_{n=1}^{N}\mathbf{X}_n\dot{\mathbf{g}}_n\right) \\
&= \left((1+\gamma)\mathbf{D} + \gamma\mathbf{M} + \beta\mathbf{P}\right)^{-1}\mathbf{p} \\
&= \mathbf{T}^{-1}\mathbf{p}. \qquad (3.15)
\end{aligned}$$

Similarly, we can design an efficient adaptive filter using

$$\mathbf{p}'_n = \eta\mathbf{x}_n \odot \dot{\mathbf{g}}_n + (1-\eta)\mathbf{p}'_{n-1}, \qquad (3.16)$$

$$\mathbf{d}_n = \eta\mathbf{x}_n \odot \mathbf{x}^*_n + (1-\eta)\mathbf{d}_{n-1}, \qquad (3.17)$$

$$\mathbf{s}_n = \eta\mathbf{x}_n + (1-\eta)\mathbf{s}_{n-1}, \qquad (3.18)$$

where the filter is initialized with the values $\mathbf{p}'_0 = \mathbf{d}_0 = \mathbf{s}_0 = 0$. The simplicity of the UCF filter implementation allows the filter to adapt in real-time to changes such as rigid-body motion, deformation, and/or lighting. The filter adapts by weighting new images more, with weights for older images decaying exponentially over time.

### *3.1.1.2 Subsets of the UCF*

All the state-of-the-art linear unconstrained correlation filters are subsets of the UCF. To show this, we first introduce some basic concepts. It is well known that the delta-like function in the space domain corresponds to a constant in the frequency domain, i.e.,

$$\check{\dot{\mathbf{g}}}_i = [0, \ldots, 0, u_i, 0, \ldots, 0]^T \quad \xleftrightarrow{\mathcal{F}} \quad \dot{\mathbf{g}}_i = u_i \mathbf{1}, \tag{3.19}$$

where $\check{\dot{\mathbf{g}}}_i$ has a value of $u_i$ at the center of the target's location and zeros elsewhere. Also note that when $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}$ for all $i$, then $\mathbf{p}$ simplifies to

$$\begin{aligned} \mathbf{p} &= \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i \\ &= \left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\right)\dot{\mathbf{g}} \\ &= \frac{1}{d}\bar{\mathbf{X}}\dot{\mathbf{g}}, \end{aligned} \tag{3.20}$$

and if $\dot{\mathbf{g}} = \mathbf{1}$ (a delta function in the space domain) then $\mathbf{p} = \bar{\mathbf{x}}$, the average of the training images in the frequency domain.

**MACH filter** The UCF filter in Eq. 3.9 reduces to the MACH filter (Section 2.8) when $\lambda = 1$ and $\mathbf{g}_i = 1$ for all $i$, i.e.,

$$\mathbf{h} = (\mathbf{D} - (1-\psi)\mathbf{M})^{-1}\bar{\mathbf{x}}. \tag{3.21}$$

In the original MACH filter implementation [45], $\psi = \frac{1}{2}$, and

$$\mathbf{h} = (\mathbf{D} - \frac{1}{2}\mathbf{M})^{-1}\bar{\mathbf{x}}. \tag{3.22}$$

**UOTSDF filter** The UCF filter in Eq. 3.9 reduces to the UOTSDF filter (Section 2.8) when $\psi = 1$ and $\dot{\mathbf{g}}_i = \mathbf{1}$ for all $i$, i.e.,

$$\mathbf{h} = (\lambda\mathbf{D} + (1-\lambda)\mathbf{P})^{-1}\bar{\mathbf{x}}. \tag{3.23}$$

**ASEF filter** The UCF filter in Eq. 3.9 reduces to the ASEF filter (Section 2.10) when $\lambda = 1$, $\psi = 1$, and $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}$ for all $i$ represents a 2-D Gaussian-function-like desired correlation output. ASEF uses one filter $\mathbf{h}_i$ per image and then averages them, i.e.,

$$\mathbf{h} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i, \tag{3.24}$$

where $\mathbf{h}_i$ is computed as in Eq. 3.9 using only one image, i.e.,

$$\mathbf{h}_i = \mathbf{D}_i^{-1}\mathbf{p}_i, \tag{3.25}$$

where the diagonal matrix $\mathbf{D}_i = \mathbf{X}_i\mathbf{X}_i^*$ contains the power spectral density of image $x_i$ along the diagonal. Eq. 3.25 can be simplified as follows,

$$\begin{aligned} \mathbf{h}_i &= \mathbf{D}_i^{-1}\mathbf{X}_i\dot{\mathbf{g}} \\ &= (\mathbf{X}_i\mathbf{X}_i^*)^{-1}\mathbf{X}_i\dot{\mathbf{g}} \\ &= (\mathbf{X}_i^*)^{-1}\dot{\mathbf{g}} \\ &= \dot{\mathbf{g}} \oslash \mathbf{x}_i^*, \end{aligned} \tag{3.26}$$

where $\oslash$ represents the Hadamard division. Note that when $\dot{\mathbf{g}} = \mathbf{1}$, $\mathbf{h}_i$ is the inverse filter.

**Extended-ASEF (EASEF) filter** The Fourier transform of the training images $\mathbf{X}_i$ usually has high frequency components with very low energy. The energy at each frequency value of the filter is

$$\begin{aligned} \mathbf{e}_i &= \mathbf{h}_i \odot \mathbf{h}_i^* \\ &= (\dot{\mathbf{g}} \oslash \mathbf{x}_i^*) \odot (\dot{\mathbf{g}} \oslash \mathbf{x}_i^*)^* \\ &= (\dot{\mathbf{g}} \odot \dot{\mathbf{g}}^*) \oslash (\mathbf{x}_i \odot \mathbf{x}_i^*)\,. \end{aligned} \tag{3.27}$$

This shows that near-zero energy frequency components in $\mathbf{x}_i$ produce very high energy (it goes to $\infty$ when $\mathbf{x}_i$ has a frequency component equal to zero) frequency components in $\mathbf{h}_i$.

Bolme et al. [13] modify the original ASEF [12] to try to fix this, and their work shows some improvements over the original ASEF. They only used the frequencies corresponding to the top $95\%$ of the total energy to construct the filter and set $\mathbf{h}_i(\omega) = 0$ for the remaining frequencies. However, in forcing the high frequency components to zero, much of discriminative information contained within these images is discarded. In most images of interest, this forces to zero the majority of the frequency components because most of the energy can be found in a small selection of the frequency components. In addition, selecting the frequencies with the top $95\%$ requires a sorting algorithm of order $O(Nd^2)$ for $N$ training images of dimension $d$ which requires additional computation.

We present the Extended-ASEF (EASEF) as a better approach to reduce the effects of near-zero energy frequencies. A small $\beta \geq 0$ is added to Eq. 3.25 as follows,

$$\mathbf{h}_i = (\mathbf{D}_i + \beta\mathbf{I})^{-1}\mathbf{p}_i. \tag{3.28}$$

The $\beta\mathbf{I}$ term prevents overemphasizing low energy components without ignoring them. In order to use a bounded scalar, $\beta$ can be replaced with $\beta = \frac{1}{\lambda}(1-\lambda)$ where $0 < \lambda \leq 1$, and $\mathbf{h}_i$ can be written as follows,

$$\begin{aligned}
\mathbf{h}_i &= \left(\mathbf{D}_i + \left(\frac{1-\lambda}{\lambda}\right)\mathbf{I}\right)^{-1}\mathbf{p}_i \\
&= \lambda\left(\lambda\mathbf{D}_i + (1-\lambda)\mathbf{I}\right)^{-1}\mathbf{p}_i \\
&= \lambda\mathbf{T}_i^{-1}\mathbf{p}_i \\
&\propto \mathbf{T}_i^{-1}\mathbf{p}_i,
\end{aligned} \tag{3.29}$$

where the scale factor $\lambda$ can be ignored because all the correlation outputs are scaled accordingly so it does not affect the filter's performance, and $\mathbf{T}_i = \lambda\mathbf{D}_i + (1-\lambda)\mathbf{I}$. Note that this modified $\mathbf{h}_i$ is also a special case of Eq. 3.9 using only one image, $\psi = 1$, and $\mathbf{P} = \mathbf{I}$.

**MOSSE filter** The UCF filter in Eq. 3.9 reduces to the MOSSE filter (Section 2.11) when $\lambda = 1$, $\psi = 1$ and $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}$ for all $i$ represents a 2-D Gaussian-function-like desired correlation output, i.e.,

$$\mathbf{h} = \mathbf{D}^{-1}\bar{\mathbf{X}}\dot{\mathbf{g}}. \tag{3.30}$$

**Extended-MOSSE (EMOSSE) filter** MOSSE suffers from a similar problem as ASEF, i.e., the training images $\mathbf{X}_i$ usually have high frequency components with very low energy which leads to very high energy frequency components in $\mathbf{h}$. Using the same techniques for EASEF, we add a small $\beta \geq 0$ to Eq. 3.30 as follows,

$$\mathbf{h} = (\mathbf{D} + \beta\mathbf{I})^{-1}\bar{\mathbf{X}}\dot{\mathbf{g}}, \tag{3.31}$$

and replaced the unbounded $\beta$ with $\beta = \frac{1}{\lambda}(1-\lambda)$ where $0 < \lambda \leq 1$, i.e.,

$$\begin{aligned} \mathbf{h} &= \left(\mathbf{D} + \left(\frac{1-\lambda}{\lambda}\right)\mathbf{I}\right)^{-1}\bar{\mathbf{X}}\dot{\mathbf{g}} \\ &= \lambda\left(\lambda\mathbf{D} + (1-\lambda)\mathbf{I}\right)^{-1}\bar{\mathbf{X}}\dot{\mathbf{g}} \\ &= \lambda\mathbf{T}^{-1}\mathbf{p} \\ &\propto \mathbf{T}^{-1}\mathbf{p}, \end{aligned} \tag{3.32}$$

where the scale factor $\lambda$ can be ignored because all the correlation outputs are scaled accordingly so it does not affect the filter's performance, $\mathbf{T} = \lambda\mathbf{D} + (1-\lambda)\mathbf{I}$ and $\mathbf{p} = \bar{\mathbf{X}}\dot{\mathbf{g}}$. Note that this modified $\mathbf{h}$ is also a special case of Eq. 3.9 using $\psi = 1$, $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}$ for all $i$, and $\mathbf{P} = \mathbf{I}$. This filter can also be expressed as

$$\mathbf{h} = \mathbf{h}_{UOTSDF} \odot \dot{\mathbf{g}}, \tag{3.33}$$

where $\mathbf{h}_{UOTSDF}$ is the UOTSDF filter $\mathbf{h}$ shown in Eq. 3.23. When the desired output is a delta function, i.e., $\dot{\mathbf{g}} = \mathbf{1}$, EMOSSE becomes UOTSDF. If $\lambda$ is also equal to 1, EMOSSE and UOTSDF becomes UMACE.

**UMACE filter** The UCF filter in Eq. 3.9 reduces to the UMACE filter (Section 2.11) when $\lambda = 1$, $\psi = 1$ and $\dot{\mathbf{g}}_i = \mathbf{1}$, i.e.,

$$\mathbf{h} = \mathbf{D}^{-1}\bar{\mathbf{x}}. \tag{3.34}$$

Note that this filter is also a special case of both the UOTSDF and MOSSE filters.

### 3.1.2 Generalized Constrained Correlation Filter (CCF)

When there are false-class images, our empirical evidence shows that constraining the correlation peak values significantly improves recognition performance. In addition to the UCF, we present the multi-objective generalized Constrained Correlation Filter (CCF)

$$\begin{aligned} \min_{\mathbf{h}} \quad & (\text{MSE, ONV, ASM}) \\ s.t. \quad & \mathbf{X}^{\dagger}\mathbf{h} = \mathbf{u}, \end{aligned} \tag{3.35}$$

where MSE, ONV, ASM $\geq 0$, $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ is a matrix where the columns represent the training images, and $\mathbf{u} = [u_1, ..., u_N]^T$ is a vector with the constrained correlation peak values for each training image $\mathbf{x}_i$. Refregier [55] (see also [15]) showed that multi-quadratic criteria are optimized (i.e., the solution yields the best for one criterion for fixed values of the others) as follows. Lagrange multipliers are used to obtain the following functional,

$$\begin{aligned} \mathcal{L}(\mathbf{h}, \gamma, \beta) &= \text{MSE} + \gamma\text{ASM} + \beta\text{ONV} - 2\Delta^{\dagger}(\mathbf{X}^{\dagger}\mathbf{h} - \mathbf{u}) \\ &= \mathbf{h}^{\dagger}\mathbf{D}\mathbf{h} - 2\mathbf{h}^{\dagger}\mathbf{p} + E_f + \gamma(\mathbf{h}^{\dagger}\mathbf{D}\mathbf{h} - \mathbf{h}^{\dagger}\mathbf{M}\mathbf{h}) + \beta\mathbf{h}^{\dagger}\mathbf{P}\mathbf{h} - 2\Delta^{\dagger}(\mathbf{X}^{\dagger}\mathbf{h} - \mathbf{u}) \\ &= \mathbf{h}^{\dagger}\left((1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}\right)\mathbf{h} - 2\mathbf{h}^{\dagger}\mathbf{p} + E_f - 2\Delta^{\dagger}(\mathbf{X}^{\dagger}\mathbf{h} - \mathbf{u}) \\ &= \mathbf{h}^{\dagger}\mathbf{T}\mathbf{h} - 2\mathbf{h}^{\dagger}\mathbf{p} + E_f - 2\Delta^{\dagger}(\mathbf{X}^{\dagger}\mathbf{h} - \mathbf{u}), \end{aligned} \tag{3.36}$$

where $\mathbf{T} = (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}$, $\gamma, \beta \geq 0$ are Lagrange multipliers, and $\Delta \neq 0$ is a vector of non-zero Lagrange multipliers. Note that when $\gamma = 0$ only MSE and ONV are optimized, when $\beta = 0$ only MSE and ASM are optimized, when $\gamma = \beta = 0$ only MSE is optimized, when $\gamma \to \infty$ and $\beta \to \infty$ only ASM and ONV are optimized, when $\gamma \to \infty$ only ASM is optimized, and when $\beta \to \infty$ only ONV is optimized.

Taking the gradient and setting it equal to zero gives,

$$\frac{\partial \mathcal{L}(\mathbf{h}, \gamma, \beta, \Delta)}{\partial \mathbf{h}} = 2\mathbf{T}\mathbf{h} - 2\mathbf{p} - 2\mathbf{X}\Delta = \mathbf{0}, \tag{3.37}$$

and solving for $\mathbf{h}$, we get

$$\begin{aligned}\mathbf{h} &= \mathbf{T}^{-1}(\mathbf{p}+\mathbf{X}\Delta) \\ &= \mathbf{T}^{-1}\mathbf{p}+\mathbf{T}^{-1}\mathbf{X}\Delta. \end{aligned} \tag{3.38}$$

Substituting $\mathbf{h}$ in Eq. 3.38 into the constraint in Eq. 3.35 gives

$$\begin{aligned}\mathbf{u} &= \mathbf{X}^{\dagger}\mathbf{h} \\ &= \mathbf{X}^{\dagger}(\mathbf{T}^{-1}\mathbf{p}+\mathbf{T}^{-1}\mathbf{X}\Delta) \\ &= \mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{p}+\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X}\Delta, \end{aligned} \tag{3.39}$$

and the Lagrange multiplier vector $\Delta$ can be computed as follows,

$$\Delta = (\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u}-\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{p}). \tag{3.40}$$

Substituting $\Delta$ in Eq. 3.40 into $\mathbf{h}$ in Eq. 3.38 gives the filter solution,

$$\begin{aligned}\mathbf{h} &= \mathbf{T}^{-1}\mathbf{p}+\mathbf{T}^{-1}\mathbf{X}\Delta \\ &= \mathbf{T}^{-1}\mathbf{p}+\mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u}-\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{p}) \\ &= \mathbf{h}_{UCF}+\mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u}-\mathbf{X}^{\dagger}\mathbf{h}_{UCF}) \\ &= \mathbf{h}_{UCF}+\mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u}-\mathbf{u}_{UCF}), \end{aligned} \tag{3.41}$$

where $\mathbf{u}$ are the constrained correlation outputs at the center, $\mathbf{h}_{UCF}$ is the UCF filter $\mathbf{h}$ shown in Eq. 3.9, and $\mathbf{u}_{UCF} = \mathbf{X}^{\dagger}\mathbf{h}_{UCF}$ are the correlation outputs at the center when using filter $\mathbf{h}_{UCF}$. In order to use bounded scalars, $\gamma$ and $\beta$ can be replaced with $\gamma = \frac{1}{\psi}(1-\psi)$ and $\beta = \frac{1}{\lambda}(1-\lambda)$, respectively, where $0 \le \psi, \lambda \le 1$, i.e.,

$$\begin{aligned}\mathbf{T} &= (1+\gamma)\mathbf{D}-\gamma\mathbf{M}+\beta\mathbf{P} \\ &= (1+\frac{1}{\psi}(1-\psi))\mathbf{D}-\frac{1}{\psi}(1-\psi)\mathbf{M}+\frac{1}{\lambda}(1-\lambda)\mathbf{P} \\ &= \frac{1}{\lambda\psi}\left(\lambda\mathbf{D}+\lambda(1-\psi)\mathbf{M}+(1-\lambda)\psi\mathbf{P}\right) \\ &= \frac{1}{\psi}\mathbf{D}-\frac{1}{\psi}\mathbf{M}+\frac{1}{\lambda}\mathbf{P}+\mathbf{M}-\mathbf{P}, \end{aligned} \tag{3.42}$$

where we define the ratio $\frac{0}{0} = 1$ for the cases when $\psi = 0$ and/or $\lambda = 0$.

**Desired peak value does not affect $\mathbf{h}$** The value at the origin of the desired correlation output $\dot{g}_i(m,n)$ does not affect the CCF solution. To show this, let the value at the origin $\dot{g}_i(m,n)$ be changed by $\alpha_i$, i.e., $\dot{g}_i(0,0)' = \dot{g}_i(0,0) + \alpha_i$. Adding a value to the origin in the space domain adds a constant to the entire frequency domain plane, i.e., $\dot{\mathbf{g}}_i' = \dot{\mathbf{g}}_i + \alpha_i\mathbf{1}$, and the new $\mathbf{p}'$ is

$$\begin{aligned}
\mathbf{p}' &= \sum_{i=1}^{N} \mathbf{X}_i \dot{\mathbf{g}}_i' \\
&= \sum_{i=1}^{N} \mathbf{X}_i (\mathbf{g}_i + \alpha_i \mathbf{1}) \\
&= \sum_{i=1}^{N} \mathbf{X}_i \mathbf{g}_i + \sum_{i=1}^{N} \alpha_i \mathbf{X}_i \mathbf{1} \\
&= \mathbf{p} + \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \\
&= \mathbf{p} + \mathbf{X}\Lambda, \qquad (3.43)
\end{aligned}$$

where $\Lambda = [\alpha_1, \ldots, \alpha_N]^T$. The CCF solution using $\mathbf{p}'$ is

$$\begin{aligned}
\mathbf{h}' &= \mathbf{T}^{-1}\mathbf{p}' + \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u} - \mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{p}') \\
&= \mathbf{T}^{-1}(\mathbf{p} + \mathbf{X}\Lambda) + \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u} - \mathbf{X}^\dagger\mathbf{T}^{-1}(\mathbf{p} + \mathbf{X}\Lambda)) \\
&= \mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{X}\Lambda + \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{u} - \mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{p} - \mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X}\Lambda) \\
&= \mathbf{h} + \mathbf{T}^{-1}\mathbf{X}\Lambda - \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X}\Lambda) \\
&= \mathbf{h} + \mathbf{T}^{-1}\mathbf{X}\Lambda - \mathbf{T}^{-1}\mathbf{X}\Lambda \\
&= \mathbf{h}, \qquad (3.44)
\end{aligned}$$

where $\mathbf{u} = \mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{p}$, which indicates that the filter does not change. This derivation is somewhat intuitive because the correlation peak value for the training images is already constrained, therefore the value at the center of the desired correlation output can be ignored. Thus, having a delta-function-like desired correlation output in the space domain, i.e., $\dot{\mathbf{g}} = \alpha\mathbf{1}$ in the frequency domain, or having a desired outputs of zeros everywhere, i.e., $\dot{\mathbf{g}} = \mathbf{0}$, gives the same filter $\mathbf{h}$.

**Extended-OTSDF** Note that when the desired output is $\dot{\mathbf{g}}_i = \mathbf{0}$ for all $n$, the CCF filter in Eq. 3.41 simplifies to

$$\mathbf{h} = \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}, \tag{3.45}$$

which we called the Extended-OTSDF (EOTSDF) because the solution looks like the OTSDF filter but in addition to the ONV and ACE criteria, it also minimizes the ASM criterion.

**Non-training images can have higher correlation peak values than training images** Linear CFs may produce a higher response for non-training images than for training images. Even if the training images have unit energy, $\mathbf{h}$ will not have unit energy. In fact $\mathbf{h}^{\dagger}\mathbf{h} > 1$ when $\mathbf{x}_i^{\dagger}\mathbf{x}_i = 1$ and $u_i = 1$. Since $\mathbf{x}_i$ has unit energy and the projection to $\mathbf{h}$ is constrained to be 1, then $\mathbf{h}^{\dagger}\mathbf{x}_i = |\mathbf{h}||\mathbf{x}_i|\cos\beta = |\mathbf{h}|\cos\beta = 1$, where $\beta$ is the angle between $\mathbf{h}$ and $\mathbf{x}_i$. Assuming $\beta > 0$ then $\cos\beta < 0$ and $|\mathbf{h}| > 1$. Since $\mathbf{h}$ is designed such that $\mathbf{h}^{\dagger}\mathbf{x}_i = 1$, there exists test chips $\mathbf{z}$ different than the training images for which $\mathbf{h}^{\dagger}\mathbf{z} > 1$, e.g., if $\mathbf{z}$ is equal or similar to $\mathbf{h}$. Assuming the test chip is normalized, i.e., $|\mathbf{z}| = 1$, then $\mathbf{h}^{\dagger}\mathbf{z} > 1$ when the angle between $\mathbf{h}$ and $\mathbf{z}$ is less than the angle between $\mathbf{h}$ and $\mathbf{x}_i$ (note that if $\mathbf{h}$ exists, the angle between $\mathbf{h}$ and any image $\mathbf{x}_i$ is equal if and only $\mathbf{h}^{\dagger}\mathbf{x}_i$ is constrained to be the same value for all $i$) as shown in Fig. 3.1. Although it is theoretically possible that a test chip that produces this response is an impostor, an image that resembles $\mathbf{h}$ that much is very likely a true-class image if the filter was well designed.

#### *3.1.2.1 Subsets of the CCF*

All the state-of-the-art linear constrained correlation filters are subsets of the CCF.

**MSESDF filter** The CCF filter in Eq. 3.41 reduces to the MSESDF filter (Section 2.7) when $\lambda = 1$, $\psi = 1$ and $\dot{\mathbf{g}}_i = \mathbf{1}$, i.e.,

$$\mathbf{h} = \mathbf{D}^{-1}\mathbf{p} + \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{D}^{-1}\mathbf{X})^{-1}(\mathbf{u} - \mathbf{X}^{\dagger}\mathbf{D}^{-1}\mathbf{p}). \tag{3.46}$$

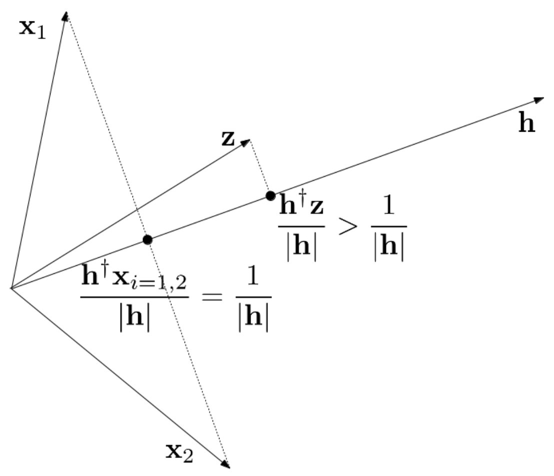

$\mathbf{x}_1$
$\mathbf{z}$
$\mathbf{h}$
$\frac{\mathbf{h}^\dagger\mathbf{z}}{|\mathbf{h}|} > \frac{1}{|\mathbf{h}|}$
$\frac{\mathbf{h}^\dagger\mathbf{x}_{i=1,2}}{|\mathbf{h}|} = \frac{1}{|\mathbf{h}|}$
$\mathbf{x}_2$

Figure 3.1: This figure shows that it is theoretically possible for a test chip $\mathbf{z}$ to have a higher inner product with filter $\mathbf{h}$ than the inner product of the training image $\mathbf{x}_n$ with $\mathbf{h}$.

**OTSDF filter** The CCF filter in Eq. 3.41 reduces to the OTSDF filter (Section 2.5) when $\psi = 1$ and $\dot{\mathbf{g}}_i = \alpha\mathbf{1}$ for all $i$ and any $\alpha \in \Re$. The CCF filter in Eq. 3.41 is simplified to

$$\mathbf{h} = \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}, \tag{3.47}$$

where

$$\begin{aligned}\mathbf{T} &= \mathbf{D} + \left(\frac{1}{\lambda} - 1\right)\mathbf{P}\\ &= \frac{1}{\lambda}(\lambda\mathbf{D} + (1-\lambda)\mathbf{P})\\ &\propto \lambda\mathbf{D} + (1-\lambda)\mathbf{P},\end{aligned} \tag{3.48}$$

where the $\frac{1}{\lambda}$ can be ignored because it does not affect the filter (note that we defined $\frac{0}{0} = 1$), i.e.,

$$\begin{aligned}\mathbf{h} &= \left(\frac{1}{\lambda}\mathbf{T}\right)^{-1}\mathbf{X}\left(\mathbf{X}^\dagger\left(\frac{1}{\lambda}\mathbf{T}\right)^{-1}\mathbf{X}\right)\mathbf{u}\\ &= \frac{\lambda}{\lambda}\mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}\\ &= \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}.\end{aligned} \tag{3.49}$$

**MACE filter** The CCF filter in Eq. 3.41 reduces to the MACE filter (Section 2.4) when $\lambda = 1$, $\psi = 1$, and $\dot{\mathbf{g}}_i = \alpha\mathbf{1}$ for all $i$ and any $\alpha \in \Re$, i.e.,

$$\mathbf{h} = \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{u}. \tag{3.50}$$

Note that this filter is also a special case of both the OTSDF and MSESDF filters.

**MVSDF filter** The CCF filter in Eq. 3.41 reduces to the MVSDF filter (Section 2.3) when $\lambda = 0$, $\psi = 1$, and $\dot{\mathbf{g}}_i = \alpha\mathbf{1}$ for all $i$ and any $\alpha \in \Re$, i.e.,

$$\mathbf{h} = \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{u}. \tag{3.51}$$

When $\mathbf{P} = \alpha\mathbf{I}$ for all $\alpha \neq 0$, the MVSDF reduces to the ECPSDF filter,

$$\mathbf{h} = \mathbf{X}(\mathbf{X}^{\dagger}\mathbf{X})^{-1}\mathbf{u}. \tag{3.52}$$

Note that the MVSDF filter is also a special case of both the OTSDF and MSESDF filters.

### 3.1.3 Modified CCF: CCF can equal UCF

When CCFs are trained, the desired response is usually chosen to be some $\dot{\mathbf{g}}_T$ for all the true-class images and $\dot{\mathbf{g}}_F$ for all false-class images. However, choosing instead a unique $\dot{\mathbf{g}}_i$ for each image can improve the filter's performance. One method to accomplish this is to choose $\dot{\mathbf{g}}_i$ for each image such that the CCF and UCF filters are the same.

Note from Eq. 3.41 that

$$\mathbf{h}_{CCF} = \mathbf{h}_{UCF} + \mathbf{h}_{EOTSDF}, \tag{3.53}$$

where $\mathbf{h}_{CCF}$ is the CCF filter $\mathbf{h}$ shown in Eq. 3.41, $\mathbf{h}_{UCF}$ is the UCF filter $\mathbf{h}$ shown in Eq. 3.9, and $\mathbf{h}_{EOTSDF}$ is the EOTSDF filter $\mathbf{h}$ shown in Eq. 3.45 with constrained correlation peak values

$$\mathbf{u}_{EOTSDF} = \mathbf{u}_{CCF} - \mathbf{u}_{UCF}, \tag{3.54}$$

where $\mathbf{u}_{CCF}$ is the peak value constraint for the CCF filter (the $\mathbf{u}$ shown in Eq. 3.41) and $\mathbf{u}_{UCF} = \mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{p}$ is the peak value for the UCF filter. In order for $\mathbf{h}_{CCF} = \mathbf{h}_{UCF}$, then $\mathbf{h}_{EOTSDF} = \mathbf{0}$, i.e., $\mathbf{u}_{CCF} = \mathbf{u}_{UCF}$. This may be accomplished by letting each desired $\dot{\mathbf{g}}_i$ be a scaled version of the others, i.e., $\dot{\mathbf{g}}_i = \tau_i \dot{\mathbf{g}}$, and determining the scalar $\tau_i$ for all $i$ as follows,

$$\begin{aligned}
\mathbf{u}_{CCF} &= \mathbf{u}_{UCF} \\
&= \mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{p} \\
&= \mathbf{X}^{\dagger}\mathbf{T}^{-1}\sum_{i=1}^{N}\mathbf{X}_i\tau_i\dot{\mathbf{g}} \\
&= \mathbf{X}^{\dagger}\mathbf{T}^{-1}\dot{\mathbf{G}}\sum_{i=1}^{N}\mathbf{x}_i\tau_i \\
&= \mathbf{X}^{\dagger}\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X}\Gamma, \qquad (3.55)
\end{aligned}$$

where $\Gamma = [\tau_1, \ldots, \tau_N]^T$, and the diagonal matrix $\dot{\mathbf{G}}$ contains $\dot{\mathbf{g}}$ along the diagonal. The vector $\Gamma$ can be computed as follows,

$$\Gamma = (\mathbf{X}^{\dagger}\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X})^{-1}\mathbf{u}, \qquad (3.56)$$

and the filter can be computed as follows,

$$\begin{aligned}
\mathbf{h} &= \mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X})^{-1}\mathbf{u} \\
&= \hat{\mathbf{T}}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\hat{\mathbf{T}}^{-1}\mathbf{X})^{-1}\mathbf{u}, \qquad (3.57)
\end{aligned}$$

where $\hat{\mathbf{T}} = \mathbf{T}\mathbf{G}^{-1}$. We call this $\mathbf{h}$ the modified Constrained Correlation Filter (mCCF). Our results usually show superior performance over the CCF and UCF filters.

### 3.1.4 Circular correlation effects

Discrete Fourier transforms (DFTs) are typically used to compute CFs. Therefore, correlations are actually *circular* (also known as *periodic*) correlations [49]. The result is that it is generally not possible to design a template (recall that terms *template* and *filter* depict whether our design is in the space or the frequency domain, respectively) that produces the desired correlation output even if we only have one training image. In the following discussion, we use 1-D notation for simplicity which can easily be adapted to 2-D.

Let $\dot{g}$ be the desired cross-correlation output of $x$ and $h$, where $x$, $h$, and $\dot{g}$ are of length $d$.

The actual cross-correlation output $g = x \otimes h$ is of length $2d - 1$. The value at the center of $g$ is $\check{\mathbf{h}}^{\dagger}\check{\mathbf{x}}_i = \sum_{n=0}^{d-1} x[n]h[n]$, and this value equals $\dot{g}[0]$, i.e., the first value of the desired cross-correlation output. This can be shown as follows (using Parseval's theorem),

$$\begin{aligned}\sum_{n=0}^{d-1} x[n]h[n] &= \frac{1}{d}\sum_{k=0}^{d-1} X[k]H^*[k] \\ &= \frac{1}{d}\sum_{k=0}^{d-1} \dot{G}[k] \\ &= \dot{g}[0], \end{aligned} \tag{3.58}$$

where the inverse DFT (IDFT) is defined as follows,

$$\dot{g}[n] = \frac{1}{d}\sum_{k=0}^{d-1} \dot{G}[k]e^{j\frac{2\pi k}{d}n}. \tag{3.59}$$

Therefore the value at the center of $g = x \otimes h$ is represented by $\dot{g}[0]$.

If a Gaussian-function-like discrete output with the mean at the center of $g$ is desired, then a Gaussian-function-like centered at $\dot{g}[0]$ should be used. This Gaussian-function-like wraps around $\dot{g}$ because of the periodicity assumption with finite discrete signals, i.e.,

$$\dot{g}[(n-m)_c] \quad \overset{\mathcal{F}}{\longleftrightarrow} \quad e^{-j\frac{2\pi k}{d}m}\dot{G}[k], \tag{3.60}$$

where $c$ represents circular periodicity and $\mathcal{F}$ represents the Fourier transform. As long as the desired Gaussian-function-like correlation output has a small variance, the effects of aliasing due to circular convolution are negligible.

One problem is that the desired correlation output generally cannot equal the actual correlation output, i.e.,

$$\dot{g} = x\tilde{\otimes}h \neq x \otimes h = g, \tag{3.61}$$

where $\tilde{\otimes}$ represents circular correlation. For example, suppose $x$ is a signal of $d = 1000$ random numbers and we find an $h$, such that $\dot{g} = [1, 0, 0, ..., 0]$ is of length $d$. Note the difference between $x\tilde{\otimes}h$ and $x \otimes h$ shown in Fig. 3.2.

If $x$ and $h$ are zero-padded to length $L = 2d - 1$, such that $x_e = [x, 0, 0, \ldots, 0]$ and $h_e =$

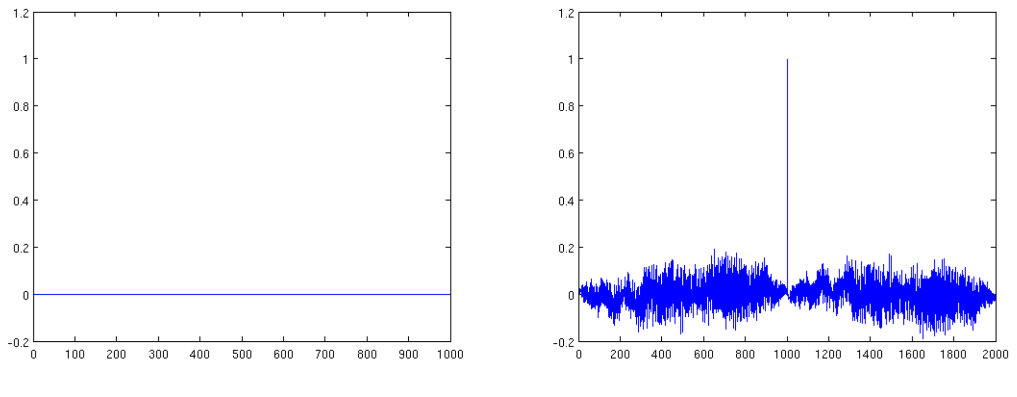


Figure 3.2: An example of (a) circular correlation $\dot{g} = x \tilde{\otimes} h$ where the peak is along the axis at $\dot{g}[0]$ and (b) regular correlation $g = x \otimes h$, where the peak is at $g[d-1] = g[999]$.

$[h, 0, 0, \ldots, 0]$, then regular correlation is equivalent to circular correlation. However, for most $x_e$ it is not possible to find a zero-padded $h_e$ such that

$$\begin{aligned} h_e &= \mathcal{F}^{-1}\left\{\mathcal{F}\{\dot{g}\} \oslash \mathcal{F}\{x_e\}\right\} \\ [h, 0, 0, \ldots, 0] &= \mathcal{F}^{-1}\left\{\mathcal{F}\{\dot{g}\} \oslash \mathcal{F}\{[x, 0, 0, ..., 0]\}\right\}, \end{aligned} \tag{3.62}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the DFT and IDFT, respectively, and $\oslash$ represents the Hadamard divide operator.

*Possible solutions*

While there is no exact solution to this problem, there are several techniques which will be described here that may be used to improve results. We use the simple inverse filter design to present these techniques. Note that in Figures 3.3, 3.4, and 3.5, $x$ is repeated as this is the DFT periodicity assumption. The green lines represent values of zero, the red lines represent real numbers, and the blue line represents the correlation peak. If this thesis is viewed in black and white, all the thin lines in $x$ represent zero values, and the thick lines represent real numbers. The desired correlation outputs $\dot{g}$ (or $g$ desired) is a circular-shifted delta function with zeros everywhere except at the peak. We now present the following four techniques:

1. Usual approach, i.e.,

$$h = \mathcal{F}^{-1}\left\{\mathcal{F}\{\dot{g}\} \oslash \mathcal{F}\{x\}\right\} \tag{3.63}$$

where $h$ is of length $d$. In this case $\dot{g}[0] = 1$ and zero elsewhere. As shown by example in Fig. 3.2, the cross-correlation of $x$ and $h$ generally does not result in zero values. Fig. 3.3 shows the training and testing phases.

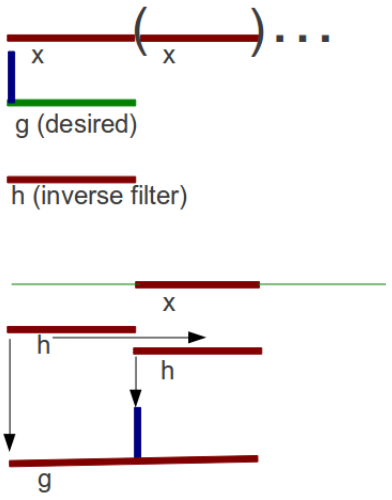


Figure 3.3: The desired correlation output and the template are of length $d$ (the same length as the training image). The desired correlation output has the peak value at the first index and the actual correlation output has the peak value at the center.

2. Zero-pad $x$, i.e.,

$$h = \mathcal{F}^{-1}\left\{\mathcal{F}\{\dot{g}\} \oslash \mathcal{F}\{x_e\}\right\} \tag{3.64}$$

where $h$ is of length $L$ (usually $L = 2d - 1$). In this case $\dot{g}\,[n] = 1$ for $n = \left\lceil \frac{d+L}{2} \right\rceil$ and 0 elsewhere, where $\lceil \cdot \rceil$ represents the ceiling operator. Fig. 3.4 shows the training and testing phases and gives some intuition as to why the peak of $\dot{g}$ is at $n = \left\lceil \frac{d+L}{2} \right\rceil$ in order to have the peak at the center of the cross-correlation of $x$ and $h$.

3. Same as #2 and then truncating $h$ to be of length $d$ (the same length as $x$ before zero-padding)

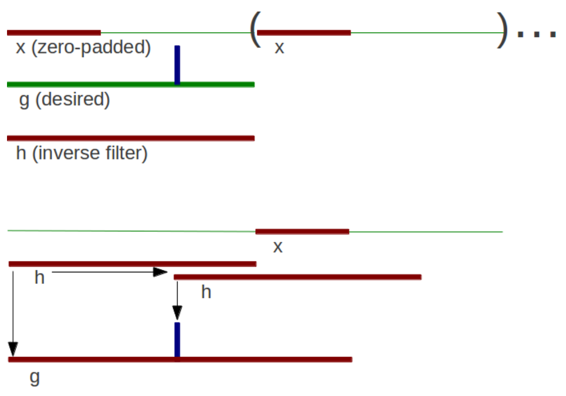


Figure 3.4: The desired correlation output and the template are of length $L$ (the same length as the zero-padded training image). The desired correlation output has the peak value at index $\lceil \frac{d+L}{2} \rceil$ and the actual correlation output has the peak value at the center.

as follows,

$$h'[n] = h\left[\left\lceil \frac{L-d}{2} \right\rceil + n\right], \tag{3.65}$$

for $n = 0, ..., d-1$. Fig. 3.5 shows the training and testing phases.

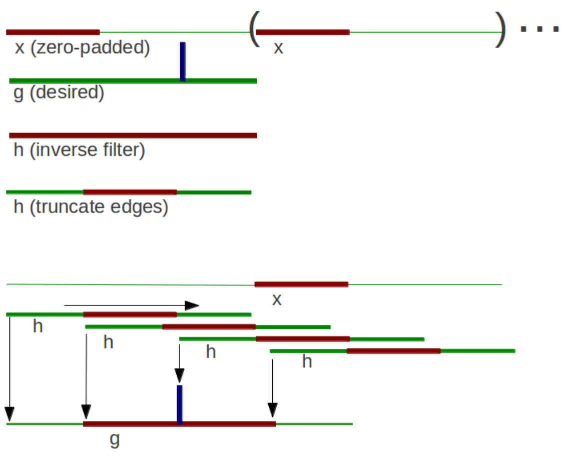


Figure 3.5: The desired correlation output is of length $L$ (the same as the zero-padded training image) and the template is truncated to be of length $d$ (the same length as the original training image). The desired correlation output has the peak value at index $\lceil \frac{d+L}{2} \rceil$ and the actual correlation output has the peak value at the center.

4. Same as #2 and then deemphasizing the edges (instead of zero-padding them completely as in #3), e.g., multiply with a sine function

$$h'[n] = h[n] \sin\left[\frac{n\pi}{L-1}\right], \tag{3.66}$$

for $n = 0, \ldots, d-1$, where $h'$ is of length $L$.

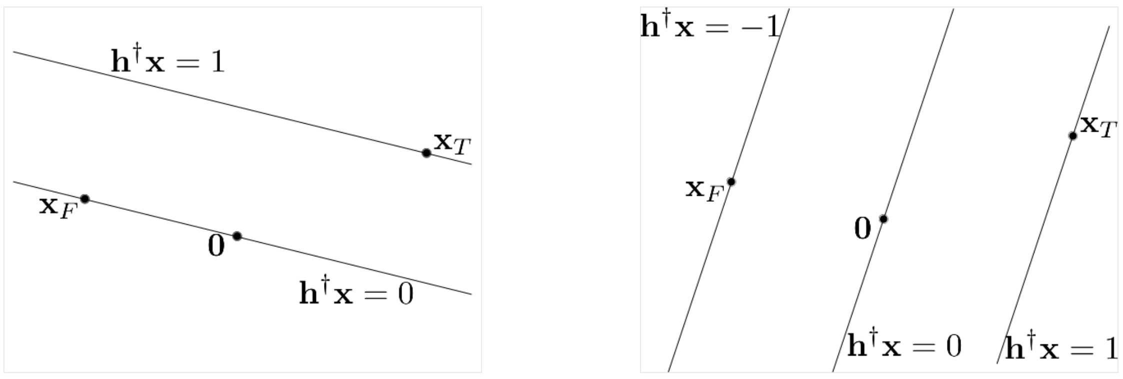
### 3.1.5 Implementation and false-class constraints

We use a Gaussian-function-like desired correlation output represented by $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}_T$ with a small variance in the space domain (or equivalently, a large variance in the frequency domain) for all $i$ corresponding to true-class images and

$$\dot{\mathbf{g}}_i = \dot{\mathbf{g}}_F = -\epsilon \dot{\mathbf{g}}_T \tag{3.67}$$

for all $i$ corresponding to false-class images, where $0 \leq \epsilon \leq 1$. In our experiments (discussed in Ch. 7) we use $\epsilon = 0.0.1$. That is, we want a distinguishable peak for true-class images and a very small negative values for false-class images. We observe empirically that this improves performance over 1) designing a filter with only true-class images and ignoring the false-class images, 2) using $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}_F = -\dot{\mathbf{g}}_T$ (i.e., $\epsilon = 1$) for all $i$ corresponding to false-class images, and 3) using $\dot{\mathbf{g}}_i = \dot{\mathbf{g}}_F = \mathbf{0}$ for all $i$ corresponding to false-class images.

We use a simple example to explain graphically why changing the value of $\epsilon$ produces a different filter. Suppose that there are only one true-class training image and one false-class training image with constrained correlation peak values 1 and 0, respectively. This means that the true-class image is in the hyperplane $\mathbf{h}^\dagger \mathbf{x} = 1$, and the false-class image is in the hyperplane $\mathbf{h}^\dagger \mathbf{x} = 0$. When linear CFs are designed, there is, however, the additional constraint that the zero image, i.e., $\mathbf{x} = \mathbf{0}$, produces a zero output, i.e., the zero image image is in the hyperplane $\mathbf{h}^\dagger \mathbf{x} = 0$. Fig. 3.6(a) shows these hyperplanes. The direction of $\mathbf{h}$ is orthogonal to these hyperplanes (proof in Appendix C). Now, suppose that the constrained correlation peak value for the false-class training image changes to $-1$. This produces a new $\mathbf{h}$ where the true-class image is in the new hyperplane $\mathbf{h}^\dagger \mathbf{x} = 1$, the false-class image is in the new hyperplane $\mathbf{h}^\dagger \mathbf{x} = -1$, and the zero image is in the new hyperplane $\mathbf{h}^\dagger \mathbf{x} = 0$. Fig. 3.6(b) shows these hyperplanes. We can see in these figures that changing the value of $\epsilon$ can produce a very different filter $\mathbf{h}$.



$\mathbf{h}^\dagger\mathbf{x} = 1$
$\mathbf{x}_T$
$\mathbf{x}_F$
$\mathbf{0}$
$\mathbf{h}^\dagger\mathbf{x} = 0$
(a)
$\mathbf{h}^\dagger\mathbf{x} = -1$
$\mathbf{x}_F$
$\mathbf{0}$
$\mathbf{x}_T$
$\mathbf{h}^\dagger\mathbf{x} = 0$
$\mathbf{h}^\dagger\mathbf{x} = 1$
(b)

Figure 3.6: Changing the constrained correlation peak value for the false-class image from $0$ in (a) to $-1$ in (b) produces a very different filter $\mathbf{h}$ (orthogonal to the hyperplane $\mathbf{h}^\dagger\mathbf{x} = 0$).

## 3.2 QUADRATIC CORRELATION FILTER ENHANCEMENTS

In Section 2.16 we reviewed quadratic correlation filters (QCFs). Generally, QCF designs assume a true class and a clutter class and maximizes the separation between these classes, which typically produces a large positive peak for the true class and a negative peak for the clutter class. This approach, however, does not utilize the full potential of QCFs. These filters are capable of distinguishing between two different classes. Linear CFs are not always trained to specifically reject a clutter class. They can reject clutter by default by producing strong peaks only for true targets. Our work [59, 61] shows that we can achieve the full potential of QCFs to recognize two different classes of targets by having one target class produce large positive correlation peaks, and the other target class produce large negative correlation peaks, and by default the clutter will usually exhibit low correlation values.

In order to facilitate recognition, we desire sharp peaks. Mahalanobis, et al. [36], suggests using shifted versions of the true class as part of the clutter class to reduce the sidelobes of the correlation plane. This cannot be implemented in our design because we do not use a clutter class. Kerekes, et al., noted that prewhitening the images can produce sharp peaks [33], and we follow a similar approach discussed in Section 3.2.1.

In order to speed up the training process we present a method to reduce the computational requirements from order $O(d^3)$ to order $O(N^3)$ for the general case when the number of training images $N$ is much smaller than the image dimension $d$. This method is discussed in Section 3.2.2.

### 3.2.1 Producing sharp peaks

The ECPSDF filter is equivalent to the OTSDF filter if the training and testing images are transformed by $\hat{\mathbf{x}}_i = \mathbf{T}^{-\frac{1}{2}}\mathbf{x}_i$ (using the OTSDF $\mathbf{T}$ shown in Eq. 2.18). Kumar and Mahalanobis [86] presented this idea for the MVSDF filter (instead of the OTSDF filter), but the same principles apply here. To verify this, the ECPSDF filter (see Eq. 2.8) is computed using transformed images as follows (note that $\mathbf{T}$ is diagonal with positive entries and is therefore symmetric and positive definite),

$$\begin{aligned} \hat{\mathbf{h}}_{ECPSDF} &= \hat{\mathbf{X}}(\hat{\mathbf{X}}^\dagger\hat{\mathbf{X}})^{-1}\mathbf{u} \\ &= \mathbf{T}^{-\frac{1}{2}}\mathbf{X}((\mathbf{T}^{-\frac{1}{2}}\mathbf{X})^\dagger\mathbf{T}^{-\frac{1}{2}}\mathbf{X})^{-1}\mathbf{u} \\ &= \mathbf{T}^{-\frac{1}{2}}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}, \end{aligned} \tag{3.68}$$

where $\hat{\mathbf{X}} = \mathbf{T}^{-\frac{1}{2}}\mathbf{X}$. Applying $\hat{\mathbf{h}}_{ECPSDF}$ to a transformed test chip $\hat{\mathbf{z}} = \mathbf{T}^{-\frac{1}{2}}\mathbf{z}$ is the same as applying the OTSDF filter $\mathbf{h}_{OTSDF}$ to the original test chip $\mathbf{z}$, i.e.,

$$\begin{aligned} \hat{\mathbf{h}}^\dagger_{ECPSDF}\hat{\mathbf{z}} &= \hat{\mathbf{h}}^\dagger_{ECPSDF}(\mathbf{T}^{-\frac{1}{2}}\mathbf{z}) \\ &= (\mathbf{T}^{-\frac{1}{2}}\hat{\mathbf{h}}_{ECPSDF})^\dagger\mathbf{z} \\ &= (\mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u})^\dagger\mathbf{z} \\ &= \mathbf{h}^\dagger_{OTSDF}\mathbf{z}, \end{aligned} \tag{3.69}$$

where $\mathbf{h}_{OTSDF} = \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}$ correctly matches the OTSDF filter in Eq. 2.17. This shows that applying the ECPSDF filter on transformed training and testing images has the same effect as applying the OTSDF filter on original (non-transformed) images. Also note from Eq. 3.69 that transforming the testing images is the same as transforming the filter itself, which can be used to achieve some computational advantages, i.e., instead of transforming all the test images, we only need to transform the filter.

Typically, most images have discriminative features at higher frequencies, i.e., at edges (object contours) within the image, and most of the energy at the lower frequencies. In the MACE filter, this $\mathbf{T}^{-\frac{1}{2}}$ transformation flattens the average power spectrum of the training images, thereby emphasizing the high frequency components. This also explains the sensitivity of the MACE filter to additive noise; it emphasizes the high frequency noise components. The MVSDF filter, on the other hand, emphasizes the low frequency components. The OTSDF filter allows for a tradeoff between ONV and ACE. In the CF literature this transformation is sometimes referred as prewhitening the images. This is somewhat abusing the terminology because each image is not strictly prewhitened, rather the spectrum of the images after the transformation is usually somewhat flattened but they are not strictly flat, especially when the OTSDF $\mathbf{T}$ is used.

In our QCF design this same concept is key in achieving sharp peaks, and we call this design the Transformed QCF (TQCF). We transform all the training images with $\mathbf{T}^{-\frac{1}{2}}$ using the $\mathbf{T}$ matrix from the OTSDF shown in Eq. 2.18, then we computed matrices $\mathbf{F}$ and $\mathbf{B}$ as shown in Section 2.16, and then transform these matrices as follows,

$$\bar{\mathbf{F}} = \mathbf{T}^{-\frac{1}{2}}\mathbf{F}, \tag{3.70}$$

and

$$\bar{\mathbf{B}} = \mathbf{T}^{-\frac{1}{2}}\mathbf{B}. \tag{3.71}$$

Note that to avoid introducing additional variables we are abusing the notation. The prewhitening is actually done in the frequency domain so the filters in $\mathbf{F}$ and $\mathbf{B}$ must first be transformed to the frequency domain (note from Eq. 2.119 that these filters are in the space domain). Also note that this transformation requires a slight increase in computation in training the filter, but requires no additional computation in applying/testing the filter. Having sharp peaks allows a QCF to recognize two different classes of targets by having one target class produce large positive correlation peaks, the other target class produce large negative correlation peaks, and by default the clutter will usually exhibit low correlation values. We present experimental results showing the QCF's ability to recognize two classes of targets in Section 3.4.

### 3.2.2 Efficient method to train QCF filters: fastQCF

We present a method to efficiently compute the QCF eigenvectors which we called fastQCF [60]. A filter with $N_1$ Class 1 and $N_2$ Class 2 training images of size $d$ requires computing the eigenvalues of a $d \times d$ matrix. The required computations are of the order $O(d^3)$. The required computations can be reduced to order $O(N^3)$, where $N = N_1 + N_2$, using the fact that the QCF only requires the eigenvectors corresponding to the non-zero eigenvalues of $\mathbf{R} = \mathbf{R}_1 - \mathbf{R}_2$ (see Eq. 2.116). This is beneficial because in most QCF applications $N \ll d$.

It is well known that given an $m \times n$ matrix $\mathbf{A}$ where $m > n$, the eigenvectors corresponding to the non-zero eigenvalues of $m \times m$ matrix $\mathbf{A}\mathbf{A}^T$ can be obtained by 1) finding the eigenvectors of $n \times n$ matrix $\mathbf{A}^T\mathbf{A}$ and 2) premultiplying those eigenvectors by $\mathbf{A}$. Both matrices will have the same non-zero eigenvalues. The trick is therefore to express matrix $\mathbf{R}$ as $\mathbf{R} = \mathbf{A}\mathbf{A}^T$ (note the transpose and not the hermitian symbol). Let the first $N_1$ columns of $\mathbf{A}$ be the lexicographically rearranged training images from Class 1 and let the next $N_2$ columns be the lexicographically rearranged training images from Class 2. Then multiply the first $N_1$ columns by $\sqrt{\frac{1}{N_1}}$ and multiply the next $N_2$ columns by $j\sqrt{\frac{1}{N_2}}$, where $j = \sqrt{-1}$, so that the columns of $\mathbf{A}$ are

$$\mathbf{A} = \left[\sqrt{\frac{1}{N_1}}\check{\mathbf{x}}_1^{(1)}, \sqrt{\frac{1}{N_1}}\check{\mathbf{x}}_2^{(1)}, \ldots, \sqrt{\frac{1}{N_1}}\check{\mathbf{x}}_{N_1}^{(1)}, j\sqrt{\frac{1}{N_2}}\check{\mathbf{x}}_1^{(2)}, j\sqrt{\frac{1}{N_2}}\check{\mathbf{x}}_2^{(2)}, \ldots, j\sqrt{\frac{1}{N_2}}\check{\mathbf{x}}_{N_2}^{(2)}\right], \quad (3.72)$$

where $\check{\mathbf{x}}_i^{(c)}$ represents the $i$th training image of Class $c$. We can verify that $\mathbf{R} = \mathbf{A}\mathbf{A}^T = \mathbf{R}_1 - \mathbf{R}_2$ (see Eq. 2.115) as follows,

$$\begin{aligned}
\mathbf{R} &= \mathbf{A}\mathbf{A}^T \\
&= \sum_{i=1}^{N_1}\left(\sqrt{\frac{1}{N_1}}\check{\mathbf{x}}_i^{(1)}\right)\left(\sqrt{\frac{1}{N_1}}\check{\mathbf{x}}_i^{(1)}\right)^T + \sum_{i=1}^{N_2}\left(j\sqrt{\frac{1}{N_2}}\check{\mathbf{x}}_i^{(2)}\right)\left(j\sqrt{\frac{1}{N_2}}\check{\mathbf{x}}_i^{(2)}\right)^T \\
&= \frac{1}{N_1}\sum_{i=1}^{N_1}\check{\mathbf{x}}_i^{(1)}\left(\check{\mathbf{x}}_i^{(1)}\right)^T - \frac{1}{N_2}\sum_{i=1}^{N_2}\check{\mathbf{x}}_i^{(2)}\left(\check{\mathbf{x}}_i^{(2)}\right)^T \\
&= \mathbf{R}_1 - \mathbf{R}_2. \qquad (3.73)
\end{aligned}$$

Table 3.1: Time to train one QCF filter using $N = 100$, $d = 2800$

| | time (s) |
|---|---|
| Traditional | 152.2 |
| fastQCF | 0.48 |

Thus, once matrix $\mathbf{A}$ is formed, the required eigenvectors of $\mathbf{R}$ can be efficiently computed.

In most of our experiments we use $N = 100$ training images of dimensions $d = 2800$. Using MATLAB on a standard Windows XP 2.91 GHz, 3.25 GB of RAM desktop, we trained a filter ten times using the traditional method and fastQCF and report the average times in Table 3.1. The table shows that our method is over 300 times faster than the traditional method to train the same filter.

## 3.3 Kalman Correlation Filter (KCF) for sequential images

The Multi-Frame Correlation Filter (MFCF) was reviewed in Section 2.17. MFCF combines information from the current correlation output with previous correlation outputs to enhance target recognition. MFCF, however, uses a fixed motion model and does not take into account the target's velocity. For example, in vehicle recognition MFCF is inadequate to represent the case where there may be multiple vehicles moving with different velocities and/or a vehicle with a varying velocity. If the vehicle's movement is large between frames, the convolution shown in Eq. 2.125 will place the actual vehicle's location under a low probability region in the posterior probability shown in Eq. 2.124. As a result, the vehicle may not be detected. In addition the MFCF may not be very robust against occlusions. When a vehicle is occluded, the correlation peak becomes small, leading to a low value in the probability image in Eq. 2.123.

These problems may be overcome by using a tracker. We introduce the Kalman Correlation Filter (KCF) [60] to address this issue using a Kalman filter (KF). Although there are more advanced trackers, the purpose of KCF is to show that combining a CF with a simple tracker can improve recognition. The KCF approach allows recursive minimum mean squared error (MMSE) estimation of a moving target's state.

### 3.3.1 Derivation

The KF consists of two models. The state model (also known as prediction model or motion model) describes how the state $\boldsymbol{\chi}$ evolves in time and the observation model (also known as update model) describes how the observations $\boldsymbol{\mathcal{Z}}$ are related to the states (note that all the notation in this derivation is in the spatial domain):

$$\begin{aligned}\boldsymbol{\chi}_t &= \mathbf{A}\boldsymbol{\chi}_{t-1} + \boldsymbol{\epsilon}_t \text{ (State model)} \\ \boldsymbol{\mathcal{Z}}_t &= \mathbf{C}\boldsymbol{\chi}_t + \boldsymbol{\delta}_t \text{ (Observation model)}\end{aligned} \tag{3.74}$$

where we use $\boldsymbol{\chi}_t$ and $\boldsymbol{\mathcal{Z}}_t$ to represent the target state and observation, respectively, at time $t$ instead of the traditional KF notation $\mathbf{x}_t$ and $\mathbf{z}_t$ to avoid confusion with previous image notation, $\mathbf{A}$ is the state transition matrix, $\mathbf{C}$ is the observation matrix, and $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\delta}_t$ are Gaussian distributed random vectors.

We use a *discrete white noise acceleration* model [8] to allow for velocity changes. The state vector is

$$\boldsymbol{\chi}_t = [p_x, p_y, v_x, v_y]^T, \tag{3.75}$$

where $p_x$ and $p_y$ represent the target location in the image plane (not world coordinates), and $v_x$ and $v_y$ represent the target's velocities in pixels per frame in the $x-$ and $y-$directions, respectively.

The state transition matrix is

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.76}$$

where $T$ (set to 1 without loss of generality) represents the time between the measurements. The

Gaussian distributed random vector $\boldsymbol{\epsilon}_t$ is modeled as

$$\boldsymbol{\epsilon}_t = \begin{bmatrix} \varepsilon_x \times \frac{1}{2}T^2 \\ \varepsilon_y \times \frac{1}{2}T^2 \\ \varepsilon_x \times T \\ \varepsilon_y \times T \end{bmatrix}, \tag{3.77}$$

where $\varepsilon_x$ and $\varepsilon_y$ are zero-mean Gaussian random variables with variance $E\{\varepsilon_x^2\} = E\{\varepsilon_y^2\} = \sigma_a^2$ chosen to represent the noise level, and it has the following covariance matrix (it is assumed that $\varepsilon_x$ and $\varepsilon_y$ are independent, i.e., $E\{\varepsilon_x\varepsilon_y\} = 0$),

$$\begin{aligned} \mathbf{R}_t &= E\{\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t^T\} \\ &= \sigma_a^2 \begin{bmatrix} \frac{1}{4}T^4 & 0 & \frac{1}{2}T^3 & 0 \\ 0 & \frac{1}{4}T^4 & 0 & \frac{1}{2}T^3 \\ \frac{1}{2}T^3 & 0 & T^2 & 0 \\ 0 & \frac{1}{2}T^3 & 0 & T^2 \end{bmatrix}. \end{aligned} \tag{3.78}$$

It has been suggested [8] that the standard deviation $\sigma_a$ be of the order of the maximum acceleration magnitude $a_M$, where $0.5a_M \leq \sigma_a \leq a_M$. This has the effect of introducing zero-mean noise propagated into the velocities and positions. Expanding the state model equations gives

$$\begin{aligned} \boldsymbol{\chi}_t &= \mathbf{A}\boldsymbol{\chi}_{t-1} + \boldsymbol{\epsilon}_t \\ \begin{bmatrix} p_{x_t} \\ p_{y_t} \\ v_{x_t} \\ v_{y_t} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x_{t-1}} \\ p_{y_{t-1}} \\ v_{x_{t-1}} \\ v_{y_{t-1}} \end{bmatrix} + \begin{bmatrix} \varepsilon_x \times \frac{1}{2}T^2 \\ \varepsilon_y \times \frac{1}{2}T^2 \\ \varepsilon_x \times T \\ \varepsilon_y \times T \end{bmatrix} \\ &= \begin{bmatrix} p_{x_{t-1}} + Tv_{x_{t-1}} + \frac{1}{2}\varepsilon_x T^2 \\ p_{y_{t-1}} + Tv_{y_{t-1}} + \frac{1}{2}\varepsilon_y T^2 \\ v_{x_{t-1}} + \varepsilon_x T \\ v_{y_{t-1}} + \varepsilon_y T \end{bmatrix}. \end{aligned} \tag{3.79}$$

The location of the maximum peak in the correlation output $g$ is used as the observed position of the target in the KF model, i.e.,

$$\mathcal{Z}_t = \begin{bmatrix} o_x \\ o_y \end{bmatrix}, \tag{3.80}$$

where $o_x$ and $o_y$ represent the location of the maximum peak in $g$ in the $x-$ and $y-$image coordinates, respectively. Because the position is the only observed parameter, the observation matrix is

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3.81}$$

Expanding the observation model equations gives,

$$\begin{aligned} \mathcal{Z}_t &= \mathbf{C}\boldsymbol{\chi}_t + \boldsymbol{\delta}_t \\ \begin{bmatrix} o_{x_t} \\ o_{y_t} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{x_t} \\ p_{y_t} \\ v_{x_t} \\ v_{y_t} \end{bmatrix} + \begin{bmatrix} \delta_{x_t} \\ \delta_{y_t} \end{bmatrix} \\ &= \begin{bmatrix} p_{x_t} + \delta_{x_t} \\ p_{y_y} + \delta_{y_t} \end{bmatrix}. \end{aligned} \tag{3.82}$$

The Gaussian distributed random vector $\check{\boldsymbol{\delta}}_t$ in the update equation is assumed to be zero-mean with covariance matrix

$$\mathbf{Q} = E\{\boldsymbol{\delta}_t\boldsymbol{\delta}_t^T\} = \begin{bmatrix} \sigma_p^2 & 0 \\ 0 & \sigma_p^2 \end{bmatrix}. \tag{3.83}$$

We relate $\sigma_p^2$ to the maximum peak value of the correlation output. A higher peak value represents more confidence in target location estimates and therefore has a small variance; a smaller peak value represents lower confidence and therefore has a large variance. The relationship between $\sigma_p^2$ and $g_{max}$ (shown in Fig. 3.7) can be modeled (somewhat arbitrarily, as other such models might work) as

$$\sigma_p^2 = 2^{(k-g_{max})} + 0.001, \tag{3.84}$$

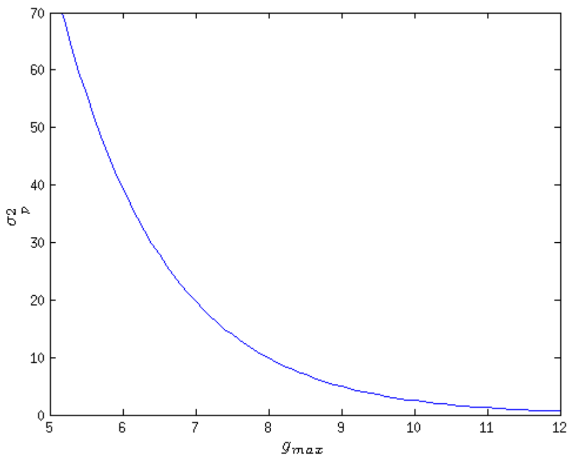


Figure 3.7: Relationship between $g_{max}$ and the uncertainty in position measurements $\sigma_p^2$.

where $g_{max}$ is the highest value in the correlation output, and $k$ is a constant chosen as the average $g_{max}$ for the visible target case. A small value of 0.001 is added to keep at least a very small degree of uncertainty even for high correlation outputs.

The KCF allows recursive MMSE estimation of a dynamic target's random vector state represented by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. Algorithm 3.1 contains the basic pseudocode for the KCF algorithm for one target. The five equations shown towards the bottom of Algorithm 3.1 are known as the KF equations. The Kalman gain matrix $\mathbf{K}$ weights confidence in the prediction versus the observation. Complete confidence in the prediction corresponds to a $\mathbf{K}$ with zero values. In our experiments, complete confidence in our measurements corresponds to a $\mathbf{K}$ with ones along the diagonal and zeros elsewhere.

This KCF algorithm provides a framework which suppresses noise by combining information from current observations with information from all previous correlation outputs. In the MMSE sense, it optimally computes the probability of a target before and after observing each correlation output. It is robust to variations in velocities and acceleration and can accommodate target occlusions. For example, if a moving tank temporarily disappears behind some other object for a few frames, the KCF can still assign a high probability of detection: it would place low confidence in the observations (since a low $g_{max}$ yields a very high $\sigma_p^2$) and more confidence in the prediction,

**Algorithm 3.1** Single-target KCF

```
1: initialize parameters: $\mathbf{A}, \mathbf{R}, \mathbf{C}$
2: while get another image $\neq$ NULL do
3:     compute $g$ (apply CF to image)
4:     observe $\boldsymbol{\mathcal{Z}} = (o_x, o_y)^T$: location of $g_{max}$
5:     compute measurement error $\sigma_p^2$ and construct $\mathbf{Q}$
6:     if not first image then $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ =UPDATE($\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}, \mathbf{Q}, \mathbf{C}$)
7:     else initialize $\boldsymbol{\mu} = [\boldsymbol{\mathcal{Z}}, \mathbf{0}]$ and $\boldsymbol{\Sigma} = \text{diag}(\sigma_p^2, \sigma_p^2, \sigma_v^2, \sigma_v^2)$
8:     $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}$ =PREDICTION($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{A}, \mathbf{R}$)
9: end while
1: function $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ =UPDATE($\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}, \mathbf{Q}, \mathbf{C}$)
2: $\mathbf{K} = \hat{\boldsymbol{\Sigma}}\mathbf{C}^T(\mathbf{C}\hat{\boldsymbol{\Sigma}}\mathbf{C}^T + \mathbf{Q})^{-1}$
3: $\boldsymbol{\mu} = \hat{\boldsymbol{\mu}} + \mathbf{K}(\boldsymbol{\mathcal{Z}} - \mathbf{C}\boldsymbol{\mu})$
4: $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}} - \mathbf{K}\mathbf{C}\hat{\boldsymbol{\Sigma}}$
1: function $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}$ =PREDICTION($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{A}, \mathbf{R}$)
2: $\hat{\boldsymbol{\mu}} = \mathbf{A}\boldsymbol{\mu}$
3: $\hat{\boldsymbol{\Sigma}} = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T + \mathbf{R}$
```

thus giving an estimated position of the target.

Algorithm 3.1, however, does not use the full information of the correlation output and therefore only works for one target. In order to take full advantage of the properties of CFs (i.e., the ability to locate multiple targets at unknown locations), a few modifications are made to Algorithm 3.1. Algorithm 3.2 contains the basic pseudocode for the KCF algorithm for multiple targets.

To locate more targets in the first image, the highest correlation peak value is found. Then, the area around the peak is set to zero. This is repeated for the next highest peak, and repeated again until all peaks above some threshold are located. Setting the region around each peak to zero is done to prevent confusing one peak that extends over a few pixels with multiple targets. For each peak, a state $\boldsymbol{\chi}_{(i)}$ is assigned with that position and with zero velocity. The uncertainty of the position depends on the peak height, and the uncertainty in velocity is chosen to be a constant ($\sigma_v^2$ in Algorithm 3.2) for the initial states. The prediction equations in Algorithm 3.1 are then applied to each state.

For subsequent images, the highest peak in an area near a state (starting with the state with the smallest uncertainty in position) is located. The size of the search area is directly proportional to the state's position uncertainty. The peak is assigned to that state and a small area around the peak is set to zero. This is repeated for all the states in order of increasing position uncertainty. The update equations in Algorithm 3.1 are then applied to each state. New targets can be detected by repeating the process for the initial image using the remaining peaks in the correlation output. Finally the update equations in Algorithm 3.1 are applied to each state. If a state's location uncertainty grows above some threshold (the standard deviation in position is larger than the image size), then the target is deleted either because it was lost or it never existed (we were tracking noise). A state $\mathbf{x}_i$ is declared to correspond to a target if the uncertainty of the state's position after the update equations is smaller than some threshold.

**Algorithm 3.2** Multi-targets KCF

```
1: initialize parameters: $\mathbf{A}, \mathbf{R}, \mathbf{C}$
2: while get another image $\neq$ NULL do
3:     if not first image then
4:         for each state $\boldsymbol{\chi}_{(i)}$ do (from smallest to largest position uncertainty)
5:             $\hat{\boldsymbol{\mu}}_{(i)}, \hat{\boldsymbol{\Sigma}}_{(i)} =$ PREDICTION($\boldsymbol{\mu}_{(i)}, \boldsymbol{\Sigma}_{(i)}, \mathbf{A}, \mathbf{R}$)
6:             observe $\boldsymbol{\mathcal{Z}}_{(i)} = [(o_x, o_y)]^T$: location of $g_{max}$ in area around $\hat{\boldsymbol{\mu}}_{(i)}$
7:             zero a small area in $g$ around and including $g_{max}$
8:             compute measurement error $\sigma_p^2$ and $\mathbf{Q}$
9:             $\boldsymbol{\mu}_{(i)}, \boldsymbol{\Sigma}_{(i)} =$ UPDATE($\hat{\boldsymbol{\mu}}_{(i)}, \hat{\boldsymbol{\Sigma}}_{(i)}, \mathbf{Q}, \mathbf{C}$)
10:        end for
11:    end if
12:    while $g_{max} >$ threshold $T'$ do /* find new states */
13:        zero a small area in $g$ around and including $g_{max}$
14:        compute measurement error $\sigma_p^2$ and $\mathbf{Q}$
15:        initialize $\boldsymbol{\mu}_{(i)} = [\boldsymbol{\mathcal{Z}}_{(i)}, \mathbf{0}]$ and $\boldsymbol{\Sigma}_{(i)} =$ diag$(\sigma_p^2, \sigma_p^2, \sigma_v^2, \sigma_v^2)$
16:    end while
17:    for all states $\boldsymbol{\chi}_{(i)}$: if $\sigma_{x_{(i)}} >$ size of image, then delete state
18:    Detection: for all states $\boldsymbol{\chi}_{(i)}$: if $\sigma_{x_{(i)}} <$ threshold $T''$, then label state as an actual target
19: end while
```

In addition to computing the correlation plane, the bottleneck of the KCF are the $4 \times 4$ matrix multiplies per state which are negligible compared to the two 2-D FFT computations required for the MFCF convolution per image frame.

## 3.4 Experiments

We test our algorithm on real and synthetic videos. The details of our experimental setup are discussed in Chapter 7. In this section we give a brief overview of our experiments and present results comparing the performance of the filters discussed in this chapter.

### 3.4.1 Vehicle recognition for filter comparison

We consider vehicle recognition on a set of $512 \times 640$ pixels 30 Hz infrared videos where the vehicle's class-label and location are unknown. Our dataset has eight vehicles (one vehicle in each video) driving in one circle, shown in Fig. 1.2. We consider 14 different types of classifiers: UCF, MOSSE, UOTSDF, MACH, UMACE, EASEF, ASEF, mCCF, CCF, OTSDF, MACE, ECPSDF, TQCF, and QCF. For each type of classifier we train eight filters. Each filter is trained to recognize one given target (there are eight targets) for all $360^\circ$ degrees of azimuth rotation. We determine target location by cross-correlating the template (recall that terms *template* and *filter* depict whether our design is in the space or the frequency domain, respectively) with the test image and determining its location by the highest value in the resulting correlation output. For the highest value, we compute the peak-to-correlation-energy (PCE) as follows,

$$\text{PCE} = \frac{g_{max}}{\sqrt{\sum_{m,n} |g(m,n)|^2 - |g_{max}|^2}}, \tag{3.85}$$

where $g_{max} = max_{n,m}(g(m,n))$ is the maximum value of the correlation plane. We select the template (out of the eight templates–one per target) that gives the highest PCE value. We declare a correct recognition when the correct template produces the maximum response to a given frame (i.e., correct classification) *and* produces the peak within a specified window centered at the correct location (i.e., correct localization). This means that it is considered an error 1) when the largest correlation peak is *not* close to the target's ground truth location *and* is from the incorrect class, or 2) when the largest correlation peak is close to the target's ground truth location but is from the incorrect class, or 3) when the largest correlation peak is from the correct class but the peak's location is not near the target's ground truth location.

We performed three sets of experiments. In Set 1, each filter is trained using 20 true-class images. In Set 2, each filter is trained using using 20 true-class images *and* 80 background (false-class) images. In Set 3, each filter is initially trained as in Set 2 and then retrained, i.e., we cross-correlate the template with the frames from which we cropped the training images, add the false positives as false-class training images, and retrain the template. Table 3.2 shows the classification (class), localization (loc), and recognition (recog) rates for each filter in Set 1. Table 3.3 shows the classification, localization, and recognition rates for each filter in Set 2, and it shows the recognition performance improvement (impr) over the recognition rates in Set 1. Improvement is computed as new performance minus old performance, and the result divided by old performance. Table 3.4 shows the recognition rates in Set 3, and it also shows the number of retraining (ret) cycles we used (after a certain number of cycles, performance does not improve), and the recognition performance improvement over the recognition rates in Set 2. We compared different $\lambda$, $\psi$, and variance $\dot{\mathbf{g}}_{\sigma^2}$ of the desired Gaussian-function-like shape correlation output parameters and report our best recognition performance findings. The values of the parameters selected are in bold and non-bold values are default parameters for those filters. The filters with bold names correspond to our filter designs.

These results show that unconstrained CFs usually have superior performance (by performance we mean *recognition* performance) over constrained CFs when only true-class images are used. Using false-class images improves the performances of all the filters except for ASEF, in particular the constrained CFs, where all of them except for ECPSDF exhibit over 150% improvement in performance. In fact, constrained CFs usually have superior performance over unconstrained CFs when we use false-class images, and this difference is more notable after retraining when more false-class images are added. We observe that $\psi = 1$ or $0.9999$ usually produces the best results and changing $\psi$ has little effect on the performance. We observe that changing $\lambda$ significantly affects the performance. This means that in our dataset the ASM criterion has little effect on the performance compared to MSE and ONV. In all of our experiments, our proposed linear CFs outperformed

Table 3.2: Filter performance (%) using true-class images only

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ |
|---|---|---|---|---|---|---|
| **TQCF** | 50.8 | 72.8 | 48.9 | **0.70** | 1 | 0 |
| QCF | 32.1 | 39.9 | 27.4 | 0 | 1 | 0 |
| **CCF** | 20.9 | 56.6 | 17.7 | **0.55** | **0.90** | **0.5** |
| OTSDF | 20.9 | 56.2 | 17.5 | **0.55** | 1 | 0 |
| MACE | 16.6 | 42.0 | 7.4 | 1 | 1 | 0 |
| ECPSDF | 14.4 | 27.0 | 10.9 | 0 | 1 | 0 |
| **mCCF** | 22.6 | 67.6 | 21.2 | **1.00** | **1.00** | **1.0** |
| **UCF** | 32.3 | 66.0 | 32.3 | **0.05** | **0.95** | **0** |
| UOTSDF | 32.2 | 65.8 | 32.2 | **0.05** | 1 | 0 |
| MOSSE | 27.2 | 62.4 | 24.5 | 1 | 1 | **1.5** |
| MACH | 29.5 | 63.6 | 29.1 | 1 | **0** | 0 |
| UMACE | 17.4 | 53.1 | 14.7 | 1 | 1 | 0 |
| **EASEF** | 34.1 | 67.0 | 31.8 | **0.05** | 1 | **0.5** |
| ASEF | 17.2 | 47.6 | 14.3 | 1 | 1 | **1.5** |

Table 3.3: Filter performance (%) using true- and false-class images before retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | impr |
|---|---|---|---|---|---|---|---|
| **TQCF** | 75.3 | 76.8 | 63.7 | **0.30** | 1 | 0 | 30.3 |
| QCF | 34.8 | 48.8 | 22.4 | 0 | 1 | 0 | -18.2 |
| **CCF** | 53.7 | 90.2 | 52.6 | **0.85** | **1.00** | **2.5** | 197.2 |
| OTSDF | 53.6 | 90.3 | 52.4 | **0.85** | 1 | 0 | 199.4 |
| MACE | 40.0 | 84.1 | 36.2 | 1 | 1 | 0 | 389.2 |
| ECPSDF | 16.1 | 63.9 | 11.1 | 0 | 1 | 0 | 1.8 |
| **mCCF** | 54.2 | 93.2 | 53.4 | **0.99** | **.9999** | **0.5** | 151.9 |
| **UCF** | 36.1 | 64.7 | 34.5 | **0.25** | **0.55** | **0.5** | 6.8 |
| UOTSDF | 33.3 | 66.3 | 33.3 | **0.15** | 1 | 0 | 3.4 |
| MOSSE | 29.7 | 64.5 | 26.1 | 1 | 1 | **2.0** | 6.5 |
| MACH | 33.4 | 68.2 | 33.1 | 1 | **0.15** | 0 | 13.7 |
| UMACE | 25.8 | 61.6 | 18.6 | 1 | 1 | 0 | 26.5 |
| **EASEF** | 35.4 | 68.7 | 31.8 | **0.05** | 1 | **0.5** | 0 |
| ASEF | 16.8 | 46.7 | 13.6 | 1 | 1 | **1.5** | -4.9 |

existing state-of-the-art linear CFs. When we only use true-class images, our UCF design performed comparably to the best performing linear CF from Chapter 2 UOTSDF with a 0.3% improvement. When we also use false-class images, our mCCF design outperformed the best performing linear CF from Chapter 2 OTSDF by 1.9% before retraining and 11.4% after retraining.

TQCF always outperforms all filters, but this comes with increase computational cost. As shown

Table 3.4: Filter performance (%) after retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | ret | impr |
|---|---|---|---|---|---|---|---|---|
| **TQCF** | 87.6 | 89.7 | 82.4 | **0.20** | 1 | 0 | 8 | 29.4 |
| QCF | 57.6 | 79.3 | 54.9 | 0 | 1 | 0 | 5 | 145.1 |
| **CCF** | 61.9 | 90.0 | 59.1 | **0.75** | **1.00** | **1.5** | 6 | 12.4 |
| OTSDF | 59.9 | 90.7 | 58.1 | **.80** | 1 | 0 | 7 | 10.9 |
| MACE | 46.2 | 85.1 | 42.7 | 1 | 1 | 0 | 3 | 18.0 |
| ECPSDF | 38.7 | 80.2 | 35.1 | 0 | 1 | 0 | 7 | 216.2 |
| **mCCF** | 65.8 | 92.4 | 64.7 | **.999** | **.9999** | **1.0** | 4 | 21.2 |
| **UCF** | 37.9 | 77.9 | 36.5 | **0.15** | **0.99** | **1.0** | 5 | 5.8 |
| UOTSDF | 35.7 | 74.6 | 35.3 | **0.35** | 1 | 0 | 2 | 6.0 |
| MOSSE | 31.9 | 76.5 | 31.1 | 1 | 1 | **2.5** | 4 | 19.2 |
| MACH | 35.7 | 77.8 | 34.9 | 1 | **0.55** | 0 | 3 | 5.4 |
| UMACE | 23.8 | 65.4 | 21.4 | 1 | 1 | 0 | 1 | 15.1 |
| **EASEF** | 33.9 | 66.5 | 32.9 | **0.05** | 1 | **0** | 3 | 3.5 |
| ASEF | 20.8 | 49.4 | 15.6 | 1 | 1 | **1.5** | 3 | 14.7 |

in Section 2.16, QCFs (and therefore TQCFs) can be written as the sum of squares of linear CF outputs. The number of linear correlations is the number of eigenvectors used. In our experiments we used 8 eigenvectors, thus the computational testing cost of QCFs is 8 times the computational cost of the linear filters. For the experiments using true-class images only, we used 8 eigenvectors corresponding 8 to the largest positive eigenvalues (there are only positive eigenvectors when only true-class images are used), and for the other experiments we used 7 eigenvectors corresponding to the 7 largest positive eigenvalues and 1 eigenvector corresponding the the largest negative eigenvalue. We observe that the recognition rate improvement of TQCF over QCF is 78%, 184%, and 79% in Sets 1, 2, and 3, respectively. Thus, in our dataset, using TQCF has a significant benefit.

### 3.4.2 QCF two target class recognition using KCF and MFCF

The previous dataset does not have videos with two targets, occluded targets, or targets moving at different velocities. To show the QCF's ability to simultaneously recognize two classes of targets and to show KCFs superiority over MFCF, we synthesized videos of moving tanks using images from three classes of toy tanks: *Abrams*, *German*, and *Missile* shown in Fig. 3.8. Former Carnegie Mellon University students Ryan Kerekes and Ramu Bhagavatula captured images of these three

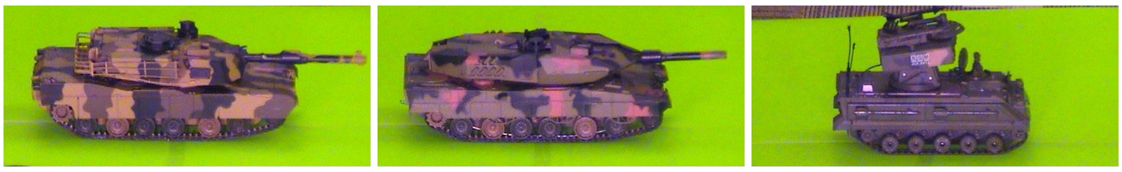


Figure 3.8: Images of toy tanks (left) Abram, (center) German, and (right) Missile used to test the algorithm

toy tanks at multiple depression angles and various azimuth rotations (covering all 360° of rotations spaced every two degrees). We adapted a system made by Kerekes [33] to create these videos. The toy tank images are segmented from the images and projected onto a plane using projective geometry to simulate a 3-D world. A simulated camera is used to capture 2-D images of the 3-D world. The collection of captured images forms the videos.

We show the results for two synthetic video sequences which we named *Straight* and *Rounded-rectangle* with additive white Gaussian noise. The first synthesized video, *Straight*, has seven frames and contains six tanks: two from each class. Three tanks are stationary while the other three move from left to right in the scene and are occluded as they pass behind the stationary tanks. The moving tanks each have a different acceleration. Fig. 3.9 shows some video frames before noise is added (for ease of visibility for the reader), and after the background mean is increased and white noise is added (which are the actual frames used in testing). To help identify the tanks in the figures we labeled them with their initials*: A*, *G*, and *M*.

The second synthesized video, *Rounded-rectangle*, has 23 frames and contains tanks *German* and *Abraham* crossing each other, turning when they reach the end of the screen, and repeating the processes completing one loop. The tanks go under a bridge and temporarily disappear. Fig. 3.10 shows some video frames.

In these experiments, the QCF is trained with *German* (Class 1) and *Abrams* (Class 2) images. The results show the capability for the QCF in the KCF framework. Fig. 3.11 shows the outputs of Frame 2 in video *Straight*. It contains the image frame, the KCF state after an observation, and the KCF state after applying the motion model. The update plot correctly shows the correct locations of the two *German* (positive peaks) and *Abrams* (negative peaks) tanks, and correctly ignores the two

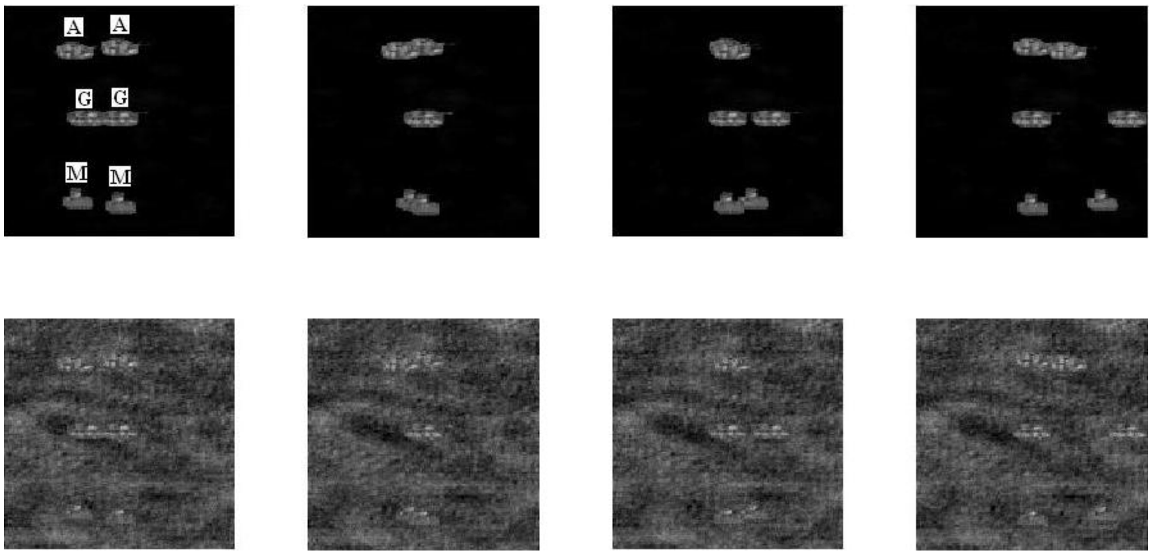


Figure 3.9: Frames 3, 4, 5, and 6 of *Straight* video. Top: before noise is added (for ease of visibility for the reader). Bottom: after the background mean is increased and AWGN is added (the actual frames used in testing).

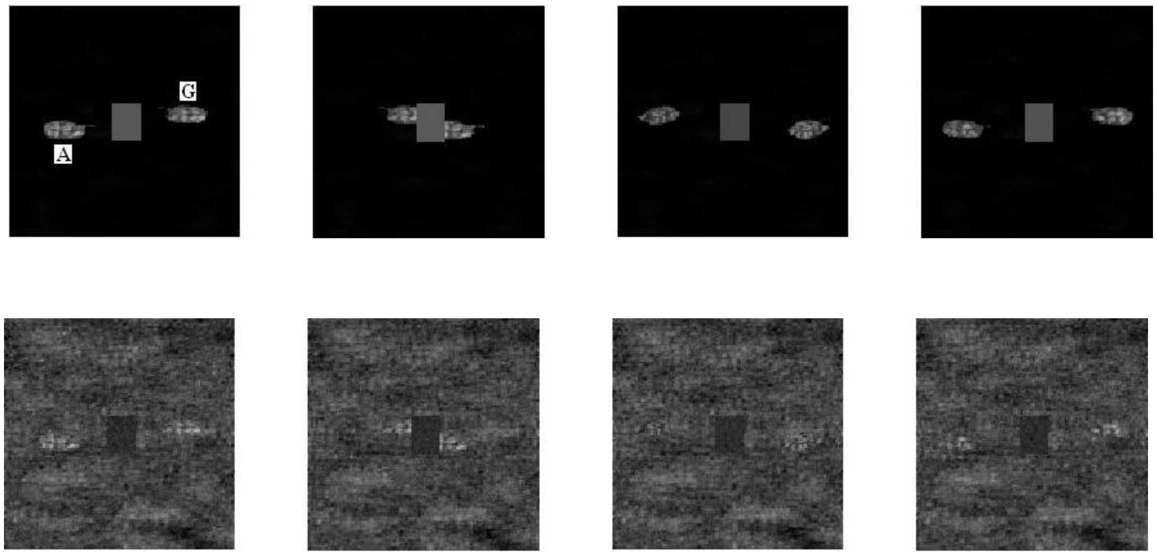


Figure 3.10: Frames 2, 7, 11, and 15 of *Rounded-rectangle* with and without noise.

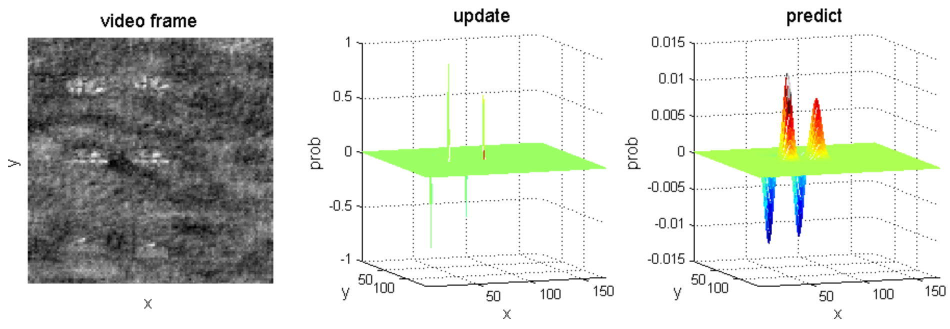


Figure 3.11: Example output from video *Straight*: (left) the frame, (center) the states after an observation, and (right) the states after applying the motion model.

Table 3.5: Recognition performance (%)

| video | KCF | MFCF |
|---|---|---|
| Straight | 100 | 92.3 |
| Rounded-rectangle | 87.0 | 71.7 |

*Missile* tanks (it was never trained to recognize the *Missile* tank). Note in Fig. 3.8 that these tanks look very similar, and in Fig. 3.11 it is impossible for most humans to differentiate them.

Table 3.5 shows the recognition rates for the QCF when used with the KCF and the MFCF. When the vehicles go through occlusions and move with changing velocities, KCF outperforms MFCF as shown in these experiments. If the velocity of the vehicles was small and known then the performance of MFCF and KCF would be comparable.

## 3.5 Summary

In this chapter we introduced two linear CFs: the CCF and UCF. UCFs can be efficiently computed and used for scenarios that required adaptive filters. Our experimental results show that our filters perform better than the state-of-the-art linear CFs. In addition we enhanced the performance of QCFs (both computational performance and recognition performance), and show its ability to simultaneously recognize multiple targets from two different classes. Finally, we presented the KCF, which improves recognition performance when using CFs in a sequence of images, i.e., in video, and show that in our dataset it outperformed the MFCF. In the next two chapters we use some of the concepts discussed here to design linear and quadratic maximum margin correlation filters.

CHAPTER 4

# MAXIMUM MARGIN CORRELATION FILTER

In this chapter we combine the design principles of support vector machines (SVMs) and correlation filters (CFs) and introduce a new type of classifier called the Maximum Margin Correlation Filter (MMCF). The MMCF classifier is less susceptible to over-fitting than traditional CFs while providing geometric shift-invariance to SVM classifiers. We show that the MMCF is an extension to both SVM and CFs by deriving MMCF first starting with an SVM design and then starting with a CF design. We provide experimental results showing that MMCF always outperforms linear SVMs and usually outperforms linear CFs in our dataset. An important contribution in this chapter to the SVM and CF research fields is to present the relationship between these two state-of-the-art classifiers.

## 4.1 MMCF: AN EXTENSION TO SVMS

In this section we show how MMCF is an extension to SVM classifiers. We review SVM classifiers, introduce the hard and soft margin SVM, formulate the SVM objective function in the frequency domain, and introduce MMCF as an SVM with a localization criterion.

### 4.1.1 SVM review

SVM classifiers [14, 19, 81] (referred to as SVMs throughout this thesis) are designed by extracting features from the training images and then using a feature vector to represent an image.

When using pixel values as features, the image is lexicographically scanned to form a feature vector. Given $N$ of these training column vectors $\check{\mathbf{x}}_i \in \mathbb{R}^{d\times 1}$ (inverted hat denotes spatial domain) and class labels $l_i \in \{-1, 1\}\ \forall i \in \{1, \dots, N\}$, the SVM approach (for a 2-class problem) finds the hyperplane that maximizes the smallest L-2 norm distance between the hyperplane and any data sample by solving

$$\max_{\check{\mathbf{h}}, b'} \left( \min_i \left( \frac{l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b')}{|\check{\mathbf{h}}|} \right) \right), \tag{4.1}$$

where $\check{\mathbf{h}}$ and $b'$ represent the hyperplane ($\check{\mathbf{h}}$ denotes the normal to the hyperplane and $b'$ is the bias or offset from the origin), and $\frac{l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b')}{|\check{\mathbf{h}}|}$ represents the distance from data sample $\check{\mathbf{x}}_i$ to the hyperplane $\check{\mathbf{h}}^T \check{\mathbf{x}} + b' = 0$ (see proof in Appendix C). We use the prime symbol for the bias $b'$ and other variables in the SVM space domain formulation, and later we remove the prime symbol in the SVM frequency domain formulation.

The problem with the formulation in Eq. 4.1 is that for any $(\check{\mathbf{h}}, b')$ solution, $(\alpha\check{\mathbf{h}}, \alpha b')$ is also a solution for all $\alpha \neq 0$. To get a unique $(\check{\mathbf{h}}, b')$ solution, one possibility is to constrain $l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b') = 1$ for the $\check{\mathbf{x}}_i$ closest to the decision boundary $\check{\mathbf{h}}^T \check{\mathbf{x}} + b' = 0$ (in this case $|\check{\mathbf{h}}|$ need not equal 1). In other words, $\min_i \left( l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b') \right) = 1$ or, equivalently, $l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b') \geq 1$ for all $i$. Using these constraints, we rewrite Eq. 4.1 as follows,

$$\begin{aligned} \max_{\check{\mathbf{h}}, b} \quad & \frac{1}{|\check{\mathbf{h}}|} \\ s.t. \quad & l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b') \geq 1, \end{aligned} \tag{4.2}$$

or equivalently,

$$\begin{aligned} \min_{\check{\mathbf{h}}, b'} \quad & \check{\mathbf{h}}^T \check{\mathbf{h}} \\ s.t. \quad & l_i(\check{\mathbf{h}}^T \check{\mathbf{x}}_i + b') \geq 1, \end{aligned} \tag{4.3}$$

noting that maximizing $\frac{1}{|\check{\mathbf{h}}|}$ is equivalent to minimizing $\check{\mathbf{h}}^T \check{\mathbf{h}}$. Eq. 4.3 is known as the SVM primal optimization problem. A more general form is

$$\begin{aligned} \min_{\check{\mathbf{h}},b'} \quad & \check{\mathbf{h}}^T\check{\mathbf{h}} \\ s.t. \quad & l_i(\check{\mathbf{h}}^T\check{\mathbf{x}}_i + b') \geq l_i u_i', \end{aligned} \tag{4.4}$$

where $u_i'$ is the peak constraint for feature vector $\check{\mathbf{x}}_i$ which allows for other values besides $l_i u_i' = 1$.

We have assumed up to this point that the data is linearly separable so that the constraint $l_i(\check{\mathbf{h}}^T\check{\mathbf{x}}_i + b') \geq l_i u_i$ for all $i$ always holds. If $u_i = u_T$ for all true-class feature vectors and $u_i = u_F$ for all false-class feature vectors, the space or distance that separates the true- and false-class data is

$$\frac{u_T - u_F}{|\check{\mathbf{h}}|} \tag{4.5}$$

and is known as the *hard margin*. We can relax the constraints to $l_i(\check{\mathbf{h}}^T\check{\mathbf{x}}_i + b') \geq l_i u_i' - \xi_i'$, where $\xi_i' \geq 0$ for all $i$. This allows for data vectors to be on the wrong size of the margin which may be misclassified. This margin is known as the *soft margin*. Data on the wrong size of the margin is penalized in the optimization problem as follows,

$$\begin{aligned} \min_{\check{\mathbf{h}},b'} \quad & \check{\mathbf{h}}^T\check{\mathbf{h}} + 2C\sum_{i=1}^{N}\xi_i' \\ s.t. \quad & l_i(\check{\mathbf{h}}^T\check{\mathbf{x}}_i + b') \geq l_i u_i' - \xi_i', \end{aligned} \tag{4.6}$$

where $\xi_i'$ is known as a *slack variable* and penalizes points that are on the wrong side of the margin, and $2C$ is a tradeoff parameter that weighs the cost (the "2" is included to avoid fractions when deriving the dual formulation). We use vector notation to be more consistent with the CF notation and write Eq. 4.6 as follows,

$$\begin{aligned} \min_{\check{\mathbf{h}},b'} \quad & \check{\mathbf{h}}^T\check{\mathbf{h}} + 2C\mathbf{1}^T\boldsymbol{\xi}' \\ s.t. \quad & \mathbf{L}(\check{\mathbf{X}}^T\check{\mathbf{h}} + b'\mathbf{1}) \geq \mathbf{L}\mathbf{u}' - \boldsymbol{\xi}', \end{aligned} \tag{4.7}$$

where $\boldsymbol{\xi}' = [\xi_1', \ldots, \xi_N']^T$, diagonal matrix $\mathbf{L}$ contains the vector labels $l_i$ along the diagonal, $\check{\mathbf{X}} = [\check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_N]$, and $\mathbf{u}' = [u_1'^{(1)}, \ldots, u_{N_1}'^{(1)}, u_1'^{(2)}, \ldots, u_{N_2}'^{(2)})]$, where $u_i'^{(1)}$ is the inner product (peak) constraint for the true-class feature vectors and $u_i'^{(2)}$ is the inner product constraint for the false-class feature vectors.

Using Lagrange multipliers, the constrained problem in Eq. 4.7 can be expressed as follows,

$$\mathcal{L}(\check{\mathbf{h}}, \boldsymbol{\xi}', b', \mathbf{a}, \boldsymbol{\mu}) = \check{\mathbf{h}}^T\check{\mathbf{h}} + 2C\mathbf{1}^\dagger\boldsymbol{\xi}' - 2\mathbf{a}^T[\mathbf{L}(\check{\mathbf{X}}^T\check{\mathbf{h}} + b'\mathbf{1}) - \mathbf{L}\mathbf{u}' + \boldsymbol{\xi}'] - 2\boldsymbol{\mu}^T\boldsymbol{\xi}', \tag{4.8}$$

where $\mathbf{a}, \boldsymbol{\mu} \geq \mathbf{0}$ are vectors of $N$ Lagrange multipliers, $\mathbf{a} \odot [\mathbf{L}(\check{\mathbf{X}}^T\check{\mathbf{h}} + b'\mathbf{1}) - \mathbf{L}\mathbf{u}' + \boldsymbol{\xi}'] = \mathbf{0}$, and $\boldsymbol{\mu} \odot \boldsymbol{\xi}' = \mathbf{0}$, where $\odot$ is the Hadamard product operator. We seek to maximize the vector $\mathbf{a}$ that minimizes $\mathcal{L}(\check{\mathbf{h}}, \boldsymbol{\xi}, b, \mathbf{a}, \boldsymbol{\mu})$ [30, 35]. Taking the partial derivative with respect to $\check{\mathbf{h}}$ and setting it to equal zero gives

$$\frac{\partial \mathcal{L}}{\partial \check{\mathbf{h}}} = \check{\mathbf{h}} - \check{\mathbf{X}}\mathbf{L}\mathbf{a} = \mathbf{0}, \tag{4.9}$$

and solving for $\check{\mathbf{h}}$ gives

$$\check{\mathbf{h}} = \check{\mathbf{X}}\mathbf{L}\mathbf{a}. \tag{4.10}$$

Taking the partial derivatives with respect to $b$ and $\xi$ and setting them to equal zero gives

$$\frac{\partial \mathcal{L}}{\partial b} = -2\mathbf{a}^T\mathbf{L}\mathbf{1} = 0, \tag{4.11}$$

and

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = 2C\mathbf{1} - 2\mathbf{a} - 2\boldsymbol{\mu} = \mathbf{0}, \tag{4.12}$$

and solving for $C\mathbf{1}$ gives,

$$C\mathbf{1} = \mathbf{a} + \boldsymbol{\mu}. \tag{4.13}$$

Given that $\mathbf{a}, \boldsymbol{\mu} \geq \mathbf{0}$ and $\mathbf{a} = C\mathbf{1} - \boldsymbol{\mu}$, then $\mathbf{0} \leq \mathbf{a} \leq C\mathbf{1}$.

Substituting Eqs. 4.10 and 4.13 into Eq. 4.8 gives,

$$\begin{aligned}
\mathcal{L}(\mathbf{a}) &= (\check{\mathbf{X}}\mathbf{L}\mathbf{a})^T\check{\mathbf{X}}\mathbf{L}\mathbf{a} + 2(\mathbf{a} + \boldsymbol{\mu})^T\boldsymbol{\xi}' - 2\mathbf{a}^T[\mathbf{L}(\check{\mathbf{X}}^T\check{\mathbf{X}}\mathbf{L}\mathbf{a} + b'\mathbf{1}) - \mathbf{L}\mathbf{u}' + \boldsymbol{\xi}'] - 2\boldsymbol{\mu}^T\boldsymbol{\xi}' \\
&= \mathbf{a}^T\mathbf{L}\check{\mathbf{X}}^T\check{\mathbf{X}}\mathbf{L}\mathbf{a} + 2\mathbf{a}^T\boldsymbol{\xi}' + 2\boldsymbol{\mu}^T\boldsymbol{\xi}' - 2\mathbf{a}^T\mathbf{L}\check{\mathbf{X}}^T\check{\mathbf{X}}\mathbf{L}\mathbf{a} - 2b'\mathbf{a}^T\mathbf{L}\mathbf{1} + 2\mathbf{a}^T\mathbf{L}\mathbf{u}' - 2\mathbf{a}^T\boldsymbol{\xi}' ... \\
&\quad -2\boldsymbol{\mu}^T\boldsymbol{\xi}' \\
&= 2\mathbf{a}^T\mathbf{L}\mathbf{u}' - \mathbf{a}^T\mathbf{L}\check{\mathbf{X}}^T\check{\mathbf{X}}\mathbf{L}\mathbf{a},
\end{aligned} \tag{4.14}$$

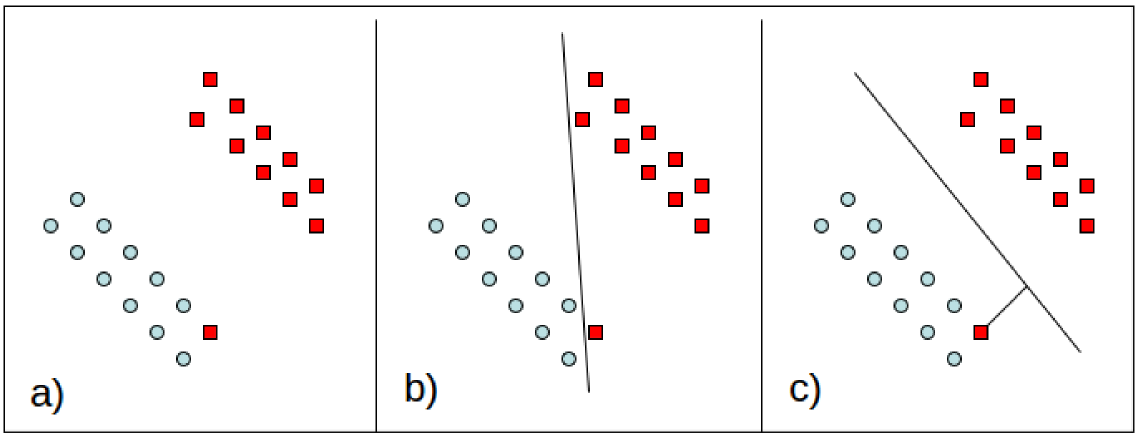


Figure 4.1: An example of a linearly separable problem that benefits from using slack variables to find a decision boundary with more generalization. a) The data points from two different classes, b) the decision boundary obtained by enforcing perfect separation, i.e., $C \to \infty$, and c) the decision boundary obtained by allowing slack variables with a $C$ value obtained using cross-validation.

where $2b'\mathbf{a}^T\mathbf{L}\mathbf{1} = 0$ using Eq. 4.11. We seek the vector $\mathbf{a}$ that maximizes $\mathcal{L}(\mathbf{a})$, i.e.,

$$\max_{\mathbf{0}\leq\mathbf{a}\leq C\mathbf{1}} \quad 2\mathbf{a}^T\mathbf{L}\mathbf{u}' - \mathbf{a}^T\mathbf{L}\check{\mathbf{X}}^T\check{\mathbf{X}}\mathbf{L}\mathbf{a}. \tag{4.15}$$

Eq. 4.15 is known as the dual formulation and can be solved by standard quadratic programming techniques (see Section 4.4.2). Note that using a hard margin is equivalent to $C = \infty$ for linearly separable data, so the dual problem is the same for the hard margin case with $\mathbf{a} \geq \mathbf{0}$ instead of $\mathbf{0} \leq \mathbf{a} \leq C\mathbf{1}$.

**Soft margin for linearly separable problems** Soft margins can be helpful even for linearly separable problems to provide better generalization. Fig. 4.1 shows one such example, where one data point is very likely an outlier (or a mislabeled data point). In this scenario, the decision boundary in Fig. 4.1(c) may be better than the decision boundary in Fig. 4.1(b). Cross-validation can be used to find an adequate $C$. Note that $C \to \infty$ forces the perfect separation shown in Fig. 4.1(b).

### 4.1.2 Space and frequency domain SVMs

Before we introduce the frequency domain SVM formulation, we review some of the notational conventions discussed in Section 1.4:

- lower case bold letters with an inverted hat denote vectors (lexicographically arranged) of 2-D spatial domain arrays, e.g, $\check{\mathbf{x}}_i$ denotes $x_i(m,n)$
- lower case bold letters sometimes denote vectors of 2-D frequency domain arrays, e.g, $\mathbf{x}_i$ denotes $X_i(u,v)$
  - we loosely refer to $\mathbf{x}_i$ as the $i$th image when we mean the vectorized 2-D DFT representation of the *i*th image
  - we loosely refer to $\mathbf{g}$ as the correlation plane when we mean the vectorized 2-D DFT representation of the correlation plane
  - we loosely refer to $\mathbf{h}$ as the template (or filter) when we mean the vectorized 2-D DFT representation of the template

Parseval's theorem [50] states that inner products in the space domain are only scaled by $\frac{1}{d}$ in the frequency domain, where $d$ is the dimension of the vectors, e.g., $\check{\mathbf{x}}_i^T\check{\mathbf{x}}_j = \frac{1}{d}\mathbf{x}_i^\dagger\mathbf{x}_j$ and $\check{\mathbf{h}}^T\check{\mathbf{x}}_i = \frac{1}{d}\mathbf{h}^\dagger\mathbf{x}_i$.

Using Parseval's theorem, it can be shown that the the solution $\mathbf{a}$ to the SVM dual problem in Eq. 4.15 is the same whether the training vectors are in the space or frequency domain, i.e.,

$$\begin{aligned}
\mathbf{a} &= \underset{\mathbf{0}\le\mathbf{a}\le C\mathbf{1}}{\operatorname{argmax}}\ 2\mathbf{a}^T\mathbf{L}\mathbf{u}' - \mathbf{a}^T\mathbf{L}\check{\mathbf{X}}^T\check{\mathbf{X}}\mathbf{L}\mathbf{a} \\
&= \underset{\mathbf{0}\le\mathbf{a}\le C\mathbf{1}}{\operatorname{argmax}}\ 2\mathbf{a}^T\mathbf{L}\left(\frac{1}{d}\mathbf{u}\right) - \mathbf{a}^T\mathbf{L}\left(\frac{1}{d}\mathbf{X}^\dagger\mathbf{X}\right)\mathbf{L}\mathbf{a} \\
&= \underset{\mathbf{0}\le\mathbf{a}\le C\mathbf{1}}{\operatorname{argmax}}\ \frac{1}{d}\left(2\mathbf{a}^T\mathbf{L}\mathbf{u} - \mathbf{a}^T\mathbf{L}\mathbf{X}^\dagger\mathbf{X}\mathbf{L}\mathbf{a}\right) \\
&= \underset{\mathbf{0}\le\mathbf{a}\le C\mathbf{1}}{\operatorname{argmax}}\ 2\mathbf{a}^T\mathbf{L}\mathbf{u} - \mathbf{a}^T\mathbf{L}\mathbf{X}^\dagger\mathbf{X}\mathbf{L}\mathbf{a}, \qquad (4.16)
\end{aligned}$$

where we choose $\mathbf{u} = d\mathbf{u}'$ so that everything is scaled appropriately.

Similarly, the SVM primal problem in Eq. 4.7 can be expressed in the frequency domain as follows,

$$\begin{aligned}\min_{\mathbf{h},b} \quad & \mathbf{h}^\dagger\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi},\end{aligned} \tag{4.17}$$

where $b = db'$ and $\boldsymbol{\xi} = d\boldsymbol{\xi}'$.

CFs usually assume $\mathbf{x}_i$ is in the frequency domain, in order to take advantage of the fact that correlation in the space domain can be computed as multiplication in the frequency domain. Since we will also use this property, the SVM notation will be in the frequency domain hereafter.

### 4.1.3 MMCF: SVM with localization criterion

The Maximum Margin Correlation Filter (MMCF) classifier combines the design principles of SVMs and CFs. In addition to maximizing the smallest L-2 distance between the hyperplane and any data, MMCF minimizes the MSE between the desired correlation output and the actual correlation output, i.e.,

$$\begin{aligned}\text{MSE} &= \frac{1}{Nd}\sum_{i=1}^{N}|\mathbf{g}_i - \dot{\mathbf{g}}_i|^2 \\ &= \frac{1}{Nd}\sum_{i=1}^{N}\left(\mathbf{g}_i^\dagger\mathbf{g}_i - 2\mathbf{g}_i^\dagger\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i\right) \\ &= \frac{1}{Nd}\sum_{i=1}^{N}\left(\mathbf{h}^\dagger\mathbf{X}_i\mathbf{X}_i^*\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{X}_i\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i\right) \\ &= \mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*\right)\mathbf{h} - 2\mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i\right) + \frac{1}{Nd}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i \\ &= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + E_f,\end{aligned} \tag{4.18}$$

where diagonal matrix $\mathbf{X}_i$ contains $\mathbf{x}_i$ along the diagonal, $\mathbf{g}_i = \mathbf{X}_i^*\mathbf{h}$ represents the correlation output for the $n$th training image, $\mathbf{D} = \frac{1}{dN}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^*$ is average power spectral density of the $N$ training images, $\mathbf{h}^\dagger\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{g}_i^\dagger\dot{\mathbf{g}}_i$ is a measurement of the similarity between the actual correlation planes and the desired correlation planes with

$$\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N} \mathbf{X}_i \dot{\mathbf{g}}_i, \tag{4.19}$$

and $E_f = \frac{1}{Nd}\sum_{i=1}^{N} \dot{\mathbf{g}}_i^\dagger \dot{\mathbf{g}}_i$ is the average energy of the desired correlation planes. We choose a desired correlation output represented by $\check{\mathbf{g}}_i$ to be Gaussian-function-like with a small variance or a delta function in order to have a sharp peak in the correlation output at the target's location. This can improve the localization capability of SVMs. Note that $\check{\mathbf{g}}_i$ can also equal $\mathbf{0}$ because the peak value is already constrained in the objective function.

For this purpose we write the MMCF multi-objective function as follows (the constant $E_f$ can be ignored here because it does not depend on $\mathbf{h}$ and does not affect the minimization problem),

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \left(\mathbf{h}^\dagger\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi}, \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p}\right) \\ s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{4.20}$$

where $\mathbf{u} = [u_1^{(1)}, \ldots, u_{N_1}^{(1)}, u_1^{(2)}, \ldots, u_{N_2}^{(2)})]$, where $u_i^{(1)}$ is the peak constraint for the true-class feature vectors and $u_i^{(2)}$ is the peak constraint for the false-class feature vectors. In most of our experiments, $u_i^{(1)} = 1$ for true-class feature vectors and $u_i^{(2)} = 0$ or other small negative value $\varepsilon$ for false-class feature vectors. That is, for true-class images, we expect a value near 1 and for false-class we expect a value that is close to 0. This allows us to detect the true targets and ignore everything else. We refer to $\mathbf{h}^\dagger\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi}$ as the *margin criterion* because minimizing that criterion maximizes the margin which facilitates classification and $\mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p}$ as the *localization criterion* because minimizing that criterion can result in sharp peaks which facilitates localization. The smaller the value of $\mathbf{h}^\dagger\mathbf{h}$, the larger the margin; a larger margin usually (but not necessarily) results in better generalization and classification performance [9]. The smaller the value of $\mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p}$, the sharper the correlation peak (assuming $\check{\mathbf{g}}_i$ is a Gaussian-function-like with a small variance or a delta-function-like or a function of all zeros). A sharp peak usually results in better localization performance.

Refregier [55] showed that two quadratic criteria are optimized (i.e., the solution gives the best performance for one criterion for a fixed value of the other) by minimizing a weighted sum of the two criteria (see also [15]), i.e.,

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \beta\left(\mathbf{h}^\dagger\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi}\right) + \left(\mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p}\right) \\ s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{4.21}$$

where $0 \leq \beta \leq \infty$. Subsuming one quadratic term into the other quadratic term, Eq. 4.21 can be written as follows,

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^\dagger\mathbf{T}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + 2C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{4.22}$$

where

$$\mathbf{T} = \mathbf{D} + \beta\mathbf{I}, \tag{4.23}$$

and $C \leftarrow \beta C$ to avoid introducing new notation. Here $\beta$ is the parameter which trades-off margin (i.e., L-2 norm margin maximization between the centered true-class and false-class training images) and object localization. Setting $\beta = \infty$ will ignore the localization criterion and result in the conventional SVM classifier for centered images. Therefore, the SVM objective function is a special case of this more general MMCF objective function with $\beta = \infty$. Smaller values of $\beta$ improve object localization by having sharper peaks in the correlation output.

#### *4.1.3.1 Delta-function-like desired output*

Our earlier work [62] assumed a delta-function-like desired output which can simplify the problem. The desired CF output in the space domain is set to $\check{\mathbf{g}}_i = [0, \ldots, 0, \check{\mathbf{h}}^T\check{\mathbf{x}}_i, 0, \ldots, 0]^T$ where we use $\check{\mathbf{h}}^T\check{\mathbf{x}}_i$ as the cross-correlation value of $\check{\mathbf{h}}$ and $\check{\mathbf{x}}_i$ at the target's location. In other words, a peak centered at the target's location and zeros everywhere else is desired. In the frequency domain, vector $\dot{\mathbf{g}}_i$ can be expressed as

$$\dot{\mathbf{g}}_i = \mathbf{1}\check{\mathbf{h}}^T\check{\mathbf{x}}_i = \frac{1}{d}\mathbf{1}\mathbf{h}^\dagger\mathbf{x}_i = \frac{1}{d}\mathbf{1}\mathbf{x}_i^\dagger\mathbf{h} \tag{4.24}$$

by using the property that a delta function in the space domain is a constant in the frequency domain. The MSE in Eq. 4.18 can be simplified to (note that $\mathbf{x}_i = \mathbf{X}_i\mathbf{1}$),

$$\begin{aligned}
\text{MSE} &= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + E_f \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - 2\mathbf{h}^\dagger\frac{1}{Nd}\sum_{i=1}^{N}(\mathbf{X}_i\dot{\mathbf{g}}_i) + \frac{1}{Nd}\sum_{i=1}^{N}\left(\dot{\mathbf{g}}_i^\dagger\dot{\mathbf{g}}_i\right) \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - \frac{2}{Nd^2}\sum_{i=1}^{N}\mathbf{h}^\dagger\mathbf{X}_i\mathbf{1}\mathbf{x}_i^\dagger\mathbf{h} + \frac{1}{Nd^3}\sum_{i=1}^{N}\mathbf{h}^\dagger\mathbf{x}_i\mathbf{1}^\dagger\mathbf{1}\mathbf{x}_i^\dagger\mathbf{h} \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - \frac{2}{Nd^2}\sum_{i=1}^{N}\mathbf{h}^\dagger\mathbf{x}_i\mathbf{x}_i^\dagger\mathbf{h} + \frac{1}{Nd^2}\sum_{i=1}^{N}\mathbf{h}^\dagger\mathbf{x}_i\mathbf{x}_i^\dagger\mathbf{h} \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - \frac{1}{d}\mathbf{h}^\dagger\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^\dagger\right)\mathbf{h} \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - \frac{1}{d}\mathbf{h}^\dagger\left(\frac{1}{Nd}\mathbf{X}\mathbf{X}^\dagger\right)\mathbf{h} \\
&= \mathbf{h}^\dagger\mathbf{D}\mathbf{h} - \frac{1}{d}\mathbf{h}^\dagger\mathbf{R}\mathbf{h} \\
&= \mathbf{h}^\dagger\mathbf{Z}\mathbf{h},
\end{aligned} \tag{4.25}$$

where $\mathbf{R} = \frac{1}{Nd}\mathbf{X}\mathbf{X}^\dagger$ is a non-diagonal matrix known as the correlation matrix, and $\mathbf{Z} = \mathbf{D} - \frac{1}{d}\mathbf{R}$. Note from Eq. 4.18 that $\mathbf{h}^\dagger\mathbf{Z}\mathbf{h}$ is a sum of squares, and thus is non-negative for any $\mathbf{h}$, and therefore $\mathbf{Z}$ is positive semidefinite.

The MMCF multi-objective function can be expressed as

$$\begin{aligned}
\min_{\mathbf{h},b} \quad & \left(\mathbf{h}^\dagger\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi}, \mathbf{h}^\dagger\mathbf{Z}\mathbf{h}\right) \\
s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}.
\end{aligned} \tag{4.26}$$

Eq. 4.26 can be expressed as a weighted sum of the two criteria, i.e.,

$$\begin{aligned}
\min_{\mathbf{h},b} \quad & \beta\left(\mathbf{h}^\dagger\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi}\right) + \mathbf{h}^\dagger\mathbf{Z}\mathbf{h} \\
s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi},
\end{aligned} \tag{4.27}$$

where $0 \leq \beta \leq \infty$, and subsuming one quadratic term into the other quadratic term Eq. 4.27 can be written as follows,

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^{\dagger}\mathbf{T}\mathbf{h} + 2C\mathbf{1}^{T}\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{X}^{\dagger}\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{4.28}$$

where $\mathbf{T} = \mathbf{Z} + \beta\mathbf{I}$, and $C \leftarrow \beta C$. Since $\mathbf{Z}$ is positive semidefinite, then $\mathbf{T}$ is positive definite for $\beta > 0$, and the data can be transformed such that

$$\tilde{\mathbf{h}} = \mathbf{T}^{\frac{1}{2}}\mathbf{h} \tag{4.29}$$

and

$$\tilde{\mathbf{x}}_i = \mathbf{T}^{-\frac{1}{2}}\mathbf{x}_i, \tag{4.30}$$

and the MMCF multi-objective function can be written as follows,

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \tilde{\mathbf{h}}^{\dagger}\tilde{\mathbf{h}} + C\mathbf{1}^{T}\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\tilde{\mathbf{X}}^{\dagger}\tilde{\mathbf{h}} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{4.31}$$

where $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N]$. This has the standard SVM form (see Eq. 4.17) which means that this MMCF design can be implemented using a standard SVM solver by using transform images to find $\tilde{\mathbf{h}}$. Then $\mathbf{h}$ can be computed as follows,

$$\mathbf{h} = \mathbf{T}^{-\frac{1}{2}}\tilde{\mathbf{h}}. \tag{4.32}$$

Note that the formulation in Eq. 4.28 is equivalent to maximizing a non-Euclidean margin or in Eq. 4.31 an Euclidean margin in a transformed space. Shivaswamy et al. [72] recently showed that the *type* of margin (e.g., L-2 norm margin) maximized is important while designing maximum margin classifiers. For example, Ashraf et al. [6] maximized a non-Euclidean margin for their task of applying Gabor filters in a lower dimensional feature space. They are motivated by reducing the computational complexity of designing classifiers on potentially infinite dimensional features extracted from infinite number of Gabor filters. However, we are motivated by a peak sharpness criterion, which improves object localization performance.

**Computing $\mathbf{T}^{-1}$** Directly computing the inverse of the non-diagonal $d \times d$ matrix $\mathbf{T}$ is computationally intensive. One method to reduce the computational complexity is to use the Woodbury matrix identity[1] [52]. Let matrix $\mathbf{T}$ be expressed as

$$\begin{aligned}
\mathbf{T} &= \mathbf{D} + \beta \mathbf{I}_d - \frac{1}{d}\mathbf{R} \\
&= \mathbf{T}_a - \frac{1}{d}\mathbf{R} \\
&= \mathbf{T}_a - \frac{1}{Nd^2}\mathbf{X}\mathbf{X}^\dagger \\
&= \mathbf{T}_a + \left(-\frac{1}{Nd^2}\mathbf{X}\right)\mathbf{I}_N\mathbf{X}^\dagger, \qquad (4.33)
\end{aligned}$$

where diagonal matrix $\mathbf{T}_a = \mathbf{D} + \beta \mathbf{I}_d$. We added a subscript to the identity matrices $\mathbf{I}_d$ and $\mathbf{I}_N$ corresponding to their $d \times d$ and $N \times N$ size, respectively. Using the Woodbury matrix identity,

$$\begin{aligned}
\mathbf{T}^{-1} &= \left(\mathbf{T}_a + \left(-\frac{1}{Nd^2}\mathbf{X}\right)\mathbf{I}_N\mathbf{X}^\dagger\right)^{-1} \\
&= \mathbf{T}_a^{-1} + \mathbf{T}_a^{-1}\left(-\frac{1}{Nd^2}\mathbf{X}\right)\left(\mathbf{I}_N^{-1} + \mathbf{X}^\dagger\mathbf{T}_a^{-1}\left(-\frac{1}{Nd^2}\mathbf{X}\right)\right)^{-1}\mathbf{X}^\dagger\mathbf{T}_a^{-1} \\
&= \mathbf{T}_a^{-1} + \mathbf{T}_a^{-1}\mathbf{X}\left(Nd^2\mathbf{I}_N - \mathbf{X}^\dagger\mathbf{T}_a^{-1}\mathbf{X}\right)^{-1}\mathbf{X}^\dagger\mathbf{T}_a^{-1}. \qquad (4.34)
\end{aligned}$$

This requires inverting two $N \times N$ matrices instead of one $d \times d$ matrix which is useful when $d \gg N$ which is often the case..

Another method to reduce the computational complexity is to ignore the correlation matrix $\mathbf{R}$ in Eq. 4.25, and approximate $\mathbf{T}$ by a diagonal matrix,

$$\begin{aligned}
\mathbf{T} &= \mathbf{Z} + \beta \mathbf{I} \\
&= \mathbf{D} - \frac{1}{d}\mathbf{R} + \beta \mathbf{I} \\
&\approx \mathbf{D} + \beta \mathbf{I}, \qquad (4.35)
\end{aligned}$$

thus avoiding the inversion of a non-diagonal matrix. This simplifies the localization criterion $\mathbf{h}^\dagger\mathbf{Z}\mathbf{h}$

[1] Thanks to Vishnu Naresh Boddeti for pointing this out. The Woodbury matrix identity is $(\mathbf{A} + \mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$.

shown in Eq. 4.26 to

$$\begin{aligned}\mathbf{h}^\dagger \mathbf{D}\mathbf{h} &= \mathbf{h}^\dagger \left( \frac{1}{Nd} \sum_{i=1}^{N} \mathbf{X}_i \mathbf{X}_i^* \right) \mathbf{h} \\ &= \frac{1}{Nd} \sum_{i=1}^{N} |\mathbf{X}_i^* \mathbf{h}|^2, \end{aligned} \tag{4.36}$$

which is a measure of the average of the energies of the correlation outputs. This localization criterion ignores the single pixel at the target's location. However, the energy contribution of the value at the target's location to the energy of the entire correlation output is negligible, and thus this approximation does not adversely affect the filter. This is equivalent to having the desired correlation output be all zeros, i.e., $\dot{\mathbf{g}}_i = \mathbf{0}$ for all $i$. The value at the target's location can be ignored because it is already constrained to be $l_i(\mathbf{h}^\dagger \mathbf{x}_i + b) \geq l_i u_i - \xi_i$. Empirically, we observed that the overall loss in filter performance is negligible when using this approximation.

## 4.2 MMCF: An extension to correlation filters

In this section we show how MMCF is a generalized CF. We first show that SVM is a general form of the ECPSDF filter. Then we use the same concepts to generalized the more advanced CCF that we introduced in Chapter 3.

### 4.2.1 From ECPSDF to SVM

In this section we show that SVM is a general form of the ECPSDF filter. We first show that the ECPSDF is the minimum norm solution for an underconstrained system, and then detail the differences between SVM and ECPSDF. The ECPSDF filter becomes an SVM by relaxing the equality constraints to inequality constraints and adding a bias.

#### *4.2.1.1 Minimum norm solutions for underconstrained systems*

Minimum norm solutions are usually used for underdetermined systems. Suppose that we have a problem with inner-product constraints of the form $\mathbf{h}^\dagger \mathbf{x}_i = u_i$ for $i = 1, \ldots, N$, where $\mathbf{h}$ is of dimension $d > N$. There are infinite vectors that satisfy those constraints. If $\mathbf{h}_s$ is one possible solution, then $\mathbf{h}_s + \alpha\mathbf{h}_0 \; \forall \alpha \in \Re$ is also a solution, where $\mathbf{h}_0$ is a vector orthogonal to the space spanned by the vectors $\mathbf{x}_i \; \forall i$, i.e., $\mathbf{h}_0 \in M^{\perp}$ where $M = \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_N)$, and therefore $\mathbf{h}_0^{\dagger}\mathbf{x}_i = 0 \; \forall i$. This can be easily verified as follows,

$$\begin{aligned}(\mathbf{h}_s + \alpha\mathbf{h}_0)^{\dagger}\mathbf{x}_i &= \mathbf{h}_s^{\dagger}\mathbf{x}_i + \alpha\mathbf{h}_0^{\dagger}\mathbf{x}_i \\ &= u_i + 0. \end{aligned} \tag{4.37}$$

The solution with minimum norm $\mathbf{h}_{\text{min}}$ lies in $\mathbf{h}_s + \alpha\mathbf{h}_0$ and is also orthogonal to $\mathbf{h}_0$. Therefore $\mathbf{h}_{\text{min}}$ lies in $M$, and therefore it is of the form

$$\mathbf{h} = \sum_{i=1}^{N} a_i\mathbf{x}_i = \mathbf{X}\mathbf{a}. \tag{4.38}$$

In other words the minimum norm problem,

$$\begin{aligned} \min_{\mathbf{h}} \quad & \mathbf{h}^{\dagger}\mathbf{h} \\ s.t. \quad & \mathbf{X}^{\dagger}\mathbf{h} = \mathbf{u} \end{aligned} \tag{4.39}$$

is equivalent to solving,

$$\mathbf{X}^{\dagger}\mathbf{h} = \mathbf{u}, \tag{4.40}$$

where

$$\mathbf{h} = \mathbf{X}\mathbf{a}. \tag{4.41}$$

This is the ECPSDF objective function in Section 2.2. In other words, ECPSDF is a minimum norm problem similar to SVM.

#### *4.2.1.2 ECPSDF and SVM differences*

The only two differences between ECPSDF and SVM are as follows: 1) ECPSDF does not use a bias $b$, and 2) ECPSDF uses equality constraints instead of inequality constraints.

**ECPSDF does not use a bias $b$** Not using a bias its equivalent to having the origin $\mathbf{x}_0 = \mathbf{0}$ as a point on the hyperplane $\mathbf{h}^\dagger\mathbf{x} + b = 0$ as an additional constraint. In other words,

$$\begin{aligned} \min_{\mathbf{h}} \quad & \mathbf{h}^\dagger\mathbf{h} \\ s.t. \quad & \mathbf{X}^\dagger\mathbf{h} + b = \mathbf{u} \\ & \mathbf{h}^\dagger\mathbf{0} + b = 0 \end{aligned} \tag{4.42}$$

and

$$\begin{aligned} \min_{\mathbf{h}} \quad & \mathbf{h}^\dagger\mathbf{h} \\ s.t. \quad & \mathbf{X}^\dagger\mathbf{h} = \mathbf{u} \end{aligned} \tag{4.43}$$

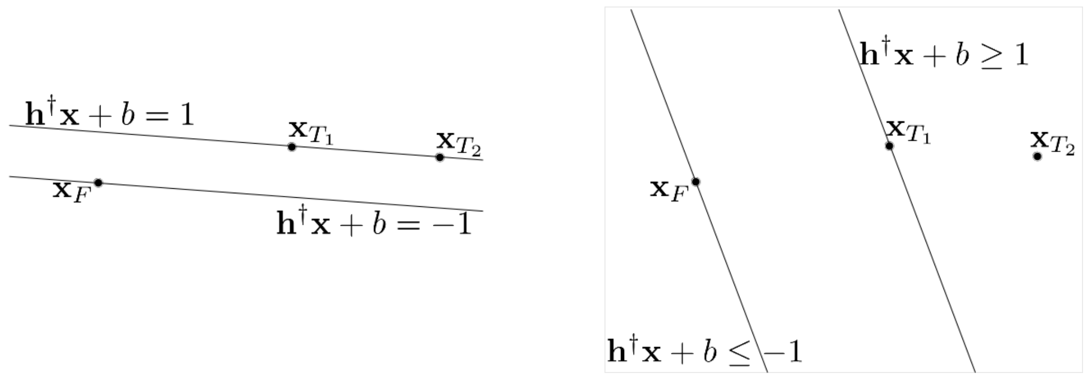
are equivalent. This presents a slight disadvantage to the ECPSDF formulation because it makes an assumption that it is not explicitly given. However, in practice, constraining the peak value for the zero vector to be zero may not be very disadvantageous. For example, this is equivalent to assuming that a false-class image with one monochromatic color with its mean subtracted, i.e., an image with zero variance where all values becomes zero after subtracting the mean as it is commonly done in preprocessing.

**ECPSDF uses equality constraints instead of inequality constraints** Assuming that ECPSDF uses a bias and that the constrained peak values are $u_i = 1$ for all $\mathbf{x}_i \in$ true-class and $u_i = -1$ for all $\mathbf{x}_i \in$ false-class, then Eq. 4.39 can be written as

$$\begin{aligned} \min_{\mathbf{h}} \quad & \mathbf{h}^\dagger\mathbf{h} \\ s.t. \quad & l_i(\mathbf{h}^\dagger\mathbf{x}_i + b) = 1, \end{aligned} \tag{4.44}$$

where $l_i = \{-1, 1\}$ is the class-label of the training vectors. Equivalently, it can be written as

$$\begin{aligned} \max_{\mathbf{h}} \quad & \frac{1}{|\mathbf{h}|} \\ s.t. \quad & l_i(\mathbf{h}^\dagger\mathbf{x}_i + b) = 1, \end{aligned} \tag{4.45}$$


$\mathbf{h}^\dagger\mathbf{x}+b=1$
$\mathbf{x}_{T_1}$
$\mathbf{x}_{T_2}$
$\mathbf{x}_F$
$\mathbf{h}^\dagger\mathbf{x}+b=-1$
$\mathbf{h}^\dagger\mathbf{x}+b\geq 1$
$\mathbf{x}_{T_1}$
$\mathbf{x}_{T_2}$
$\mathbf{x}_F$
$\mathbf{h}^\dagger\mathbf{x}+b\leq -1$

(a) (b)

Figure 4.2: The effects of relaxing the constraints from equality to inequality where $\mathbf{x}_{T_1}$ and $\mathbf{x}_{T_2}$ are true-class data points and $\mathbf{x}_F$ is a false-class data point. a) The hyperplane obtained with equality constraints leads to a small margin that may not generalize well. b) The hyperplane obtained by relaxing the constraints leads to a larger margin that may provide better generalization.

or as

$$\max_{\mathbf{h}} \quad \frac{l_i(\mathbf{h}^\dagger\mathbf{x}_i+b)}{|\mathbf{h}|}, \tag{4.46}$$

where $\frac{l_i(\mathbf{h}^\dagger\mathbf{x}_i+b)}{|\mathbf{h}|}$ is the distance from the hyperplane $\mathbf{h}^\dagger\mathbf{x}+b=0$ to the training vector $\mathbf{x}_i$ (see proof in Appendix C). That is, the ECPSDF is implicitly designed to maximize the distance between the hyperplanes $\mathbf{h}^\dagger\mathbf{x}+b=1$ containing all the true-class points and $\mathbf{h}^\dagger\mathbf{x}+b=-1$ containing all the false-class points. This means that the ECPSDF is a maximum margin filter. In general, if $\mathbf{h}^\dagger\mathbf{x}_i+b=u_T$ for all $\mathbf{x}_i\in$ true-class and $\mathbf{h}^\dagger\mathbf{x}_i+b=u_F$ for all $\mathbf{x}_i\in$ false-class, then the maximum margin between these hyperplanes is

$$\frac{\mathbf{h}^\dagger\mathbf{x}_T+b}{|\mathbf{h}|}-\frac{\mathbf{h}^\dagger\mathbf{x}_F+b}{|\mathbf{h}|}=\frac{u_T-u_F}{|\mathbf{h}|}. \tag{4.47}$$

SVM (ignoring slack variables) maximizes the distance between two hyperplanes that contain at least one (although it could contain more) of the true-class and false-class points, respectively. That is, SVM only takes into account the data points from each class that are closest in distance to the data points from the opposite class to come up with a decision boundary. This results in a larger margin and can lead to better generalization as shown in Fig. 4.2.

When not using a bias and using $u_i=u_T$ for all $\mathbf{x}_i\in$ true-class and $u_i=u_F$ for all $\mathbf{x}_i\in$

false-class, ECPSDF maximizes the distance between the hyperplanes $\mathbf{h}^\dagger \mathbf{x} = u_T$ and $\mathbf{h}^\dagger \mathbf{x} = u_F$ with the implicit constraint that the zero image has peak value $u_0 = 0$. This margin is

$$\frac{\mathbf{h}^\dagger \mathbf{x}_T}{|\mathbf{h}|} - \frac{\mathbf{h}^\dagger \mathbf{x}_F}{|\mathbf{h}|} = \frac{u_T - u_F}{|\mathbf{h}|}, \tag{4.48}$$

as in Eq. 4.47 but with a different $|\mathbf{h}|$. Fig. 4.3 shows the difference in the margin when a bias is used versus when it is not used. From this figure we see that when a bias is used, changing the false-class constraint value $u_i$ does not change the direction of $\mathbf{h}$; however, when a bias is not used, changing the constraint value $u_i$ produces a different $\mathbf{h}$ (see Section 3.1.5 for a discussion on different $u_i$ values for false-class feature vectors). If there are no explicit false-class feature vectors, the ECPSDF solution $\mathbf{h}$ maximizes the distance from the hyperplane $\mathbf{h}^\dagger \mathbf{x} = 1$ to the origin (it assumes the zero image has $u_i = 0$).

Another ECPSDF disadvantage is that it usually fails when $N \geq d + 1$ (except for the rare case when some points of a given class are collinear). Fig. 4.4 helps to make this point. When $d \leq N$ a hyperplane that perfectly separates these two classes can usually be found as is shown next.

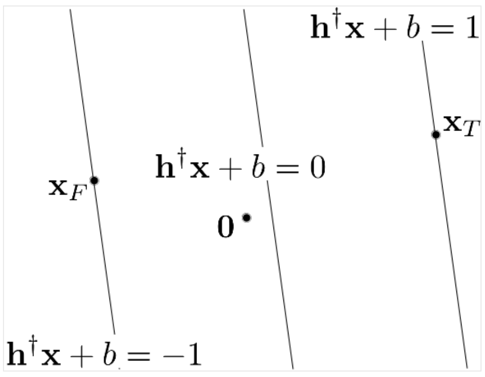
**When $d \geq N$ there usually exists a hyperplane that perfectly separates these two classes** In many applications, the dimension of the training vectors $d$ is greater than the number of training vectors, $N$. When $d \geq N$ there exists a hyperplane that contains the data of Class 1 that is parallel to a different hyperplane that contains the data of Class 2 *if and only if* the columns of

$$\mathbf{X}^{(2-\boldsymbol{\mu}_1)} = \mathbf{X}^{(2)} - \boldsymbol{\mu}_1 = [\mathbf{x}_1^{(2)} - \boldsymbol{\mu}_1, \mathbf{x}_2^{(2)} - \boldsymbol{\mu}_1, \ldots, \mathbf{x}_N^{(2)} - \boldsymbol{\mu}_1] \tag{4.49}$$

are not in the span of the columns of

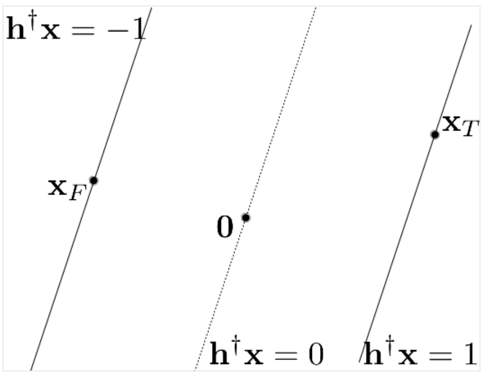
$$\mathbf{X}^{(1-\boldsymbol{\mu}_1)} = \mathbf{X}^{(1)} - \boldsymbol{\mu}_1 = [\mathbf{x}_1^{(1)} - \boldsymbol{\mu}_1, \mathbf{x}_2^{(1)} - \boldsymbol{\mu}_1, \ldots, \mathbf{x}_N^{(1)} - \boldsymbol{\mu}_1], \tag{4.50}$$

where $\mathbf{X}^{(1)}$ is a matrix containing the feature vectors of Class 1, $\mathbf{X}^{(2)}$ is a matrix containing the feature vectors of Class 2, and $\boldsymbol{\mu}_1$ is the mean of the feature vectors in $\mathbf{X}^{(1)}$. If this condition is met then the two parallel hyperplanes exist, and therefore there exists a hyperplane between these two hyperplanes that perfectly separates the two classes.



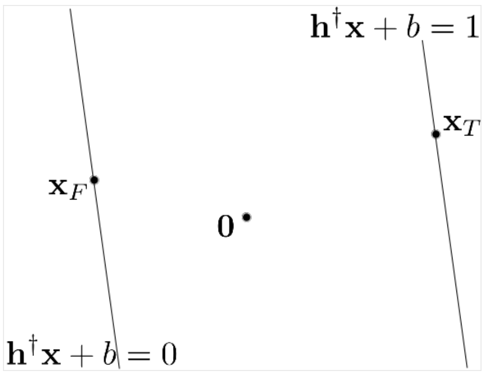
$\mathbf{h}^\dagger\mathbf{x}+b=1$
$\mathbf{x}_T$
$\mathbf{h}^\dagger\mathbf{x}+b=0$
$\mathbf{x}_F$
$\mathbf{0}$
$\mathbf{h}^\dagger\mathbf{x}+b=-1$

(a)



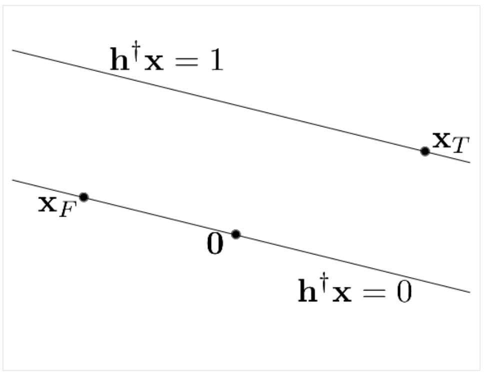
$\mathbf{h}^\dagger\mathbf{x}=-1$
$\mathbf{x}_T$
$\mathbf{x}_F$
$\mathbf{0}$
$\mathbf{h}^\dagger\mathbf{x}=0$
$\mathbf{h}^\dagger\mathbf{x}=1$

(b)



$\mathbf{h}^\dagger\mathbf{x}+b=1$
$\mathbf{x}_T$
$\mathbf{x}_F$
$\mathbf{0}$
$\mathbf{h}^\dagger\mathbf{x}+b=0$

(c)



$\mathbf{h}^\dagger\mathbf{x}=1$
$\mathbf{x}_T$
$\mathbf{x}_F$
$\mathbf{0}$
$\mathbf{h}^\dagger\mathbf{x}=0$

(d)

Figure 4.3: The difference in the margin when a bias is used versus when it is not used. Figs. (a) and (b) both have peak constraints $u_i = -1$ for the false-class features but produce a different $\mathbf{h}$ because (a) uses a bias $b$. Similarly, Figs. (c) and (d) both have peak constraints $u_i = 0$ for false-class features but also produce different $\mathbf{h}$ because (c) uses a bias $b$. When a bias is used as in Figs. (a) and (c), the peak constraints $u_i$ for false-class features do not affect the direction of $\mathbf{h}$, whereas in Figs. (b) and (d) when a bias is not used, different peak constraints $u_i$ produce different $\mathbf{h}$ vectors.

*Axiom 1*: If $d \geq N$ there exists at least one hyperplane that contains all $N$ points.

*Lemma 1*: A hyperplane $\mathbf{h}^\dagger\mathbf{x} + b_1 = 0$, or equivalently, $\mathbf{h}^\dagger\mathbf{x} = -b_1$ that contains all the data points in $\mathbf{X}^{(1)}$ must also contain its mean $\boldsymbol{\mu}_1$.

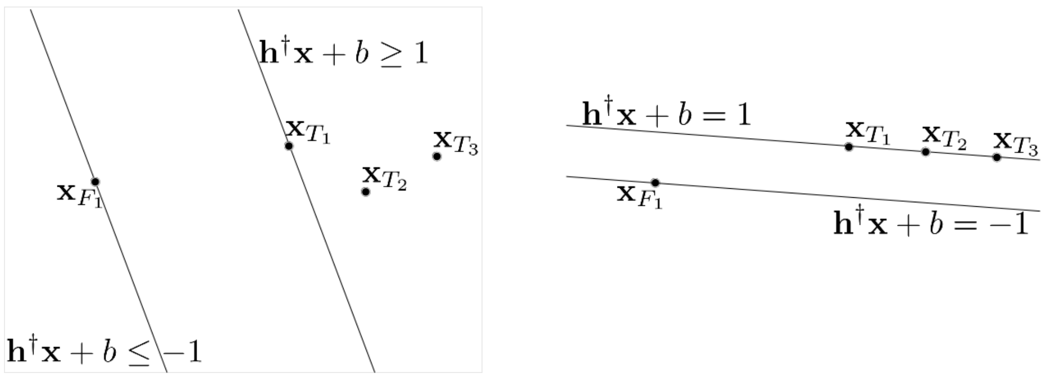


Figure 4.4: In (a), when $N \geq d+1$, ECPSDF cannot find one hyperplane that goes through all the true-class data points (the SVM margin is shown), except for the rare case when some points of a given class are collinear as shown in (b). In (b), the ECPSDF margin is shown. Note that both (a) and (b) have the same SVM margin.

The proof is as follows,

$$\begin{aligned}\mathbf{h}^\dagger\boldsymbol{\mu}_1 &= \mathbf{h}^\dagger\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^{(1)}\right)\\ &= \frac{1}{N}(\mathbf{h}^\dagger\mathbf{x}_1^{(1)}+\cdots+\mathbf{h}^\dagger\mathbf{x}_N^{(1)})\\ &= \frac{1}{N}((-b_1)+\cdots+(-b_1))\\ &= -b_1. \end{aligned} \tag{4.51}$$

Thus, $\boldsymbol{\mu}_1$ is also in the hyperplane $\mathbf{h}^\dagger\mathbf{x}=-b_1$.

*Lemma 2*: The vector $\mathbf{h}$ is orthogonal to the vector $\mathbf{x}_i^{(1)}-\boldsymbol{\mu}_1$.

The proof is as follows. Since $\mathbf{h}$ is orthogonal to the hyperplane $\mathbf{h}^\dagger\mathbf{x}=-b_1$ (see proof in Appendix C), then $\mathbf{h}$ must be orthogonal to any vector in that hyperplane. The vector $\mathbf{x}_i^{(1)}-\boldsymbol{\mu}_1$ is a vector on that hyperplane because $\mathbf{x}_i^{(1)}$ and $\boldsymbol{\mu}_1$ (by Lemma 1) are points on the hyperplane.

*Lemma 3*: The vector $\mathbf{h}$ is orthogonal to any vector in the span of the columns of $\mathbf{X}^{(1-\boldsymbol{\mu}_1)}$.

The proof is as follows. Let $\mathbf{v}$ be a vector in the span of the columns of $\mathbf{X}^{(1-\boldsymbol{\mu}_1)}$, i.e.,

$$\mathbf{v}=\alpha_1(\mathbf{x}_1^{(1)}-\boldsymbol{\mu}_1)+\cdots+\alpha_N(\mathbf{x}_N^{(1)}-\boldsymbol{\mu}_1). \tag{4.52}$$

Using Lemma 2,

$$\begin{aligned}\mathbf{h}^\dagger\mathbf{v} &= \mathbf{h}^\dagger\left(\alpha_1(\mathbf{x}_1^{(1)}-\boldsymbol{\mu}_1)+\cdots+\alpha_N(\mathbf{x}_N^{(1)}-\boldsymbol{\mu}_1)\right)\\ &= \alpha_1\mathbf{h}^\dagger(\mathbf{x}_1^{(1)}-\boldsymbol{\mu}_1)+\cdots+\alpha_N\mathbf{h}^\dagger(\mathbf{x}_N^{(1)}-\boldsymbol{\mu}_1)\\ &= \alpha_1 0+\cdots+\alpha_N 0 = 0. \end{aligned} \tag{4.53}$$

*Lemma 4*: Any vector $\mathbf{v}+\boldsymbol{\mu}_1$ is in the hyperplane $\mathbf{h}^\dagger\mathbf{x}=-b_1$, where $\mathbf{v}$ is a vector in the span of the columns of $\mathbf{X}^{(1-\boldsymbol{\mu}_1)}$.

The proof is as follows. Using Lemmata 2 and 3,

$$\mathbf{h}^\dagger(\mathbf{v}+\boldsymbol{\mu}_1)=\mathbf{h}^\dagger\mathbf{v}+\mathbf{h}^\dagger\boldsymbol{\mu}_1=0+(-b_1)=-b_1. \tag{4.54}$$

*Lemma 5*: The vector $\mathbf{w}$ is in the hyperplane $\mathbf{h}^\dagger\mathbf{x}=-b_1$ if $\mathbf{w}-\boldsymbol{\mu}_1$ is in the span of the columns of $\mathbf{X}^{(1-\boldsymbol{\mu}_1)}$. This can be expressed as $\mathbf{h}^\dagger\mathbf{x}=-b_1|\mathbf{x}\in \text{span}\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right)+\boldsymbol{\mu}_1$.

The proof is as follows. Let $\mathbf{v}=\mathbf{w}-\boldsymbol{\mu}_1$, and by Lemma 4 $\mathbf{v}+\boldsymbol{\mu}_1=\mathbf{w}$ is in the hyperplane $\mathbf{h}^\dagger\mathbf{x}=-b_1$.

*Lemma* 6: The vector $\mathbf{x}_i^{(2)}$ is linearly independent of the columns of $\mathbf{X}^{(1)}$ if $\mathbf{x}_i^{(2)}-\boldsymbol{\mu}_1$ is linearly independent of the columns of $\mathbf{X}^{(1-\boldsymbol{\mu}_1)}$, i.e., $rank\left(\left[\mathbf{X}^{(1)},\mathbf{x}_i^{(2)}\right]\right)=rank\left(\mathbf{X}^{(1)}\right)+rank\left(\mathbf{x}_i^{(2)}\right)$ if $rank\left(\left[\mathbf{X}^{(1-\boldsymbol{\mu}_1)},\mathbf{x}_i^{(2)}-\boldsymbol{\mu}_1\right]\right)=rank\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right)+rank\left(\mathbf{x}_i^{(2)}-\boldsymbol{\mu}_1\right)$.

The proof is as follows. Linearly independence implies that

$$\mathbf{x}_i^{(2)}-\boldsymbol{\mu}_1\neq\alpha_1(\mathbf{x}_1^{(1)}-\boldsymbol{\mu}_1)+...+\alpha_N(\mathbf{x}_N^{(1)}-\boldsymbol{\mu}_1), \tag{4.55}$$

for all $\alpha_i \in \Re$. Therefore,

$$\begin{aligned}
\mathbf{x}_i^{(2)} &\neq \alpha_1(\mathbf{x}_1^{(1)} - \boldsymbol{\mu}_1) + \cdots + \alpha_N(\mathbf{x}_N^{(1)} - \boldsymbol{\mu}_1) + \boldsymbol{\mu}_1 \\
&\neq \alpha_1\mathbf{x}_1^{(1)} + \cdots + \alpha_N\mathbf{x}_N^{(1)} + \boldsymbol{\mu}_1(-\alpha_1 - \cdots - \alpha_N + 1) \\
&\neq \alpha_1\mathbf{x}_1^{(1)} + \cdots + \alpha_N\mathbf{x}_N^{(1)} + \boldsymbol{\mu}_1\hat{\alpha} \\
&\neq \alpha_1\mathbf{x}_1^{(1)} + \cdots + \alpha_N\mathbf{x}_N^{(1)} + \frac{1}{N}(\mathbf{x}_1^{(1)} + \cdots + \mathbf{x}_N^{(1)})\hat{\alpha} \\
&\neq \alpha_1\mathbf{x}_1^{(1)} + \frac{1}{N}\hat{\alpha}\mathbf{x}_1^{(1)} + \cdots + \alpha_N\mathbf{x}_N + \frac{1}{N}\hat{\alpha}\mathbf{x}_N^{(1)} \\
&\neq \left(\alpha_1 + \frac{1}{N}\hat{\alpha}\right)\mathbf{x}_1^{(1)} + \cdots + \left(\alpha_N + \frac{1}{N}\hat{\alpha}\right)\mathbf{x}_N^{(1)} \\
&\neq \bar{\alpha}_1\mathbf{x}_1^{(1)} + \cdots + \bar{\alpha}_N\mathbf{x}_N^{(1)}, \qquad (4.56)
\end{aligned}$$

where $\bar{\alpha}_i = \alpha_i + \frac{1}{N}\hat{\alpha}$, $\hat{\alpha} = -\alpha_1 - \cdots - \alpha_N + 1$.

*Theorem 1*: Two different parallel hyperplanes $\mathbf{h}^\dagger\mathbf{x} = -b_1$ and $\mathbf{h}^\dagger\mathbf{x} = -b_2$ contain all the data in $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, respectively, if and only if

$$rank\left(\left[\mathbf{X}^{(1-\boldsymbol{\mu}_1)}, \mathbf{X}^{(2-\boldsymbol{\mu}_1)}\right]\right) = rank\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + rank\left(\mathbf{X}^{(2-\boldsymbol{\mu}_1)}\right) \qquad (4.57)$$

and

$$\boldsymbol{\mu}_1 \notin \mathbf{X}^{(2)}. \qquad (4.58)$$

For example, Theorem 1 is violated if the same sample is labeled as a Class 1 sample and as a Class 2 sample, i.e., $\mathbf{x}_i^{(1)} = \mathbf{x}_j^{(2)}$ for some sample $i$ and $j$ in Class 1 and 2, respectively.

The proof is as follows. By definition, $\mathbf{h}^\dagger\mathbf{x}_i^{(2)} = -b_2 \ \forall i$, and if $b_1 \neq b_2$, then the data are in different parallel hyperplanes. From Lemma 5, $\mathbf{h}^\dagger\mathbf{x} = -b_1 | \mathbf{x} \in \text{span}\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + \boldsymbol{\mu}_1$, and other vectors not in span $\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + \boldsymbol{\mu}_1$ can be constrained to be in a different hyperplane, e.g., $\mathbf{h}^\dagger\mathbf{x} = -b_2$. That is, if the vector $\mathbf{x}_i^{(2)} \ \forall i \notin \text{span}\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + \boldsymbol{\mu}_1$, then $\mathbf{x}_i^{(2)}$ can be constrained to be in the hyperplane $\mathbf{h}^\dagger\mathbf{x} = -b_2$. If vector $\mathbf{x}_i^{(2)} - \boldsymbol{\mu}_1$ is not in span $\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right)$, then by definition, it is linearly independent of the vectors $\mathbf{X}^{(1-\boldsymbol{\mu}_1)} = [\mathbf{x}_1^{(1)} - \boldsymbol{\mu}_1, \mathbf{x}_2^{(1)} - \boldsymbol{\mu}_1, ..., \mathbf{x}_N^{(1)} - \boldsymbol{\mu}_1]$, and therefore,

$$rank\left(\left[\mathbf{X}^{(1-\boldsymbol{\mu}_1)}, \mathbf{x}_i^{(2)} - \boldsymbol{\mu}_1\right]\right) = rank\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + rank\left(\mathbf{x}_i^{(2)} - \boldsymbol{\mu}_1\right) = rank\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + 1, \qquad (4.59)$$

given that $\mathbf{x}_i^{(2)} \neq \boldsymbol{\mu}_1$ if the vector $\mathbf{x}_i^{(2)} - \boldsymbol{\mu}_1$ is not in span $\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right)$.

To prove the reverse, if

$$rank\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) = rank\left(\left[\mathbf{X}^{(1-\boldsymbol{\mu}_1)}, \mathbf{x}_i^{(2)} - \boldsymbol{\mu}_1\right]\right) \tag{4.60}$$

then the vector $\mathbf{x}_i^{(2)} - \boldsymbol{\mu}_1$ is in span $\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right)$ and from Lemma 5, $\mathbf{x}_i^{(2)}$ has to be in the hyperplane $\mathbf{h}^\dagger\mathbf{x} = -b_1$. Therefore, $b_1 = b_2$ and two different hyperplanes do not exist.

*Theorem 2*: Two hyperplanes contain all the data in $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, respectively, if

$$rank\left(\left[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}\right]\right) = rank\left(\mathbf{X}^{(1)}\right) + rank\left(\mathbf{X}^{(2)}\right). \tag{4.61}$$

However, the reverse is not necessarily true. This is a more strict requirement than Theorem 1, i.e., if it is true, then two different hyperplanes exist, and if it is false, it does *not* mean that two different hyperplanes do not exist. For example, $\mathbf{x}_1^{(1)} = [1,0,0]^T$, $\mathbf{x}_2^{(1)} = [0,1,0]^T$, and $\mathbf{x}_1^{(2)} = [1,1,0]^T$ does not pass Theorem 2 requirements but it passes Theorem 1 requirements.

The proof is as follows. Using Theorem 1 and Lemma 6, if

$$rank\left(\left[\mathbf{X}^{(1-\boldsymbol{\mu}_1)}, \mathbf{X}^{(2-\boldsymbol{\mu}_1)}\right]\right) = rank\left(\mathbf{X}^{(1-\boldsymbol{\mu}_1)}\right) + rank\left(\mathbf{X}^{(2-\boldsymbol{\mu}_1)}\right) \tag{4.62}$$

then

$$rank\left(\left[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}\right]\right) = rank\left(\mathbf{X}^{(1)}\right) + rank\left(\mathbf{X}^{(2)}\right). \tag{4.63}$$

### 4.2.2 Relaxing equality constraints to inequality constraints

As discussed earlier, ECPSDF is a maximum margin classifier, and it can be reformulated as an SVM by including a bias and relaxing the equality constraints to inequality constraints. CFs, in general, are maximum margin classifiers. Including a bias and relaxing the equality constraints to inequality constraints can improved generalization and lead to better performance. This comes at some computational cost. ECPSDF and other CFs usually have a closed-form solution that can be efficiently computed (e.g., see Eqs. 2.8 and 2.17). Including a bias and/or using inequality constraints in the formulation prevents a closed-form solution, and requires quadratic programming techniques (see Section 4.4.2). However, since the testing time does not increase, a small increase in training computation that results in improved performance is usually a good tradeoff.

### 4.2.3 From CCF to MMCF

In Section 3.1.2, we introduced our CCF design. CCF outperformed all the other linear CFs when false-class features are available. In Section 4.2.1, we showed how ECPSDF produces a larger margin by using a bias and inequality constraints in the objective function. In fact, doing this transforms ECPSDF into the SVM, which is known to generalize well. ECPSDF is one of the first CF designed, and there have been several advances leading up to our CCF design which outperformed ECPSDF by 62%, 374%, and 68% in the three sets of experiments in Section 3.4.1.

Similarly, we designed the Maximum Margin Correlation Filter (MMCF) by using a bias and inequality constraints on CCF. That is, we seek the $\mathbf{h}$ that minimizes the CCF objective function $\mathbf{h}^\dagger\mathbf{T}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + \dot{\mathbf{g}}^\dagger\dot{\mathbf{g}}$ subject to the linear inequality constraints $\mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u}$. Including slack variables, the MMCF objective function can be expressed as follows,

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^\dagger\mathbf{T}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + 2C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{4.64}$$

where $\mathbf{T} = (1-\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}$, $\mathbf{p} = \frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\dot{\mathbf{g}}_i$ , and $\mathbf{u} = [u_1^{(1)}, \ldots, u_{N_1}^{(1)}, u_1^{(2)}, \ldots, u_{N_2}^{(2)}]$ (note that the peak constraints belonging to Class 2 are negated because of the $\mathbf{L}$ matrix).

## 4.3 MMCF SOLUTION

The MMCF objective functions in Eq. 4.22 (derived from SVM) and in Eq. 4.64 (derived from CCF) are the same, and can be solved as follows. Using Lagrange multipliers, the constrained problem in Eq. 4.64 can be expressed as follows,

$$\mathcal{L}(\mathbf{h}, \boldsymbol{\xi}, b, \mathbf{a}, \boldsymbol{\mu}) = \mathbf{h}^\dagger\mathbf{T}\mathbf{h} - 2\mathbf{h}^\dagger\mathbf{p} + 2C\mathbf{1}^T\boldsymbol{\xi} - 2\mathbf{a}^T[\mathbf{L}(\mathbf{X}^\dagger\mathbf{h} + b\mathbf{1}) - \mathbf{L}\mathbf{u} + \boldsymbol{\xi}] - 2\boldsymbol{\mu}^T\boldsymbol{\xi}, \tag{4.65}$$

where $\mathbf{a}, \boldsymbol{\mu} \geq 0$ are vectors of Lagrange multipliers. Taking the partial derivative with respect to $\mathbf{h}$ and setting it equal to zero gives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}} = 2\mathbf{T}\mathbf{h} - 2\mathbf{p} - 2\mathbf{X}\mathbf{L}\mathbf{a} = \mathbf{0}, \tag{4.66}$$

and solving for $\mathbf{h}$ gives

$$\mathbf{h} = \mathbf{T}^{-1}(\mathbf{p} + \mathbf{X}\mathbf{L}\mathbf{a}), \tag{4.67}$$

which can be expanded as

$$\begin{aligned} \mathbf{h} &= \mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{X}\mathbf{L}\mathbf{a} \\ &= \mathbf{h}_{UCF} + \mathbf{T}^{-1}\mathbf{X}\mathbf{L}\mathbf{a}, \end{aligned} \tag{4.68}$$

where $\mathbf{h}_{UCF}$ is the UCF $\mathbf{h}$ in Eq. 3.9. Taking the partial derivatives with respect to $b$ and $\xi$ and setting them equal to zero gives

$$\frac{\partial \mathcal{L}}{\partial b} = -2\mathbf{a}^T\mathbf{L}\mathbf{1} = 0, \tag{4.69}$$

and

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = 2C\mathbf{1} - 2\mathbf{a} - 2\boldsymbol{\mu} = \mathbf{0}, \tag{4.70}$$

and solving for $C\mathbf{1}$ gives

$$C\mathbf{1} = \mathbf{a} + \boldsymbol{\mu}. \tag{4.71}$$

Given that $\mathbf{a}, \boldsymbol{\mu} \geq 0$ and $\mathbf{a} = C\mathbf{1} - \boldsymbol{\mu}$ , then $\mathbf{0} \leq \mathbf{a} \leq C\mathbf{1}$.

Substituting Eqs. 4.67 and 4.71 into Eq. 4.65 gives

$$\begin{aligned}
\mathcal{L}(\mathbf{a}) &= (\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa}))^{\dagger}\mathbf{T}(\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa})) - 2(\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa}))^{\dagger}\mathbf{p} + 2(\mathbf{a}+\boldsymbol{\mu})^{T}\boldsymbol{\xi}\ldots \\
&\quad -2\mathbf{a}^{T}[\mathbf{L}(\mathbf{X}^{\dagger}(\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa}))+b\mathbf{1}) - \mathbf{Lu} + \boldsymbol{\xi}] - 2\boldsymbol{\mu}^{T}\boldsymbol{\xi} \\
&= (\mathbf{p}+\mathbf{XLa})^{\dagger}\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa}) - 2(\mathbf{p}+\mathbf{XLa})^{\dagger}\mathbf{T}^{-1}\mathbf{p} + 2\mathbf{a}^{T}\boldsymbol{\xi} + 2\boldsymbol{\mu}^{T}\boldsymbol{\xi}\ldots \\
&\quad -2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa}) - 2b\mathbf{a}^{T}\mathbf{L1} + 2\mathbf{a}^{T}\mathbf{Lu} - 2\mathbf{a}^{T}\boldsymbol{\xi} - 2\boldsymbol{\mu}^{T}\boldsymbol{\xi} \\
&= (\mathbf{p}^{\dagger}+\mathbf{a}^{T}\mathbf{LX}^{\dagger})\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa}) - 2(\mathbf{p}^{\dagger}+\mathbf{a}^{T}\mathbf{LX}^{\dagger})\mathbf{T}^{-1}\mathbf{p} - 2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}(\mathbf{p}+\mathbf{XLa})\ldots \\
&\quad +2\mathbf{a}^{T}\mathbf{Lu} \\
&= \mathbf{p}^{\dagger}\mathbf{T}^{-1}\mathbf{p} + 2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{p} + \mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{XLa} - 2\mathbf{p}^{\dagger}\mathbf{T}^{-1}\mathbf{p} - 2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{p}\ldots \\
&\quad -2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{p} - 2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{XLa} + 2\mathbf{a}^{T}\mathbf{Lu} \\
&= -\mathbf{p}^{\dagger}\mathbf{T}^{-1}\mathbf{p} - 2\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{p} - \mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{XLa} + 2\mathbf{a}^{T}\mathbf{Lu} \\
&= -\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{XLa} + 2\mathbf{a}^{T}(\mathbf{Lu} - \mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{p}) + \kappa, \\
&= -\mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{XLa} + 2\mathbf{a}^{T}\mathbf{L}(\mathbf{u} - \mathbf{u}_{UCF}) + \kappa, \qquad (4.72)
\end{aligned}$$

where $2b\mathbf{a}^{T}\mathbf{L1} = 0$ using Eq. 4.69, $\mathbf{0} \le \mathbf{a} \le C\mathbf{1}$, $\kappa = -\mathbf{p}^{\dagger}\mathbf{T}^{-1}\mathbf{p}$ does not depend on $\mathbf{a}$ and does not affect the $\mathbf{a}$ that maximizes $\mathcal{L}(\mathbf{a})$, and

$$\mathbf{u}_{UCF} = \mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{p} = \mathbf{X}^{\dagger}\mathbf{h}_{UCF} \qquad (4.73)$$

represents the peak correlation output value at the center for the UCF filter. We seek the $\mathbf{a}$ that maximizes $\mathcal{L}(\mathbf{a})$, i.e.,

$$\max_{\mathbf{0}\le\mathbf{a}\le C\mathbf{1}} \quad 2\mathbf{a}^{T}\mathbf{L}(\mathbf{u}-\mathbf{u}_{UCF}) - \mathbf{a}^{T}\mathbf{LX}^{\dagger}\mathbf{T}^{-1}\mathbf{XLa}. \qquad (4.74)$$

Eq. 4.74 is known as the dual formulation and can be solved by standard quadratic programming techniques (see Section 4.4.2).

Finally, note from Eq. 4.68 that

$$\begin{aligned}
\mathbf{h} &= \mathbf{h}_{UCF} + \mathbf{T}^{-1}\mathbf{XLa} \\
&= \mathbf{h}_{UCF} + \mathbf{h}_{0}, \qquad (4.75)
\end{aligned}$$

where $\mathbf{h}_0 = \mathbf{T}^{-1}\mathbf{X}\mathbf{L}\mathbf{a}$ is the MMCF filter when the desired correlation output $\dot{\mathbf{g}}_i = \mathbf{0}\ \forall i$.

### 4.3.1 Modified MMCF

We can design a UCF that produces the MMCF peak constraints by letting each desired $\dot{\mathbf{g}}_i$ be a scaled version of the others, i.e., $\dot{\mathbf{g}}_i = \tau_i\dot{\mathbf{g}}$, and determining the scalar $\tau_i$ for all $i$ as follows,

$$\begin{aligned}
\mathbf{u}_{MMCF} &= \mathbf{u}_{UCF} \\
&= \mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{p} \\
&= \mathbf{X}^\dagger\mathbf{T}^{-1}\left(\frac{1}{Nd}\sum_{i=1}^{N}\mathbf{X}_i\tau_i\dot{\mathbf{g}}\right) \\
&= \frac{1}{Nd}\mathbf{X}^\dagger\mathbf{T}^{-1}\dot{\mathbf{G}}\sum_{i=1}^{N}\mathbf{x}_i\tau_i \\
&= \frac{1}{Nd}\mathbf{X}^\dagger\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X}\Gamma, \qquad (4.76)
\end{aligned}$$

where $\Gamma = [\tau_1, \ldots, \tau_N]^T$, and diagonal matrix $\dot{\mathbf{G}}$ contains $\dot{\mathbf{g}}$ along the diagonal. The vector $\Gamma$ can be computed as follows,

$$\Gamma = Nd(\mathbf{X}^\dagger\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X})^{-1}\mathbf{u}. \qquad (4.77)$$

Then $\mathbf{u} = \mathbf{u}_{UCF}$ and Eq. 4.74 can be simplified to

$$\max_{\mathbf{0}\le\mathbf{a}\le C\mathbf{1}} \quad -\mathbf{a}^T\mathbf{L}\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X}\mathbf{L}\mathbf{a}. \qquad (4.78)$$

Since $\mathbf{T}$ is positive definite, then $\mathbf{L}\mathbf{X}^\dagger\mathbf{T}^{-1}\mathbf{X}\mathbf{L}$ is also positive definite and Eq. 4.78 is maximized when $\mathbf{a} = \mathbf{0}$. The filter is

$$\begin{aligned}
\mathbf{h} &= \mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{X}\mathbf{L}\mathbf{a} \\
&= \mathbf{T}^{-1}\mathbf{p} \\
&= \frac{1}{Nd}\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X}\Gamma \\
&= \mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X}(\mathbf{X}^\dagger\mathbf{T}^{-1}\dot{\mathbf{G}}\mathbf{X})^{-1}\mathbf{u} \\
&= \hat{\mathbf{T}}^{-1}\mathbf{X}(\mathbf{X}^\dagger\hat{\mathbf{T}}^{-1}\mathbf{X})^{-1}\mathbf{u}, \qquad (4.79)
\end{aligned}$$

where $\hat{\mathbf{T}} = \mathbf{T}\mathbf{G}^{-1}$. Thus, the modify MMCF solution is equivalent to mCCF from Section 3.1.2. This solution is intuitively; once $\mathbf{u} = \mathbf{u}_{UCF}$ , then all the constraints are already satisfied, and the non-UCF portion of $\mathbf{h}$ is not needed.

## 4.4 IMPLEMENTATION

### 4.4.1 Unbounded scalar variables

The $\mathbf{T}$ used in the CCF to MMCF formulation is $\mathbf{T} = (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}$ which accounts for the MSE, ASM, and ONV criteria discussed in Section 3.1. In the SVM to MMCF formulation, there is no ASM criterion used ($\gamma = 0$), and we implicitly assumed that $\mathbf{P} = \frac{1}{d}\mathbf{I}$, so that $\mathbf{T} = \mathbf{D}+\frac{\beta}{d}\mathbf{I}$ (see Eq. 4.23). In our experiments we use the more general $\mathbf{T} = (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}$, where $\gamma, \beta \geq 0$. In order to use bounder scalars, $\gamma$ and $\beta$ are replaced with bounded scalars: $\gamma$ is replaced with $\gamma = \frac{1}{\psi}(1-\psi)$ and $\beta$ with $\beta = \frac{1}{\lambda}(1-\lambda)$, where $0 \leq \psi, \lambda \leq 1$, i.e.,

$$\begin{aligned}
\mathbf{T} &= (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P} \\
&= (1+\frac{1}{\psi}(1-\psi))\mathbf{D} - \frac{1}{\psi}(1-\psi)\mathbf{M} + \frac{1}{\lambda}(1-\lambda)\mathbf{P} \\
&= \frac{1}{\lambda\psi}\left(\lambda\mathbf{D} + \lambda(1-\psi)\mathbf{M} + (1-\lambda)\psi\mathbf{P}\right) \\
&= \frac{1}{\psi}\mathbf{D} - \frac{1}{\psi}\mathbf{M} + \frac{1}{\lambda}\mathbf{P} + \mathbf{M} - \mathbf{P}, \qquad (4.80)
\end{aligned}$$

where we define the ratio $\frac{0}{0} = 1$ for the cases when $\psi = 0$ and/or $\lambda = 0$. Note that when $\gamma = 0$ only MSE and ONV are optimized, when $\beta = 0$ only MSE and ASM are optimized, when $\gamma = \beta = 0$ only MSE is optimized, when $\gamma \to \infty$ and $\beta \to \infty$ only ASM and ONV are optimized, when $\gamma \to \infty$ only ASM is optimized, and when $\beta \to \infty$ only ONV is optimized.

### 4.4.2 Sequential minimal optimization

There are several algorithms to minimize the quadratic term in Eq. 4.74. The most popular algorithm is the sequential minimal optimization (SMO) [53]. Instead of simultaneously solving for the entire $\mathbf{a} = [a_1, \ldots, a_N]^T$ vector, SMO recursively solves for different $(a_n, a_m)_{n\neq m}$ pairs and can be implemented efficiently.

## 4.5 EXPERIMENTS

We test our algorithm on real videos of vehicles. The details of our experimental setup are discussed in Chapter 7. In this section we give a brief overview of our experiments and present results comparing the performance of the filters discussed in this chapter.

We consider vehicle recognition on a set of $512 \times 640$ pixels 30 Hz infrared videos where the vehicle's class-label and location are unknown. Our dataset has eight vehicles (one vehicle in each video) driving in one circle, shown in Fig. 1.2. We consider the quadratic CFs, the best unconstrained CFs, UCF and EASEF, and all the constrained CFs from Chapter 3 using both equality and inequality constraints. We referred to the filters that have the same objective function as OTSDF and MACE but have instead inequality constraints and a bias as inequality OTSDF (iOTSDF) and inequality MACE (iMACE) respectively. Note that inequality ECPSDF is the same as SVM, and inequality CCF is the same as MMCF. In total we consider 15 different types of classifiers: UCF, EASEF, mCCF, MMCF, CCF, iOTSDF, OTSDF, iMACE, MACE, SVM, ECPSDF, TQCF, and QCF, where MMCF is the generalized CCF, and SVM is the generalized ECPSDF. For each type of classifier we train eight filters. Each filter is trained to recognize one given target (there are eight targets) for all 360° degrees of azimuth rotation. We determine target location by cross-correlating the template with the test image and determining its location by the highest value in the resulting correlation output. For the highest value, we compute the peak-to-correlation-energy (PCE) (see Eq. 3.85) and select the template (out of the eight templates–one per target) that gives the highest PCE value. We declare a correct recognition when the correct template produces the maximum response to a given frame (i.e., correct classification) *and* produces the peak within a specified window centered at the correct location (i.e., correct localization). This means that it is considered an error 1) when the largest correlation peak is *not* close to the target's ground truth location *and* is from the incorrect class, or 2) when the largest correlation peak is close to the target's ground truth location but is from the incorrect class, or 3) when the largest correlation peak is from the correct class but the peak's location is not near the target's ground truth location.

We performed three sets of experiments. In Set 1, each filter is trained using 20 true-class images. In Set 2, each filter is trained using using 20 true-class images *and* 80 background (false-class) images. In Set 3, each filter is initially trained as in Set 2 and then retrained, i.e., we cross-correlate the template with the frames from which we cropped the training images, add the false positives as false-class training images, and retrain the template. Table 4.1 shows the classification (class), localization (loc), and recognition (recog) rates for each filter in Set 1. Table 4.2 shows the classification, localization, and recognition rates for each filter in Set 2, and it shows the recognition performance improvement (impr) over the recognition rates in Set 1. Improvement is computed as new performance minus old performance, and the result divided by old performance. Table 4.3 shows the recognition rates in Set 3, and it also shows the number of retraining (ret) cycles we used (after a certain number of cycles, performance does not improve), and the recognition performance improvement over the recognition rates in Set 2. We compared different $\lambda$, $\psi$, and variance $\dot{\mathbf{g}}_{\sigma^2}$ of the desired Gaussian-function-like shape correlation output parameters and report our best recognition performance findings. The values of the parameters selected are in bold and non-bold values are default parameters for those filters.

These results show that generalizing the constrained CFs improves performance when retraining is used. As the number of false class images increases in retraining, relaxing the equality constraints to inequality constraints and including a bias improves all the constrained CFs, i.e., i.e., SVM shows a 35% improvement over ECPSDF, iMACE shows a 3% improvement over iMACE, iOTSDF shows a 9% improvement over OTSDF, and MMCF shows a 7% improvement over CCF. We observe that when few training images are used (i.e., before retraining) it is usually better to use equality constraints. This is somewhat surprising because inequality constraints have lower or equal objective function values than having equality constraints. We conjecture that in a large dimensional space when there are few images, having two hyperplanes that contains all the images may generalized better than having two hyperplanes that maximize the separation between the two classes.

Table 4.1: Filter performance (%) using true-class images only

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{g}_{\sigma^2}$ |
|---|---|---|---|---|---|---|
| TQCF | 50.8 | 72.8 | 48.9 | **0.70** | 1 | 0 |
| QCF | 32.1 | 39.9 | 27.4 | 0 | 1 | 0 |
| MMCF | 32.7 | 66.4 | 29.6 | **0.01** | **0.20** | **1.5** |
| iOTSDF | 20.9 | 55.5 | 17.4 | **0.55** | 1 | 0 |
| iMACE | 16.4 | 42.6 | 7.8 | 1 | 1 | 0 |
| SVM | 12.1 | 25.8 | 9.9 | 0 | 1 | 0 |
| CCF | 21.1 | 57.8 | 17.7 | **0.55** | **0.90** | **0.5** |
| OTSDF | 20.9 | 56.2 | 17.5 | **0.55** | 1 | 0 |
| MACE | 16.6 | 42.0 | 7.4 | 1 | 1 | 0 |
| ECPSDF | 14.4 | 27.0 | 10.9 | 0 | 1 | 0 |
| mCCF | 22.6 | 67.6 | 21.2 | **1.00** | **1.00** | **1.0** |
| UCF | 32.3 | 66.0 | 32.3 | **0.05** | **0.95** | **0** |
| EASEF | 34.1 | 67.0 | 31.8 | **0.05** | 1 | **0.5** |

Table 4.2: Filter performance (%) using true- and false-class images before retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{g}_{\sigma^2}$ | impr |
|---|---|---|---|---|---|---|---|
| TQCF | 75.3 | 76.8 | 63.7 | **0.30** | 1 | 0 | 30.3 |
| QCF | 34.8 | 48.8 | 22.4 | 0 | 1 | 0 | -34.3 |
| MMCF | 51.1 | 87.7 | 49.6 | **0.95** | **1.00** | **2.0** | 67.6 |
| iOTSDF | 50.5 | 87.6 | 49.0 | **0.95** | 1 | 0 | 181.6 |
| iMACE | 36.2 | 81.8 | 32.6 | 1 | 1 | 0 | 318.0 |
| SVM | 23.1 | 55.7 | 17.6 | 0 | 1 | 0 | 77.8 |
| CCF | 50.5 | 87.6 | 49.0 | **0.95** | 1 | 0 | 181.6 |
| OTSDF | 36.2 | 81.8 | 32.6 | 1 | 1 | 0 | 318.0 |
| MACE | 40.0 | 84.1 | 36.2 | 1 | 1 | 0 | 389.2 |
| ECPSDF | 16.1 | 63.9 | 11.1 | 0 | 1 | 0 | 1.8 |
| mCCF | 54.2 | 93.2 | 53.4 | **0.99** | **.9999** | **0.5** | 151.9 |
| UCF | 36.1 | 64.7 | 34.5 | **0.25** | **0.55** | **0.5** | 6.8 |
| EASEF | 35.4 | 68.7 | 31.8 | **0.05** | 1 | **1.0** | 0 |

The linear CFs MMCF and mCCF outperformed all the linear CFs and QCF. Linear CF require one cross-correlation in testing. TQCF always outperforms all filters, but this comes with increase computational cost. As shown in Section 2.16, QCFs (and therefore TQCFs) can be written as the sum of squares of linear CF outputs. The number of linear correlations is the number of eigenvectors used. In our experiments we used 8 eigenvectors, thus the computational testing cost of QCFs is 8 times the computational cost of the linear filters. In the next chapter, we introduced the quadratic

Table 4.3: Filter performance (%) after retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | ret | impr |
|---|---|---|---|---|---|---|---|---|
| TQCF | 87.6 | 89.7 | 82.4 | **0.20** | 1 | 0 | 8 | 29.4 |
| QCF | 57.6 | 79.3 | 54.9 | 0 | 1 | 0 | 5 | 145.1 |
| MMCF | 63.9 | 95.3 | 63.2 | **0.25** | **1.00** | **0** | 9 | 27.4 |
| iOTSDF | 63.9 | 95.3 | 63.2 | **0.25** | 1 | 0 | 9 | 29.0 |
| iMACE | 47.4 | 84.3 | 44.0 | 1 | 1 | 0 | 1 | 35.0 |
| SVM | 47.4 | 87.3 | 47.4 | 0 | 1 | 0 | 4 | 169.3 |
| CCF | 61.9 | 90.0 | 59.1 | **0.75** | **1.00** | **1.5** | 6 | 12.4 |
| OTSDF | 59.9 | 90.7 | 58.1 | **0.80** | 1 | 0 | 7 | 10.9 |
| MACE | 46.2 | 85.1 | 42.7 | 1 | 1 | 0 | 3 | 18.0 |
| ECPSDF | 38.7 | 80.2 | 35.1 | 0 | 1 | 0 | 7 | 216.2 |
| mCCF | 65.8 | 92.4 | 64.7 | **.999** | **.9999** | **1.0** | 4 | 21.2 |
| UCF | 37.9 | 77.9 | 36.5 | **0.15** | **0.99** | **1.0** | 5 | 5.8 |
| EASEF | 33.9 | 66.5 | 32.9 | **0.05** | 1 | **0** | 3 | 3.5 |

MMCF (QMMCF), and we will show that it can outperforms all the previously mentioned filters including TQCF.

We also observed that retraining improves the performance of ECPSDF by 216% and SVM by 169%. This is more improvement than the observed from other filters. ECPSDF and SVM used the ONV criterion and ignore the MSE and ASM criteria. In general, we observe that retraining improves the performance when the ONV criterion is emphasized. That is, when there are few training images, choosing a $\lambda$ that is less than but close to 1 usually gives the best performance. Recall that $\lambda = 1$ ignores the ONV (or margin) criterion and $\lambda = 0$ only uses the ONV criterion and ignores the other criteria (see Eq. 4.80). As the number of training images increases (in our experiments this happens in retraining), performance usually increases when keeping $\lambda$ constant, and lowering the value of $\lambda$ further increases the performance. For example, before retraining, MMCF's best performance is when $\lambda = 0.95$ with 49.6%. After retraining using $\lambda = 0.95$, MMCF's performance increases to 54.3% (not shown in the tables above); however, the overall best performance after retraining is when $\lambda = 0.25$ with 63.2%. This observation is more apparent in the generalized constrained CFs than the regular constrained CFs, although it usually happens in both. This happens because emphasizing ONV results in a larger margin of separation between true- and false-classes.

When ONV is emphasized, the localization criterion is de-emphasized. This results in many false-positive peaks before retraining. After retraining, many of the false-positive are included as false-class images. This improves classification and localization and reduces (but does not eliminates) the need for the localization criterion.

## 4.6 Summary

We presented the similarities between SVMs and CFs, and we connected these two state-of-the-art algorithms with the MMCF. The MMCF classifier is less susceptible to over-fitting than traditional CFs while providing geometric shift-invariance to SVM classifiers. We concluded this chapter providing experimental results demonstrating that MMCF outperforms SVM, UCF, CCF, and QCF filters.

CHAPTER **5**

# QUADRATIC MMCF

In the previous chapter we showed the relation between linear CFs and SVMs. In this chapter we show the relation between quadratic correlation filters (QCFs) and Quadratic SVMs (QSVMs) also known as second order polynomial Kernel SVMs. We combine the design principles of linear CFs and QCFs and introduce the Quadratic Maximum Margin Correlation Filter (QMMCF) of which QSVM is a subset. QMMCF is better able to exploit the higher-order statistics of the data resulting in superior performance. This improved performance comes at the cost of added computation in testing. Our experiments show that QMMCF outperforms all previously discussed filters including QSVMs. Note that all notation in this chapter is in the spatial domain. Thus, we avoid the usual inverse hat symbol (e.g., $\check{x}$) to clean up the notation because there is no need to differentiate between frequency and space domain.

## 5.1 FROM QCF TO QSVM

As discussed in Section 2.16, the QCF design maximizes the difference between the mean values $E_c\{\mathbf{x}^T\mathbf{Q}\mathbf{x}\}$ of each class $c$. An approach that may generalized better is to maximize the difference between the minimum value of $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ when $\mathbf{x}$ belongs to Class 1 and the maximum value of $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ when $\mathbf{x}$ belongs to Class 2. This is a maximum margin problem that has a loss function which does not penalize correct values, and hence it may provide superior performance over QCFs. This can be written as

$$\max_{\mathbf{Q}}\left(\frac{1}{|\mathbf{Q}|_F}\left(\min\left(\mathbf{x}_1^{(1)T}\mathbf{Q}\mathbf{x}_1^{(1)},\ldots,\mathbf{x}_{N_1}^{(1)T}\mathbf{Q}\mathbf{x}_{N_1}^{(1)},-\mathbf{x}_1^{(2)T}\mathbf{Q}\mathbf{x}_1^{(2)},\ldots,-\mathbf{x}_{N_2}^{(2)T}\mathbf{Q}\mathbf{x}_{N_2}^{(2)}\right)\right)\right), \quad (5.1)$$

where $\mathbf{Q}$ is a $d \times d$ matrix, $\mathbf{x}_i^{(1)T}\mathbf{Q}\mathbf{x}_i^{(1)}$ $\forall i = 1,\ldots,N_1$ is the output of the $i$th training image in Class 1, $N_1$ is the number of Class 1 training images, $\mathbf{x}_i^{(2)T}\mathbf{Q}\mathbf{x}_i^{(2)}$ $\forall i = 1,\ldots,N_2$ is the output of the $i$th training image in Class 2, $N_2$ is the number of Class 2 training images, and $|\cdot|_F$ is the Frobenius norm. Note that we divide by the Frobenius norm of $\mathbf{Q}$ because otherwise for some possible $\mathbf{Q}$, $\kappa\mathbf{Q}$ (with $\kappa > 1$) would gives a higher value in Eq. 5.1.

Eq. 5.1 can be written as follows,

$$\max_{\mathbf{Q}}\left(\min_i\left(\frac{l_i(\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i)}{|\mathbf{Q}|_F}\right)\right), \quad (5.2)$$

where $l_i \in \{-1,1\}$ $\forall i \in \{1,\ldots,N\}$ corresponds to the class-label of $\mathbf{x}_i$, and $N = N_1 + N_2$. This is a convex problem ($\mathbf{Q}$ is the unknown and therefore does *not* need to be restricted to be positive definite to make the problem convex). There exists a Frobenius norm $|\mathbf{Q}|_F$ such that $\min_i\ l_i(\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i) = 1$, or equivalently, $l_i\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i \geq 1$ for all $i$. Using these constraints, we rewrite Eq. 5.2 as follows,

$$\begin{aligned}\max_{\mathbf{Q}} \quad & \frac{1}{|\mathbf{Q}|_F} \\ s.t. \quad & l_i\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i \geq 1,\end{aligned} \quad (5.3)$$

or equivalently,

$$\begin{aligned}\min_{\mathbf{Q}} \quad & |\mathbf{Q}|_F^2 \\ s.t. \quad & l_i\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i \geq 1.\end{aligned} \quad (5.4)$$

A more general form is

$$\begin{aligned}\min_{\mathbf{Q}} \quad & |\mathbf{Q}|_F^2 + 2C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & l_i\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i + b \geq l_iu_i - \xi_i,\end{aligned} \quad (5.5)$$

where $\xi_i$ is a penalty term known as a *slack variable* that allows some points to be on the wrong side of the margin at some cost, and $2C$ is a tradeoff parameter that weighs the cost (the “2” is included to avoid fractions when deriving the dual formulation). To simplify notation we show the solution to the simpler formulation in Eq. 5.4 and note that the solution to Eq. 5.5 requires only a few additional steps.

Using Lagrange multipliers, the constrained problem in Eq. 5.4 can be expressed as follows,

$$\mathcal{L}(\mathbf{Q}, \mathbf{a}) = |\mathbf{Q}|_F^2 - \sum_{i=1}^{N} 2a_i(l_i \mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i - 1), \tag{5.6}$$

where $a_i \geq 0$ for all $i$. Eq. 5.6 can be re-written as

$$\mathcal{L}(\mathbf{Q}, \mathbf{a}) = \sum_{n,m=1}^{d} Q^2[n,m] - \sum_{i=1}^{N} 2a_i \left( l_i \sum_{n,m=1}^{d} x_i[n]x_i[m]Q_{ij} - 1 \right), \tag{5.7}$$

where

$$\mathbf{Q} = \begin{bmatrix} Q_{11} & \cdots & Q_{1d} \\ \vdots & \ddots & \vdots \\ Q_{d1} & \cdots & Q_{dd} \end{bmatrix}, \tag{5.8}$$

$Q[n,m]$ refers to the $(n,m)$ value of matrix $\mathbf{Q}$,

$$|\mathbf{Q}|_F^2 = \sum_{n,m=1}^{d} Q^2[n,m] \tag{5.9}$$

is the Frobenius norm, and $x_i[m]$ refers to the the $m$th value of vector $\mathbf{x}_i$. Note that

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Q}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial Q[1,1]} & \cdots & \frac{\partial \mathcal{L}}{\partial Q[1,d]} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial Q[d,1]} & \cdots & \frac{\partial \mathcal{L}}{\partial Q[d,d]} \end{bmatrix}. \tag{5.10}$$

Taking the derivative with respect to $Q_{nm}$ and setting it equal to zero gives

$$\frac{\partial \mathcal{L}}{\partial Q[n,m]} = 2Q[n,m] - \sum_{i=1}^{N} 2a_i l_i x_i[n]x_i[m] = 0, \tag{5.11}$$

and solving for $Q[n,m]$ gives

$$Q[n,m] = \sum_{i=1}^{N} a_i l_i x_i[n]x_i[m]. \tag{5.12}$$

Plugging Eq. 5.12 into Eq. 5.7 gives the dual formulation,

$$\begin{aligned}
\mathcal{L}(\mathbf{a}) &= \sum_{n,m}\left(\sum_i a_i l_i x_i[n]x_i[m]\right)^2 - \sum_i 2a_i\left(l_i\sum_{n,m} x_i[n]x_i[m]\sum_j a_j l_j x_j[n]x_j[m] - 1\right) \\
&= \sum_{n,m}\sum_{i,j} a_i a_j l_i l_j x_i[n]x_j[n]x_i[m]x_j[m] - \sum_{n,m}\sum_{i,j} 2a_i a_j l_i l_j x_i[n]x_j[n]x_i[m]x_j[m] ... \\
&\quad +2\sum_i a_i \qquad (5.13) \\
&= -\sum_{i,j=1}^{N} a_i a_j l_i l_j \sum_{n=1}^{d} x_i[n]x_j[n]\sum_{m=1}^{d} x_i[m]x_j[m] + 2\sum_{i=1}^{N} a_i \\
&= 2\sum_{i=1}^{N} a_i - \sum_{i,j=1}^{N} a_i a_j l_i l_j (\mathbf{x}_i^T\mathbf{x}_j)^2 \\
&= 2\mathbf{a}^T\mathbf{1} - \mathbf{a}^T\boldsymbol{\mathcal{X}}\mathbf{a}, \qquad (5.14)
\end{aligned}$$

where

$$\boldsymbol{\mathcal{X}} = \begin{bmatrix} l_1 l_1(\mathbf{x}_1^T\mathbf{x}_1)^2 & \cdots & l_1 l_N(\mathbf{x}_1^T\mathbf{x}_N)^2 \\ \vdots & \ddots & \vdots \\ l_N l_1(\mathbf{x}_N^T\mathbf{x}_1)^2 & \cdots & l_N l_N(\mathbf{x}_N^T\mathbf{x}_N)^2 \end{bmatrix}. \qquad (5.15)$$

Eq. 5.14 is the dual formulation and can be solved by standard quadratic programming techniques (see Section 4.4.2). Once we compute $\mathbf{a}$, matrix $\mathbf{Q}$ can be computed using Eq. 5.12 as follows,

$$\mathbf{Q} = \sum_{(i:a_i\neq 0)} a_i l_i \mathbf{x}_i\mathbf{x}_i^T. \qquad (5.16)$$

In testing, matrix $\mathbf{Q}$ can be applied to a vectorized image $\mathbf{x}'$ (equal in size to a training image) as follows,

$$\begin{aligned}
\mathbf{x}'^T\mathbf{Q}\mathbf{x}' &= \mathbf{x}'^T\left(\sum_{(i:a_i\neq 0)} a_i l_i \mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{x}' \\
&= \sum_{(i:a_i\neq 0)} a_i l_i \mathbf{x}'^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{x}' \\
&= \sum_{(i:a_i\neq 0)} a_i l_i (\mathbf{x}_i^T\mathbf{x}')^2 \\
&= \sum_{(i:a_i\neq 0)} a_i l_i K(\mathbf{x}_i, \mathbf{x}'), \qquad (5.17)
\end{aligned}$$

where $K(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1^T \mathbf{v}_2)^2$ is known as a second-order polynomial kernel. From this observation we note that this maximum margin QCF is equivalent to solving a Quadratic SVM (QSVM) also known as a second-order polynomial kernel SVM. Although using QSVMs in not novel, making the connection between QCFs and QSVMs is novel.

Testing QSVMs with a large query image requires the same number of cross-correlations as the number of support vectors. Let $\mathbf{z}$ be a test image, and $\mathbf{x}_i$ be the $n$th training image. The correlation output is

$$\mathbf{g} = \sum_{(i:a_i \neq 0)} a_i l_i \left(\mathbf{z} \otimes \mathbf{x}_i\right)^2 . \tag{5.18}$$

The main computational load is equal to $s + 1$ 2-D DFTs, where $s$ is the number of support vectors.

## 5.2 QMMCF: Generalized Maximum Margin Quadratic Correlation Filters

**Expressing the quadratic correlation output using matrix notation** In this section we give an overview on how to express the quadratic correlation output using matrix notation. The mathematical details can be found in Appendix D. The QCF output for the $i$th training feature vector, $\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i$, can be expressed as follows,

$$\begin{aligned} \mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i &= \mathbf{x}_i^T [\mathbf{q}_1, \ldots, \mathbf{q}_d] \mathbf{x}_i \\ &= [\mathbf{x}_i^T \mathbf{q}_1, \ldots, \mathbf{x}_i^T \mathbf{q}_d] \mathbf{x}_i, \end{aligned} \tag{5.19}$$

where $\mathbf{q}_1$ to $\mathbf{q}_d$ are the column vectors of $d \times d$ matrix $\mathbf{Q}$. If $\mathbf{x}_i$ represents a 1-D feature vector (instead of a vectorized 2-D feature image), the correlation output can be computed as follows (the first index of $\mathbf{q}_1$ is represented by $q[l, k] = q[0, 0]$),

$$\sum_{k=0}^{d-1} \sum_{l=0}^{d-1} x_i[n + k] q[l, k] x_i[n + l]. \tag{5.20}$$

Eq. 5.20 cannot be trivially expressed in the frequency domain, and therefore the derivation is done in the spatial domain.

When $\mathbf{x}_i$ represents a 1-D feature, the quadratic correlation output in Eq. 5.20 can be written as,

$$\mathbf{g}_i = \mathbf{X}_{\mathbf{C}i}^T \mathbf{h}, \tag{5.21}$$

where

$$\mathbf{h} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_d \end{bmatrix} \tag{5.22}$$

is a vector of length $d^2$,

$$\mathbf{X}_{\mathbf{C}i} = \begin{bmatrix} \pi_1 \mathbf{x}_i & \pi_1 \mathbf{x}_i^{(1\downarrow)} & \dots & \pi_1 \mathbf{x}_i^{(d-1\downarrow)} \\ \pi_2 \mathbf{x}_i & \pi_2 \mathbf{x}_i^{(1\downarrow)} & \dots & \pi_2 \mathbf{x}_i^{(d-1\downarrow)} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_d \mathbf{x}_i & \pi_d \mathbf{x}_i^{(1\downarrow)} & \dots & \pi_d \mathbf{x}_i^{(d-1\downarrow)} \end{bmatrix} \tag{5.23}$$

is a $d^2 \times d$ matrix, $\mathbf{g}_i$ is a vector of length $d$, $\mathbf{x}_i^{(r\downarrow)}$ represents the vector $\mathbf{x}_i$ shifted by $r$ pixels, and $\pi_l \mathbf{x}_i^{(r\downarrow)}$ represents the shifted vector multiplied by its $l$th vector value, i.e., $\pi_l \mathbf{x}_i^{(r\downarrow)} = \mathbf{x}_i^{(r\downarrow)}[l]\mathbf{x}_i^{(r\downarrow)}$.

In our experiments, $\mathbf{x}_i$ represents a 2-D feature image. In this case, the quadratic correlation output when $x[n,m]$ is a 2-D $R \times C$ feature image is

$$g_i[n,m] = \sum_{l=0}^{R-1} \sum_{k=0}^{R-1} \sum_{u=0}^{C-1} \sum_{v=0}^{C-1} x_i[n+l, m+u] x_i[n+k, m+v] q_{l+1,u+1}[k,v], \tag{5.24}$$

where the columns of $\mathbf{Q}$ are each rearrange as 2-D $R \times C$ arrays. The quadratic correlation output in Eq. 5.24 can be written as

$$\mathbf{g}_i = \mathbf{X}_{\mathbf{C}i}^T \mathbf{h}, \tag{5.25}$$

where

$$
\mathbf{X}_{\mathbf{C}i} = \begin{bmatrix}
\pi_1\mathbf{x}_i & \dots & \pi_1\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_1\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_1\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\pi_2\mathbf{x}_i & \dots & \pi_2\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_2\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_2\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\vdots & & \vdots & & \vdots & & \vdots \\
\pi_{C-1}\mathbf{x}_i & \dots & \pi_{C-1}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{C-1}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{C-1}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\pi_{C}\mathbf{x}_i & \dots & \pi_{C}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{C}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{C}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\pi_{C+1}\mathbf{x}_i & \dots & \pi_{C+1}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{C+1}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{C+1}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\vdots & & \vdots & & \vdots & & \vdots \\
\pi_{2C-1}\mathbf{x}_i & \dots & \pi_{2C-1}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{2C-1}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{2C-1}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\pi_{2C}\mathbf{x}_i & \dots & \pi_{2C}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{2C}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{2C}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\pi_{2C+1}\mathbf{x}_i & \dots & \pi_{2C+1}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{2C+1}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{2C+1}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\
\vdots & & \vdots & & \vdots & & \vdots \\
\pi_{d}\mathbf{x}_i & \dots & \pi_{d}\mathbf{x}_i^{(R-1\downarrow)} & \dots & \pi_{d}\mathbf{x}_i^{(\overrightarrow{C-1})} & \dots & \pi_{d}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})}
\end{bmatrix} \tag{5.26}
$$

is a $d^2 \times d$ matrix, $\mathbf{x}_i^{(r\downarrow)(c\rightarrow)}$ represents the vectorized 2-D $R \times C$ feature image shifted by $r$ pixels down and $c$ pixels to the right. The vector that starts at the $(n, m)$ entry of this $\mathbf{X}_{\mathbf{C}i}^T$ matrix is

$$
\pi_i \mathbf{x}_i^{\left(m\%R\downarrow,\overrightarrow{\lfloor \frac{m}{R} \rfloor - 1}\right)}, \tag{5.27}
$$

where $\%$ represents the modulus (or remainder) operator, and $\lfloor \cdot \rfloor$ represents the floor operator.

**QMMCF criteria** We use the criteria discussed in Section 3.1. The MSE can be computed as follows,

$$\begin{aligned}
\text{MSE} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{g}_i - \dot{\mathbf{g}}_i)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{g}_i^T\mathbf{g}_i - 2\mathbf{g}_i^T\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^T\dot{\mathbf{g}}_i\right) \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{h}^T\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T\mathbf{h} - 2\mathbf{h}^T\mathbf{X}_{\mathbf{C}i}\dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^T\dot{\mathbf{g}}_i\right) \\
&= \mathbf{h}^T\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T\right)\mathbf{h} - 2\mathbf{h}^T\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\dot{\mathbf{g}}_i\right) + \frac{1}{N}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^T\dot{\mathbf{g}}_i \\
&= \mathbf{h}^T\mathbf{D}\mathbf{h} - 2\mathbf{h}^T\mathbf{p} + E_f, \qquad (5.28)
\end{aligned}$$

where $d^2 \times d^2$ matrix $\mathbf{D} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T$ is the average of the autocorrelation matrices for the training vectors, $\mathbf{h}^T\mathbf{p} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\dot{\mathbf{g}}_i$ is a measurement of the similarity between the actual correlation planes and the desired correlation planes with

$$\mathbf{p} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\dot{\mathbf{g}}_i, \qquad (5.29)$$

and $E_f = \frac{1}{N}\sum_{i=1}^{N}\dot{\mathbf{g}}_i^T\dot{\mathbf{g}}_i$ is the average energy of the desired correlation planes.

The ASM can be computed as follows,

$$\begin{aligned}
\text{ASM} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{g}_i - \bar{\mathbf{g}})^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{g}_i^T\mathbf{g}_i - 2\mathbf{g}_i^T\bar{\mathbf{g}} + \bar{\mathbf{g}}^T\bar{\mathbf{g}}) \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\mathbf{g}_i - 2\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\right)\bar{\mathbf{g}} + \bar{\mathbf{g}}^T\bar{\mathbf{g}} \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\mathbf{g}_i - \bar{\mathbf{g}}^T\bar{\mathbf{g}} \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}^T\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T\mathbf{h} - \mathbf{h}^T\bar{\mathbf{X}}_{\mathbf{C}}\bar{\mathbf{X}}_{\mathbf{C}}^T\mathbf{h} \\
&= \mathbf{h}^T\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T\right)\mathbf{h} - \mathbf{h}^T\bar{\mathbf{X}}_{\mathbf{C}}\bar{\mathbf{X}}_{\mathbf{C}}^T\mathbf{h} \\
&= \mathbf{h}^T\mathbf{D}\mathbf{h} - \mathbf{h}^T\mathbf{M}\mathbf{h}, \qquad (5.30)
\end{aligned}$$

where

$$\bar{\mathbf{g}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}^T\mathbf{h} = \bar{\mathbf{X}}_{\mathbf{C}}^T\mathbf{h}, \qquad (5.31)$$

and $d^2 \times d^2$ matrix $\mathbf{M} = \bar{\mathbf{X}}_{\mathbf{C}}\bar{\mathbf{X}}_{\mathbf{C}}^T$.

The ONV can be computed as follows,

$$\begin{aligned}
\text{ONV} &= \sigma^2 \\
&= E\{(\mathbf{h}^T\boldsymbol{\eta})^2\} \\
&= \mathbf{h}^T E\{\boldsymbol{\eta}\boldsymbol{\eta}^T\}\mathbf{h} \\
&= \mathbf{h}^T\mathbf{P}\mathbf{h}, \qquad (5.32)
\end{aligned}$$

where $d^2 \times d^2$ matrix $\mathbf{P}$ is the autocorrelation matrix of the additive noise $\boldsymbol{\eta}$. In many scenarios, unit-variance white noise is assumed, i.e., $\mathbf{P} = \mathbf{I}$.

The QCF output for the $i$th training feature vector, $\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i$, can be written as an inner product as follows,

$$\begin{aligned}\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i &= \mathbf{x}_i^T[\mathbf{q}_1,\ldots,\mathbf{q}_d]\mathbf{x}_i \\ &= \mathbf{h}^T\mathbf{y}_i, \end{aligned} \tag{5.33}$$

where

$$\mathbf{h} = \begin{bmatrix}\mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_d\end{bmatrix}, \quad \mathbf{y}_i = \begin{bmatrix}\pi_1\mathbf{x}_i \\ \pi_2\mathbf{x}_i \\ \vdots \\ \pi_d\mathbf{x}_i\end{bmatrix}, \tag{5.34}$$

where $\pi_i\mathbf{x}_i$ represents $\mathbf{x}_i$ multiplied by its $i$th value, and $\mathbf{q}_1$ to $\mathbf{q}_d$ are the column vectors of matrix $\mathbf{Q}$. Note that the vectors $\mathbf{y}_i$ and $\mathbf{h}$ are of dimension $d^2$, and vector $\mathbf{x}_i$ is of dimension $d$.

**QMMCF objective function** We designed the QMMCF minimizing the criteria MSE, ASM, and ONV subject to the linear constraints $l_i(\mathbf{h}^T\mathbf{y}_i + b) \geq l_i u_i$. Refregier [55] showed (and we also show this in Eq. 3.36) that an optimal tradeoff among quadratic criteria can be obtained by minimizing a weighted sum of the criteria. Thus, the QMMCF objective function

$$\begin{aligned}\min_{\mathbf{h}} \quad & (\text{MSE, ONV, ASM}) \\ s.t. \quad & \mathbf{L}(\mathbf{Y}^T\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u},\end{aligned} \tag{5.35}$$

can be expressed as follows,

$$\begin{aligned}\min_{\mathbf{h},b} \quad & \mathbf{h}^T\mathbf{T}\mathbf{h} - 2\mathbf{h}^T\mathbf{p} \\ s.t. \quad & \mathbf{L}(\mathbf{Y}^T\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u},\end{aligned} \tag{5.36}$$

where $\mathbf{T} = (1-\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}$ (see Eq. 3.36), diagonal matrix $\mathbf{L}$ contains the vector labels $l_i$ along the diagonal, $\mathbf{Y} = [\mathbf{y}_1,\ldots,\mathbf{y}_N]$, and $\mathbf{u} = [u_1^{(1)},\ldots,u_{N_1}^{(1)}, u_1^{(2)},\ldots,u_{N_2}^{(2)}]$ (note that the peak constraints belonging to Class 2 are negated because of the $\mathbf{L}$ matrix). Including slack variables, the objective function is

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^T\mathbf{T}\mathbf{h} - 2\mathbf{h}^T\mathbf{p} + 2C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{Y}^T\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{5.37}$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^T$. This is the same objective function as Eq. 4.64 (replacing the $\mathbf{X}$ by $\mathbf{Y}$). The dual formulation is (see derivation leading to Eq. 4.74),

$$\max_{\mathbf{0}\leq\mathbf{a}\leq C\mathbf{1}} \quad 2\mathbf{a}^T\mathbf{L}(\mathbf{u} - \mathbf{u}_{UCF}) - \mathbf{a}^T\mathbf{L}\mathbf{Y}^\dagger\mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a}, \tag{5.38}$$

where

$$\mathbf{u}_{UCF} = \mathbf{Y}^T\mathbf{T}^{-1}\mathbf{p} = \mathbf{Y}^\dagger\mathbf{h}_{UCF} \tag{5.39}$$

represents the peak correlation output value at the center for the UCF filter.

### 5.2.1 Zero desired correlation output

When the desired correlation output $\dot{\mathbf{g}}_i = \mathbf{0}$ for all $i$, then $\mathbf{p} = \mathbf{0}$, and the primal objective function can be expressed as

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^T\mathbf{T}\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{Y}^T\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{5.40}$$

where

$$\mathbf{h} = \mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a} \tag{5.41}$$

and the dual can be expressed as

$$\max_{\mathbf{0}\leq\mathbf{a}\leq C\mathbf{1}} \quad 2\mathbf{a}^T\mathbf{L}\mathbf{u} - \mathbf{a}^T\mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a}. \tag{5.42}$$

Since $\mathbf{T}$ is positive definite, the data can be transformed such that

$$\tilde{\mathbf{h}} = \mathbf{T}^{\frac{1}{2}}\mathbf{h} \tag{5.43}$$

and

$$\tilde{\mathbf{y}}_i = \mathbf{T}^{-\frac{1}{2}}\mathbf{y}_i, \tag{5.44}$$

and the QMMCF primal objective function can be written as

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \tilde{\mathbf{h}}^T\tilde{\mathbf{h}} + C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\tilde{\mathbf{Y}}^T\tilde{\mathbf{h}} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}, \end{aligned} \tag{5.45}$$

and the dual as

$$\max_{\mathbf{0}\leq\mathbf{a}\leq C\mathbf{1}} \quad 2\mathbf{a}^T\mathbf{L}\mathbf{u} - \mathbf{a}^T\mathbf{L}\tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}\mathbf{L}\mathbf{a}, \tag{5.46}$$

where $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_N]$. This means that this QMMCF design can be implemented using a standard SVM solver by using transform images to find $\tilde{\mathbf{h}}$. Then $\mathbf{h}$ can be computed as follows,

$$\mathbf{h} = \mathbf{T}^{-\frac{1}{2}}\tilde{\mathbf{h}}. \tag{5.47}$$

### 5.2.2 QSVM: A special case of QMMCF

When $\mathbf{T} = \mathbf{I}$ and $\mathbf{p} = \mathbf{0}$ (e.g., when $\beta \to \infty$ and $\mathbf{P} = \mathbf{I}$ which means that we only consider the ONV criterion and assume additive white noise), then the QMMCF primal objective function can be expressed as

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^T\mathbf{h} + 2C\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & \mathbf{L}(\mathbf{Y}^T\mathbf{h} + b\mathbf{1}) \geq \mathbf{L}\mathbf{u} - \boldsymbol{\xi}. \end{aligned} \tag{5.48}$$

We can show that this is the same formulation as Eq. 5.5 by rewriting Eq. 5.5 as follows,

$$\begin{aligned} \min_{\mathbf{Q}} \quad & \sum_{i,j=1}^{d} Q_{ij}^2 + 2C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & l_i(\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i + b) \geq l_i u_i - \xi_i. \end{aligned} \tag{5.49}$$

The scalars $\sum_{i,j=1,}^{d} Q_{ij}^2$ can be expressed as $\mathbf{h}^T\mathbf{h}$, and $\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i$ can be expressed as $\mathbf{h}^T\mathbf{y}$ (see Eq. 5.33). Thus, Eqs. 5.5 and 5.48 are equivalent. The dual formulations are therefore equivalent, but this can also be shown as follows. When $\mathbf{T} = \mathbf{I}$, $\mathbf{p} = \mathbf{0}$, $\mathbf{u} = \mathbf{L1}$, and $C \to \infty$, Eq. 5.38 can be written as

$$\max_{\mathbf{a} \geq \mathbf{0}} \quad 2\mathbf{a}^T\mathbf{1} - \mathbf{a}^T\mathbf{L}\mathbf{Y}^T\mathbf{Y}\mathbf{L}\mathbf{a}, \tag{5.50}$$

or equivalently,

$$\max_{\mathbf{a} \geq \mathbf{0}} \quad 2\mathbf{a}^T\mathbf{1} - \mathbf{a}^T\boldsymbol{\mathcal{X}}\mathbf{a}, \tag{5.51}$$

where

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \mathbf{L}\mathbf{Y}^T\mathbf{Y}\mathbf{L} \\ &= \begin{bmatrix} l_1 l_1 \mathbf{y}_1^T \mathbf{y}_1 & \cdots & l_1 l_N \mathbf{y}_1^T \mathbf{y}_N \\ \vdots & \ddots & \vdots \\ l_N l_1 \mathbf{y}_N^T \mathbf{y}_1 & \cdots & l_N l_N \mathbf{y}_N^T \mathbf{y}_N \end{bmatrix}. \end{aligned} \tag{5.52}$$

The inner product

$$\begin{aligned} \mathbf{y}_i^T\mathbf{y}_j &= \begin{bmatrix} x_i^{(1)}\mathbf{x}_i \\ x_i^{(2)}\mathbf{x}_i \\ \vdots \\ x_i^{(d)}\mathbf{x}_i \end{bmatrix}^T \begin{bmatrix} x_j^{(1)}\mathbf{x}_j \\ x_j^{(2)}\mathbf{x}_j \\ \vdots \\ x_j^{(d)}\mathbf{x}_j \end{bmatrix} \\ &= \sum_{m=1}^{d} x_i^{(m)} x_j^{(m)} \mathbf{x}_i^T \mathbf{x}_j \\ &= \sum_{m=1}^{d} x_i^{(m)} x_j^{(m)} \sum_{p=1}^{d} x_i^{(p)} x_j^{(p)} \\ &= \left( \sum_{m=1}^{d} x_i^{(m)} x_j^{(m)} \right)^2 \\ &= (\mathbf{x}_i^T\mathbf{x}_j)^2. \end{aligned} \tag{5.53}$$

Substituting Eq. 5.53 into Eq. 5.52 gives,

$$\boldsymbol{\mathcal{X}} = \begin{bmatrix} l_1 l_1 (\mathbf{x}_1^T \mathbf{x}_1)^2 & \cdots & l_1 l_N (\mathbf{x}_1^T \mathbf{x}_N)^2 \\ \vdots & \ddots & \vdots \\ l_N l_1 (\mathbf{x}_N^T \mathbf{x}_1)^2 & \cdots & l_N l_N (\mathbf{x}_N^T \mathbf{x}_N)^2 \end{bmatrix}. \tag{5.54}$$

which is equivalent to 5.15, thus showing that the two dual formulations are equivalent.

### 5.2.3 Closed form solution (no quadratic programming)

The QMMCF solution can be computed without quadratic programming if the inequality constraints are replaced by equality constraints and the bias is ignored. The objective function simplifies to

$$\begin{aligned} \min_{\mathbf{h},b} \quad & \mathbf{h}^T \mathbf{T} \mathbf{h} - 2\mathbf{h}^T \mathbf{p} \\ s.t. \quad & \mathbf{L} \mathbf{Y}^T \mathbf{h} = \mathbf{u}. \end{aligned} \tag{5.55}$$

Using Lagrange multipliers, the constrained problem in Eq. 5.55 can be expressed as follows,

$$\mathcal{L}(\mathbf{h}, \mathbf{a}) = \mathbf{h}^T \mathbf{T} \mathbf{h} - 2\mathbf{h}^T \mathbf{p} - 2\mathbf{a}^T (\mathbf{L} \mathbf{Y}^T \mathbf{h} - \mathbf{u}), \tag{5.56}$$

where $\mathbf{a} \neq \mathbf{0}$ is a vector of $N$ non-zero Lagrange multipliers. We seek to maximize the vector $\mathbf{a}$ that minimizes $\mathcal{L}(\mathbf{h}, \mathbf{a})$. Taking the gradient and setting it equal to zero gives,

$$\frac{\partial \mathcal{L}(\mathbf{h}, \mathbf{a})}{\partial \mathbf{h}} = 2\mathbf{T}\mathbf{h} - 2\mathbf{p} - 2\mathbf{Y}\mathbf{L}\mathbf{a} = \mathbf{0}, \tag{5.57}$$

and solving for $\mathbf{h}$ gives

$$\begin{aligned} \mathbf{h} &= \mathbf{T}^{-1}(\mathbf{p} + \mathbf{Y}\mathbf{L}\mathbf{a}) \\ &= \mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a}. \end{aligned} \tag{5.58}$$

Substituting $\mathbf{h}$ in Eq. 5.58 into the objective function constraints in Eq. 5.55 gives

$$\begin{aligned}\mathbf{u} &= \mathbf{L}\mathbf{Y}^T\mathbf{h} \\ &= \mathbf{L}\mathbf{Y}^T(\mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a}) \\ &= \mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{p} + \mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a}, \end{aligned} \tag{5.59}$$

and the Lagrange multiplier vector $\mathbf{a}$ can be computed as follows,

$$\mathbf{a} = (\mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{Y}\mathbf{L})^{-1}(\mathbf{u} - \mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{p}). \tag{5.60}$$

Substituting $\mathbf{a}$ in Eq. 5.60 back into $\mathbf{h}$ in Eq. 5.58 gives the QMMCF closed form solution,

$$\begin{aligned}\mathbf{h} &= \mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{X}\mathbf{a} \\ &= \mathbf{T}^{-1}\mathbf{p} + \mathbf{T}^{-1}\mathbf{Y}\mathbf{L}(\mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{Y}\mathbf{L})^{-1}(\mathbf{u} - \mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{p}). \end{aligned} \tag{5.61}$$

## 5.3 Implementation

The draw back of QMMCF is constructing a non-diagonal $d^2 \times d^2$ matrix $\mathbf{T}$ and taking the inverse. Efficient implementation is a topic of future research. In this section we discuss possible methods to implement QMMCF.

Recall that matrix $\mathbf{T}$ is

$$\mathbf{T} = (1-\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}, \tag{5.62}$$

where $\mathbf{D} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T$, $\mathbf{M} = \bar{\mathbf{X}}_{\mathbf{C}}\bar{\mathbf{X}}_{\mathbf{C}}^T$, and $\mathbf{P} = \mathbf{I}$ are $d^2 \times d^2$ matrices, and $\mathbf{X}_{\mathbf{C}i}$ and $\bar{\mathbf{X}}_{\mathbf{C}}$ are $d^2 \times d$ matrices. Matrix $\mathbf{D}$ can be written as

$$\mathbf{D} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{\mathbf{C}i}\mathbf{X}_{\mathbf{C}i}^T = \frac{1}{N}\mathbf{X}_{\mathbf{C}}\mathbf{X}_{\mathbf{C}}^T, \tag{5.63}$$

where

$$\mathbf{X}_{\mathbf{C}} = [\mathbf{X}_{\mathbf{C}1}, \ldots, \mathbf{X}_{\mathbf{C}N}] \tag{5.64}$$

is a $d^2 \times dN$ matrix.

In our experiments we use $d = 2800$. Therefore building a $d^2 \times d^2$ matrix requires on the order of $d^4 = 62$ TB of memory. This is impractical. One method to reduce the memory requirements is to set $\gamma = 0$ which is equivalent to ignoring the ASM criterion. The ASM criterion has not been shown to provide significant performance improvement compared to the MSE and ONV criteria. This reduces matrix $\mathbf{T}$ to

$$\mathbf{T} = \frac{1}{N}\mathbf{X_C}\mathbf{X_C}^T + \beta\mathbf{I}. \tag{5.65}$$

Using the Woodbury matrix identity[1] ,

$$\begin{aligned}\mathbf{T}^{-1} &= \left(\beta\mathbf{I}_{d^2} + \left(\frac{1}{N}\mathbf{X_C}\right)\mathbf{I}_{dN}\mathbf{X_C}^T\right)^{-1} \\ &= \left(\beta\mathbf{I}_{d^2}\right)^{-1} + \left(\beta\mathbf{I}_{d^2}\right)^{-1}\left(\frac{1}{N}\mathbf{X_C}\right)\left(\mathbf{I}_{dN}^{-1} + \mathbf{X_C}^T\left(\beta\mathbf{I}_{d^2}\right)^{-1}\left(\frac{1}{N}\mathbf{X_C}\right)\right)^{-1}\mathbf{X_C}^T\left(\beta\mathbf{I}_{d^2}\right)^{-1} \\ &= \frac{1}{\beta}\left(\mathbf{I}_{d^2} + \mathbf{X_C}\left(\beta N\mathbf{I}_{dN} + \mathbf{X_C}^T\mathbf{X_C}\right)^{-1}\mathbf{X_C}^T\right), \end{aligned} \tag{5.66}$$

where $\mathbf{I}_{d^2}$ and $\mathbf{I}_{dN}$ are identity matrices of size $d^2 \times d^2$ and $dN \times dN$, respectively. This requires inverting a $Nd \times Nd$ matrix instead of a $d^2 \times d^2$ matrix.

Recall that the dual can be expressed as

$$\max_{\mathbf{0}\leq\mathbf{a}\leq C\mathbf{1}} \quad 2\mathbf{a}^T\mathbf{u} - \mathbf{a}^T\mathbf{L}\mathbf{Y}^T\mathbf{T}^{-1}\mathbf{Y}\mathbf{L}\mathbf{a}, \tag{5.67}$$

where $\mathbf{T}^{-1}\mathbf{Y}$ can be computed as

$$\begin{aligned}\mathbf{T}^{-1}\mathbf{Y} &= \frac{1}{\beta}\left(\mathbf{I}_{d^2} + \mathbf{X_C}\left(\beta N\mathbf{I}_{dN} + \mathbf{X_C}^T\mathbf{X_C}\right)^{-1}\mathbf{X_C}^T\right)\mathbf{Y} \\ &= \frac{1}{\beta}\left(\mathbf{Y} + \mathbf{X_C}\left(\beta N\mathbf{I}_{dN} + \mathbf{X_C}^T\mathbf{X_C}\right)^{-1}\mathbf{X_C}^T\mathbf{Y}\right). \end{aligned} \tag{5.68}$$

Thus, the $d^2 \times d^2$ matrix $\mathbf{T}$ never needs to be computed. The largest matrix needed is an $Nd \times Nd$ matrix which requires on the order of $d^2N^2 = 7.8N^2$ MB of memory which is practical for small values of $N$. However, in our experiments, $N = 100$ (requiring on the order of 78 GB of memory) before retraining and it can be up to $N = 1000$ (requiring on the order of 7.8 TB of memory) after retraining, making this method also impractical to implement.

For computational reasons, in our implementation we use the same $\mathbf{T}$ computed in MMCF.

[1] The Woodbury matrix identity is $(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$.

We premultiplied the image $\mathbf{x}_i$ instead of $\mathbf{y}_i$ as in Eq. 4.30. Recent work [33, 59] shows that premultiplying the training images by matrix $\mathbf{T}^{-\frac{1}{2}}$ shown in Eq. 4.30 results in sharper peaks and enhanced performance. This is because higher frequency components (which is usually where most of the discriminative information is) are emphasized by premultiplying the image with this matrix. As our experimental results will show, premultiplying the training images by matrix $\mathbf{T}^{-\frac{1}{2}}$ enhances performance. We conjecture that using the correct $\mathbf{T}$ matrix will further improve the results.

## 5.4 EXPERIMENTS

We test our algorithm on real videos of vehicles. The details of our experimental setup are discussed in Chapter 7. In this section we give a brief overview of our experiments and present results comparing the performance of the filters discussed in this chapter.

We consider vehicle recognition on a set of $512 \times 640$ pixels 30 Hz infrared videos where the vehicle's class-label and location are unknown. Our dataset has eight vehicles (one vehicle in each video) driving in one circle, shown in Fig. 1.2. We consider QMMCF, QSVM, Quadratic MACE (QMACE) which is QMMCF with $\lambda = 1$, TQCF, MMCF, SVM, CCF, mCCF, UCF, and EASEF. For each type of classifier we train eight filters. Each filter is trained to recognize one given target (there are eight targets) for all 360° degrees of azimuth rotation. We determine target location by cross-correlating the template with the test image and determining its location by the highest value in the resulting correlation output. For the highest value, we compute the peak-to-correlation-energy (PCE) (see Eq. 3.85) and select the template (out of the eight templates–one per target) that gives the highest PCE value. We declare a correct recognition when the correct template produces the maximum response to a given frame (i.e., correct classification) *and* produces the peak within a specified window centered at the correct location (i.e., correct localization). This means that it is considered an error 1) when the largest correlation peak is *not* close to the target's ground truth location *and* is from the incorrect class, or 2) when the largest correlation peak is close to the target's ground truth location but is from the incorrect class, or 3) when the largest correlation peak is from

Table 5.1: Filter performance (%) using true-class images only

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ |
|---|---|---|---|---|---|---|
| QMMCF | 46.1 | 70.8 | 37.3 | **0.60** | **1.00** | 0 |
| QMACE | 15.0 | 28.9 | 2.1 | 1 | 1 | 0 |
| QSVM | 20.4 | 29.3 | 16.9 | 0 | 1 | 0 |
| TQCF | 50.8 | 72.8 | 48.9 | **0.70** | 1 | 0 |
| MMCF | 32.7 | 66.4 | 29.6 | **0.01** | **0.20** | **1.5** |
| SVM | 12.1 | 25.8 | 9.9 | 0 | 1 | 0 |
| CCF | 21.1 | 57.8 | 17.7 | **0.55** | **0.90** | **0.5** |
| mCCF | 22.6 | 67.6 | 21.2 | **1.00** | **1.00** | **1.0** |
| UCF | 32.3 | 66.0 | 32.3 | **0.05** | **0.95** | **0** |
| EASEF | 34.1 | 67.0 | 31.8 | **0.05** | 1 | **0.5** |

the correct class but the peak's location is not near the target's ground truth location.

We performed three sets of experiments. In Set 1, each filter is trained using 20 true-class images. In Set 2, each filter is trained using using 20 true-class images *and* 80 background (false-class) images. In Set 3, each filter is initially trained as in Set 2 and then retrained, i.e., we cross-correlate the template with the frames from which we cropped the training images, add the false positives as false-class training images, and retrain the template. Table 5.1 shows the classification (class), localization (loc), and recognition (recog) rates for each filter in Set 1. Table 5.2 shows the classification, localization, and recognition rates for each filter in Set 2, and it shows the recognition performance improvement (impr) over the recognition rates in Set 1. Improvement is computed as new performance minus old performance, and the result divided by old performance. Table 5.3 shows the recognition rates in Set 3, and it also shows the number of retraining (ret) cycles we used (after a certain number of cycles, performance does not improve), and the recognition performance improvement over the recognition rates in Set 2. We compared different $\lambda$, $\psi$, and variance $\dot{\mathbf{g}}_{\sigma^2}$ of the desired Gaussian-function-like shape correlation output parameters and report our best recognition performance findings. The values of the parameters selected are in bold and non-bold values are default parameters for those filters.

These results show that QMMCF improves performance when retraining is used. As the number of false class images increases in retraining, using QMMCF improves over QSVM by 33%,

Table 5.2: Filter performance (%) using true- and false-class images before retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | impr |
|---|---|---|---|---|---|---|---|
| QMMCF | 59.2 | 88.6 | 55.8 | **0.50** | **1.00** | 0 | 49.6 |
| QMACE | 24.5 | 62.5 | 14.9 | 1 | 1 | 0 | 609.5 |
| QSVM | 24.3 | 57.9 | 22.6 | 0 | 1 | 0 | 33.7 |
| TQCF | 75.3 | 76.8 | 63.7 | **0.30** | 1 | 0 | 30.3 |
| MMCF | 51.1 | 87.7 | 49.6 | **0.95** | **1.00** | **2.0** | 67.6 |
| SVM | 23.1 | 55.7 | 17.6 | 0 | 1 | 0 | 77.8 |
| CCF | 53.7 | 90.2 | 52.6 | **0.85** | **1.00** | **2.5** | 197.2 |
| mCCF | 54.2 | 93.2 | 53.4 | **0.99** | **.9999** | **0.5** | 151.9 |
| UCF | 36.1 | 64.7 | 34.5 | **0.25** | **0.55** | **0.5** | 6.8 |
| EASEF | 35.4 | 68.7 | 31.8 | **0.05** | 1 | **1.0** | 0 |

Table 5.3: Filter performance (%) after retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | ret | impr |
|---|---|---|---|---|---|---|---|---|
| QMMCF | 89.5 | 96.6 | 86.4 | **0.05** | **1.00** | 0 | 4 | 54.8 |
| QMACE | 45.1 | 73.2 | 36.7 | 1 | 1 | 0 | 3 | 146.3 |
| QSVM | 65.62 | 92.2 | 64.9 | 0 | 1 | 0 | 7 | 187.2 |
| TQCF | 87.6 | 89.7 | 82.4 | **0.20** | 1 | 0 | 8 | 29.4 |
| MMCF | 63.9 | 95.3 | 63.2 | **0.25** | **1.00** | **0** | 9 | 27.4 |
| SVM | 47.4 | 87.3 | 47.4 | 0 | 1 | 0 | 4 | 169.3 |
| CCF | 61.9 | 90.0 | 59.1 | **0.75** | **1.00** | **1.5** | 6 | 12.4 |
| mCCF | 65.8 | 92.4 | 64.7 | **.999** | **.9999** | **1.0** | 4 | 21.2 |
| UCF | 37.9 | 77.9 | 36.5 | **0.15** | **0.99** | **1.0** | 5 | 5.8 |
| EASEF | 33.9 | 66.5 | 32.9 | **0.05** | 1 | **0** | 3 | 3.5 |

over TQCF by 5%, over mCCF by 34%, and over MMCF by 37%. Keep in mind that these improved results were obtained using the MMCF matrix $\mathbf{T}$ and not the QMMCF matrix $\mathbf{T}$ derived in this chapter. A topic of future research is to find a computationally feasible method to implement QMMCF using the matrix $\mathbf{T}$ derived in this chapter. We conjecture that using the correct $\mathbf{T}$ matrix will further improve the results.

We also observed that retraining improves more of the performance when the ONV criterion is emphasized. That is, when there are few training images (before retraining), choosing $\lambda = 0.5$ gave the best performance. Recall that $\lambda = 1$ ignores the ONV criterion and $\lambda = 0$ only uses the ONV criterion and ignores the MSE criterion (see Eq. 4.80). As the number of training images increases (after retraining), performance usually increases keeping $\lambda$ constant. Lowering the value of $\lambda$, however, further increases the performance. For example, before retraining QMMCF's best performance is 55.8% when $\lambda = 0.5$. After retraining using $\lambda = 0.5$, QMMCF's performance increases to 73.4% (not shown in the tables above) which is a 32% improvement. However, the overall best performance after retraining is when $\lambda = 0.05$ with 86.4% which is a 118% improvement over the 39.6% performance (not shown in the tables above) before retraining. When ONV is emphasized, the MSE criterion is de-emphasized. This results in many false-positive peaks before retraining. After retraining, a lot of the false-positive are included as false-class images. This improves classification and localization and reduces (but not eliminates) the need for the MSE criterion.

## 5.5 Summary

In this chapter we showed the relation between QCFs and QSVMs. We combined the design principles of CFs and QSVMs and introduced the Quadratic Maximum Margin Correlation Filter (QMMCF). Our results show superior performance to all the other filters previously mentioned. This improved performance comes at the cost of added computation in the testing.

CHAPTER **6**

# VECTOR FEATURES

In the previous chapters we have used scalar features, i.e., gray-scaled pixels, in our designs. In this chapter we discuss vector features and adapt our filter designs to include vector features. We focus on the Histograms of Oriented Gradients (HOG) feature [21, 24] due to its recent growth in popularity. The results show our CFs maintain superior performance over other state-of-the-art algorithms when we use HOG features.

## 6.1 HISTOGRAMS OF ORIENTED GRADIENTS

The CF designs that have been introduced in the previous chapters use scalar features, i.e., gray-scaled pixels. There are algorithms that extract features from different regions in the image. We call these features *vector features*. A simple example is using color pixels. Every pixel in the image is consider a different region with three values: red, green, and blue. Another example is using HOG features. They were originally proposed by Dalal and Triggs [21] and later modified by Felzenszwalb, et al. [24]. In Felzenszwalb, et al.'s work, HOG features are obtained by taking the gradient at every pixel, computing the magnitude and orientation of each gradient value, quantizing each *unsigned* orientation to the closest multiple of 20 degrees, i.e., $20°, 40°, \cdots, 180°$ (there are nine quantized values), and quantizing each *signed* orientation to the closest multiple of 20 degrees, i.e., $20°, 40°, \cdots, 360°$ (there are eighteen quantized values). Each pixel in the image is associated

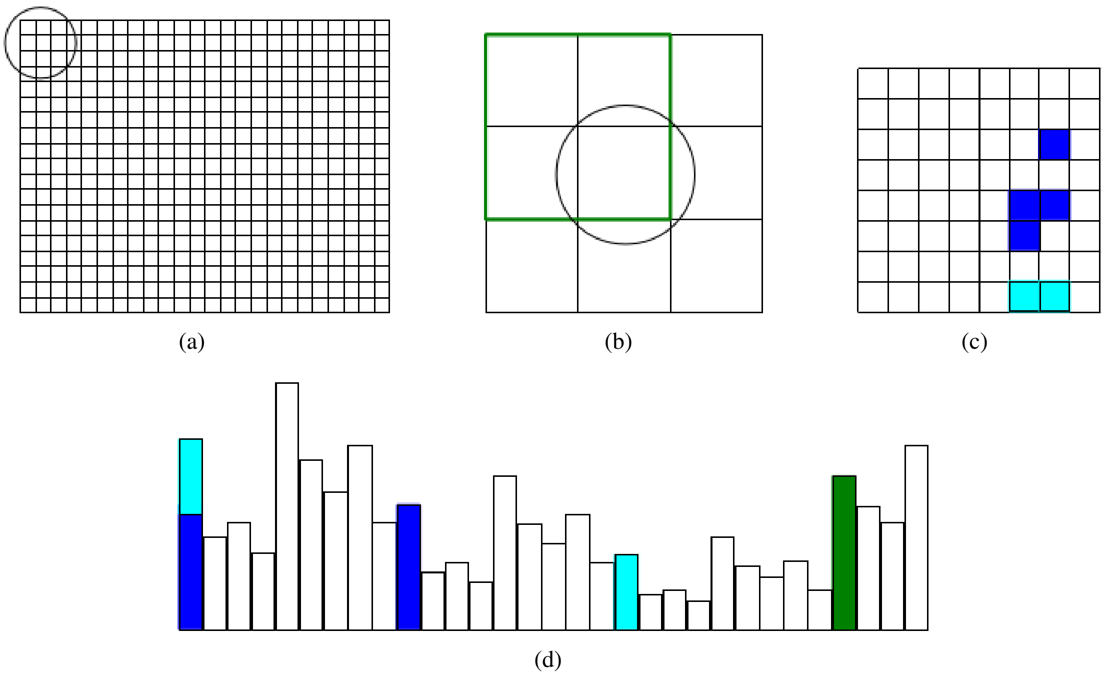


Figure 6.1: An image (a) is broken into non-overlapping cells. The circled cells in (a) are shown in (b). In (b) the green block in made of 2 $\times$ 2 cells. The circled cell in (b) in shown in (c). In (c), each cell is composed of 8 $\times$ 8 pixels each with a gradient magnitude and quantized orientation. In this figure, the pixels with a quantized *unsigned* orientation of 20° are shown in blue, where those with a quantized *signed* orientation of 20° are shown in dark blue and those with a quantized *signed* orientation of 200° are shown in light blue. The sum of the magnitudes corresponding to the *unsigned* orientations of 20°, the *signed* orientation of 20°, and the *signed* orientation of 200° are represented by the height of the 1st, 10th, and 19th entry in the histogram in (d). Thus, the 1st histogram value equals the sum of the 10th and 19th histogram values. The gradient energy of the first block in green is represented by the height of the 28th entry in the histogram. The gradient of the other blocks in (b) are represented by the 29th, 30th, and 31st histogram values.

with a gradient magnitude, an unsigned quantized orientation, and a signed quantized orientation. The image is broken into *non-overlapping* cells each composed of 8 $\times$ 8 pixels and *overlapping* blocks each composed of 2 $\times$ 2 cells. Each cell is represented by a 31-D vector. The first 27 entries of the 31-D HOG vector are the sum of the magnitudes corresponding to each orientation (9 unsigned and 18 signed). The last 4 entries of the 31-D vector contain the gradient energy of the four overlapping blocks that contain that cell. Therefore an $R \times C$ training image is represented by an $\lfloor \frac{R}{8} \rfloor \times \lfloor \frac{C}{8} \rfloor \times 31$ HOG feature cube. This process is shown in Figure 6.1 and is the same for training and test images. In our experiments, we used Felzenszwalb, et al.'s code [25] to compute the HOG features.

The HOG feature cubes from the training images are used to designed a 3-D filter that can be cross-correlated with the HOG feature cubes from the test images. If the correlation output produces a value above some threshold, then the test image is labeled as one containing a true target and that location of the value represents the location of the target.

## 6.2 Correlation output using matrix notation

When computing HOG features from a 2-D image, the result is a 3-D feature cube, with the number of features being the third dimension. This requires computing a 3-D filter. However, 3-D correlation is not required because correlation along the feature dimension is not needed. Rather, the overall correlation output is computed by doing a series of 2-D correlations and adding them. More specifically, let $\check{h}_{(p)}(m,n)$ be the $p$th point spread function (i.e., 2-D impulse response) and $\check{x}_{i(p)}(m,n)$ be the $p$th feature of the $i$th image. The overall correlation output for $i$th image is

$$\check{g}_i(m,n) = \sum_{p=1}^{P} \check{x}_{i(p)}(m,n) \otimes \check{h}_{(p)}(m,n), \tag{6.1}$$

where $P$ is the number of HOG features, i.e., the dimension of the histogram (in our experiments $P = 31$). In the frequency domain and using vector notation, Eq. 6.1 can be written as

$$\begin{aligned} \mathbf{g}_i &= \sum_{p=1}^{P} \mathbf{g}_{i(p)} \\ &= \sum_{p=1}^{P} \mathbf{X}^*_{i(p)} \mathbf{h}_{(p)}, \end{aligned} \tag{6.2}$$

where $d \times d$ diagonal matrix $\mathbf{X}_{i(p)}$ contains the vectorized $p$th HOG feature of the $i$th image along the diagonal, $\mathbf{h}_{(p)}$ represents the $p$th filter, and $d$ is the dimension of the HOG feature (which is usually smaller than the dimension of the image). Thornton [77] pointed out (for a different problem with the same format as Eq. 6.2) that this summation can be written as

$$\mathbf{g}_i = \mathbf{X}_i^{\dagger} \mathbf{h}, \tag{6.3}$$

where $\mathbf{g}_i$ is a vector of length $d$, $\mathbf{h} = [\mathbf{h}^\dagger_{(1)}, ..., \mathbf{h}^\dagger_{(P)}]^\dagger$ is a vector of length $dP$, and $\mathbf{X}_i = [\mathbf{X}_{i(1)}, ..., \mathbf{X}_{i(P)}]^T$ is a $dP \times d$ block matrix of diagonal matrices.

We use the criteria discussed in Section 3.1. The only difference is that $\mathbf{X}_i$ is not diagonal and therefore $\mathbf{X}_i^\dagger \neq \mathbf{X}_i^*$ as in previous chapters. The MSE can be computed as follows,

$$\begin{aligned}
\text{MSE} &= \frac{1}{dN}\sum_{i=1}^{N} |\mathbf{g}_i - \dot{\mathbf{g}}_i|^2 \\
&= \frac{1}{dN}\sum_{i=1}^{N} \left(\mathbf{g}_i^\dagger \mathbf{g}_i - 2\mathbf{g}_i^\dagger \dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger \dot{\mathbf{g}}_i\right) \\
&= \frac{1}{dN}\sum_{i=1}^{N} \left(\mathbf{h}^\dagger \mathbf{X}_i \mathbf{X}_i^\dagger \mathbf{h} - 2\mathbf{h}^\dagger \mathbf{X}_i \dot{\mathbf{g}}_i + \dot{\mathbf{g}}_i^\dagger \dot{\mathbf{g}}_i\right) \\
&= \mathbf{h}^\dagger \mathbf{D} \mathbf{h} - 2\mathbf{h}^\dagger \mathbf{p} + E_f, \qquad (6.4)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{D} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_i \mathbf{X}_i^\dagger \\
&= \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_{i(1)}\mathbf{X}^*_{i(1)} & \cdots & \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_{i(1)}\mathbf{X}^*_{i(P)} \\ \vdots & \ddots & \vdots \\ \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_{i(P)}\mathbf{X}^*_{i(1)} & \cdots & \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_{i(P)}\mathbf{X}^*_{i(P)} \end{bmatrix} \qquad (6.5)
\end{aligned}$$

is a $dN \times dN$ block matrix of diagonal matrices, $\mathbf{h}^\dagger \mathbf{p} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}_i^\dagger \dot{\mathbf{g}}_i$ is a measurement of the similarity between the actual correlation planes and the desired correlation planes with

$$\mathbf{p} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_i \dot{\mathbf{g}}_i, \qquad (6.6)$$

and $E_f = \frac{1}{N}\sum_{i=1}^{N} \dot{\mathbf{g}}_i^\dagger \dot{\mathbf{g}}_i$ is the average energy of the desired correlation planes.

The ASM can be computed as follows,

$$\begin{aligned}
\text{ASM} &= \frac{1}{N}\sum_{i=1}^{N}|\mathbf{g}_i - \bar{\mathbf{g}}|^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{g}_i^T\mathbf{g}_i - 2\mathbf{g}_i^T\bar{\mathbf{g}} + \bar{\mathbf{g}}^T\bar{\mathbf{g}}) \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\mathbf{g}_i - 2\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\right)\bar{\mathbf{g}} + \bar{\mathbf{g}}^T\bar{\mathbf{g}} \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i^T\mathbf{g}_i - \bar{\mathbf{g}}^T\bar{\mathbf{g}} \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}^T\mathbf{X}_i\mathbf{X}_i^\dagger\mathbf{h} - \mathbf{h}^T\bar{\mathbf{X}}\bar{\mathbf{X}}^\dagger\mathbf{h} \\
&= \mathbf{h}^T\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i\mathbf{X}_i^\dagger\right)\mathbf{h} - \mathbf{h}^T\bar{\mathbf{X}}\bar{\mathbf{X}}^\dagger\mathbf{h} \\
&= \mathbf{h}^T\mathbf{D}\mathbf{h} - \mathbf{h}^T\mathbf{M}\mathbf{h}, \qquad (6.7)
\end{aligned}$$

where

$$\bar{\mathbf{g}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i^\dagger\mathbf{h} = \bar{\mathbf{X}}^\dagger\mathbf{h}, \qquad (6.8)$$

and $\mathbf{M} = \bar{\mathbf{X}}\bar{\mathbf{X}}^\dagger$.

The ONV can be computed as follows,

$$\begin{aligned}
\text{ONV} &= \sigma^2 \\
&= \sum_{k=0}^{d-1}\mathbf{P}_{\mathbf{c}_{\tilde{\eta}}}[k] \\
&= \mathbf{h}^\dagger\mathbf{P}\mathbf{h}, \qquad (6.9)
\end{aligned}$$

where $\mathbf{P}_{\mathbf{c}_{\tilde{\eta}}}$ is the power spectral density of $\mathbf{c}_{\tilde{\eta}}$ (which represents the inner product between the filter and the input noise), and $\mathbf{P}$ is the power spectral density of the input noise. In many scenarios, unit-variance white noise is assumed, i.e., $\mathbf{P} = \frac{1}{d}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

### 6.2.1 Implementation

Recall that our CF designs require inverting

$$\mathbf{T} = (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P}. \tag{6.10}$$

When using $P$ features, matrix $\mathbf{T}$ can be up to $dP \times dP$ which may be computational infeasible to invert. However, matrix $\mathbf{T}$ is a block matrix of diagonal matrices, and can be inverted iteratively as follows. Let

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_A & \mathbf{T}_B \\ \mathbf{T}_C & \mathbf{T}_D \end{bmatrix}, \tag{6.11}$$

where $\mathbf{T}_D$ is a $d \times d$ diagonal matrix. Using the Banachiewicz inversion formula [20],

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{T}_A^{-1} + \mathbf{T}_A^{-1}\mathbf{T}_B(\mathbf{T}_D - \mathbf{T}_C\mathbf{T}_A^{-1}\mathbf{T}_B)^{-1}\mathbf{T}_C\mathbf{T}_A^{-1} & -\mathbf{T}_A^{-1}\mathbf{T}_B(\mathbf{T}_D - \mathbf{T}_C\mathbf{T}_A^{-1}\mathbf{T}_B)^{-1} \\ -(\mathbf{T}_D - \mathbf{T}_C\mathbf{T}_A^{-1}\mathbf{T}_B)^{-1}\mathbf{T}_C\mathbf{T}_A^{-1} & (\mathbf{T}_D - \mathbf{T}_C\mathbf{T}_A^{-1}\mathbf{T}_B)^{-1} \end{bmatrix}. \tag{6.12}$$

The $d(P-1) \times d(P-1)$ matrix $\mathbf{T}_A^{-1}$ can then be iteratively computed using the Banachiewicz inversion formula. Using this method only diagonal matrices are inverted.

For convenience in our experiments, in order to use bounded scalars the unbounded scalars $\gamma$ and $\beta$ are replaced with $\gamma = \frac{1}{\psi}(1-\psi)$ and $\beta = \frac{1}{\lambda}(1-\lambda)$, where $0 \leq \psi, \lambda \leq 1$. Eq. 6.10 can be written as

$$\begin{aligned} \mathbf{T} &= (1+\gamma)\mathbf{D} - \gamma\mathbf{M} + \beta\mathbf{P} \\ &= (1 + \frac{1}{\psi}(1-\psi))\mathbf{D} - \frac{1}{\psi}(1-\psi)\mathbf{M} + \frac{1}{\lambda}(1-\lambda)\mathbf{P} \\ &= \frac{1}{\lambda\psi}\left(\lambda\mathbf{D} + \lambda(1-\psi)\mathbf{M} + (1-\lambda)\psi\mathbf{P}\right), \end{aligned} \tag{6.13}$$

where we define the ratio $\frac{0}{0} = 1$ for the cases when $\psi = 0$ and/or $\lambda = 0$ (this allows for the scalars $\gamma \to \infty$ and $\beta \to \infty$).

## 6.3 Experiments

We repeat the same set of experiments as in Chapter 5 but using HOG features instead of gray-scaled pixel features. The details of our experimental setup are discussed in Chapter 7. We consider vehicle recognition on a set of $512 \times 640$ pixels 30 Hz infrared videos where the vehicle's class-label and location are unknown. Our dataset has eight vehicles (one vehicle in each video) driving in one circle, shown in Fig. 1.2. We consider all the classifiers from Chapter 5. For each type of classifier we train eight filters. Each filter is trained to recognize one given target (there are eight targets) for all $360°$ degrees of azimuth rotation. We determine target location by cross-correlating the template with the test image and determining its location by the highest value in the resulting correlation output. For the highest value, we compute the peak-to-correlation-energy (PCE) (see Eq. 3.85) and select the template (out of the eight templates–one per target) that gives the highest PCE value. We declare a correct recognition when the correct template produces the maximum response to a given frame (i.e., correct classification) *and* produces the peak within a specified window centered at the correct location (i.e., correct localization). This means that it is considered an error 1) when the largest correlation peak is *not* close to the target's ground truth location *and* is from the incorrect class, or 2) when the largest correlation peak is close to the target's ground truth location but is from the incorrect class, or 3) when the largest correlation peak is from the correct class but the peak's location is not near the target's ground truth location.

We performed three sets of experiments. In Set 1, each filter is trained using 20 true-class images. In Set 2, each filter is trained using using 20 true-class images *and* 80 background (false-class) images. In Set 3, each filter is initially trained as in Set 2 and then retrained, i.e., we cross-correlate the template with the frames from which we cropped the training images, add the false positives as false-class training images, and retrain the template. Table 6.1 shows the classification (class), localization (loc), and recognition (recog) rates for each filter in Set 1. Table 6.2 shows the classification, localization, and recognition rates for each filter in Set 2, and it shows the recognition performance improvement (impr) over the recognition rates in Set 1. Improvement is computed

Table 6.1: Filter performance (%) using true-class images only

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ |
|---|---|---|---|---|---|---|
| QMMCF | 70.2 | 87.1 | 68.8 | $1 \times 10^{-6}$ | **0.99** | 0 |
| QSVM | 18.1 | 42.4 | 7.7 | 0 | 1 | 0 |
| TQCF | 30.4 | 38.5 | 21.7 | $1 \times 10^{-9}$ | 1 | 0 |
| MMCF | 41.3 | 73.6 | 40.3 | $1 \times 10^{-5}$ | **1.00** | **0d** |
| SVM | 34.3 | 41.0 | 14.9 | 0 | 1 | 0 |
| CCF | 30.4 | 68.9 | 28.1 | $3 \times 10^{-6}$ | **0.4** | **0.2** |
| mCCF | 30.5 | 69.5 | 27.6 | $1 \times 10^{-5}$ | **0.99** | **0.1** |
| UCF | 41.3 | 73.6 | 40.3 | $1 \times 10^{-5}$ | **1** | **0** |
| EASEF | 36.2 | 69.1 | 32.7 | $1 \times 10^{-4}$ | 1 | **0** |

Table 6.2: Filter performance (%) using true- and false-class images before retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | impr |
|---|---|---|---|---|---|---|---|
| QMMCF | 72.4 | 94.1 | 72.2 | $1 \times 10^{-5}$ | **0.80** | 0 | 4.9 |
| QSVM | 13.6 | 53.8 | 11.4 | 0 | 1 | 0 | 48.1 |
| TQCF | 24.9 | 47.5 | 23.6 | $3 \times 10^{-8}$ | 1 | 0 | 8.8 |
| MMCF | 44.9 | 75.5 | 42.6 | $3 \times 10^{-5}$ | **0.99** | **0d** | 5.7 |
| SVM | 16.4 | 57.2 | 9.8 | 0 | 1 | 0 | -34.2 |
| CCF | 40.3 | 66.9 | 34.5 | $1 \times 10^{-5}$ | **0.20** | **0.05** | 22.8 |
| mCCF | 40.3 | 66.9 | 34.5 | $1 \times 10^{-5}$ | **0.20** | **0.05** | 25.0 |
| UCF | 44.8 | 75.5 | 42.6 | $3 \times 10^{-5}$ | **0.99** | 0 | 5.7 |
| EASEF | 34.3 | 71.5 | 32.2 | $3 \times 10^{-5}$ | 1 | **0** | -1.5 |

as new performance minus old performance, and the result divided by old performance. Table 6.3 shows the recognition rates in Set 3, and it also shows the number of retraining (ret) cycles we used (after a certain number of cycles, performance does not improve), and the recognition performance improvement over the recognition rates in Set 2. We compared different $\lambda$, $\psi$, and variance $\dot{\mathbf{g}}_{\sigma^2}$ of the desired Gaussian-function-like shape correlation output parameters and report our best recognition performance findings. The values of the parameters selected are in bold and non-bold values are default parameters for those filters.

The results show that 1) our MMCF filter outperforms all the other linear filters (ties with the UCF filter), and 2) our QMMCF filter outperforms all the linear and quadratic filters. The values for the MMCF and UCF filter become equal as $\lambda \rightarrow 0$ and $\dot{\mathbf{g}} = \mathbf{0}$ because the UCF filter portion of the MMCF filter gets emphasized. MMCF outperforms SVM by 170%, 335%, and 172% and CCF

Table 6.3: Filter performance (%) after retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | ret | impr |
|---|---|---|---|---|---|---|---|---|
| QMMCF | 76.6 | 95.6 | 76.5 | $1 \times 10^{-5}$ | **0.70** | 0 | **1** | 6.0 |
| QSVM | 21.4 | 80.5 | 20.4 | 0 | 1 | 0 | **7** | 79.0 |
| TQCF | 55.3 | 87.8 | 54.9 | $3 \times 10^{-8}$ | 1 | 0 | **5** | 132.6 |
| MMCF | 57.6 | 87.1 | 54.7 | $1 \times 10^{-4}$ | **1.00** | **0.2** | **4** | 25.4 |
| SVM | 24.5 | 72.1 | 19.6 | 0 | 1 | 0 | **5** | 100.0 |
| CCF | 50.8 | 77.7 | 46.8 | $3 \times 10^{-5}$ | **1.00** | **0** | **2** | 35.7 |
| mCCF | 50.8 | 77.6 | 46.8 | $3 \times 10^{-5}$ | **1.00** | **0** | **2** | 35.7 |
| UCF | 57.6 | 87.1 | 54.7 | $1 \times 10^{-4}$ | **1.00** | **0.2** | **4** | 25.4 |
| EASEF | 34.3 | 71.5 | 32.2 | $3 \times 10^{-5}$ | 1 | **0** | **0** | 0 |

by 43%, 23%, and 14%, and QMMCF outperforms QSVM by 794%, 533%, and 275% and TQCF by 217%, 206%, 39% in Sets 1, 2, and 3, respectively. This shows that using other features such as HOG maintains the superior performance of our algorithms in our dataset.

## 6.4 Summary

In this chapter we adapted our MMCF filter design to include vector features. We tested our designed with HOG features. Our results shows that in our dataset, our design outperformed all the other state-of-the-art filters when HOG features are used.

CHAPTER 7

# EXPERIMENTS

In the previous chapters we have shown some experimental results. In this chapter we explain in more detail the dataset used in our experiments. We discuss common techniques that may be used in the pre- and post-processing stages. We explain our method for selecting parameters. We give more details on the results presented in previous chapters, and present our conclusions.

## 7.1 TEST DATASET

We test our algorithm on real videos. We consider vehicle recognition on a set of infrared videos where the vehicle's class-label and location are unknown. We use the recently approved for public release *ATR Algorithm Development Image Database* [17] produced by the Military Sensing Information Analysis Center. The database contains infrared videos of $512 \times 640$ pixels/frame from eight military vehicles (one vehicle in each video), shown in Fig. 1.2, taken at multiple ranges during day and night time at 30 Hz. We used videos from each vehicle collected during day time at a range of 1000 meters. The ground truth provided in the dataset varies by about 10 pixels from the target's center. To improve the ground truth data, we manually selected the location of the target's center for each frame for each vehicle.

The vehicles were driven at about 10 mph making a circle of diameter of about 100 meters, therefore exhibiting full $360°$ of azimuth rotation. Each video is originally 60 seconds long, (i.e., 1800 frames) allowing the vehicle to complete at least one full circle. We truncated the videos so that the vehicles in the video only make one full circle, and we downsampled each video to 1000 frames so that all the videos contain the same number of frames.

From these 1000 frames, we selected 20 frames uniformly sampled for training and from the reminder 980 frames we selected 200 frames uniformly sample for testing. We only selected 200 frames for testing to reduce the testing time and because we observed in preliminary experiments that using all 980 frames does not significantly change the results. We referred to these 20 frames as training frames. From these frames we truncated a $70 \times 40$ pixel area centered at the target's center that contains the entire target region, and truncated 80 non-target regions. These truncated areas are the images used to train the template. The 20 training images with the target are the true-class images and the 80 images with background are the false-class images.

## 7.2 Pre- and Post-processing data

Pre- and post-processing techniques may be used to reduce the effect of different illumination, to register and center the targets in the training images, to select an adequate background for the training images, to emphasize certain frequencies over others, and to measure the peak-sharpness in the correlation plane. Different scenarios may benefit from different techniques. The following discussion summarizes the common techniques that are sometimes used.

### 7.2.1 Training images preprocessing

Some of the techniques that may be used to preprocess the images are:

- To reduce edge effects that arise at the boundaries of the image
  - have the background of the training images be equal to the mean of the image
  - set all the values along the entire $u$- and $v$-axis in the frequency domain of the training images to zero

  - apply a cosine window to the training images [12]

- To reduce the effects of shadows and intense lighting

  - apply the logarithmic function $\log(x+1)$ to the image pixels for both training and testing [12, 66]
  - zero-mean the training images and then normalize the energy
  - use adaptive histogram equalization (similar to *imadjust* command in MATLAB)

An alternative to unit energy training images is unit variance training images. The only difference is that the autocorrelation peak in the training images is one when they have unit energy and $d$ when they have unit variance, where $d$ is the dimension of the training images. In our experiments, we subtracted the mean of each training images (this reduces the aliasing caused by circular correlation) and normalized the energy to be unit energy. This is equivalent to setting to zero the origin in the frequency domain and normalizing the energy in the frequency domain.

### 7.2.2 Training images selection and registration

It is important to choose an adequate set of images to train the filter to represent the possible distortions of the target. Using too many or too few images may lower the filter's recognition performance. In addition the training images require registration, i.e., alignment, for better performance. In fact, poor registration significantly decreases the filter's performance.

Suppose that a filter capable of recognizing 360 degrees of azimuth rotation for a tank is required for ATR, and a set of images that sample this range is available. One well-known technique to train the filter is to first choose one image from the set and train the filter with just that one image. Then that filter is tested against all the images in the set. The image where the filter performed the worst is selected as an additional training image, and the two training images are registered. To register, the second image is shifted so that the cross-correlation peak is centered. Then a new filter is trained using these two images, and tested against all the images in the set. The image where the filter performs the worst is selected and shifted so that the correlation peak is centered (i.e., registered), and added to the training set. The filter is retrained with these three images and this process is repeated until the filter recognizes all the images in the set, i.e., until the filter's output for all the images in the set is above some threshold value. The number of training images is usually much less than the number of images in the training set.

During this process, however, the filter's recognition performance may decrease with additional images before reaching this threshold. When this happens, it means that the filter is not capable of capturing all the different distortions in the image set. In this case, the set of images is divided into two or more groups, e.g., four filters at every 90 degrees of azimuth rotation. Then, for each subset, a different filter is trained.

In our experiments, we used 20 true-class training images spaced at approximately $\frac{360^\circ}{20} = 18^\circ$ apart of azimuth rotation. Even though we had manually selected the target's center, we also registered the images using the above method to further improve registration accuracy. Although some of the filter's performance may have improved with more training images, we choose a constant value for the training images for all filters so that no one filter would could benefit from more training images.

### 7.2.3 Background selection

There are different options in selecting the background for the training images. Some methods use a white background, others use a black background, and others use the mean value of the target region. It is important to determine an appropriate background since a mismatched background may cause the filter to discriminate based on the background's pattern rather than the target's pattern. We investigated elsewhere [58] the effects of different backgrounds on the filter performance in different scenarios using both synthetic (pixels in the background chosen from a Gaussian distribution) and real backgrounds (photographs of different sceneries) for testing. In our comparisons we did not restrict ourselves to using a background with constant pixel intensity for training but also included backgrounds with varying pixel intensity with a mean and a standard deviation equal to the mean and the standard deviation of the target region. Our experiments showed that without a prior knowledge of the background in the testing images, training the filters using a background with the mean and variance of all the desired objects tends to give better results. In the experiments of this thesis, this technique would have required manually truncating each target region along its sihouette and was not used. Rather, we truncated a small square region that contains the target, and subtracted the mean from that entire region.

### 7.2.4 Image and filter transformation

In Section 3.2.1 we showed that the ECPSDF filter equals the OTSDF filter if the training and testing images are transformed by $\mathbf{T}^{-\frac{1}{2}}$ (see Eq. 2.18 for a description of $\mathbf{T}$). Equivalent, the MMCF filter can equal the SVM filter with this transformation. Some filters such as the ECPSDF filter and QCF are not designed to produce sharp peaks. Based on previous work [33, 59, 61], we expect that transforming the images in these filter results in OTSDF-like behavior, i.e., sharp peaks with built-in noise tolerance. In our experiments, we explicitly transformed the images in the QCF and QMMCF filters to have OTSDF-like behavior, and our results show significant improvement in performance. Other filters used this transformation implicitly in their designs such as the OTSDF, and MMCF filters.

### 7.2.5 Correlation output normalization

The zero-mean/unit-energy techniques, and/or the PCE or PSR metrics provide some form of normalization to the correlation output, e.g., due to illumination changes in the test image scene (e.g., when one part of the scene is lighted while another part is the shadows) and/or in different test images. In addition, PCE and PSR are useful metrics for quantifying the sharpness of the correlation peak. Peak sharpness is a measure of how high a peak is in relation to the surrounding values. In this section we discuss these techniques in more detail. In our experiments, we computed the PCE for each correlation output.

#### *7.2.5.1 Zero-mean and unit energy test chips*

In Section 7.2.1 the importance of zero-meaning and normalizing the energy in the training images was discussed. In testing it is not always desirable to do this for the entire scene. If one area of the scene has more illumination, those values will be unfairly emphasized. Instead it may be better to normalize every test chip, i.e., every test region equal in size to the training images.

We use 1-D notation for simplicity to show how this is done. The mean of each test chip (of length $d$) in image $x(n)$ is computed as follows,

$$\begin{aligned}\mu_x(n) &= \frac{1}{d}\sum_{k=0}^{d-1} x(n+k) \\ &= \frac{1}{d}\sum_{k=0}^{d-1} x(n+k)z(k) \\ &= \frac{1}{d}x(n)\otimes z(n), \end{aligned} \tag{7.1}$$

where

$$z(n) = \begin{cases} 1 & 0 \le n \le d-1 \\ 0 & n > d-1 \end{cases}. \tag{7.2}$$

In 2-D, $d$ represents the dimensions of the template $h(m,n)$ before zero-padding (the template is usually zero-padded to be greater than or equal to the test image size in order to do correlation in the frequency domain). The energy of zero-mean test chips is computed as follows,

$$\begin{aligned} e(n) &= \sum_{k=0}^{d-1} (x(n+k) - \mu_x(n))^2 \\ &= \sum_{k=0}^{d-1} x^2(n+k) - \sum_{k=0}^{d-1} 2x(n+k)\mu_x(n) + \sum_{k=0}^{d-1} \mu_x^2(n) \\ &= \sum_{k=0}^{d-1} x^2(n+k)z(k) - 2d\mu_x^2(n) + d\mu_x^2(n) \\ &= x^2(n)\otimes z(n) - d\mu_x^2(n) \\ &= x^2(n)\otimes z(n) - \frac{1}{d}(x(n)\otimes z(n))^2. \end{aligned} \tag{7.3}$$

The correlation output can be computed using zero-mean unit-energy test chips as follows

$$\begin{aligned} g(n) &= \sum_{k=0}^{d-1}\left[\frac{x(n+k)-\mu_x(n)}{(e(n))^{\frac{1}{2}}}\right]h(k) \\ &= \frac{\sum_{k=0}^{d-1}x(n+k)h(k)-\mu_x(n)\sum_{k=0}^{d-1}h(k)}{(e(n))^{\frac{1}{2}}} \\ &= \frac{x(n)\otimes h(n)-\mu_x(n)h_s}{(e(n))^{\frac{1}{2}}} \\ &= \frac{x(n)\otimes h(n)-\frac{1}{d}\left(x(n)\otimes z(n)\right)h_s}{\left(x^2(n)\otimes z(n)-\frac{1}{d}\left(x(n)\otimes z(n)\right)^2\right)^{\frac{1}{2}}}, \end{aligned} \tag{7.4}$$

where $h_s = \sum_{k=0}^{d-1} h(k)$ is the sum of the template's values. Thus, using zero-mean unit-energy test chips requires two additional correlations (or four DFTs). Although we have found this method useful in other experiments, in the set of experiments used in this thesis we found that doing these additional computations does not improve performance. Therefore, we do not compute the zero-mean unit-energy of the test chips.

#### 7.2.5.2 *Peak-to-Correlation Energy (PCE)*

When there is only *one* authentic target in the image, the Peak-to-Correlation Energy (PCE) [85] is a good metric to use, and it is computed for each test image as follows:

$$\text{PCE} = \frac{g_{max}}{\sqrt{\sum_{m,n}|g(m,n)|^2-|g_{max}|^2}}, \tag{7.5}$$

where $g_{max} = max_{n,m}(g(m,n))$ is the maximum value of the correlation plane. In our experiments, we found PCE to be a very useful metric. When there is more than one target in the image or when there is significant variation in illumination computing the peak-to-sidelobe ratio (explained below) is a better metric for target recognition.

#### 7.2.5.3 *Peak-to-Sidelobe Ratio (PSR)*

Another commonly used correlation peak sharpness metric is the peak-to-sidelobe ratio (PSR) [67] defined as

$$\text{PSR} \triangleq \frac{\text{peak}-\mu}{\sigma}, \tag{7.6}$$

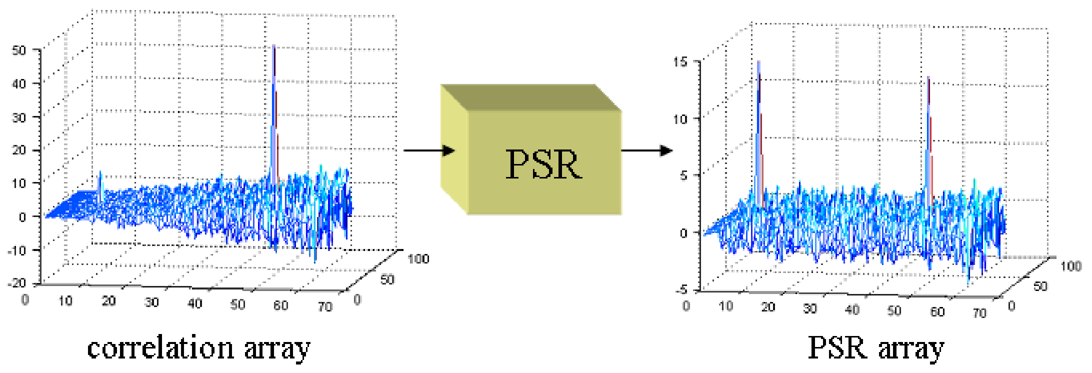


Figure 7.1: In this correlation output array, there is one large peak and another one small. The PSR compares the peak values to the surrounding values. We observe in this example that the PSR values of these two correlation peaks are similar, even though the correlation peak values are not similar. This can be caused when one target is in an illuminated region and the other one in a darker region. (Figure from Kereke's 2008 SPIE presentation [32]).

where $\mu$ and $\sigma$ are the mean and the standard deviation, respectively, of the region surrounding the correlation peak (we refer to the size of this region as "PSR outer window size") possibly excluding a small area around the peak (we refer to the size of this region as "PSR inner window size") to avoid contributions from broad correlation peaks. A PSR value of $\eta$ indicates that the peak is $\eta$ standard deviations above the mean of the surrounding values. For example, a value of $g = 6$ or greater is unlikely to be the result of an impostor target.

This metric can be efficiently computed for an entire correlation output using two correlations (or four DFTs) for each image [34]. A more efficient method is to compute the PSR only for the highest correlation values. However, computing PSR at only high correlation outputs may overlook some (not so high) peak values that stand out in comparison to their surrounding values (e.g., an authentic image in a dark (low intensity) location in the large scene) as shown in Fig. 7.2.5.3.

The PSR for each value can be efficiently computed as follows [34]. We use 1-D notation for simplicity. The mean of each region surrounding each correlation value in image $g(n)$ can be computed as follows,

$$\begin{aligned}\mu(n) &= \frac{1}{K}\sum_{k=0}^{K-1} g(n+k) \\ &= \sum_{k=0}^{K-1} g(n+k)w(k) \\ &= g(n)\otimes w(n), \end{aligned} \tag{7.7}$$

where

$$w(n) = \begin{cases} \frac{1}{K} & 0 \le n \le K-1 \\ 0 & n > K-1 \end{cases}. \tag{7.8}$$

In 2-D, $K$ represents the dimensions of the "PSR outer window" possibly excluding the "PSR inner window". The variance can be computed as follows,

$$\begin{aligned}\sigma^2(n) &= \frac{1}{K}\sum_{k=0}^{K-1}\left(g(n+k)-\mu(n)\right)^2 \\ &= \frac{1}{K}\sum_{k=0}^{K-1} g^2(n+k) - \frac{1}{K}\sum_{k=0}^{K-1} 2g(n+k)\mu(n) + \frac{1}{K}\sum_{k=0}^{K-1}\mu^2(n) \\ &= \sum_{k=0}^{K-1} g^2(n+k)w(k) - 2\mu^2(n) + \mu^2(n) \\ &= g^2(n)\otimes w(n) - \mu^2(n) \\ &= g^2(n)\otimes w(n) - \left(g(n)\otimes w(n)\right)^2. \end{aligned} \tag{7.9}$$

The entire PSR plane can be computed as follows,

$$\begin{aligned} g_{PSR}(n) &= \frac{g(n)-\mu(n)}{\sigma(n)} \\ &= \frac{g(n)-g(n)\otimes w(n)}{\left(g^2(n)\otimes w(n) - \left(g(n)\otimes w(n)\right)^2\right)^{\frac{1}{2}}} \end{aligned} \tag{7.10}$$

which requires two additional correlations (or four DFTs).

### 7.2.6 Retraining

Dalal and Triggs [21] proposed cross-correlating the 2-D template (recall that terms *template* and *filter* depict whether our design is in the space or the frequency domain, respectively) with training frames, adding the false positives as false-class training images, and retraining the template. In our experiments, we cross-correlated the template with the 20 training frames of each class ($20 \times 8 = 160$ training frames total). A false positive occurs when the highest peak is not at the correct location *or* when the highest peak corresponds to the wrong class. For each frame, we selected at most one false positive by truncating an area in the training frame centered at the location of the highest false-positive peak of the correlation output. These truncated areas were added to the set of false-class images and the filter was retrained. We repeated this process nine times. We observe that the performance improves during the first three retraining iterations but it usually does not change much after that. We observe that this retraining method always improves recognition performance particularly in the constrained CFs.

### 7.2.7 Selecting parameters

Most CFs require the selection of parameters. In our experiments, we conducted hundreds of tests with different parameters and chose the parameters that gave the best recognition performance. Other methods that are commonly used is to use a validation set of images (images not used in training or testing) and choose parameters that maximize recognition performance on this set. Another method that is commonly used [32, 74, 80] is to select parameters that produce a large Fisher ratio (FR). The FR is a measurement of the separation between two sets of filter responses expressed as

$$FR = \frac{|\mu_2 - \mu_1|^2}{\sigma_1^2 + \sigma_2^2}, \tag{7.11}$$

where $\mu_1, \mu_2, \sigma_1^2$ and $\sigma_2^2$ are the means and variances of the two sets. Set 1 is defined as the correlation peak values in response to true-class images and Set 2 as the correlation peak values in response to false-class images. The correlation peak values can either be the highest peak in the correlation plane, the highest PSR value, or the highest PCE value.

In our experiments, some parameters are chosen to be the same for all filters such as the number of true-class training images, the number of false-class training images, and the constrained peak value for the training images. We call these parameters *constant* parameters. Other parameters are chosen to be unique to each filter such as the $\lambda$, $\psi$, and $\mathbf{g}_{\sigma^2}$ that provide a tradeoff between the MSE, ASM, and ONV criteria and control the desired correlation plane peak sharpness. We call these parameters *varying* parameters.

Constant parameters were chosen by varying one parameter while keeping the other ones constant and selecting a value that produces overall good performance across all the filters. Before explaining the details of the process, recall that the constrained peak is usually set to $u_T = 1$ for true-class images and $u_F = -\epsilon$ for false class images, where $\epsilon$ is some small value. This means that for true-class images we desire a filter response with a high value and for the false-class images we desire a filter response with a value close to zero. As explained in Section 3.1.5, the value of $\epsilon$ can affect the filter. Initially we use $\epsilon = 0.1$ and 100 background (false-class) images. We trained filters with 5, 10, 15,..., 45, and 50 true-class training images. After comparing the results, we found that using 20 true-class training images produces good results across all the filters and using more than 20 training images did not significantly improve the results. Using 20 true-class training images and 100 background images, we trained filters with $\epsilon = 0$, $10^{-4}$,$10^{-3}$,$10^{-2}$, $10^{-1}$, and 1. After comparing the results, we find that using $\epsilon = 10^{-2}$ produces good results across all the filters. Using 20 true-class training images, and $\epsilon = 10^{-2}$, we trained filters with 10, 20, 30, ..., 90, and 100 background (false-class) images. After comparing the results, we find that using 80 background training images produces good results across all the filters and using more than 80 training images did not significantly improve the results. Thus, the constant parameters that were used in all filters were 20 true-class training images, 80 background (false-class) training images, and $\epsilon = 10^{-2}$.

Varying parameters were chosen individually for each filter by simultaneously varying $\lambda$ and $\mathbf{g}_{\sigma^2}$ parameter while keeping $\psi = 1$, selecting the best $\lambda$ and $\mathbf{g}_{\sigma^2}$ combination and then varying the $\psi$ parameter. We observed that the $\psi$ of the varying parameters has little effect on performance compared to the $\lambda$. We tested the filters against $\lambda$ =1, 0.9999, 0.999, 0.99, 0.95, 0.90, 0.85,...,0.15, 0.10, 0.05, 0.03, 0.01, $3 \times 10^{-3}$, $1 \times 10^{-3}$, $3 \times 10^{-4}$, $1 \times 10^{-4}$, $3 \times 10^{-5}$, $1 \times 10^{-5}$, $3 \times 10^{-6}$, $1 \times 10^{-6}$, $3 \times 10^{-7}$, $1 \times 10^{-7}$, $3 \times 10^{-8}$, $1 \times 10^{-8}$, $3 \times 10^{-9}$, $1 \times 10^{-9}$, $3 \times 10^{-10}$, $1 \times 10^{-10}$, and 0 (as we will see, experiments using gray-scaled pixel features usually performed best with a $\lambda$ value close to 1, and experiments using HOG features usually performed best with a $\lambda$ value close to 0). For each $\lambda$ value we tested different variance parameters of the Gaussian-function-like desired correlation output using $\mathbf{g}_{\sigma^2}$ =0, 0.001, 0.01, 0.02, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, and 2.5. We selected the best $\lambda$ and $\mathbf{g}_{\sigma^2}$ value and then tested against different $\psi$ values using $\psi$ =1, 0.99999, 0.9999, 0.999, 0.99, 0.95, 0.90, 0.85, 0.80, 0.75, ...,0.15, 0.10, 0.05, 0.03, and 0.01.

## 7.3 Experimental setup

To demonstrate the efficacy of our algorithm, we consider 18 classifiers: QMMCF, QSVM, QMACE, TQCF, QCF, MMCF, SVM, CCF, OTSDF, mCCF, UCF, MOSSE, UOTSDF, MACH, UMACE, ECPSDF, EASEF, and ASEF. For each type of classifier we train eight filters. Each filter is trained to recognize one given target (there are eight targets) for all 360° degrees of azimuth rotation. We determine target location by cross-correlating the template (recall that terms *template* and *filter* depict whether our design is in the space or the frequency domain, respectively) with the test image and determining its location by the highest value in the resulting correlation output. For the highest value, we compute the peak-to-correlation-energy (PCE) (see Eq. 7.5) and select the template (out of the eight templates–one per target) that gives the highest PCE value. We declare a correct recognition when the correct template produces the maximum response to a given frame (i.e., correct classification) *and* produces the peak within a specified window centered at the correct location (i.e., correct localization). This means that it is considered an error 1) when the largest correlation peak is *not* close to the target's ground truth location *and* is from the incorrect class, or 2) when the largest correlation peak is close to the target's ground truth location but is from the incorrect class, or 3) when the largest correlation peak is from the correct class but the peak's location is not near the target's ground truth location. Note that the low quality frame and the general

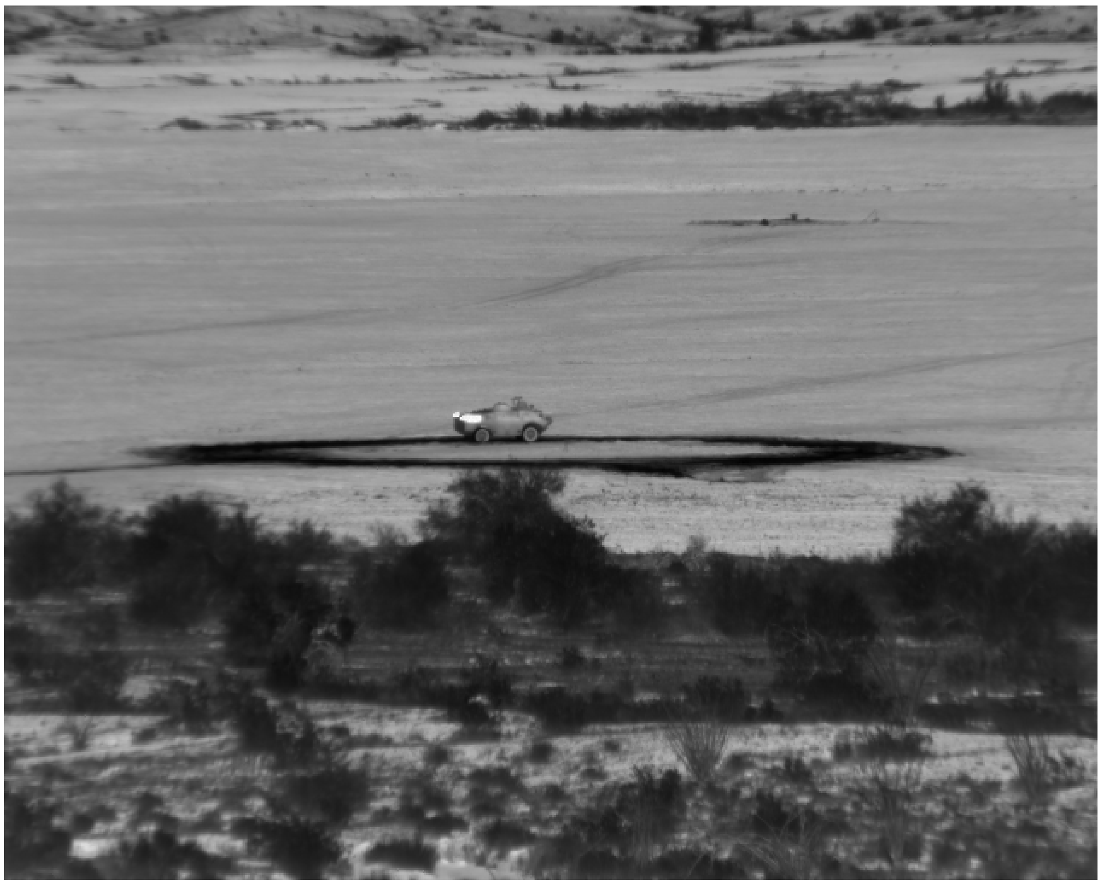


Figure 7.2: An example of a test frame containing the BRDM2 target.

background (see Fig. 7.2) makes the problem of localization and classification challenging.

It is important to note that we did not include any shifted images of authentic class as impostor class as commonly done in some localization approaches [22, 36, 70]. Including every possible shift would have required an additional $20 \times \{(2 \times 70 - 1) \times (2 \times 40 - 1) - 1\} = 219600$ false-class images per filter. One of the strengths of the algorithms in this work is avoiding the need to use these shifted images during training and still being able to locate the vehicle with high precision.

To investigate the ability of the various approaches to localize and classify targets, we did *not* use any tracker in these experiments but assumed that the vehicle can be anywhere in each frame, and we treated each frame independently from other frames. Including a tracker may improve localization performance but is omitted from our experiments in order to analyze the performance of the unaided filters.

We declare a correct recognition when the correct template produces the maximum response to a given frame (i.e., correct classification) *and* produces the peak within a specified window centered at the correct location (i.e., correct localization). This means that it is considered an error 1) when the largest correlation peak is close to the target's ground truth location but is from the incorrect class, or 2) when the largest correlation peak is from the correct class but the peak's location is not near the target's ground truth location. In these experiments, we defined the window as follows,

$$window = \left( \frac{|P_x - \hat{P}_x|}{w} \leq D \right) \cap \left( \frac{|P_y - \hat{P}_y|}{h} \leq D \right), \tag{7.12}$$

where $P_x$ and $P_y$ are the ground truth location coordinates, $\hat{P}_x$ and $\hat{P}_y$ are correlation peak location coordinates, and $0 \leq D \leq 1$ is the normalized distance. Recall that in our experiments, width $w = 70$ and height $h = 40$. $D = 0$ requires that the correlation peak location be the same as the ground truth location. $D = 0.5$ requires that the peak location be within 35 and 20 pixels of the ground truth location in the $x-$ and $y-$directions, respectively. $D = 1$ requires that the peak location be within 70 and 40 pixels of the ground truth location in the $x-$ and $y-$directions, respectively.

In addition, we repeat the experiments using HOG features. Each $70 \times 40$ pixel image is transformed to an $8 \times 4$ HOG feature image where each pixel is a 31-D vector; i.e., an $8 \times 4 \times 31$ HOG feature cube. This process is shown in Figure 6.1 and is the same for training and test images. In our experiments, we used Felzenszwalb, et al.'s code [25] to compute the HOG features.

## 7.4 COMPUTATIONAL COMPARISON

The theoretical and measured complexity of the QMMCF, QSVM, TQCF, MMCF, CCF, UCF, and ASEF classifiers is shown in Table 7.1. The other filters are not considered because they are subsets of one of these classifiers. These comparisons were done using MATLAB on a 2.91 GHz, 3.25 GB RAM Dual Core Windows XP desktop. To measure the training and testing time, we used $N = 100$ training images (gray-scaled pixels and not HOG features) of dimension $d =$

Table 7.1: Computational complexity big O and measured (sec.)

| | Training one template | | Testing one image | |
|---|---|---|---|---|
| Template | Big O | time (s) | Big O | time (s) |
| QMMCF | $\min(N^3, N^2d^2) + Nd\log d$ | 0.66 | $Nd_s \log d_s$ | 10.9 |
| QSVM | $\min(N^3, N^2d)$ | 0.31 | $Nd_s \log d_s$ | 10.9 |
| TQCF | $N^3 + Nd\log d$ | 0.92 | $vd_s \log d_s$ | 0.96 |
| MMCF | $\min(N^3, N^2d) + Nd\log d$ | 0.62 | $d_s \log d_s$ | 0.19 |
| SVM | $\min(N^3, N^2d)$ | 0.27 | $d_s \log d_s$ | 0.19 |
| CCF | $N^3 + Nd\log d$ | 0.35 | $d_s \log d_s$ | 0.19 |
| UCF | $Nd\log d$ | 0.39 | $d_s \log d_s$ | 0.19 |
| EASEF | $Nd\log d$ | 0.35 | $d_s \log d_s$ | 0.19 |

$40 \times 70 = 2800$, and 200 testing images of dimension $d_s = 512 \times 640 = 327680$ and report the average time (in seconds) per image. QMMCF, TQCF, MMCF, CCF, UCF, and EASEF are designed by transforming the images into the frequency domain thereby requiring $N$ FFTs of size $d$, i.e., $O(Nd\log d)$. In addition to the Fourier transforms of the images, QMMCF, QSVM, MMCF, and SVM solve the quadratic optimization problem using SMO which has a computational complexity of $O(\min(N^3, N^2d))$, and CCF requires a matrix inversion of complexity $O(N^3)$. The computation required to test the linear filters is exactly the same, i.e., it involves the cross-correlation of the query image with the template which is computed efficiently in the frequency domain which has a computational complexity of $O(d_s \log d_s)$. Testing TQCF requires $v$ cross-correlations, where $v < N$ is the number of eigenvalues used, and therefore has a computational complexity of $O(vd_s \log d_s)$ (in Table 7.1 $v = 8$). QMMCF and QSVM can required up to $N$ (the number of support vectors) cross-correlations per frame, and therefore have a computational complexity of at most $O(Nd_s \log d_s)$ (in Table 7.1 $N = 100$ support vectors). The computational complexity of QMMCF, QSVM, and TQCF can be reduced by computing FFTs in different cores in a multi-core platform.

## 7.5 Experimental results

We performed three sets of experiments. In Set 1, each filter is trained using 20 true-class images. In Set 2, each filter is trained using using 20 true-class images *and* 80 background (false-class) images. In Set 3, each filter is initially trained as in Set 2 and then retrained, i.e., we cross-correlate the template with the frames from which we truncated the training images, add the false positives as false-class training images, and retrain the template. Table 7.2 shows the classification (class), localization (loc), and recognition (recog) rates for each filter in Set 1. Table 7.3 shows the classification, localization, and recognition rates for each filter in Set 2, and it shows the recognition performance improvement (impr) over the recognition rates in Set 1. Improvement is computed as new performance minus old performance, and the result divided by old performance. Table 7.4 shows the recognition rates in Set 3, and it also shows the number of retraining (ret) cycles we used (after a certain number of cycles, performance does not improve), and the recognition performance improvement over the recognition rates in Set 2. We compared different $\lambda$, $\psi$, and variance $\dot{\mathbf{g}}_{\sigma^2}$ of the desired Gaussian-function-like shape correlation output parameters and report our best recognition performance findings. The values of the parameters selected are in bold and non-bold values are default parameters for those filters. The first value is obtained using gray-scaled pixel features and the second value (separated by a comma) is obtained using HOG features. The SDF acronym is removed in order to fit all the results in each table, e.g., OTSDF is represented by OT.

We also investigate the classification performance of each classifier using confusion matrices and receiver operating characteristic (ROC) curves. For QMMCF, QSVM, TQCF, MMCF, SVM, CCF, mCCF, UCF and EASEF we show the confusion matrices using gray-scaled pixels in Tables 7.5 through 7.13, respectively, and using HOG features in Tables 7.14 through 7.22, respectively. Each column of the matrices represents the instances in a predicted class, while each row represents the instances in an actual class. The target classes Pickup, SUV, BTR70, BRDM2, BMP2, T72, ZSU23-4, and 2S3 shown in Fig. 1.2 are represented in these tables as targets 1 through 8, respectively. For QMMCF, QSVM, TQCF, MMCF, SVM, CCF, mCCF, UCF and EASEF we show the ROC curves in Figs. 7.3 through 7.11. The ROC curves are computed as follows. For each test frame, the correct filter is applied (i.e., if the test frame contains target 3, the filter trained with target 3 images is applied). For each correlation output, we compute the number of false alarms (FA) de-

Table 7.2: Filter performance (%) using true-class images only

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ |
|---|---|---|---|---|---|---|
| QMMCF | 46, 70 | 71, 87 | 37, 69 | **0.60, 1n6** | **1, 0.99** | 0 |
| QMACE | 15, 04 | 29, 09 | 02, 01 | 1 | 1 | 0 |
| QSVM | 20, 18 | 29, 42 | 17, 08 | 0 | 1 | 0 |
| TQCF | 51, 30 | 73, 39 | 49, 22 | **0.70, 1n9** | 1 | 0 |
| QCF | 32, 25 | 40, 45 | 27, 18 | 0 | 1 | 0 |
| MMCF | 33, 41 | 66, 74 | 30, 40 | **0.01, 1n5** | **0.2, 1** | $\frac{3}{2}$, **0d** |
| iOT | 21, 41 | 56, 74 | 17, 40 | **0.55, 1n5** | 1 | 0 |
| iMACE | 16, 02 | 43, 10 | 08, 01 | 1 | 1 | 0 |
| SVM | 12, 34 | 26, 41 | 10, 15 | 0 | 1 | 0 |
| CCF | 21, 30 | 58, 69 | 18, 28 | **0.55, 3n6** | **0.9, 0.4** | $\frac{1}{2}, \frac{1}{5}$ |
| OT | 21, 31 | 56, 65 | 18, 28 | **0.55, 3n6** | 1 | 0 |
| MACE | 17, 02 | 42, 10 | 07, 01 | 1 | 1 | 0 |
| ECP | 14, 12 | 27, 56 | 11, 08 | 0 | 1 | 0 |
| mCCF | 23, 31 | 67, 70 | 21, 28 | **1.00, 1n5** | **1, 0.99** | **1,** $\frac{1}{10}$ |
| UCF | 32, 41 | 66, 74 | 32, 40 | **0.05, 1n5** | **0.95, 1** | **0, 0** |
| UOT | 32, 41 | 66, 74 | 32, 40 | **0.05, 1n6** | 1 | 0 |
| MOSSE | 27, 03 | 62, 01 | 25, 01 | 1 | 1 | $\frac{3}{2}$, **0** |
| MACH | 30, 03 | 64, 01 | 29, 01 | 1 | **0, 1** | 0 |
| UMACE | 17, 03 | 53, 01 | 15, 01 | 1 | 1 | 0 |
| EASEF | 34, 36 | 67, 69 | 32, 33 | **0.05, 1n4** | 1 | $\frac{1}{2}$, **0** |
| ASEF | 17, 22 | 48, 38 | 14, 17 | 1 | 1 | $\frac{3}{2}$, **0** |

fined as the number of peaks that are greater than the peak value corresponding to the target (setting to zero a small area around each FA peak to prevent double counting values from peaks that extend over a few pixels). For each filter we plot eight ROC curves (one per target) as the localization rate (over the entire 200 test frames) versus the number of FA per frame. The ROC curves of the average over the eight targets are shown in Fig. 7.12.

Table 7.3: Filter performance (%) using true- and false-class images before retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | impr |
|---|---|---|---|---|---|---|---|
| QMMCF | 59, 72 | 89, 94 | 56, 72 | **0.50, 1n5** | **1, 0.8** | 0 | 50, 5 |
| QMACE | 25, 20 | 63, 44 | 15, 6 | 1 | 1 | 0 | 610, 433 |
| QSVM | 24, 14 | 58, 54 | 23, 11 | 0 | 1 | 0 | 34, 48 |
| TQCF | 75, 25 | 77, 48 | 64, 24 | **0.30, 3n8** | 1 | 0 | 30, 9 |
| QCF | 35, 38 | 49, 54 | 22, 20 | 0 | 1 | 0 | -18, 6 |
| MMCF | 51, 45 | 88, 76 | 50, 43 | **0.95, 3n5** | **1, 0.99** | **2, 0d** | 68, 6 |
| iOT | 51, 41 | 88, 75 | 49, 40 | **0.95, 3n5** | 1 | 0 | 182, -1 |
| iMACE | 36, 10 | 82, 46 | 33, 02 | 1 | 1 | 0 | 318, 156 |
| SVM | 23, 16 | 56, 57 | 18, 10 | 0 | 1 | 0 | 78, -34 |
| CCF | 54, 40 | 90, 67 | 53, 35 | **0.85, 1n5** | **1, 0.2** | $\frac{5}{2}, \frac{1}{20}$ | 197, 23 |
| OT | 54, 42 | 90, 69 | 52, 34 | **0.85, 1n5** | 1 | 0 | 199, 25 |
| MACE | 40, 16 | 84, 56 | 36, 12 | 1 | 1 | 0 | 389, 1244 |
| ECP | 16, 30 | 64, 64 | 11, 26 | 0 | 1 | 0 | 2, 225 |
| mCCF | 54, 40 | 93, 67 | 53, 35 | **0.99, 1n5** | **.9999, 0.2** | $\frac{1}{2}, \frac{1}{20}$ | 152, 25 |
| UCF | 36, 45 | 65, 76 | 35, 43 | **0.25, 3n5** | **0.55, 0.99** | $\frac{1}{2}$**, 0** | 7, 6 |
| UOT | 33, 45 | 66, 76 | 33, 42 | **0.15, 3n5** | 1 | 0 | 3, 5 |
| MOSSE | 30, 11 | 65, 47 | 26, 03 | 1 | 1 | **2, 0** | 7, 200 |
| MACH | 33, 11 | 68, 47 | 33, 03 | 1 | **0.15, 1** | 0 | 14, 200 |
| UMACE | 26, 11 | 62, 47 | 19, 03 | 1 | 1 | 0 | 27, 200 |
| EASEF | 35, 34 | 69, 72 | 32, 32 | **0.05, 3n5** | 1 | $\frac{1}{2}$**, 0** | 0, -3 |
| ASEF | 17, 19 | 47, 32 | 14, 15 | 1 | 1 | $\frac{3}{2}$**, 0** | -5, -12 |

Table 7.4: Filter performance (%) after retraining

| | class | local | recog | $\lambda$ | $\psi$ | $\dot{\mathbf{g}}_{\sigma^2}$ | ret | impr |
|---|---|---|---|---|---|---|---|---|
| QMMCF | 90, 77 | 97, 96 | 86, 77 | **0.05, 1n5** | **1, 0.7** | 0 | 4,1 | 55, 6 |
| QMACE | 45, 67 | 73, 80 | 37, 61 | 1 | 1 | 0 | 3, 3 | 146, 853 |
| QSVM | 66, 21 | 92, 81 | 65, 20 | 0 | 1 | 0 | 7, 7 | 187, 79 |
| TQCF | 88, 55 | 90, 88 | 82, 55 | **0.20, 3n8** | 1 | 0 | 8, 5 | 29, 133 |
| QCF | 58, 28 | 79, 68 | 55, 22 | 0 | 1 | 0 | 5, 6 | 145, 11 |
| MMCF | 64, 58 | 95, 87 | 63, 55 | **0.25, 1n4** | **1, 1** | **0,** $\frac{1}{5}$ | 9, 4 | 27, 25 |
| iOT | 64, 54 | 95, 86 | 63, 51 | **0.25, 3n5** | 1 | 0 | 9, 4 | 29, 29 |
| iMACE | 47, 53 | 84, 74 | 44, 47 | 1 | 1 | 0 | 1, 2 | 35, 1935 |
| SVM | 47, 25 | 87, 72 | 47, 20 | 0 | 1 | 0 | 4, 5 | 169, 100 |
| CCF | 62, 51 | 90, 78 | 59, 47 | **0.75, 3n5** | **1, 1** | $\frac{3}{2}$**, 0** | 6, 2 | 12, 36 |
| OT | 60, 51 | 91, 78 | 58, 47 | **0.80, 3n5** | 1 | 0 | 7, 2 | 11, 36 |
| MACE | 46, 41 | 85, 66 | 43, 33 | 1 | 1 | 0 | 3, 2 | 18, 169 |
| ECP | 39, 30 | 80, 61 | 35, 26 | 0 | 1 | 0 | 7, 5 | 216, 0 |
| mCCF | 66, 51 | 92, 78 | 65, 47 | **.999, 3n5** | **.9999, 1** | **1, 0** | 4, 2 | 21, 36 |
| UCF | 38, 58 | 78, 87 | 37, 55 | **0.15, 1n4** | **0.99, 1** | **1,**$\frac{1}{5}$ | 5, 4 | 06, 25 |
| UOT | 36, 57 | 75, 85 | 35, 53 | **0.35, 1n4** | 1 | 0 | 2, 4 | 6, 26 |
| MOSSE | 32, 57 | 77, 79 | 31, 51 | 1 | 1 | $\frac{5}{2}$**,**$\frac{1}{5}$ | 4, 3 | 19, 1500 |
| MACH | 36, 55 | 78, 76 | 35, 48 | 1 | **0.55, 1** | 0 | 3, 3 | 5, 1500 |
| UMACE | 24, 55 | 65, 76 | 21, 48 | 1 | 1 | 0 | 1, 3 | 15, 1500 |
| EASEF | 34, 34 | 67, 72 | 33, 32 | **0.05, 3n5** | 1 | **0, 0** | 3, 0 | 4, 0 |
| ASEF | 21, 19 | 49, 32 | 16, 15 | 1 | 1 | $\frac{3}{2}$**, 0** | 3, 0 | 15, 0 |

Table 7.5: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QMMCF.

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 107 | 72 | 0 | 17 | 4 | 0 | 0 | 0 |
| 2 | 0 | 196 | 0 | 1 | 3 | 0 | 0 | 0 |
| 3 | 0 | 25 | 134 | 38 | 3 | 0 | 0 | 0 |
| 4 | 0 | 33 | 5 | 160 | 2 | 0 | 0 | 0 |
| 5 | 0 | 13 | 2 | 38 | 76 | 2 | 0 | 69 |
| 6 | 64 | 38 | 17 | 40 | 20 | 14 | 0 | 7 |
| 7 | 19 | 69 | 19 | 59 | 2 | 0 | 32 | 0 |
| 8 | 23 | 64 | 0 | 94 | 0 | 0 | 0 | 19 |

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 196 | 0 | 3 | 0 | 0 | 0 | 0 | 1 |
| 2 | 98 | 36 | 38 | 0 | 2 | 0 | 14 | 12 |
| 3 | 0 | 0 | 165 | 0 | 0 | 0 | 0 | 35 |
| 4 | 0 | 0 | 37 | 163 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 60 | 0 | 65 | 0 | 0 | 72 |
| 6 | 70 | 0 | 9 | 34 | 0 | 74 | 0 | 13 |
| 7 | 59 | 0 | 35 | 20 | 0 | 0 | 86 | 0 |
| 8 | 4 | 0 | 0 | 34 | 0 | 0 | 0 | 162 |

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 181 | 4 | 13 | 0 | 0 | 0 | 0 | 2 |
| 2 | 1 | 167 | 32 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 197 | 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 15 | 184 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 7 | 0 | 176 | 0 | 2 | 15 |
| 6 | 0 | 15 | 3 | 0 | 0 | 177 | 0 | 5 |
| 7 | 7 | 0 | 20 | 2 | 0 | 0 | 150 | 21 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200 |

Table 7.6: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QSVM.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 199 |
| 2 | 0 | 16 | 0 | 73 | 0 | 4 | 0 | 107 |
| 3 | 0 | 0 | 0 | 0 | 0 | 2 | 6 | 192 |
| 4 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 184 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 199 |
| 6 | 0 | 0 | 0 | 0 | 0 | 76 | 4 | 120 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 179 |
| 8 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 198 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 21 | 168 | 0 | 0 | 0 | 10 | 0 |
| 2 | 0 | 103 | 93 | 0 | 0 | 0 | 4 | 0 |
| 3 | 0 | 62 | 20 | 8 | 0 | 0 | 110 | 0 |
| 4 | 0 | 137 | 13 | 46 | 0 | 0 | 4 | 0 |
| 5 | 0 | 170 | 2 | 0 | 5 | 0 | 26 | 0 |
| 6 | 0 | 29 | 16 | 51 | 0 | 0 | 20 | 84 |
| 7 | 0 | 138 | 0 | 1 | 0 | 0 | 56 | 5 |
| 8 | 0 | 26 | 13 | 0 | 0 | 0 | 1 | 160 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 162 | 1 | 31 | 0 | 0 | 0 | 2 |
| 2 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 43 | 140 | 12 | 0 | 0 | 5 | 0 |
| 4 | 0 | 18 | 0 | 182 | 0 | 0 | 0 | 0 |
| 5 | 0 | 21 | 126 | 7 | 46 | 0 | 0 | 0 |
| 6 | 0 | 33 | 7 | 18 | 0 | 94 | 5 | 43 |
| 7 | 0 | 4 | 1 | 5 | 0 | 0 | 186 | 4 |
| 8 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 198 |

Table 7.7: Confusion matrices using gray-scaled pixel for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the TQCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 95 | 9 | 0 | 94 | 0 | 0 | 1 | 1 |
| 2 | 0 | 144 | 0 | 41 | 0 | 3 | 0 | 12 |
| 3 | 0 | 53 | 28 | 46 | 0 | 73 | 0 | 0 |
| 4 | 0 | 28 | 0 | 172 | 0 | 0 | 0 | 0 |
| 5 | 0 | 175 | 0 | 25 | 0 | 0 | 0 | 0 |
| 6 | 3 | 55 | 11 | 12 | 0 | 54 | 1 | 64 |
| 7 | 5 | 114 | 1 | 3 | 0 | 0 | 65 | 12 |
| 8 | 0 | 19 | 0 | 8 | 0 | 0 | 0 | 173 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 195 | 0 | 0 | 0 | 0 | 0 | 5 | 0 |
| 2 | 48 | 149 | 3 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 188 | 3 | 0 | 0 | 9 | 0 |
| 4 | 0 | 0 | 5 | 176 | 0 | 16 | 3 | 0 |
| 5 | 0 | 0 | 49 | 49 | 55 | 0 | 47 | 0 |
| 6 | 0 | 2 | 3 | 0 | 0 | 167 | 11 | 17 |
| 7 | 7 | 0 | 98 | 0 | 0 | 0 | 95 | 0 |
| 8 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 180 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 170 | 0 | 26 | 0 | 0 | 0 | 4 | 0 |
| 2 | 9 | 167 | 15 | 9 | 0 | 0 | 0 | 0 |
| 3 | 5 | 1 | 164 | 14 | 0 | 11 | 2 | 0 |
| 4 | 0 | 0 | 0 | 200 | 0 | 0 | 0 | 0 |
| 5 | 14 | 2 | 22 | 3 | 153 | 0 | 6 | 0 |
| 6 | 0 | 3 | 0 | 9 | 0 | 187 | 1 | 0 |
| 7 | 2 | 0 | 5 | 10 | 0 | 0 | 183 | 0 |
| 8 | 1 | 0 | 10 | 7 | 0 | 0 | 4 | 178 |

Table 7.8: Confusion matrices using gray-scaled pixel for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the MMCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 154 | 0 | 46 | 0 | 0 | 0 | 0 |
| 2 | 0 | 146 | 0 | 54 | 0 | 0 | 0 | 0 |
| 3 | 0 | 80 | 0 | 120 | 0 | 0 | 0 | 0 |
| 4 | 0 | 48 | 0 | 152 | 0 | 0 | 0 | 0 |
| 5 | 0 | 157 | 0 | 43 | 0 | 0 | 0 | 0 |
| 6 | 0 | 58 | 0 | 142 | 0 | 0 | 0 | 0 |
| 7 | 0 | 98 | 0 | 102 | 0 | 0 | 0 | 0 |
| 8 | 0 | 91 | 0 | 109 | 0 | 0 | 0 | 0 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 150 | 45 | 0 | 4 | 0 | 0 | 0 | 1 |
| 2 | 35 | 163 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 27 | 93 | 49 | 11 | 0 | 1 | 10 | 9 |
| 4 | 2 | 24 | 0 | 160 | 0 | 0 | 1 | 13 |
| 5 | 5 | 75 | 0 | 6 | 34 | 11 | 54 | 15 |
| 6 | 28 | 7 | 4 | 56 | 0 | 51 | 22 | 32 |
| 7 | 25 | 33 | 3 | 21 | 0 | 0 | 100 | 18 |
| 8 | 27 | 20 | 0 | 42 | 0 | 0 | 0 | 111 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 166 | 24 | 0 | 3 | 0 | 1 | 3 | 3 |
| 2 | 18 | 178 | 0 | 3 | 0 | 0 | 1 | 0 |
| 3 | 7 | 13 | 54 | 101 | 0 | 13 | 9 | 3 |
| 4 | 0 | 1 | 0 | 186 | 0 | 1 | 12 | 0 |
| 5 | 6 | 19 | 0 | 129 | 28 | 11 | 6 | 1 |
| 6 | 2 | 5 | 0 | 8 | 0 | 179 | 0 | 6 |
| 7 | 33 | 17 | 0 | 21 | 1 | 14 | 114 | 0 |
| 8 | 14 | 22 | 0 | 32 | 0 | 14 | 0 | 118 |

Table 7.9: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the SVM.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200 |
| 2 | 0 | 9 | 0 | 6 | 0 | 0 | 0 | 185 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200 |
| 4 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 189 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 199 |
| 6 | 0 | 0 | 0 | 0 | 0 | 33 | 0 | 167 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 190 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 150 | 29 | 0 | 2 | 0 | 1 | 0 | 18 |
| 2 | 143 | 30 | 0 | 2 | 0 | 10 | 2 | 13 |
| 3 | 3 | 44 | 0 | 75 | 0 | 1 | 77 | 0 |
| 4 | 21 | 90 | 0 | 84 | 0 | 0 | 5 | 0 |
| 5 | 5 | 165 | 0 | 21 | 0 | 9 | 0 | 0 |
| 6 | 41 | 15 | 0 | 52 | 0 | 16 | 6 | 70 |
| 7 | 70 | 41 | 0 | 33 | 0 | 38 | 0 | 18 |
| 8 | 76 | 16 | 0 | 19 | 0 | 0 | 0 | 89 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 120 | 19 | 0 | 7 | 0 | 45 | 0 | 9 |
| 2 | 17 | 81 | 1 | 0 | 0 | 100 | 0 | 1 |
| 3 | 0 | 8 | 21 | 16 | 0 | 155 | 0 | 0 |
| 4 | 0 | 0 | 0 | 135 | 0 | 53 | 0 | 12 |
| 5 | 0 | 0 | 0 | 14 | 0 | 186 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 194 | 0 | 6 |
| 7 | 1 | 0 | 0 | 13 | 0 | 136 | 30 | 20 |
| 8 | 0 | 0 | 0 | 4 | 0 | 13 | 0 | 183 |

Table 7.10: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the CCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 189 | 0 | 11 | 0 | 0 | 0 | 0 |
| 2 | 0 | 199 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 118 | 0 | 82 | 0 | 0 | 0 | 0 |
| 4 | 0 | 65 | 0 | 135 | 0 | 0 | 0 | 0 |
| 5 | 0 | 179 | 0 | 21 | 0 | 0 | 0 | 0 |
| 6 | 0 | 120 | 0 | 80 | 0 | 0 | 0 | 0 |
| 7 | 0 | 165 | 0 | 35 | 0 | 0 | 0 | 0 |
| 8 | 0 | 137 | 0 | 63 | 0 | 0 | 0 | 0 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 129 | 62 | 0 | 6 | 1 | 1 | 0 | 1 |
| 2 | 25 | 171 | 0 | 1 | 0 | 0 | 1 | 2 |
| 3 | 17 | 50 | 53 | 61 | 5 | 3 | 4 | 7 |
| 4 | 2 | 20 | 0 | 163 | 2 | 0 | 3 | 10 |
| 5 | 20 | 88 | 1 | 11 | 56 | 9 | 7 | 8 |
| 6 | 10 | 13 | 1 | 46 | 0 | 76 | 14 | 40 |
| 7 | 21 | 21 | 1 | 17 | 2 | 0 | 93 | 45 |
| 8 | 11 | 33 | 0 | 39 | 0 | 0 | 0 | 117 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 156 | 33 | 0 | 4 | 1 | 0 | 1 | 5 |
| 2 | 79 | 115 | 0 | 1 | 1 | 0 | 0 | 4 |
| 3 | 16 | 14 | 75 | 31 | 7 | 2 | 17 | 38 |
| 4 | 3 | 6 | 0 | 162 | 4 | 2 | 16 | 7 |
| 5 | 40 | 13 | 1 | 20 | 95 | 5 | 15 | 11 |
| 6 | 28 | 5 | 0 | 18 | 3 | 119 | 15 | 12 |
| 7 | 17 | 13 | 2 | 6 | 4 | 1 | 127 | 30 |
| 8 | 21 | 4 | 0 | 32 | 0 | 1 | 0 | 142 |

Table 7.11: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the mCCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 185 | 15 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 71 | 125 | 0 | 0 | 0 | 0 | 4 | 0 |
| 3 | 137 | 61 | 0 | 0 | 0 | 0 | 2 | 0 |
| 4 | 126 | 12 | 0 | 39 | 0 | 0 | 0 | 23 |
| 5 | 23 | 69 | 0 | 57 | 4 | 0 | 33 | 14 |
| 6 | 175 | 10 | 0 | 0 | 0 | 0 | 0 | 15 |
| 7 | 181 | 8 | 0 | 1 | 0 | 0 | 10 | 0 |
| 8 | 136 | 1 | 0 | 0 | 0 | 0 | 0 | 69 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 151 | 48 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 197 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 4 | 162 | 0 | 0 | 12 | 0 | 0 | 0 | 26 |
| 5 | 96 | 3 | 0 | 4 | 11 | 0 | 86 | 0 |
| 6 | 156 | 0 | 0 | 0 | 0 | 0 | 0 | 44 |
| 7 | 192 | 0 | 0 | 1 | 0 | 0 | 5 | 2 |
| 8 | 117 | 0 | 0 | 0 | 0 | 0 | 0 | 83 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 195 | 2 | 0 | 0 | 0 | 0 | 1 | 2 |
| 2 | 141 | 54 | 0 | 0 | 0 | 0 | 4 | 1 |
| 3 | 196 | 0 | 0 | 0 | 0 | 0 | 4 | 0 |
| 4 | 132 | 0 | 0 | 32 | 0 | 0 | 0 | 36 |
| 5 | 65 | 5 | 0 | 7 | 90 | 0 | 33 | 0 |
| 6 | 132 | 0 | 0 | 1 | 0 | 0 | 0 | 67 |
| 7 | 177 | 0 | 0 | 6 | 0 | 0 | 12 | 5 |
| 8 | 97 | 0 | 0 | 0 | 0 | 0 | 0 | 103 |

Table 7.12: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the UCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 63 | 75 | 0 | 59 | 0 | 0 | 1 | 2 |
| 2 | 0 | 172 | 0 | 25 | 0 | 0 | 0 | 3 |
| 3 | 82 | 17 | 0 | 60 | 0 | 0 | 41 | 0 |
| 4 | 39 | 18 | 0 | 141 | 0 | 0 | 0 | 2 |
| 5 | 0 | 26 | 0 | 146 | 0 | 0 | 31 | 0 |
| 6 | 8 | 16 | 0 | 82 | 0 | 0 | 19 | 75 |
| 7 | 0 | 60 | 0 | 60 | 0 | 0 | 34 | 46 |
| 8 | 0 | 33 | 0 | 48 | 0 | 0 | 3 | 116 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 142 | 0 | 0 | 13 | 0 | 0 | 5 | 40 |
| 2 | 8 | 108 | 0 | 12 | 0 | 0 | 20 | 52 |
| 3 | 20 | 6 | 0 | 29 | 0 | 2 | 143 | 0 |
| 4 | 4 | 0 | 0 | 137 | 0 | 10 | 9 | 40 |
| 5 | 12 | 52 | 0 | 65 | 0 | 0 | 71 | 0 |
| 6 | 32 | 0 | 0 | 76 | 0 | 2 | 5 | 85 |
| 7 | 49 | 0 | 0 | 40 | 0 | 1 | 84 | 26 |
| 8 | 51 | 0 | 0 | 38 | 0 | 0 | 6 | 105 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 121 | 19 | 0 | 17 | 0 | 0 | 0 | 43 |
| 2 | 1 | 132 | 0 | 16 | 0 | 0 | 3 | 48 |
| 3 | 5 | 41 | 0 | 97 | 0 | 49 | 8 | 0 |
| 4 | 0 | 0 | 0 | 151 | 0 | 7 | 0 | 42 |
| 5 | 0 | 85 | 0 | 91 | 0 | 0 | 20 | 4 |
| 6 | 1 | 9 | 0 | 80 | 0 | 10 | 0 | 100 |
| 7 | 10 | 22 | 0 | 59 | 0 | 0 | 47 | 62 |
| 8 | 5 | 11 | 0 | 40 | 0 | 0 | 0 | 144 |

Table 7.13: Confusion matrices using gray-scaled pixels for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the EASEF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 66 | 15 | 0 | 118 | 0 | 0 | 1 | 0 |
| 2 | 0 | 115 | 0 | 85 | 0 | 0 | 0 | 0 |
| 3 | 10 | 19 | 0 | 154 | 0 | 0 | 17 | 0 |
| 4 | 5 | 7 | 0 | 188 | 0 | 0 | 0 | 0 |
| 5 | 0 | 12 | 0 | 183 | 0 | 0 | 5 | 0 |
| 6 | 0 | 7 | 0 | 129 | 0 | 0 | 27 | 37 |
| 7 | 14 | 9 | 0 | 126 | 0 | 0 | 47 | 4 |
| 8 | 0 | 6 | 0 | 99 | 0 | 0 | 10 | 85 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 62 | 107 | 0 | 30 | 0 | 0 | 1 | 0 |
| 2 | 0 | 182 | 0 | 18 | 0 | 0 | 0 | 0 |
| 3 | 47 | 19 | 0 | 82 | 0 | 0 | 52 | 0 |
| 4 | 22 | 21 | 0 | 157 | 0 | 0 | 0 | 0 |
| 5 | 0 | 25 | 0 | 134 | 0 | 0 | 41 | 0 |
| 6 | 7 | 19 | 0 | 86 | 0 | 0 | 27 | 61 |
| 7 | 0 | 70 | 0 | 56 | 0 | 0 | 40 | 34 |
| 8 | 0 | 28 | 0 | 44 | 0 | 0 | 14 | 114 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 70 | 20 | 0 | 109 | 0 | 0 | 1 | 0 |
| 2 | 0 | 142 | 0 | 58 | 0 | 0 | 0 | 0 |
| 3 | 3 | 43 | 0 | 141 | 0 | 0 | 13 | 0 |
| 4 | 3 | 9 | 0 | 188 | 0 | 0 | 0 | 0 |
| 5 | 0 | 100 | 0 | 95 | 0 | 0 | 5 | 0 |
| 6 | 4 | 15 | 0 | 109 | 0 | 0 | 30 | 42 |
| 7 | 16 | 16 | 0 | 112 | 0 | 0 | 54 | 2 |
| 8 | 0 | 7 | 0 | 91 | 0 | 0 | 8 | 94 |

Table 7.14: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QMMCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 143 | 30 | 4 | 8 | 5 | 2 | 8 | 0 |
| 2 | 5 | 193 | 0 | 0 | 0 | 0 | 0 | 2 |
| 3 | 4 | 0 | 176 | 2 | 4 | 1 | 0 | 13 |
| 4 | 4 | 0 | 1 | 193 | 0 | 0 | 0 | 2 |
| 5 | 44 | 12 | 0 | 3 | 70 | 14 | 0 | 57 |
| 6 | 4 | 2 | 0 | 8 | 56 | 105 | 0 | 25 |
| 7 | 6 | 6 | 0 | 69 | 1 | 2 | 84 | 32 |
| 8 | 11 | 2 | 0 | 8 | 20 | 0 | 0 | 159 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 128 | 53 | 0 | 8 | 2 | 0 | 2 | 7 |
| 2 | 1 | 197 | 0 | 0 | 0 | 1 | 0 | 1 |
| 3 | 5 | 1 | 177 | 6 | 5 | 0 | 6 | 0 |
| 4 | 0 | 1 | 0 | 199 | 0 | 0 | 0 | 0 |
| 5 | 30 | 36 | 2 | 12 | 86 | 21 | 1 | 12 |
| 6 | 21 | 8 | 0 | 17 | 31 | 110 | 0 | 13 |
| 7 | 15 | 8 | 0 | 32 | 0 | 0 | 136 | 9 |
| 8 | 45 | 11 | 1 | 7 | 6 | 1 | 4 | 125 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 134 | 41 | 1 | 9 | 1 | 1 | 2 | 11 |
| 2 | 0 | 198 | 0 | 0 | 0 | 1 | 0 | 1 |
| 3 | 1 | 3 | 175 | 5 | 1 | 0 | 13 | 2 |
| 4 | 0 | 0 | 0 | 199 | 0 | 0 | 1 | 0 |
| 5 | 24 | 15 | 5 | 16 | 110 | 16 | 3 | 11 |
| 6 | 1 | 20 | 0 | 25 | 19 | 116 | 2 | 17 |
| 7 | 4 | 16 | 0 | 26 | 0 | 0 | 147 | 7 |
| 8 | 7 | 24 | 1 | 9 | 0 | 4 | 8 | 147 |

Table 7.15: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QSVM.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 24 | 77 | 1 | 68 | 0 | 30 | 0 | 0 |
| 2 | 10 | 65 | 3 | 53 | 6 | 62 | 0 | 1 |
| 3 | 20 | 24 | 2 | 23 | 58 | 55 | 0 | 18 |
| 4 | 33 | 8 | 1 | 64 | 5 | 65 | 6 | 18 |
| 5 | 3 | 39 | 11 | 45 | 73 | 25 | 0 | 4 |
| 6 | 0 | 109 | 0 | 27 | 0 | 41 | 17 | 6 |
| 7 | 0 | 169 | 0 | 24 | 0 | 0 | 6 | 1 |
| 8 | 6 | 46 | 2 | 110 | 1 | 15 | 5 | 15 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 159 | 1 | 27 | 7 | 0 | 0 | 1 |
| 2 | 0 | 192 | 0 | 3 | 5 | 0 | 0 | 0 |
| 3 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 13 | 172 | 0 | 9 | 0 | 0 | 0 | 6 |
| 5 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 195 | 0 | 5 | 0 | 0 | 0 | 0 |
| 7 | 0 | 182 | 0 | 3 | 0 | 0 | 11 | 4 |
| 8 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 23 | 108 | 0 | 0 | 0 | 11 | 45 | 13 |
| 2 | 7 | 161 | 0 | 0 | 0 | 0 | 22 | 10 |
| 3 | 0 | 45 | 0 | 0 | 0 | 0 | 119 | 36 |
| 4 | 0 | 14 | 0 | 0 | 0 | 3 | 119 | 64 |
| 5 | 0 | 96 | 0 | 0 | 2 | 27 | 63 | 12 |
| 6 | 0 | 71 | 0 | 0 | 0 | 46 | 64 | 19 |
| 7 | 0 | 44 | 0 | 0 | 0 | 0 | 155 | 1 |
| 8 | 19 | 155 | 0 | 0 | 0 | 2 | 19 | 5 |

Table 7.16: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the TQCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 48 | 19 | 131 | 2 | 0 | 0 | 0 |
| 2 | 0 | 99 | 1 | 71 | 29 | 0 | 0 | 0 |
| 3 | 0 | 8 | 74 | 0 | 118 | 0 | 0 | 0 |
| 4 | 0 | 96 | 1 | 50 | 53 | 0 | 0 | 0 |
| 5 | 0 | 63 | 21 | 60 | 54 | 1 | 0 | 1 |
| 6 | 0 | 58 | 68 | 70 | 4 | 0 | 0 | 0 |
| 7 | 0 | 148 | 19 | 32 | 0 | 1 | 0 | 0 |
| 8 | 0 | 118 | 12 | 52 | 1 | 1 | 0 | 16 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 97 | 39 | 60 | 1 | 0 | 0 | 0 |
| 2 | 27 | 39 | 24 | 109 | 1 | 0 | 0 | 0 |
| 3 | 0 | 37 | 109 | 54 | 0 | 0 | 0 | 0 |
| 4 | 0 | 2 | 0 | 198 | 0 | 0 | 0 | 0 |
| 5 | 0 | 23 | 43 | 85 | 49 | 0 | 0 | 0 |
| 6 | 5 | 21 | 23 | 143 | 4 | 0 | 0 | 4 |
| 7 | 0 | 25 | 0 | 175 | 0 | 0 | 0 | 0 |
| 8 | 0 | 11 | 6 | 183 | 0 | 0 | 0 | 0 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 133 | 47 | 2 | 0 | 7 | 0 | 0 | 11 |
| 2 | 18 | 166 | 15 | 1 | 0 | 0 | 0 | 0 |
| 3 | 2 | 0 | 180 | 12 | 6 | 0 | 0 | 0 |
| 4 | 9 | 11 | 10 | 170 | 0 | 0 | 0 | 0 |
| 5 | 17 | 58 | 12 | 6 | 107 | 0 | 0 | 0 |
| 6 | 54 | 101 | 0 | 0 | 0 | 43 | 1 | 1 |
| 7 | 2 | 185 | 0 | 0 | 0 | 0 | 13 | 0 |
| 8 | 99 | 28 | 0 | 0 | 0 | 0 | 0 | 73 |

Table 7.17: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the MMCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 102 | 12 | 7 | 6 | 2 | 1 | 36 | 34 |
| 2 | 31 | 129 | 2 | 2 | 0 | 0 | 0 | 36 |
| 3 | 10 | 37 | 45 | 28 | 2 | 3 | 52 | 23 |
| 4 | 11 | 7 | 1 | 121 | 2 | 19 | 3 | 36 |
| 5 | 44 | 4 | 9 | 43 | 0 | 59 | 5 | 36 |
| 6 | 39 | 16 | 2 | 21 | 1 | 63 | 12 | 46 |
| 7 | 31 | 20 | 6 | 11 | 1 | 17 | 75 | 39 |
| 8 | 53 | 13 | 0 | 4 | 0 | 0 | 4 | 126 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 109 | 32 | 0 | 32 | 1 | 1 | 3 | 22 |
| 2 | 19 | 162 | 10 | 0 | 1 | 0 | 5 | 3 |
| 3 | 3 | 63 | 73 | 22 | 4 | 3 | 31 | 1 |
| 4 | 15 | 12 | 1 | 156 | 1 | 10 | 0 | 5 |
| 5 | 68 | 16 | 27 | 14 | 8 | 63 | 0 | 4 |
| 6 | 56 | 26 | 1 | 26 | 3 | 65 | 6 | 17 |
| 7 | 27 | 31 | 3 | 12 | 6 | 37 | 69 | 15 |
| 8 | 67 | 46 | 0 | 3 | 0 | 2 | 8 | 74 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 111 | 19 | 1 | 57 | 2 | 0 | 8 | 2 |
| 2 | 16 | 175 | 0 | 9 | 0 | 0 | 0 | 0 |
| 3 | 3 | 41 | 104 | 29 | 5 | 0 | 18 | 0 |
| 4 | 8 | 16 | 6 | 165 | 0 | 4 | 1 | 0 |
| 5 | 29 | 36 | 21 | 30 | 51 | 6 | 26 | 1 |
| 6 | 21 | 40 | 2 | 28 | 2 | 87 | 9 | 11 |
| 7 | 21 | 2 | 7 | 6 | 9 | 15 | 135 | 5 |
| 8 | 21 | 67 | 2 | 11 | 0 | 2 | 16 | 81 |

Table 7.18: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the SVM.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 20 | 100 | 0 | 52 | 0 | 27 | 1 | 0 |
| 2 | 14 | 78 | 1 | 49 | 2 | 56 | 0 | 0 |
| 3 | 21 | 15 | 1 | 20 | 62 | 36 | 0 | 18 |
| 4 | 34 | 6 | 1 | 58 | 3 | 70 | 9 | 19 |
| 5 | 3 | 36 | 10 | 38 | 68 | 14 | 0 | 4 |
| 6 | 0 | 98 | 0 | 18 | 0 | 36 | 44 | 4 |
| 7 | 0 | 132 | 0 | 19 | 0 | 0 | 49 | 0 |
| 8 | 8 | 59 | 1 | 87 | 0 | 15 | 30 | 0 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 64 | 134 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 4 | 190 | 0 | 6 | 0 | 0 | 0 | 0 |
| 3 | 3 | 197 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 83 | 98 | 0 | 5 | 0 | 0 | 0 | 14 |
| 5 | 56 | 144 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 3 | 184 | 0 | 2 | 0 | 3 | 0 | 8 |
| 7 | 1 | 181 | 0 | 0 | 0 | 0 | 0 | 18 |
| 8 | 8 | 187 | 0 | 5 | 0 | 0 | 0 | 0 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 18 | 106 | 0 | 0 | 1 | 10 | 3 | 62 |
| 2 | 17 | 165 | 0 | 1 | 1 | 12 | 3 | 1 |
| 3 | 39 | 26 | 0 | 0 | 0 | 6 | 2 | 127 |
| 4 | 0 | 13 | 0 | 0 | 0 | 56 | 2 | 129 |
| 5 | 19 | 122 | 0 | 2 | 4 | 34 | 0 | 19 |
| 6 | 0 | 24 | 0 | 3 | 0 | 68 | 14 | 91 |
| 7 | 3 | 40 | 0 | 3 | 0 | 56 | 31 | 67 |
| 8 | 8 | 74 | 0 | 3 | 0 | 9 | 0 | 106 |

Table 7.19: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the CCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 91 | 29 | 8 | 12 | 3 | 2 | 0 | 55 |
| 2 | 37 | 136 | 21 | 4 | 0 | 0 | 0 | 3 |
| 3 | 9 | 8 | 48 | 34 | 9 | 0 | 0 | 92 |
| 4 | 0 | 3 | 1 | 60 | 0 | 0 | 0 | 136 |
| 5 | 23 | 7 | 46 | 107 | 0 | 0 | 1 | 16 |
| 6 | 18 | 29 | 5 | 25 | 13 | 43 | 11 | 56 |
| 7 | 52 | 42 | 9 | 34 | 8 | 5 | 28 | 22 |
| 8 | 47 | 55 | 6 | 6 | 2 | 1 | 3 | 80 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 85 | 31 | 2 | 18 | 3 | 3 | 3 | 0 |
| 2 | 23 | 126 | 19 | 14 | 1 | 1 | 15 | 0 |
| 3 | 6 | 24 | 89 | 10 | 3 | 3 | 32 | 0 |
| 4 | 9 | 7 | 17 | 0 | 2 | 2 | 30 | 1 |
| 5 | 71 | 24 | 27 | 7 | 14 | 14 | 27 | 0 |
| 6 | 19 | 25 | 4 | 28 | 79 | 79 | 13 | 8 |
| 7 | 20 | 23 | 7 | 8 | 21 | 21 | 89 | 17 |
| 8 | 54 | 65 | 3 | 7 | 6 | 6 | 17 | 27 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 70 | 51 | 11 | 54 | 10 | 3 | 1 | 0 |
| 2 | 27 | 155 | 9 | 2 | 0 | 0 | 6 | 1 |
| 3 | 9 | 36 | 126 | 19 | 0 | 2 | 8 | 0 |
| 4 | 8 | 12 | 19 | 150 | 1 | 7 | 1 | 2 |
| 5 | 38 | 76 | 17 | 33 | 23 | 1 | 11 | 1 |
| 6 | 9 | 26 | 2 | 38 | 2 | 99 | 7 | 17 |
| 7 | 14 | 11 | 16 | 9 | 11 | 7 | 116 | 16 |
| 8 | 9 | 46 | 3 | 53 | 1 | 3 | 12 | 73 |

Table 7.20: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the mCCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 103 | 11 | 7 | 5 | 2 | 1 | 37 | 34 |
| 2 | 32 | 129 | 2 | 1 | 0 | 0 | 0 | 36 |
| 3 | 10 | 33 | 44 | 29 | 2 | 3 | 56 | 12 |
| 4 | 11 | 7 | 1 | 121 | 2 | 19 | 4 | 35 |
| 5 | 42 | 4 | 9 | 44 | 0 | 59 | 5 | 37 |
| 6 | 39 | 16 | 2 | 21 | 1 | 62 | 12 | 47 |
| 7 | 32 | 20 | 5 | 12 | 1 | 17 | 74 | 39 |
| 8 | 54 | 12 | 0 | 4 | 0 | 0 | 4 | 126 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 108 | 32 | 2 | 32 | 2 | 2 | 2 | 20 |
| 2 | 18 | 161 | 17 | 0 | 0 | 0 | 1 | 3 |
| 3 | 6 | 73 | 71 | 21 | 3 | 4 | 21 | 1 |
| 4 | 10 | 11 | 1 | 155 | 1 | 17 | 0 | 5 |
| 5 | 53 | 16 | 20 | 13 | 8 | 87 | 0 | 5 |
| 6 | 55 | 23 | 1 | 24 | 5 | 71 | 5 | 16 |
| 7 | 22 | 28 | 4 | 15 | 6 | 50 | 64 | 11 |
| 8 | 64 | 46 | 0 | 3 | 0 | 2 | 8 | 77 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 111 | 29 | 1 | 25 | 0 | 1 | 3 | 30 |
| 2 | 13 | 163 | 0 | 0 | 0 | 0 | 0 | 24 |
| 3 | 1 | 59 | 98 | 15 | 0 | 0 | 2 | 25 |
| 4 | 10 | 17 | 4 | 159 | 0 | 8 | 1 | 1 |
| 5 | 19 | 26 | 5 | 15 | 40 | 15 | 3 | 77 |
| 6 | 23 | 35 | 2 | 31 | 5 | 72 | 5 | 27 |
| 7 | 14 | 36 | 4 | 5 | 11 | 9 | 92 | 29 |
| 8 | 19 | 70 | 1 | 2 | 0 | 0 | 12 | 96 |

Table 7.21: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the UCF.

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 102 | 12 | 7 | 6 | 2 | 1 | 36 | 34 |
| 2 | 31 | 129 | 2 | 2 | 0 | 0 | 0 | 36 |
| 3 | 10 | 37 | 45 | 28 | 2 | 3 | 52 | 23 |
| 4 | 11 | 7 | 1 | 121 | 2 | 18 | 3 | 36 |
| 5 | 44 | 4 | 9 | 43 | 0 | 59 | 5 | 36 |
| 6 | 39 | 16 | 2 | 21 | 1 | 36 | 12 | 46 |
| 7 | 31 | 20 | 6 | 11 | 1 | 17 | 75 | 39 |
| 8 | 53 | 13 | 0 | 4 | 0 | 0 | 4 | 126 |

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 109 | 32 | 0 | 32 | 1 | 1 | 3 | 22 |
| 2 | 19 | 162 | 10 | 0 | 1 | 0 | 5 | 3 |
| 3 | 3 | 63 | 73 | 22 | 4 | 3 | 31 | 1 |
| 4 | 15 | 12 | 1 | 156 | 1 | 10 | 0 | 5 |
| 5 | 68 | 16 | 27 | 14 | 8 | 63 | 0 | 4 |
| 6 | 56 | 26 | 1 | 26 | 3 | 65 | 6 | 17 |
| 7 | 27 | 31 | 3 | 12 | 6 | 37 | 69 | 15 |
| 8 | 67 | 46 | 0 | 3 | 0 | 2 | 8 | 74 |

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 111 | 19 | 1 | 57 | 2 | 0 | 8 | 2 |
| 2 | 16 | 175 | 0 | 9 | 0 | 0 | 0 | 0 |
| 3 | 3 | 41 | 104 | 29 | 5 | 0 | 18 | 0 |
| 4 | 8 | 16 | 6 | 165 | 0 | 4 | 1 | 0 |
| 5 | 29 | 36 | 21 | 30 | 51 | 6 | 26 | 1 |
| 6 | 21 | 40 | 2 | 28 | 2 | 87 | 9 | 11 |
| 7 | 21 | 2 | 7 | 6 | 9 | 15 | 135 | 5 |
| 8 | 21 | 67 | 2 | 11 | 0 | 2 | 16 | 81 |

Table 7.22: Confusion matrices using HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the EASEF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 88 | 57 | 2 | 42 | 0 | 1 | 9 | 1 |
| 2 | 24 | 110 | 29 | 36 | 0 | 0 | 0 | 1 |
| 3 | 0 | 21 | 79 | 75 | 0 | 4 | 21 | 0 |
| 4 | 21 | 12 | 16 | 128 | 1 | 17 | 5 | 0 |
| 5 | 5 | 15 | 110 | 50 | 10 | 1 | 9 | 0 |
| 6 | 38 | 45 | 10 | 27 | 0 | 65 | 14 | 1 |
| 7 | 17 | 30 | 39 | 29 | 0 | 0 | 78 | 7 |
| 8 | 126 | 24 | 0 | 16 | 0 | 0 | 12 | 22 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 85 | 62 | 3 | 44 | 0 | 0 | 5 | 1 |
| 2 | 18 | 116 | 18 | 47 | 0 | 1 | 0 | 0 |
| 3 | 0 | 13 | 86 | 101 | 0 | 0 | 0 | 0 |
| 4 | 18 | 10 | 19 | 140 | 0 | 13 | 0 | 0 |
| 5 | 2 | 29 | 85 | 74 | 8 | 2 | 0 | 0 |
| 6 | 35 | 46 | 11 | 42 | 0 | 64 | 2 | 0 |
| 7 | 20 | 63 | 37 | 42 | 0 | 0 | 32 | 6 |
| 8 | 133 | 29 | 1 | 19 | 0 | 0 | 0 | 18 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 85 | 62 | 3 | 44 | 0 | 0 | 5 | 1 |
| 2 | 18 | 116 | 18 | 47 | 0 | 1 | 0 | 0 |
| 3 | 0 | 13 | 86 | 101 | 0 | 0 | 0 | 0 |
| 4 | 18 | 10 | 19 | 140 | 0 | 13 | 0 | 0 |
| 5 | 2 | 29 | 85 | 74 | 8 | 2 | 0 | 0 |
| 6 | 35 | 46 | 11 | 42 | 0 | 64 | 2 | 0 |
| 7 | 20 | 63 | 37 | 42 | 0 | 0 | 32 | 6 |
| 8 | 133 | 29 | 1 | 19 | 0 | 0 | 0 | 18 |

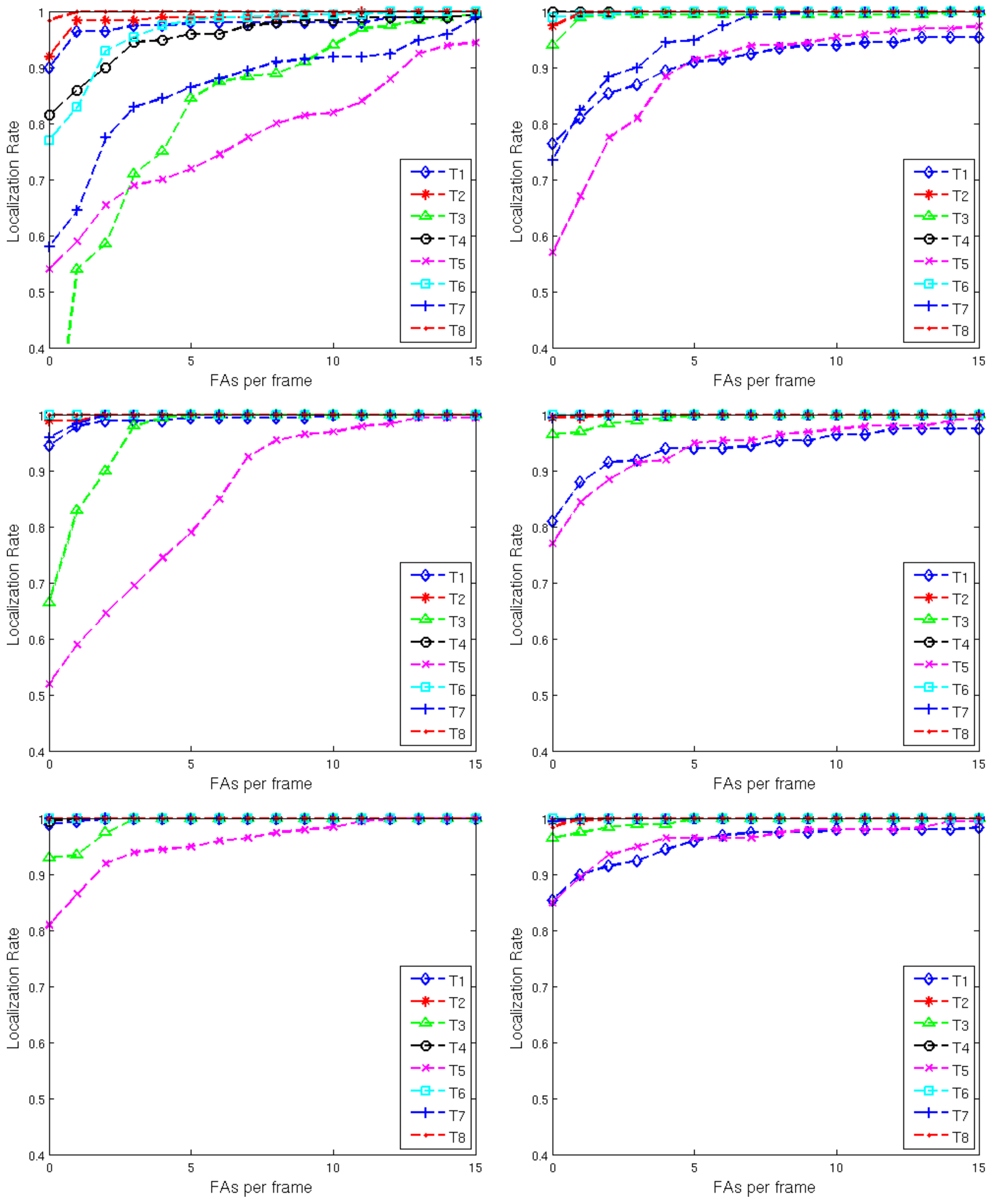


Figure 7.3: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QMMCF.

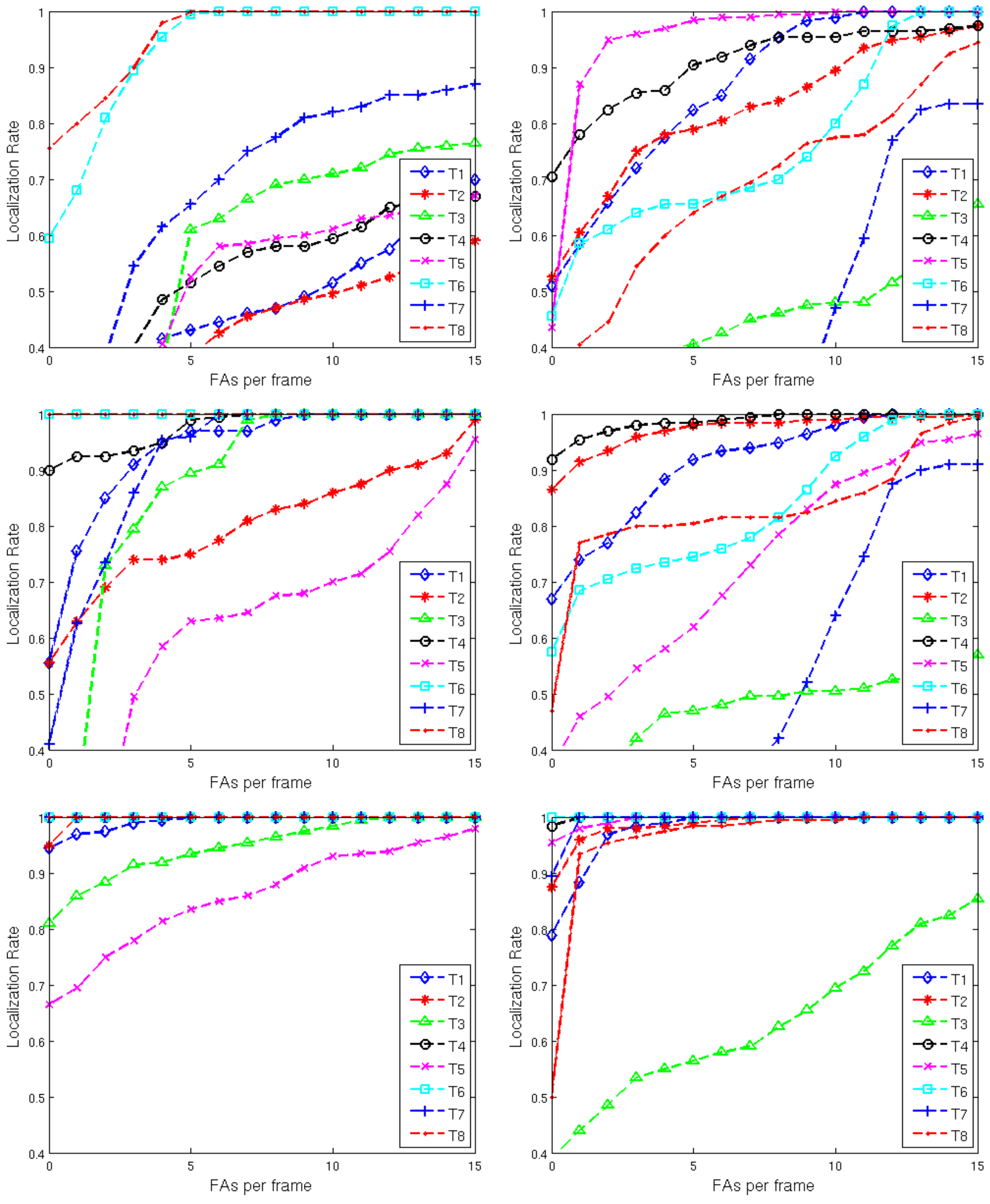


Figure 7.4: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QSVM.

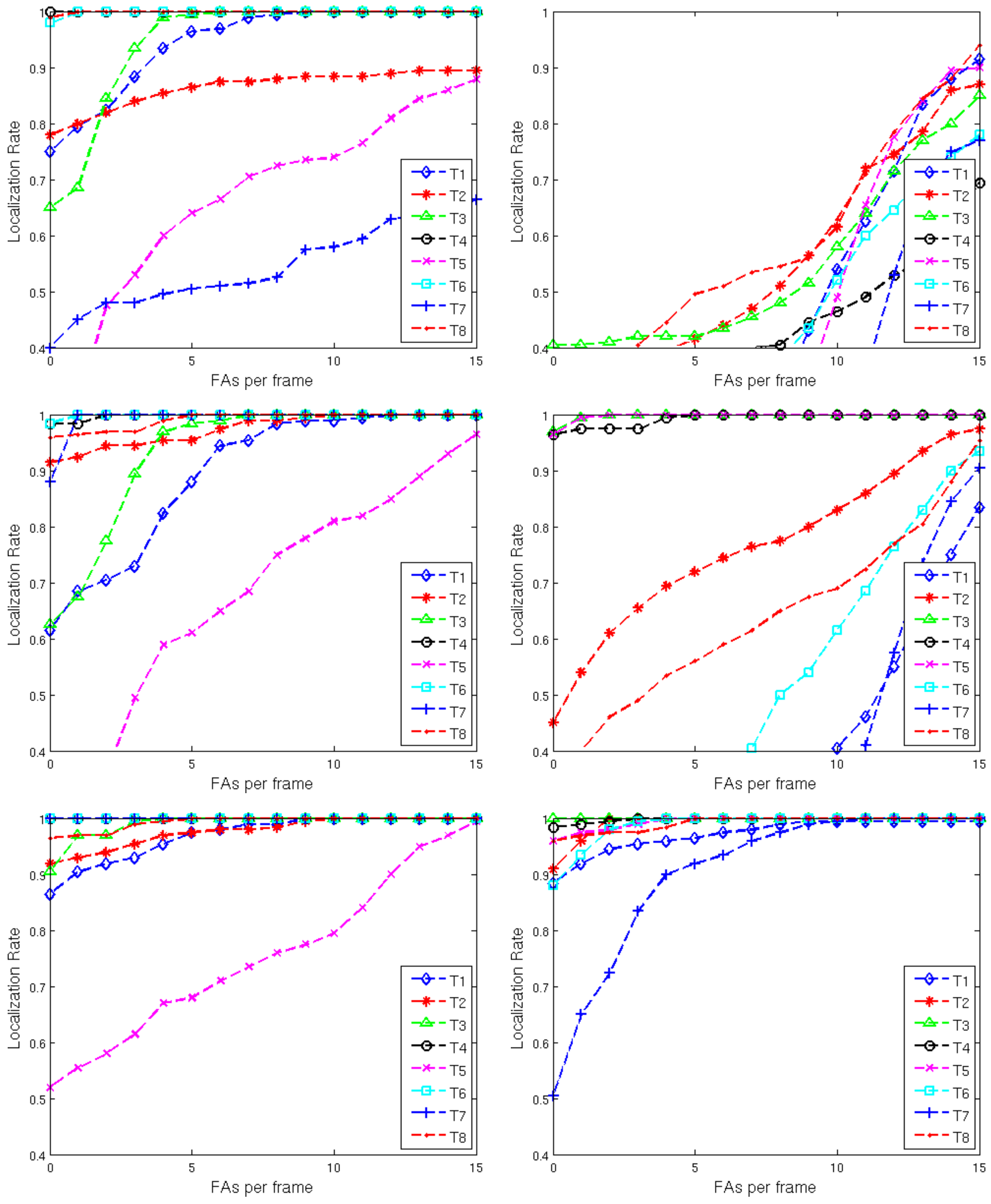


Figure 7.5: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the TQCF.

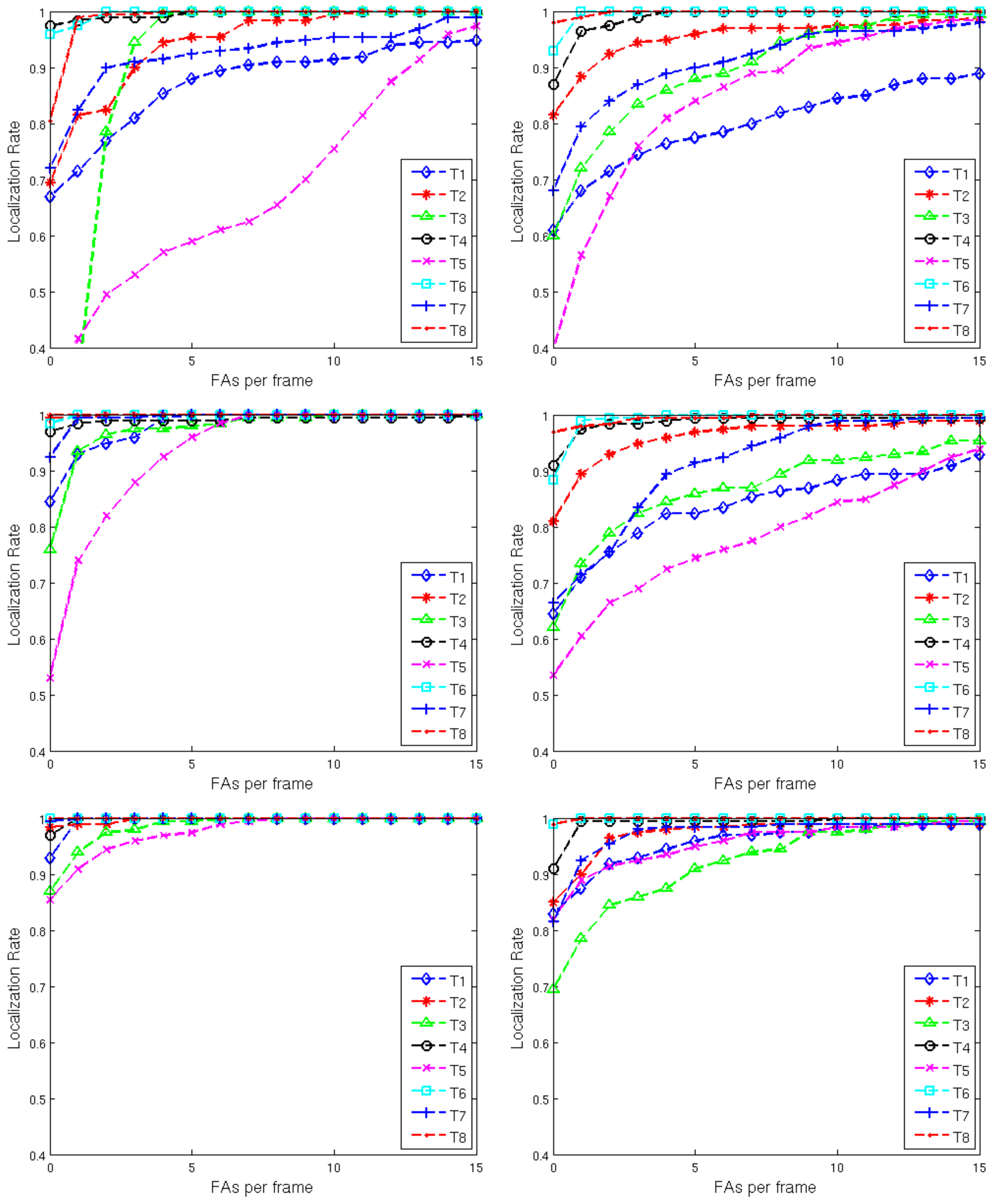


Figure 7.6: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the MMCF.

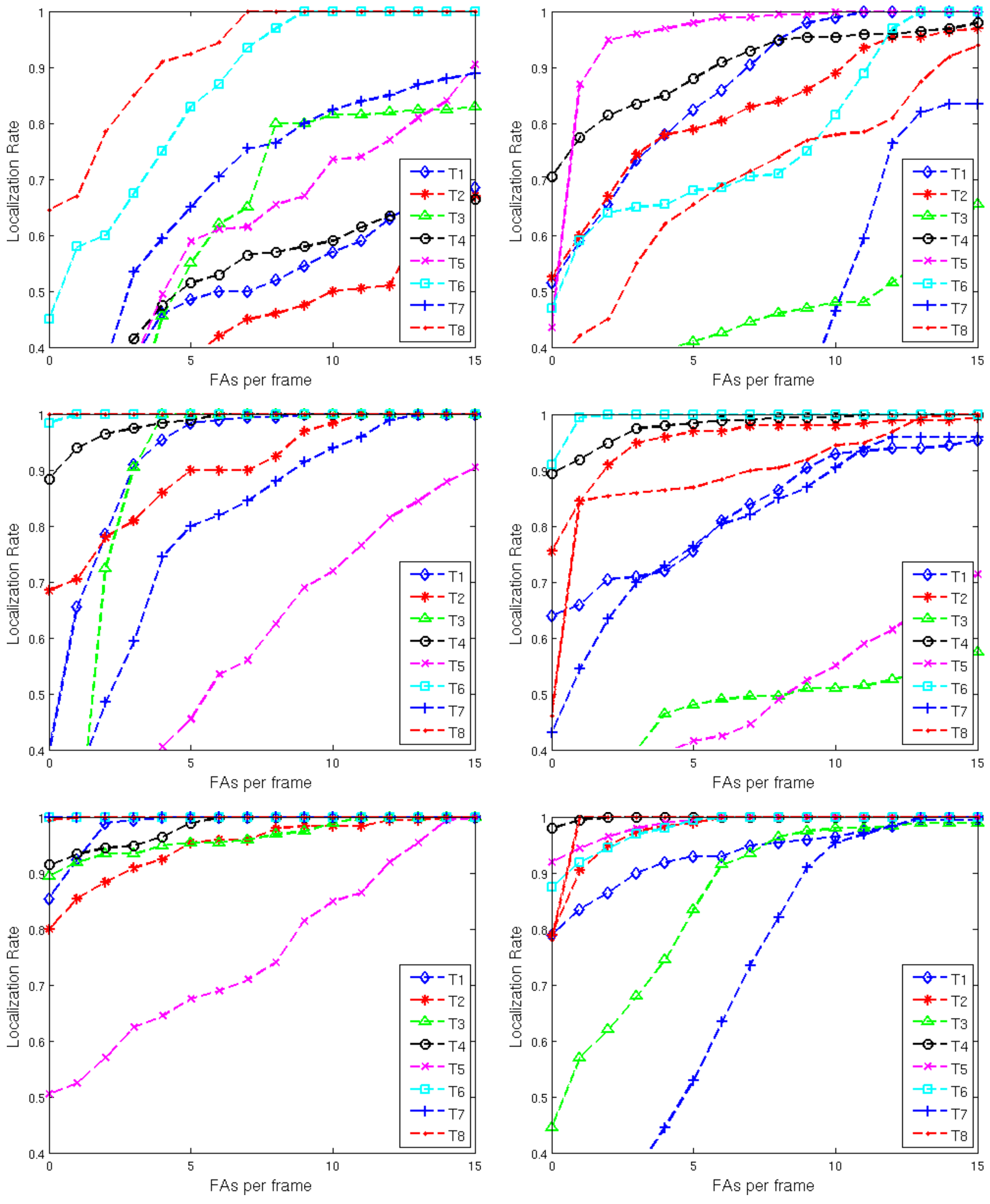


Figure 7.7: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the SVM.

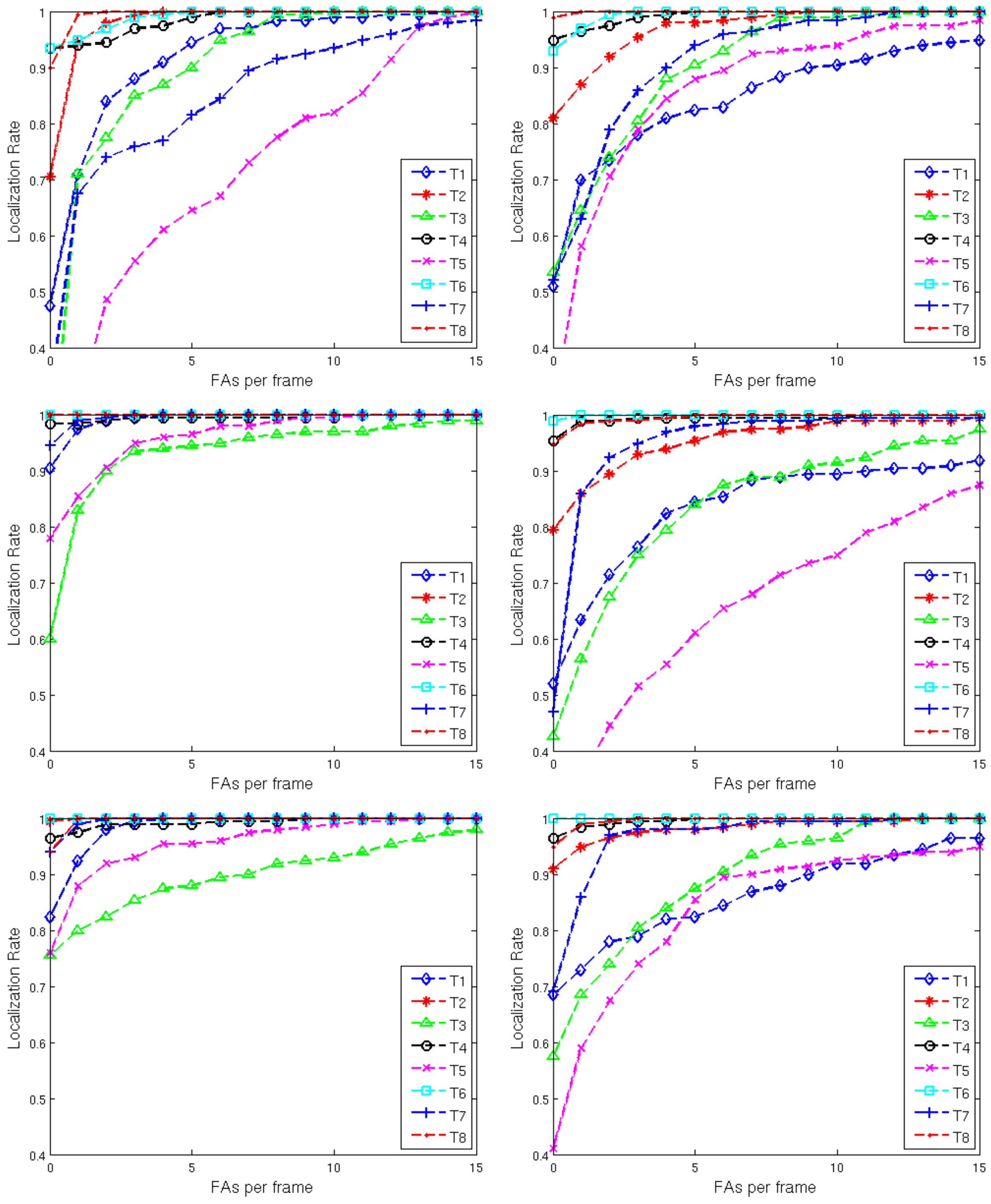


Figure 7.8: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the CCF.

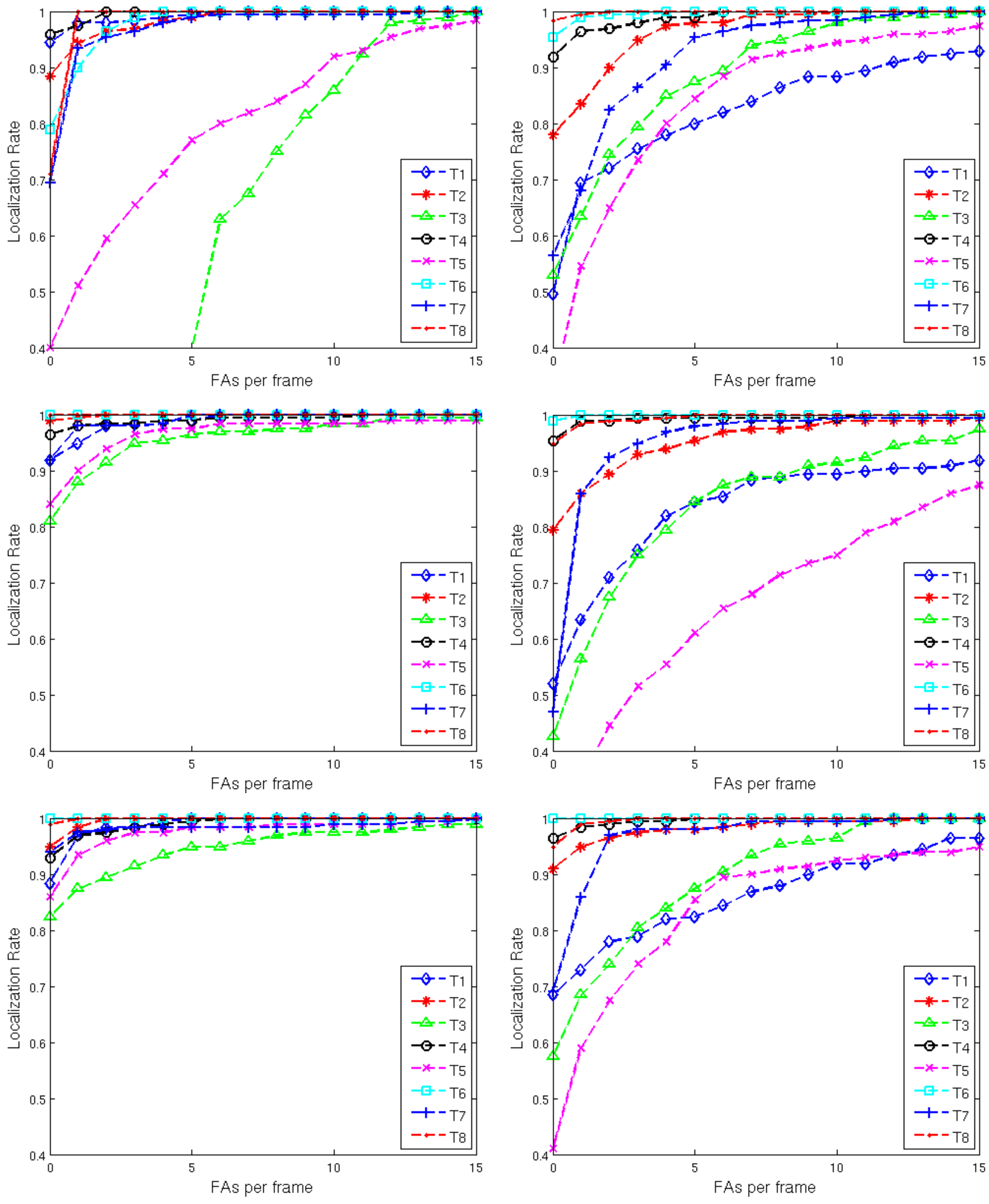


Figure 7.9: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the mCCF.

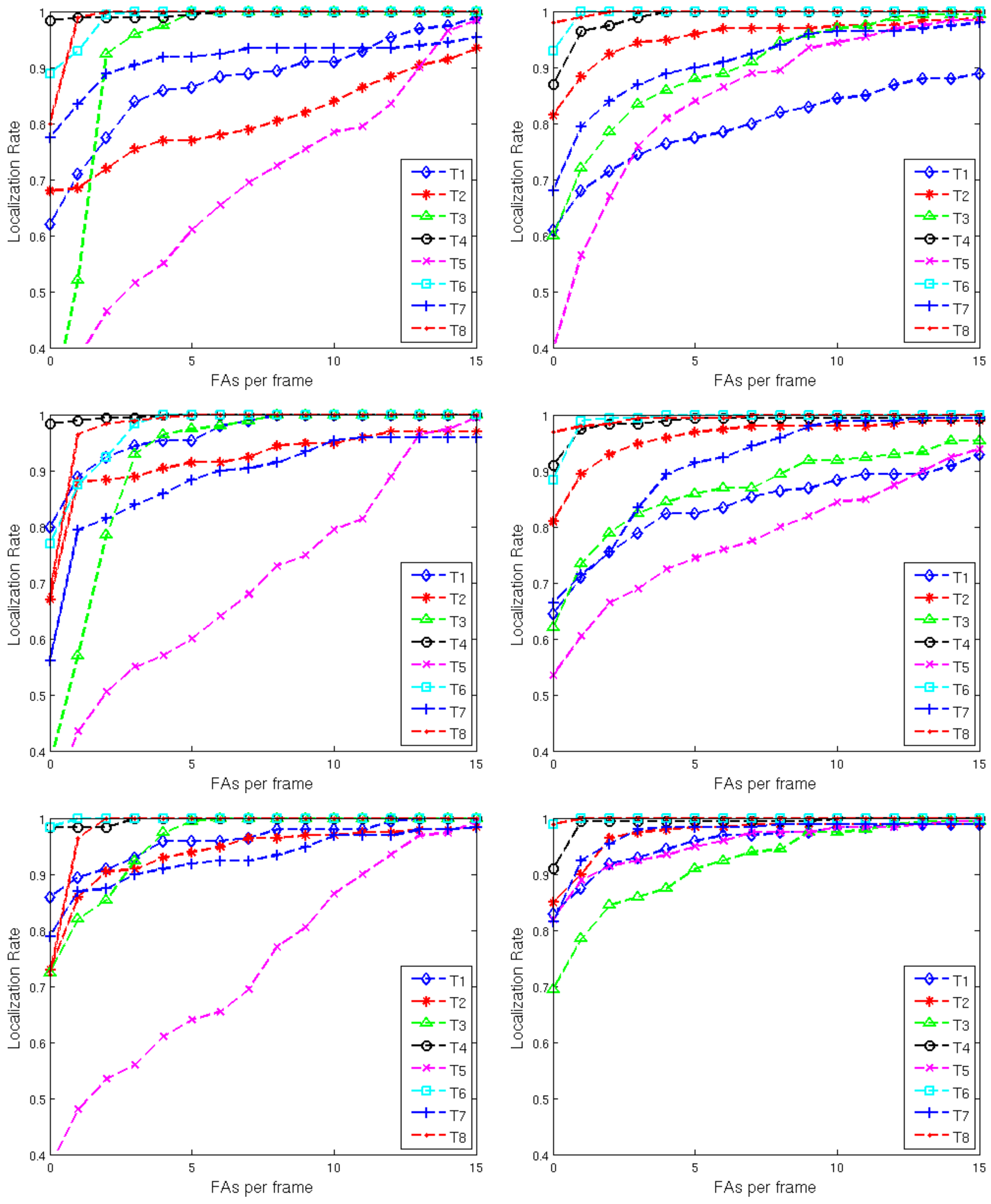


Figure 7.10: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the UCF.

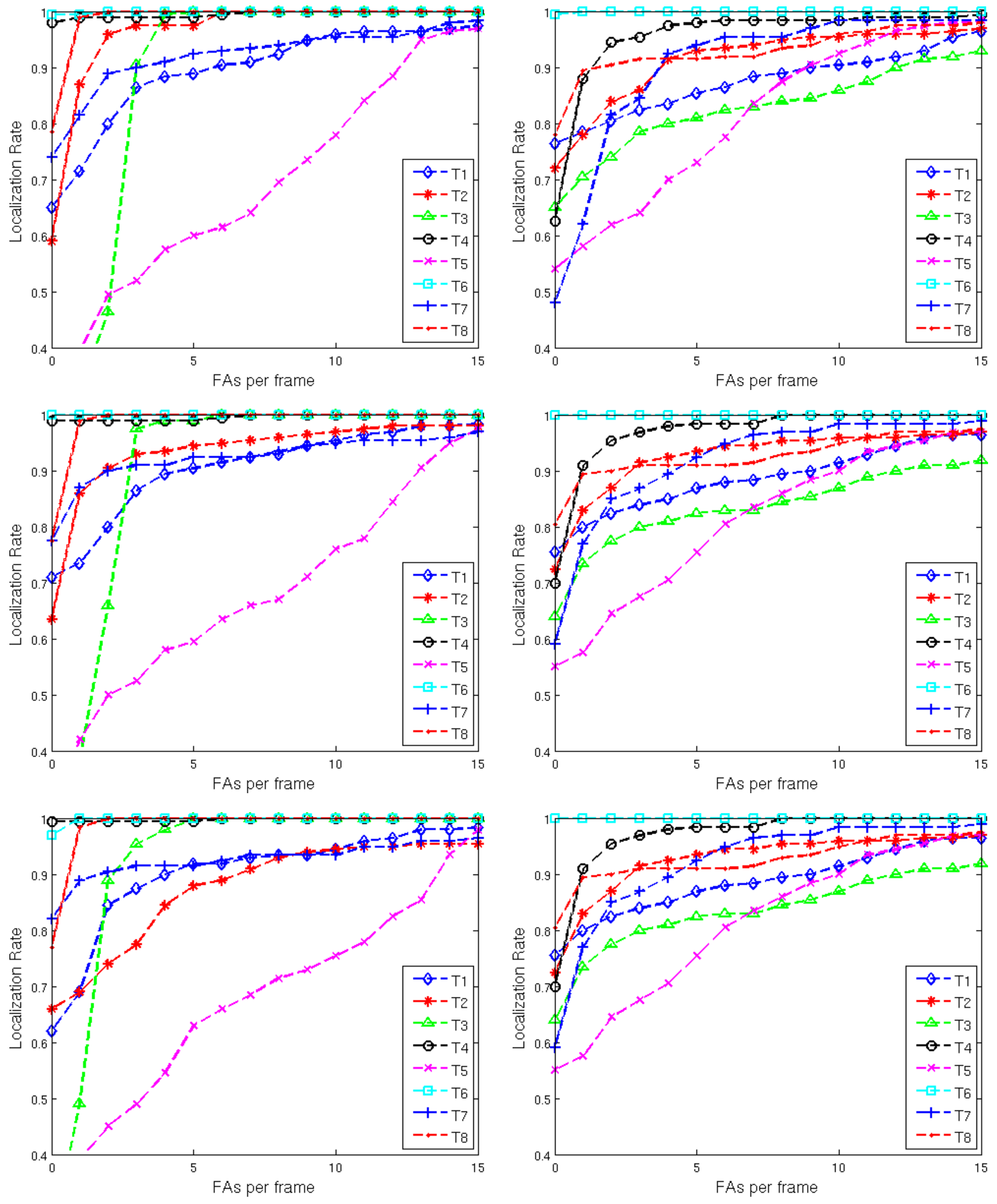


Figure 7.11: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the EASEF.

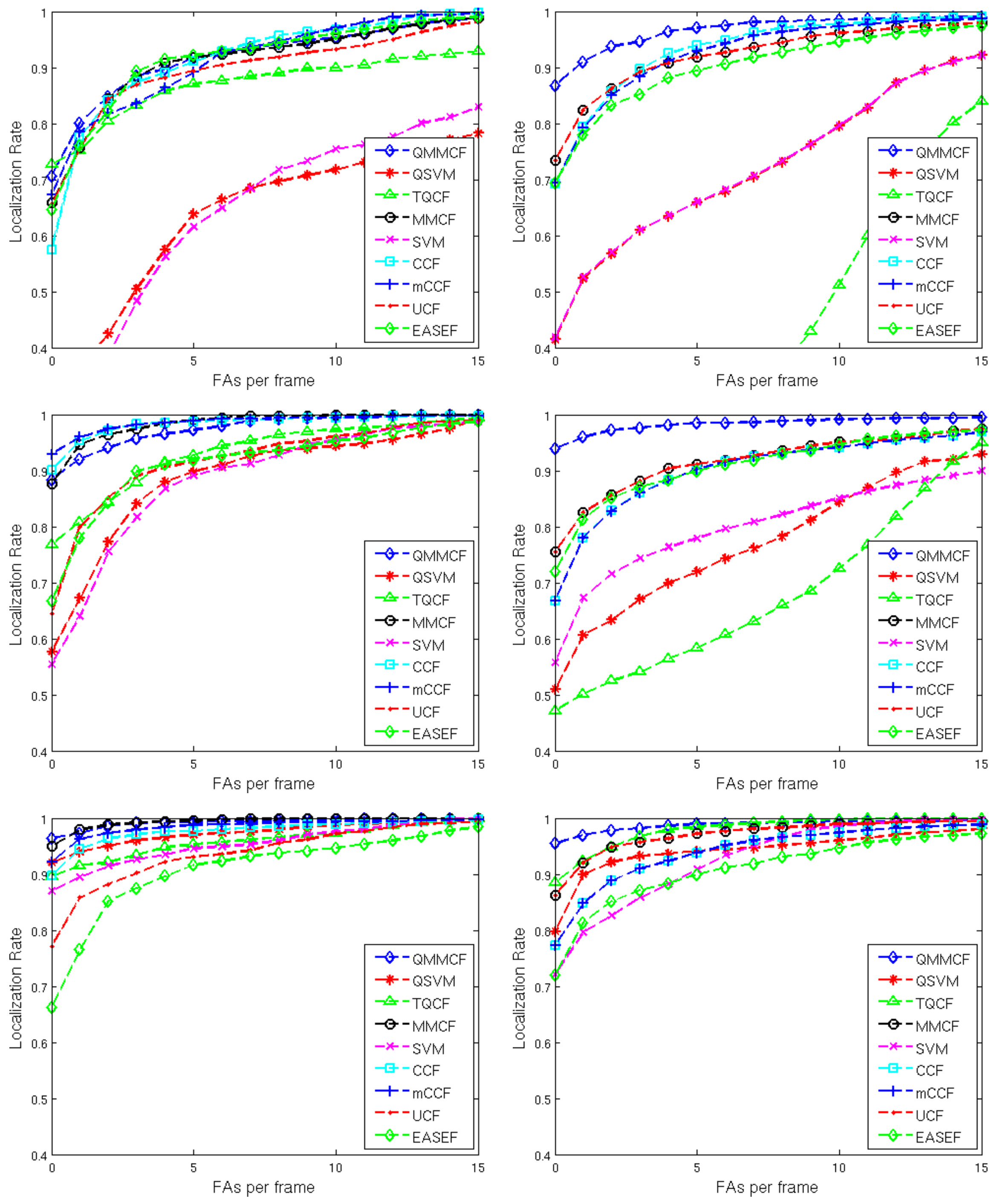


Figure 7.12: Localization rate as a function of the number of FAs per frame using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3.

These results show that using false-class images and retraining can significantly improve performance. In Set 1, most confusion matrices show that some targets are never detected (i.e., they have zeros along the column), but this rarely happens in Set 3. In addition, the ROC curves show that the localization performance also improves from Set 1 to Set 3. The results also show that our linear correlation filter designs can outperform all linear correlation filter, and our quadratic correlation filter designs outperform existing quadratic and all linear designs. In addition, they show that using the MSE and ONV criteria can significantly improve results. In Tables 7.2, 7.3, and 7.4, QMMCF outperforms QSVM using gray-scaled pixels by 118%, 143%, and 32%, and using HOG features by 763%, 555%, and 285%, respectively, TQCF outperforms QCF using gray-scaled pixels by 81%, 190%, and 49%, and using HOG features by 22%, 20%, and 150%, respectively, MMCF outperforms SVM using gray-scaled pixels by 200%, 178%, and 34%, and using HOG features by 167%, 330%, and 175%, respectively, in Sets 1, 2, and 3.

To observe the effect of each criterion, we compute the recognition rates using different $\lambda$ values and keep $\psi$ and $\mathbf{g}_{\sigma}^{2}$ constant; using different $\psi$ values and keep $\lambda$ and $\mathbf{g}_{\sigma^2}$ constant; and using different $\dot{\mathbf{g}}_{\sigma^2}$ value and keep $\lambda$ and $\psi$ constant. Figures 7.13, 7.14, and 7.15 (best viewed in color) shows the effect of changing $\lambda$, $\psi$, and $\dot{\mathbf{g}}_{\sigma^2}$ values, respectively, while keeping the other values constant. Note that the plots in Figure 7.13 corresponding to Set 3 include QSVM and SVM. Earlier, we discussed that MMCF is equivalent to SVM, and QMMCF is equivalent to QSVM when $\lambda = 1$ (and $\psi = 1$ and $\mathbf{g}_{\sigma^2} = 0$ for MMCF). Before retraining, all the filters have the same training images, but as we retrain, each filter selects a different set of false-class training images. Therefore, MMCF usually has a different set of false-class training images for each $\lambda$ value after retraining. However, to investigate the effect of changing $\lambda$ for a fixed set of training images, we designed the MMCF using the same set of training images obtained after retraining for the $\lambda$ that gave the best recognition rate (in our experiments this was $\lambda = 0.25$ for gray-scaled pixels and $\lambda = 10^{-4}$ for HOG features). In the plots in Set 3, the graphs corresponding to SVM are actually MMCF plots with varying $\lambda$ using the SVM set of images obtained after retraining.

From these figures we observe that the ASM criterion does not significantly affect our results.

Setting $\psi = 1$ (i.e., ignoring the ASM criterion) usually gives the best results (on a few filters setting $\psi = 0.99$ or $\psi = 0.999$ may give a small improvement of less than $1\%$). Although the MACH filter benefited from varying $\psi$, UCF with $\psi = 1$ always outperformed the MACH filter in our experiments. We also observe that varying the variance of the desired Gaussian-function-like correlation output affects the unconstrained filters and mCCF when using gray-scaled pixels but it has negligible effects otherwise. Furthermore, when false-class training images are used, constrained CFs usually outperformed unconstrained CF. Therefore, if false-class training images are available, then setting $\mathbf{g}_{\sigma^2} = 0$ usually gives the best performance. We observe that varying the $\lambda$ value, which represents a tradeoff between ONV and ACE, has a significant effect on the recognition rates . In all the plots, the recognition rate is lowest at the edges, i.e., when $\lambda = 0$ or $\lambda = 1$, and highest in between. Therefore, we conclude that a tradeoff between ONV and MSE is very beneficial.

We also compare performance as a function of the normalized distance $D$ in Fig. 7.16 for all sets of experiments using the QMMCF, QSVM, TQCF, MMCF, SVM, CCF, mCCF, UCF and EASEF classifiers. In all sets of experiments, QMMCF outperforms all the other classifiers for all values of $D$. We observe that the improvement when $D > 0.3$ is insignificant over $D = 0.3$. This means when the target is correctly recognized, the classifier usually puts the target's location within 21 and 12 pixels in the $x$- and $y$-direction, respectively, of the ground truth location.

We next investigated the relation between the number of support vectors that the QMMCF, QSVM, MMCF, SVM, and CCF filters have (using $\mathbf{g}_{\sigma^2} = 0$ and $\psi = 1$ for MMCF and CCF). MMCF , SVM, and CCF can all be formulated as a weighted sum of the training images (or transformed images) as in Eqs. 4.67 (when $\mathbf{p} = \mathbf{0}$), 4.10, and 3.38 (when $\mathbf{p} = \mathbf{0}$), respectively, and QMMCF and QSVM can all be formulated as a weighted sum of the squared inner product of the training images (or transformed images) as in Eqs. 5.41 and 5.16, respectively. The vectors $\mathbf{a}$ used in QMMCF, QSVM, MMCF and SVM have many zero elements while CCF does not have any zero elements. We referred to the training vectors corresponding to non-zero elements of $\mathbf{a}$ as support vectors. Fig. 7.17 compares the average percentage of supports vectors to the number of training images over the eight (one per vehicle) templates. We see from this figure that SVM (i.e., MMCF with $\lambda = 0$) has the fewest number of support vectors. As the value of $\lambda$ increases, the number of support vectors that QMMCF and MMCF have slightly increases, and CCF uses all the support vectors for all $\lambda$ values.. This trend was observed for all QMMCFs, QSVMs, MMCFs, SVMs and CCFs in all our experiments. We note that using all training images as support vectors (as is the case in CCF), usually decreases performances as the number of training images increases because the classifier overfits to the training images. MMCF and QMMCF use more support vectors than SVM and QSVM, respectively, to improve localization (and therefore recognition) while usually maintaining a smaller number of support vectors than CCF which avoids overfitting.

In addition, we investigated how the margin criterion $\mathbf{h}^\dagger\mathbf{h}$ (i.e., the ONV criterion when $\mathbf{P} = \alpha\mathbf{I}$ for all $\alpha \neq 0$) and the localization criterion $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ (i.e., the MSE criterion when $\dot{\mathbf{g}} = \mathbf{0}$) vary as a function of $\lambda$. As discussed earlier, when $\lambda = 1$ we minimize only the localization criterion $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$, and when $\lambda = 0$ we minimize only the margin criterion $\mathbf{h}^\dagger\mathbf{h}$. For QMMCF, QSVM, QCF, MMCF, SVM, CCF, mCCF, UCF and EASEF we show the average (over the eight filters–one per vehicle) values of criteria $\mathbf{h}^\dagger\mathbf{h}$ and $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ as function of $\lambda$ in Figs. 7.18 through 7.24, respectively. Notice from Figs. 7.20, 7.21, and 7.22 that MMCF, CCF, and mCCF, respectively, are the only filters for which as $\lambda$ increases, $\mathbf{h}^\dagger\mathbf{h}$ monotonically increases and $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ monotonically decreases. This is because these are the only filters that are designed to optimally trades-off between these two criteria. Note that QMMCF is also designed as a tradeoff between these criteria but, do to computational constraints, our actual implementation is not (as discussed in Ch. 5, QMMCF is implemented as a QSVM with the training vectors premultiplied by the MMCF matrix $\mathbf{T}$). A good filter usually has a $\lambda$ value that produces a low value for both of these criteria. We observe that for MMCF, CCF, and mCCF, a slight increase in one criterion can significantly improve the other criterion. For example, in the MMCF experiment in Set 1, increasing the value of $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ from 21.0 at $\lambda = 1$ to 38.1 at $\lambda = 0.9$, decreases the value of $\mathbf{h}^\dagger\mathbf{h}$ from $7.1 \times 10^6$ to $3.4 \times 10^5$. This is the reasons for QMMCF outperforming QSVM, MMCF outperforming SVM, and CCF outperforming both ECPSDF and MACE filters.

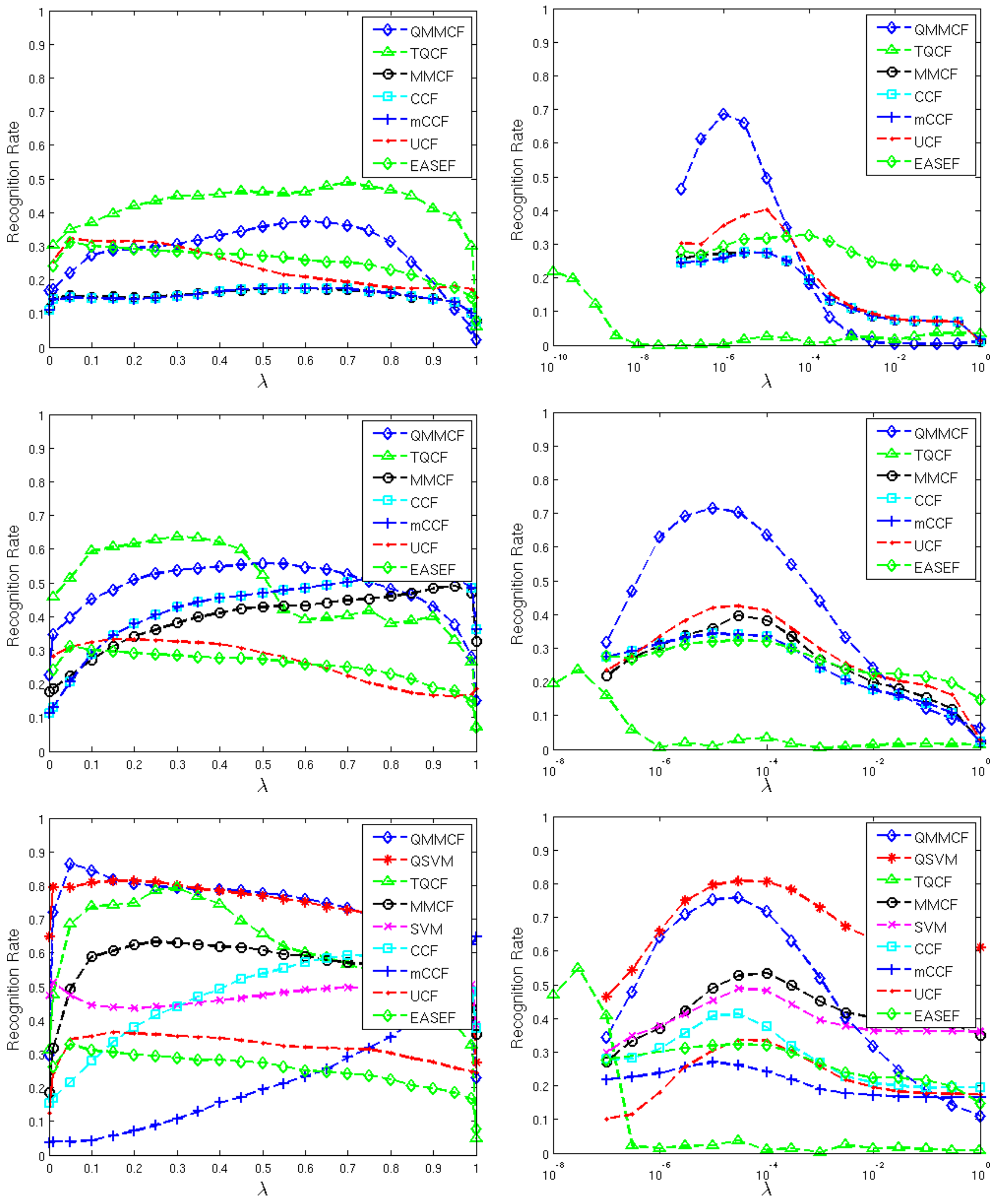


Figure 7.13: Recognition rate as a function of lambda $\lambda$ for different classifiers using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3. In all the plots, the recognition rate is lowest at the edges, i.e., when $\lambda = 0$ or $\lambda = 1$, and has the highest values in between.

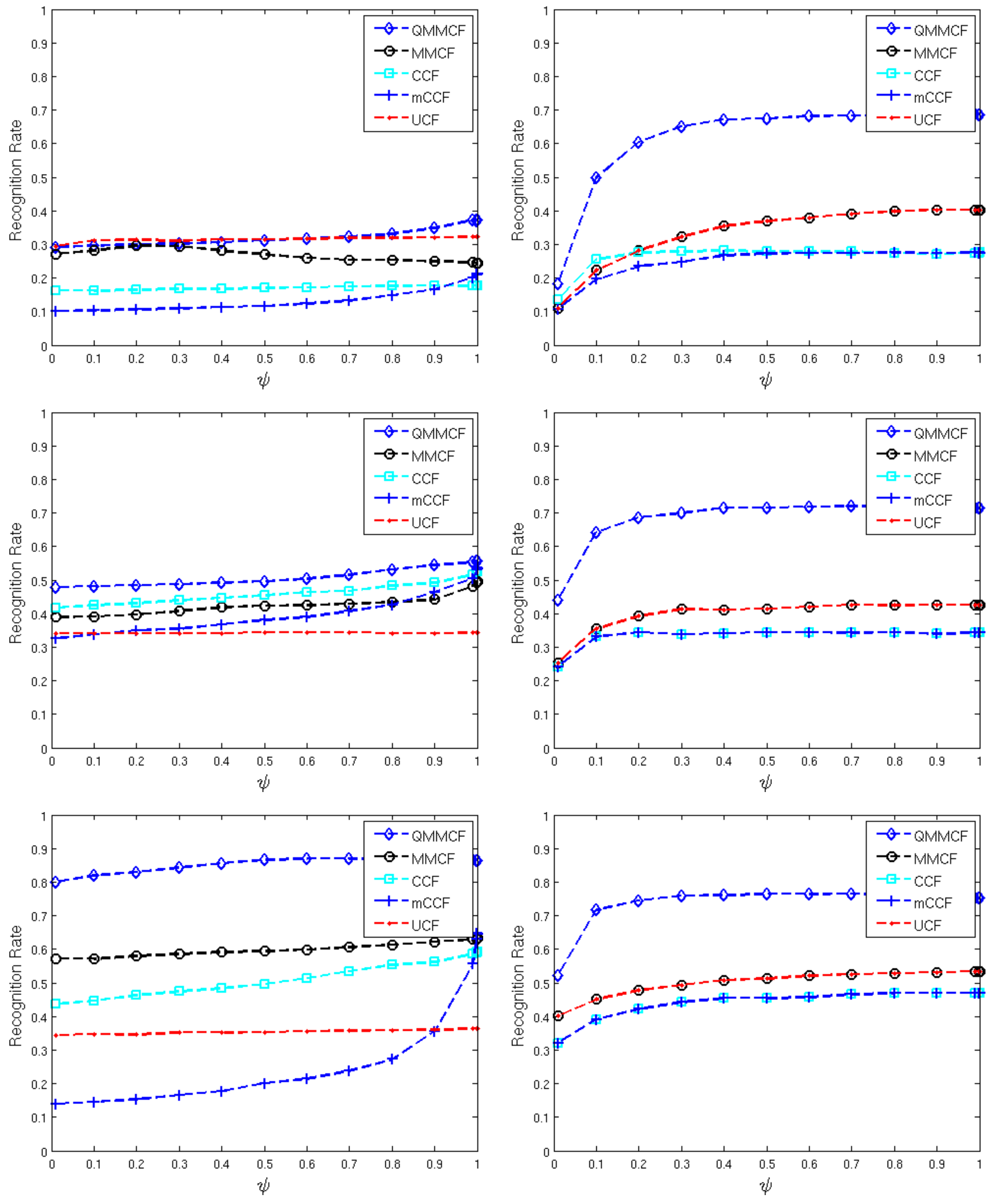


Figure 7.14: Recognition rate as a function of psi $\psi$ for different classifiers using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3.

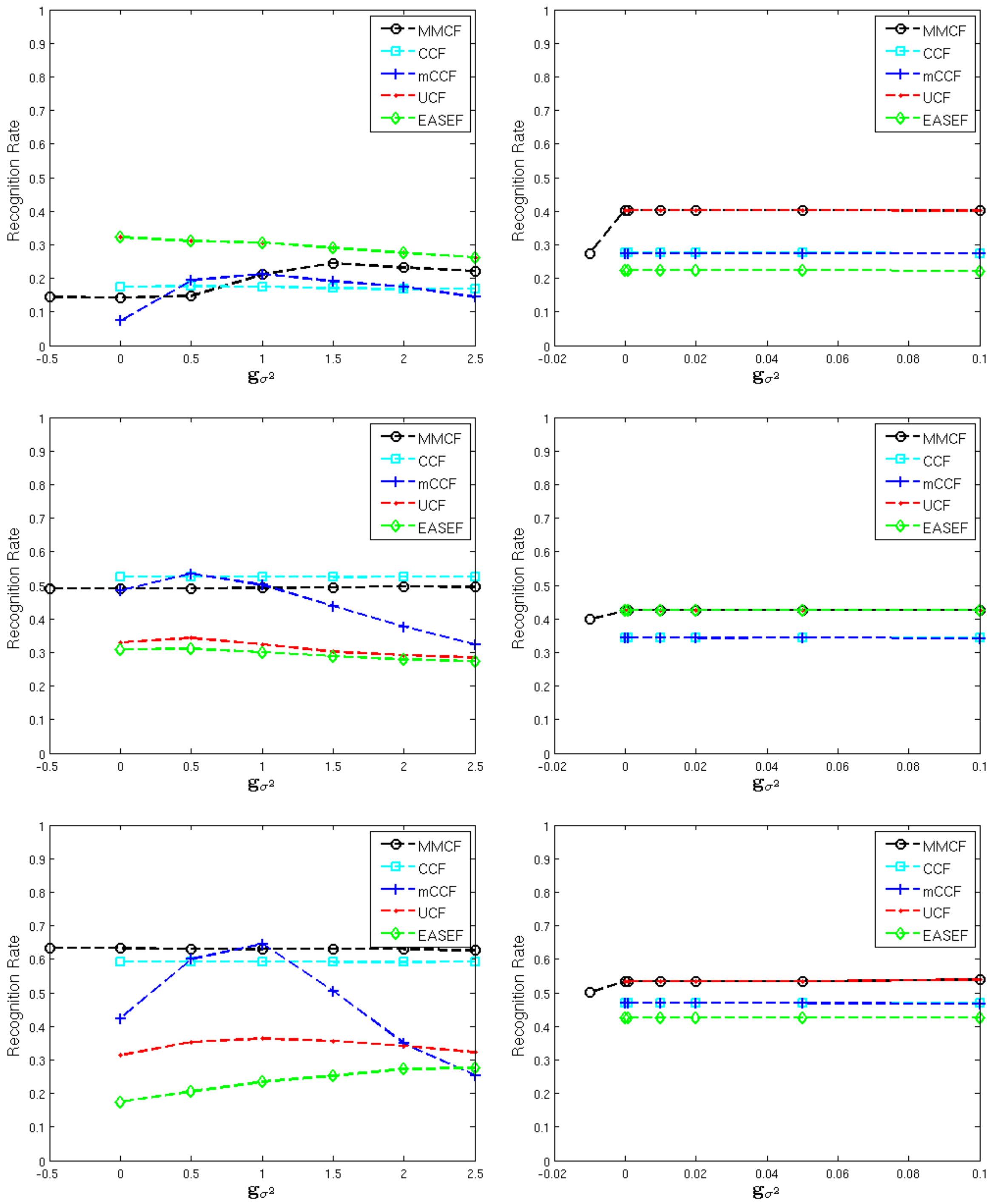


Figure 7.15: Recognition rate as a function of variance $\mathbf{g}_{\sigma^2}$ for different classifiers using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3. Note that a negative variance for MMCF means that the desired correlation plane is zero, i.e., $\mathbf{g} = \mathbf{0}$, and a zero variance means that the desired correlation plane is a delta function.

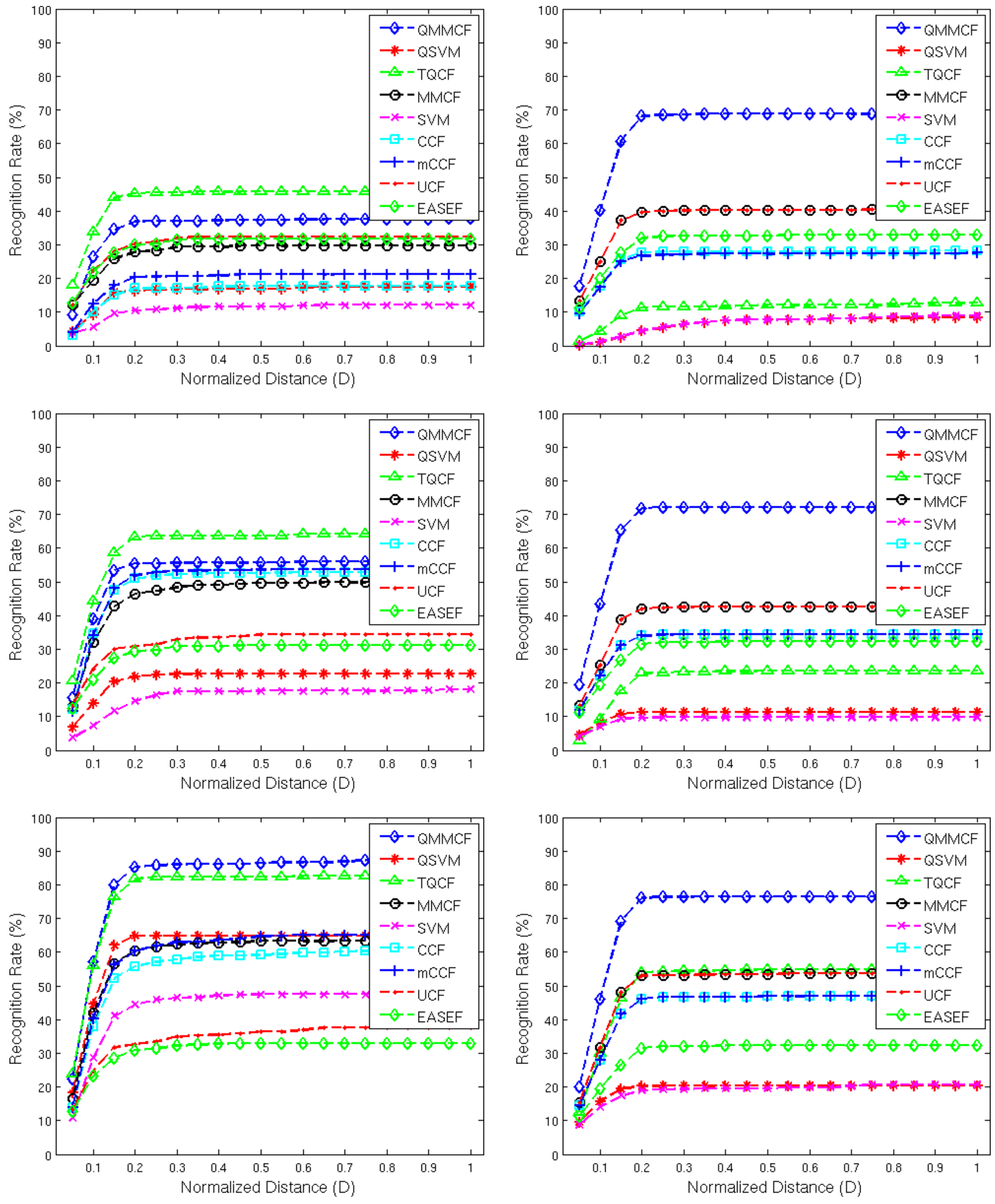


Figure 7.16: Recognition rate as a function of normalized localization error $D$ for different classifiers using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3. MMCF outperforms all linear classifiers (except mCCF in (a) with slightly lower performance), and QMMCF outperforms all quadratic classifiers. Usually performance does not improve when $D > 0.3$.

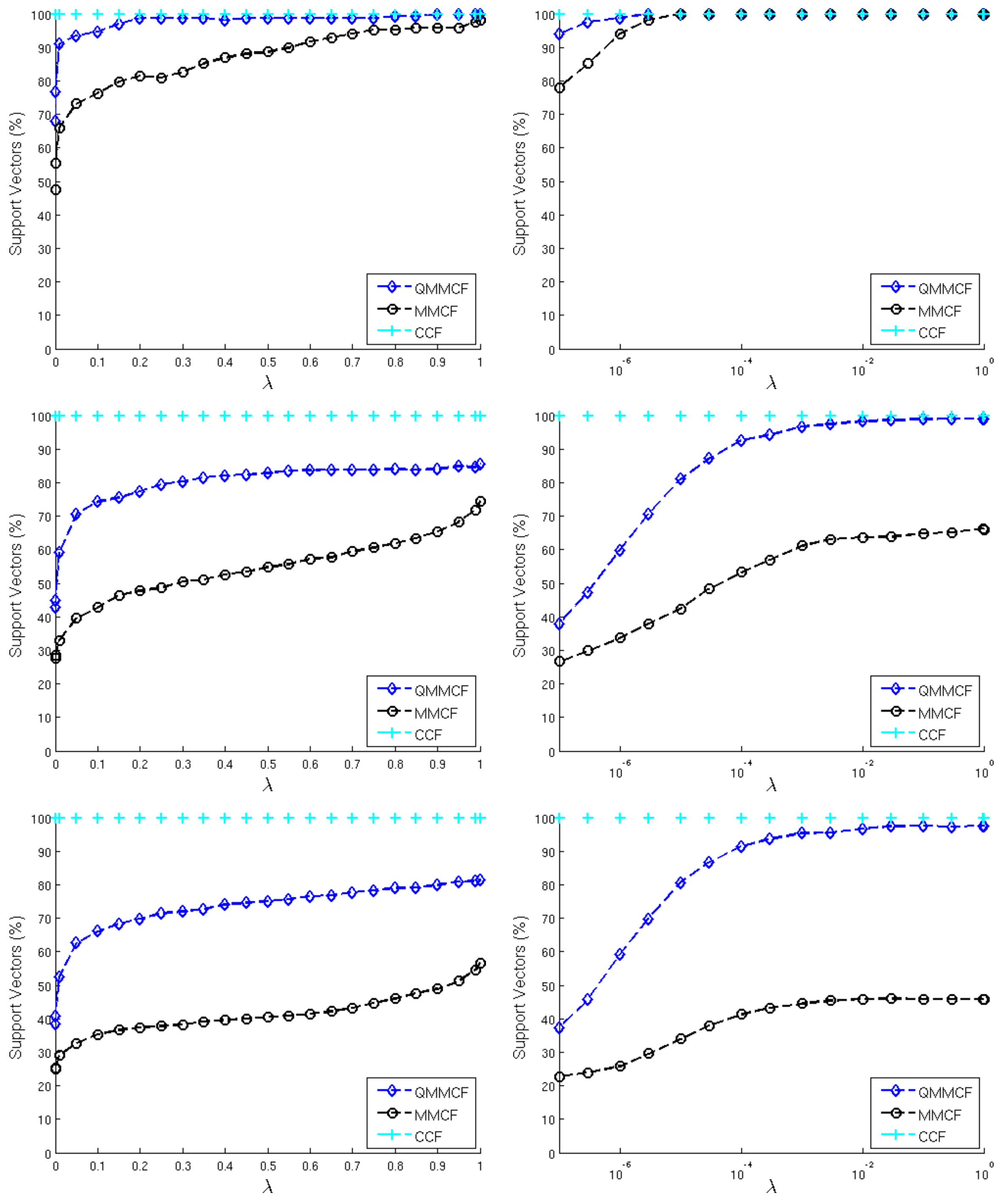


Figure 7.17: The percent of support vectors (i.e., support vectors over number of training images) for QMMCF, MMCF, and CCF (using $g_{\sigma^2} = 0$ and $\psi = 1$) as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3.

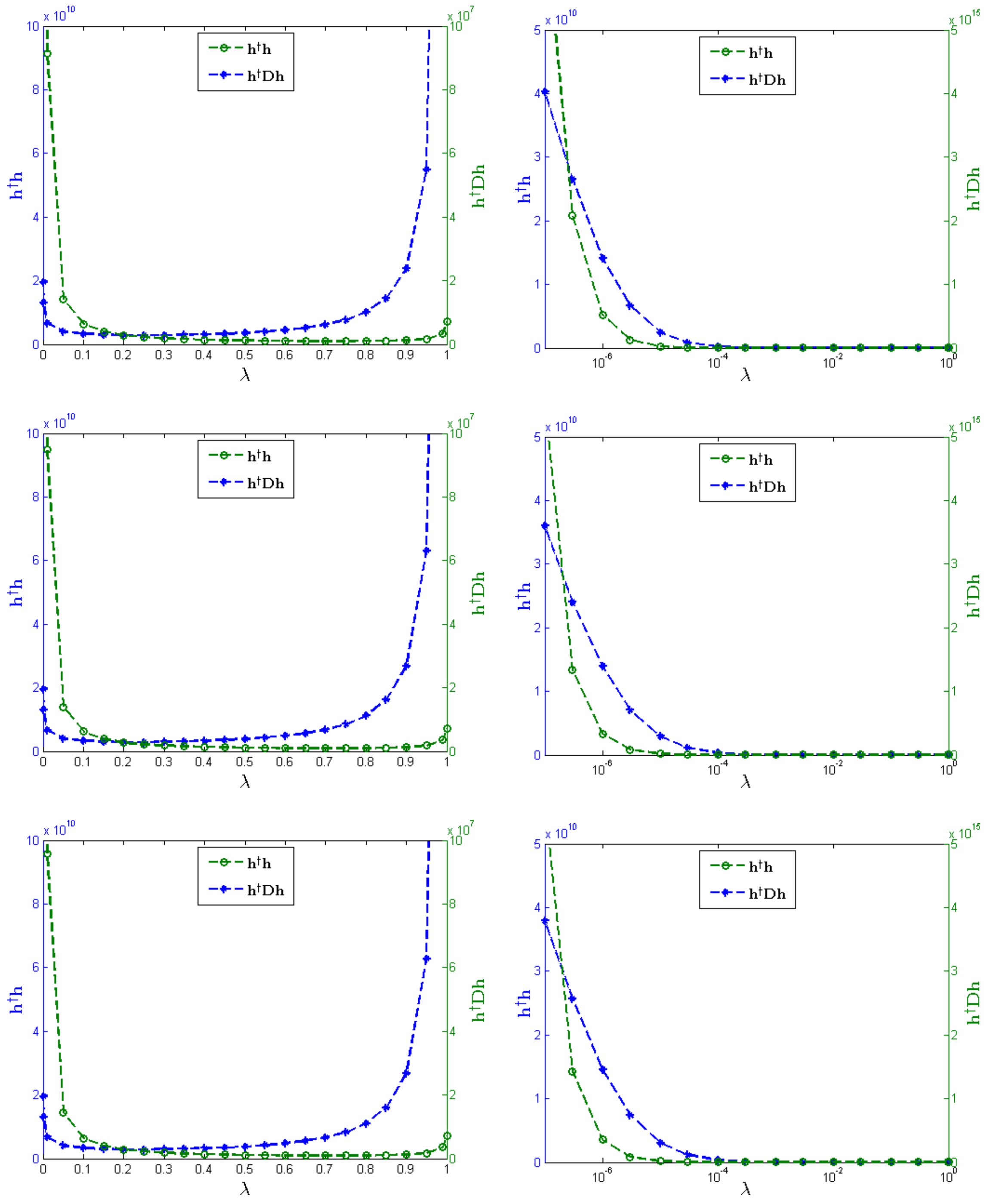


Figure 7.18: The classification criterion $\mathbf{h}^\dagger\mathbf{h}$ and localization criterion $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the QMMCF.

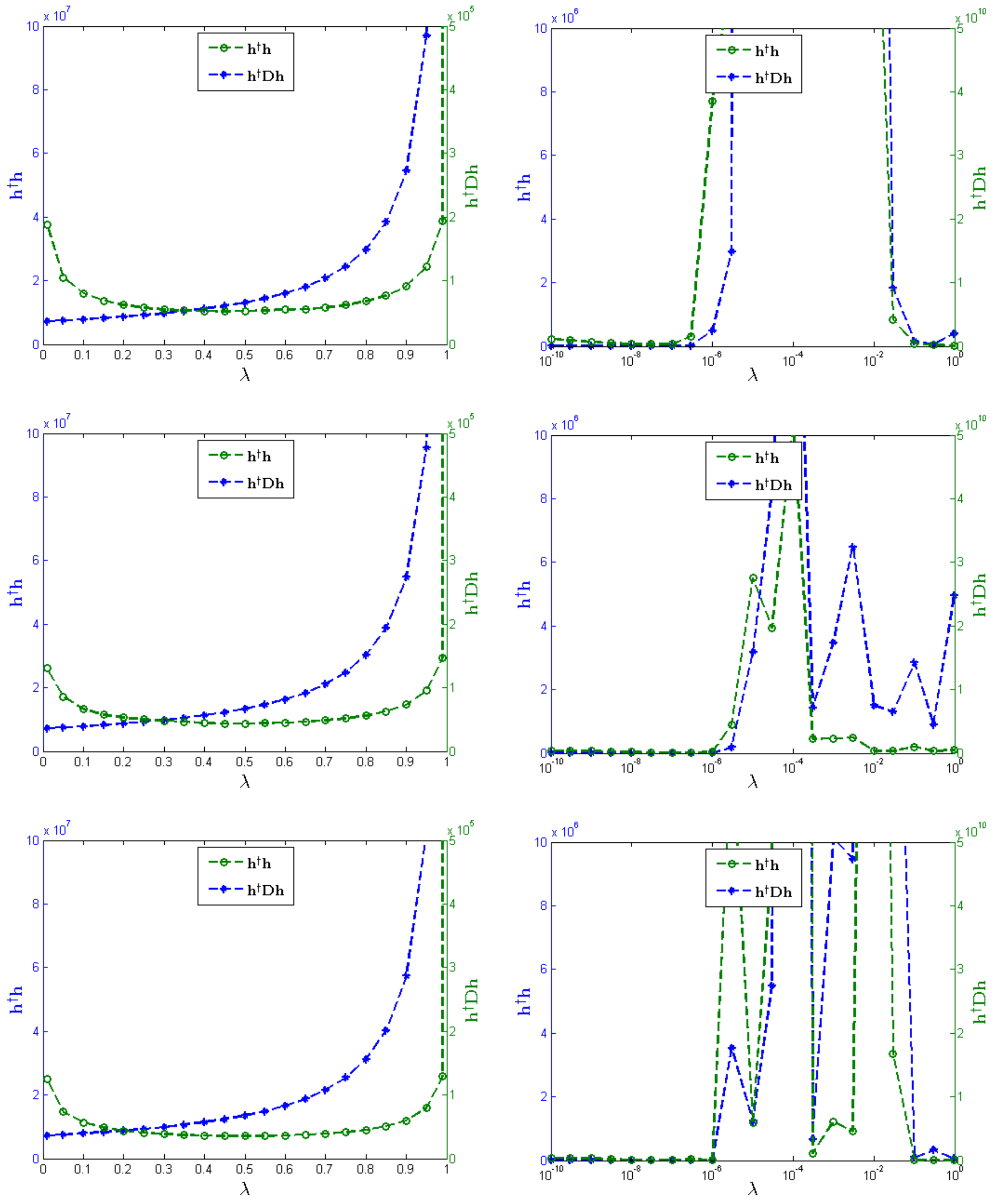


Figure 7.19: The classification criterion $\mathbf{h}^{\dagger}\mathbf{h}$ and localization criterion $\mathbf{h}^{\dagger}\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the TQCF.

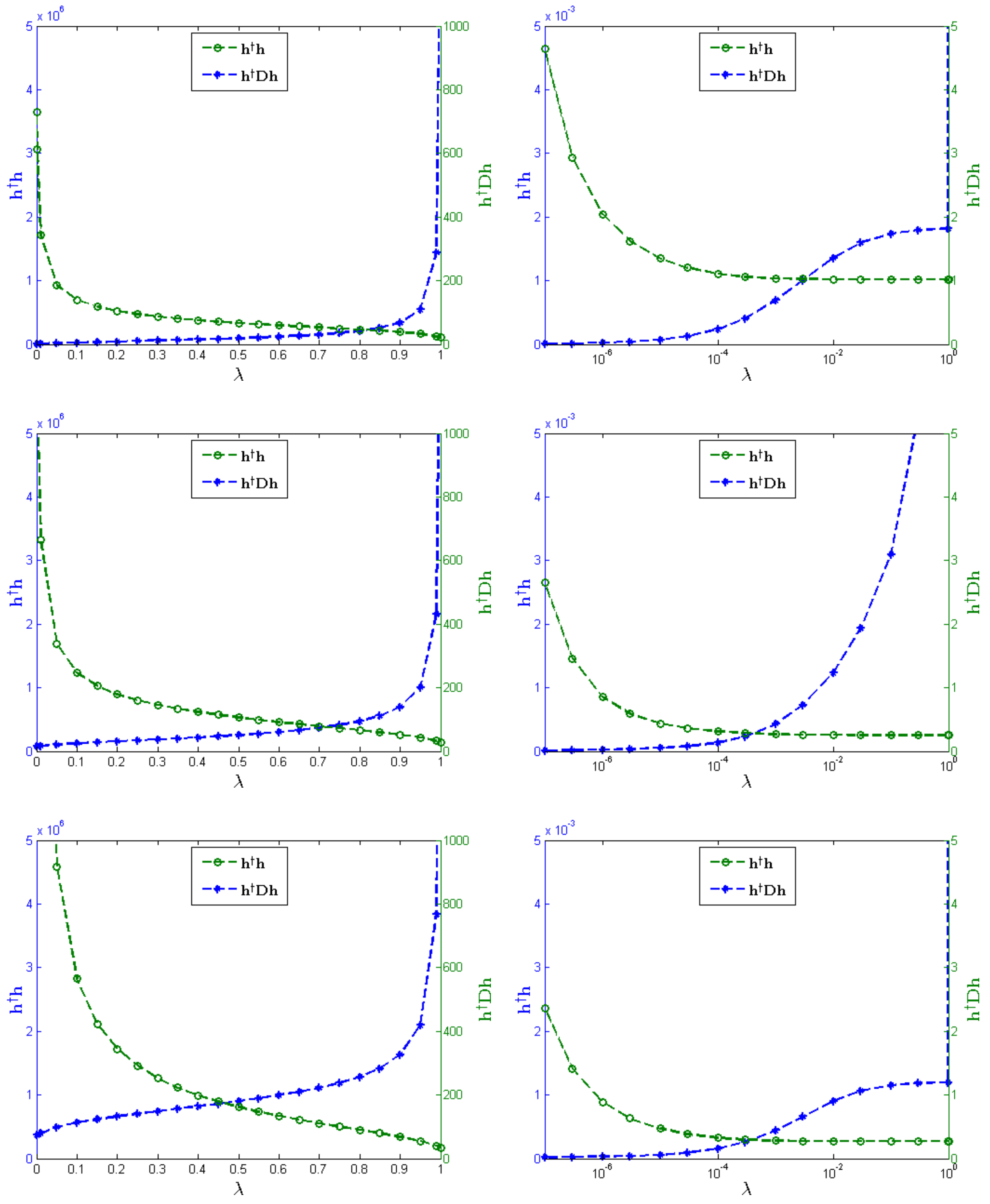


Figure 7.20: The classification criterion $\mathbf{h}^{\dagger}\mathbf{h}$ and localization criterion $\mathbf{h}^{\dagger}\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the MMCF.

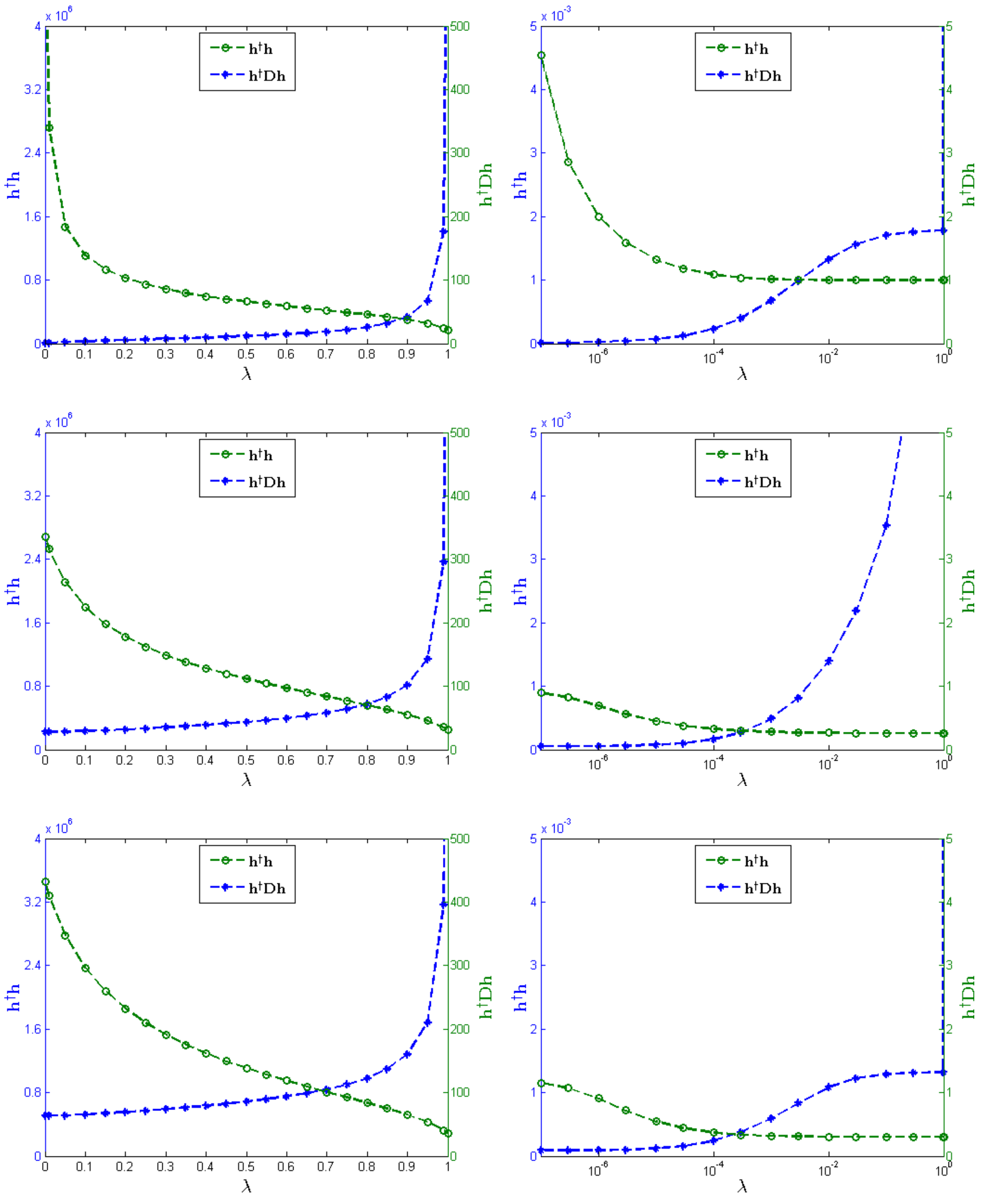


Figure 7.21: The classification criterion $\mathbf{h}^{\dagger}\mathbf{h}$ and localization criterion $\mathbf{h}^{\dagger}\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the CCF.

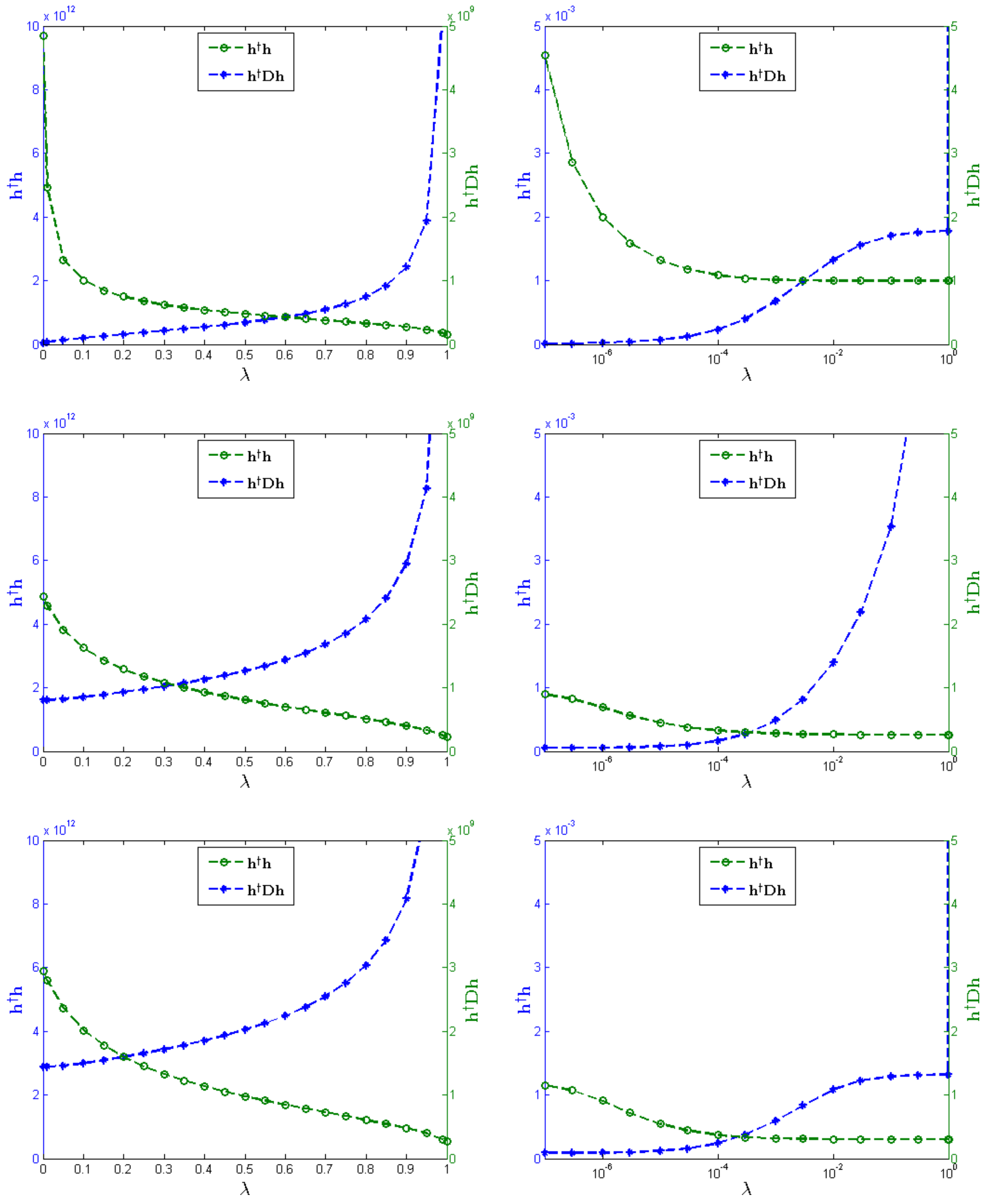


Figure 7.22: The classification criterion $\mathbf{h}^\dagger\mathbf{h}$ and localization criterion $\mathbf{h}^\dagger\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the mCCF.

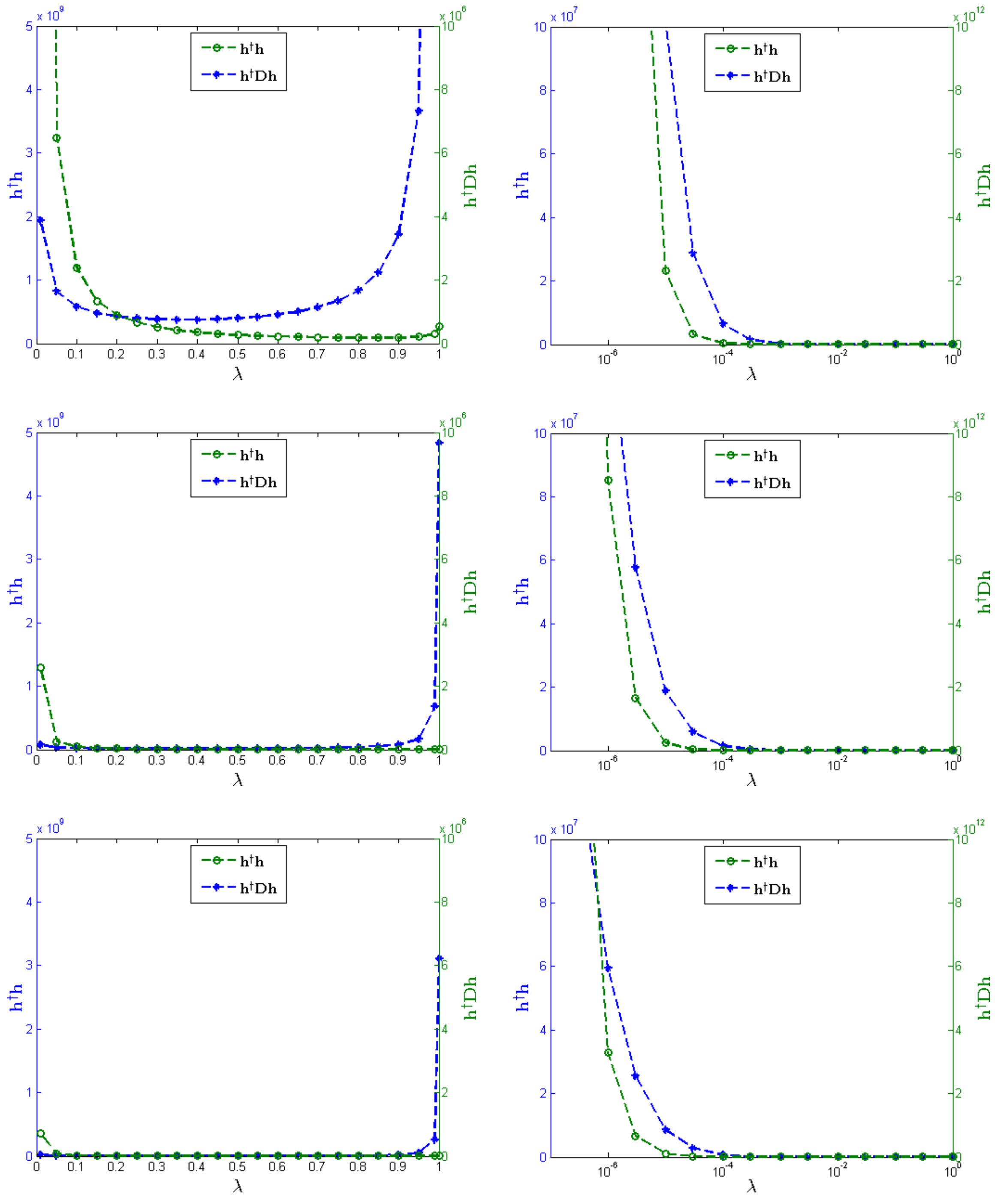


Figure 7.23: The classification criterion $\mathbf{h}^{\dagger}\mathbf{h}$ and localization criterion $\mathbf{h}^{\dagger}\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the UCF.

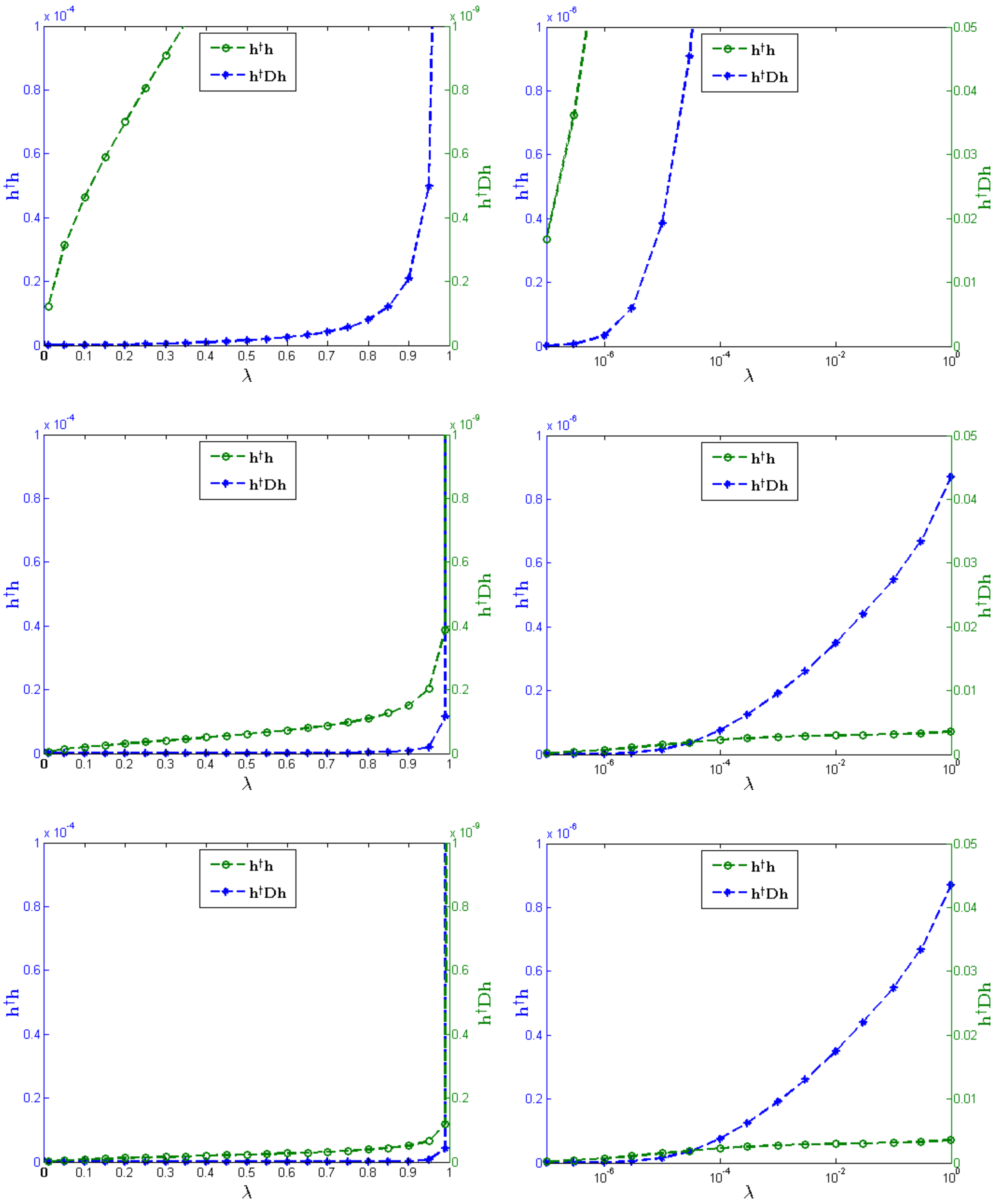


Figure 7.24: The classification criterion $\mathbf{h}^{\dagger}\mathbf{h}$ and localization criterion $\mathbf{h}^{\dagger}\mathbf{D}\mathbf{h}$ as a function of $\lambda$ using (left) gray-scaled pixel and (right) HOG features for (top) Set 1, (middle) Set 2, and (bottom) Set 3 for the EASEF.

Next, we show examples of the different templates and their responses to the same test image (containing the BRDM2 target) in Figs. 7.25 through 7.33 using the QMMCF, QSVM, TQCF, MMCF, SVM, CCF, mCCF, UCF and EASEF classifiers, respectively. From these figures we observe that our quadratic correlation filter designs QMMCF and TQCF usually produce a very sharp peak at the target's location. We also observed that QMMCF and MMCF produce a sharper peak than QSVM and SVM, respectively. As discussed earlier, this is because QMMCF and MMCF includes the localization criterion which can help improve overall recognition performance.

We conclude by testing our classifiers with test images taken at 1500 meters range. As was mentioned earlier, the *ATR Algorithm Development Image Database* contains IR videos taken at multiple ranges. Using the previous set up, we trained the QMMCF, QSVM, TQCF, MMCF, SVM, CCF, mCCF, UCF and EASEF classifiers using images taken at 1000 meters range. Given that the training frames are taken at 2/3 the distance of the testing frames, we scaled the training images by 0.68. We observed that scaling the training images by 0.68 gave a slight improvement over scaling the images by exactly 2/3. We tested the classifiers using different $\lambda$ and $\dot{\mathbf{g}}_{\sigma^2}$ values and report our best results. We used $\psi = 1$ for all classifiers because our previous results showed that varying $\psi$ does not significantly improve the results. Table 7.23 shows the classification, localization, and recognition rates using gray-scaled pixels and retraining. The confusion matrices are shown in Tables 7.24 through 7.32, and the ROC curves are shown in Figures 7.34 through 7.42 for the QMMCF, QSVM, TQCF, MMCF, SVM, CCF, mCCF, UCF and EASEF classifiers, respectively. The ROC curves of the average over the eight targets are shown in Fig. 7.43. The results show that the recognition performance is lower than our previous set up. This is because the training images and test images comes from two different videos taken at different ranges making the ATR task more challenging, but also because the resolution of the test images is much lower (the targets are less than $26 \times 46$ pixels). Many of the targets are very similar and require high resolution for accurate classification (the confusion matrices show that some filters are not even able to detect some targets). Thus, this decrease in resolution greatly decreases the classification and recognition performance of all filters. We also observe that in this set, QMMCF does not outperform MMCF. We conjecture

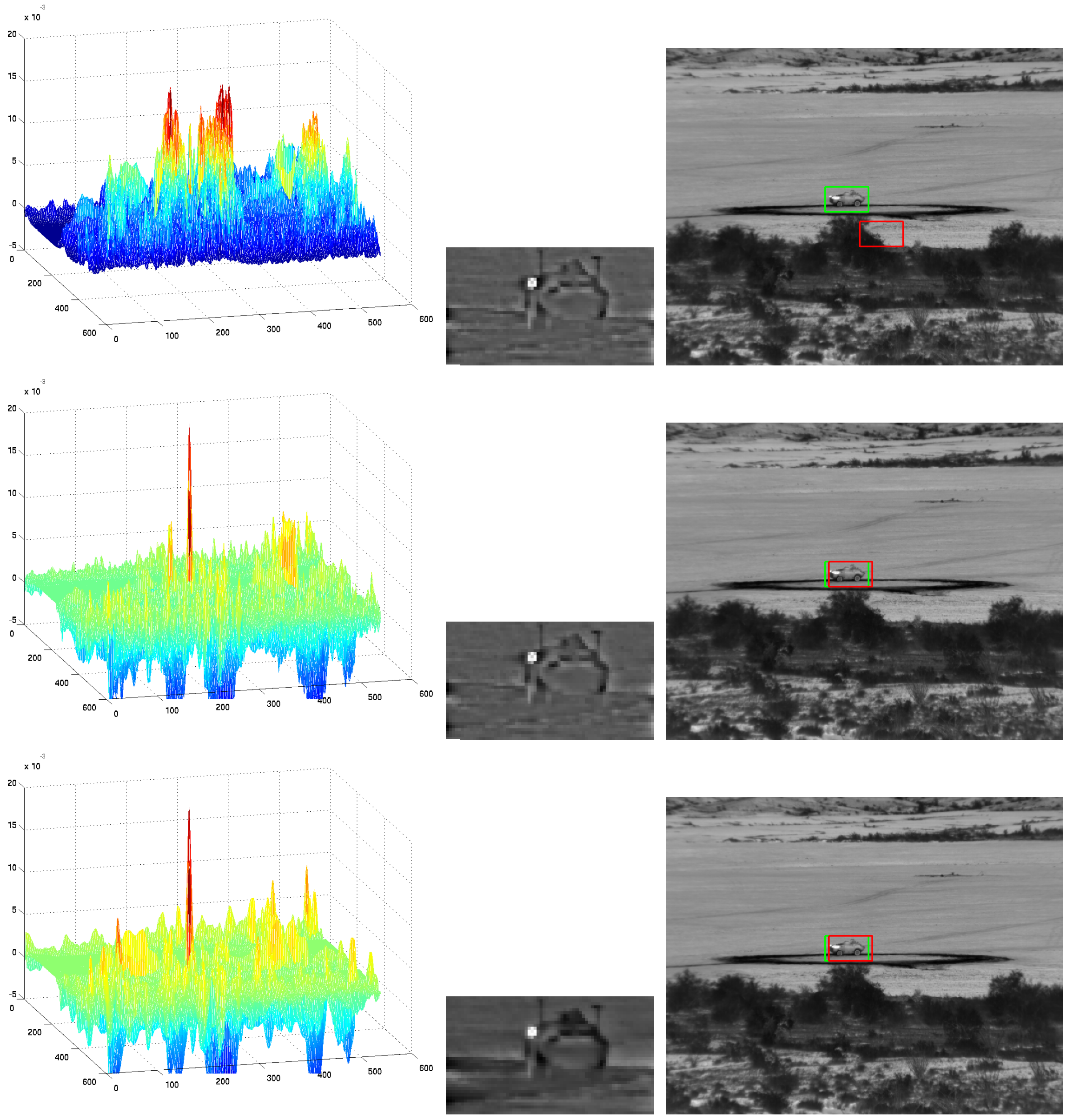


Figure 7.25: The correlation output response (left) when the QMMCF classifier (center: the training image with the largest coefficient is shown) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

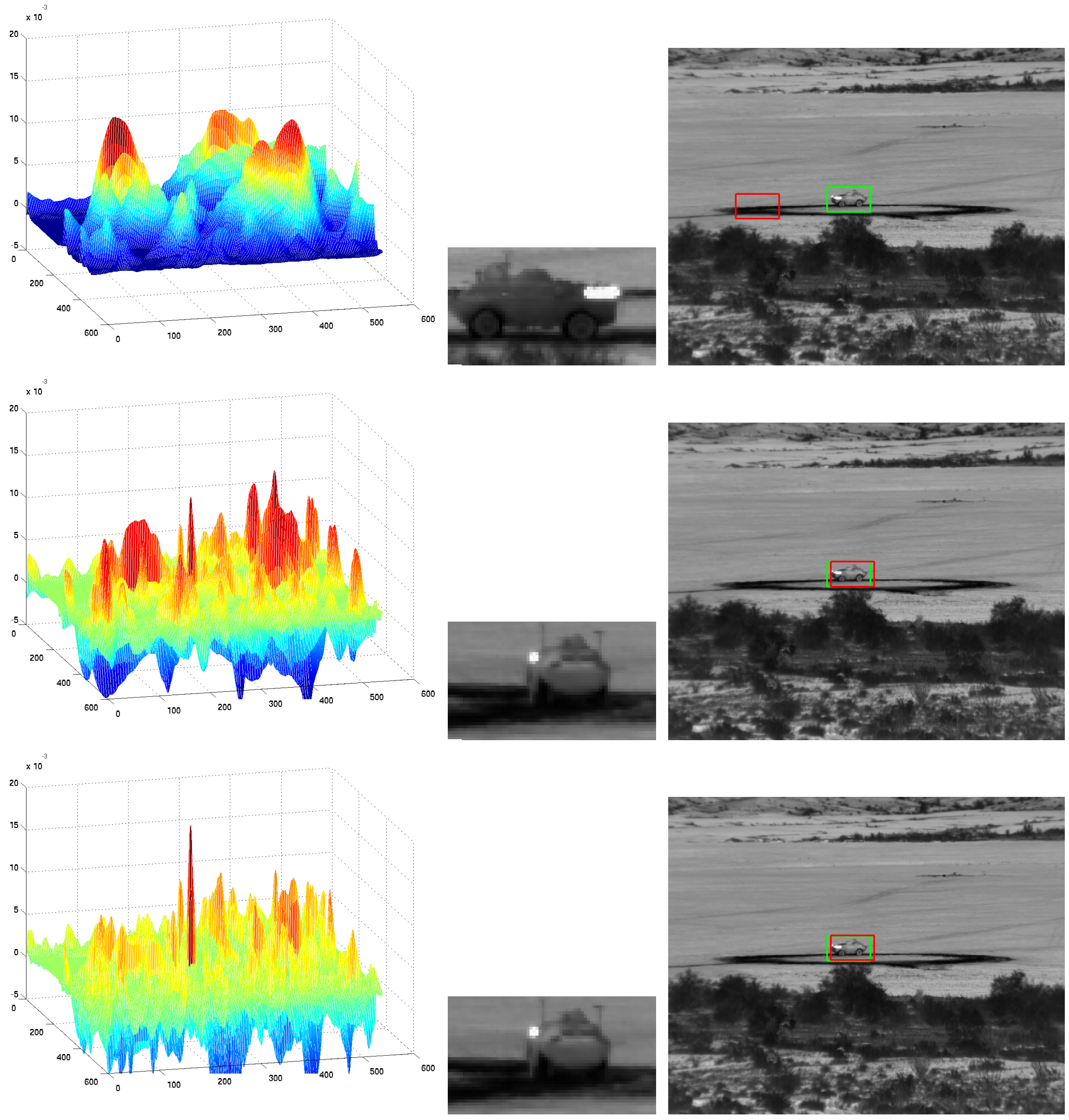


Figure 7.26: The correlation output response (left) when the QSVM classifier (center: the training image with the largest coefficient is shown) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

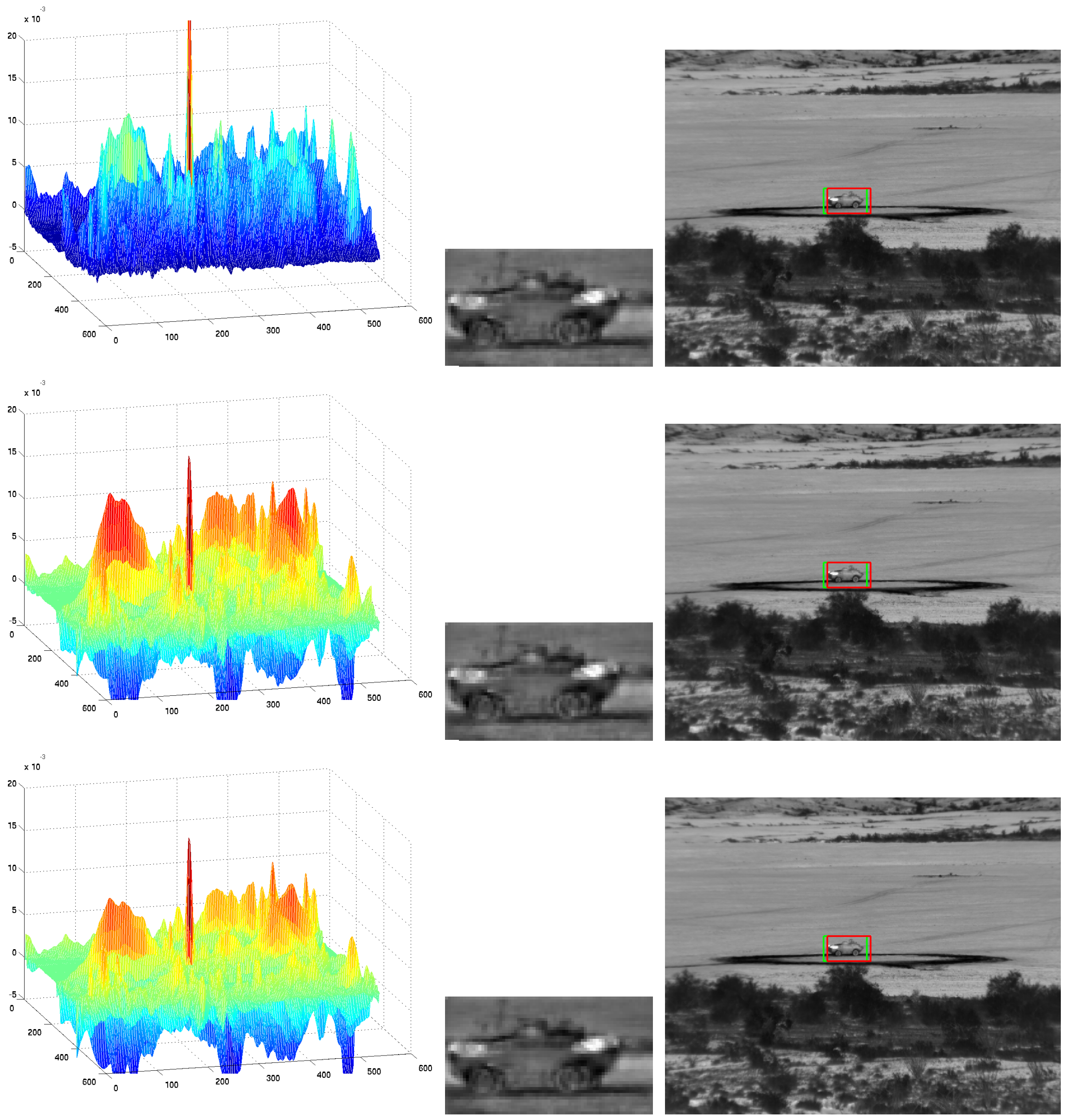


Figure 7.27: The correlation output response (left) when the TQCF classifier (center: the training image with the largest coefficient is shown) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

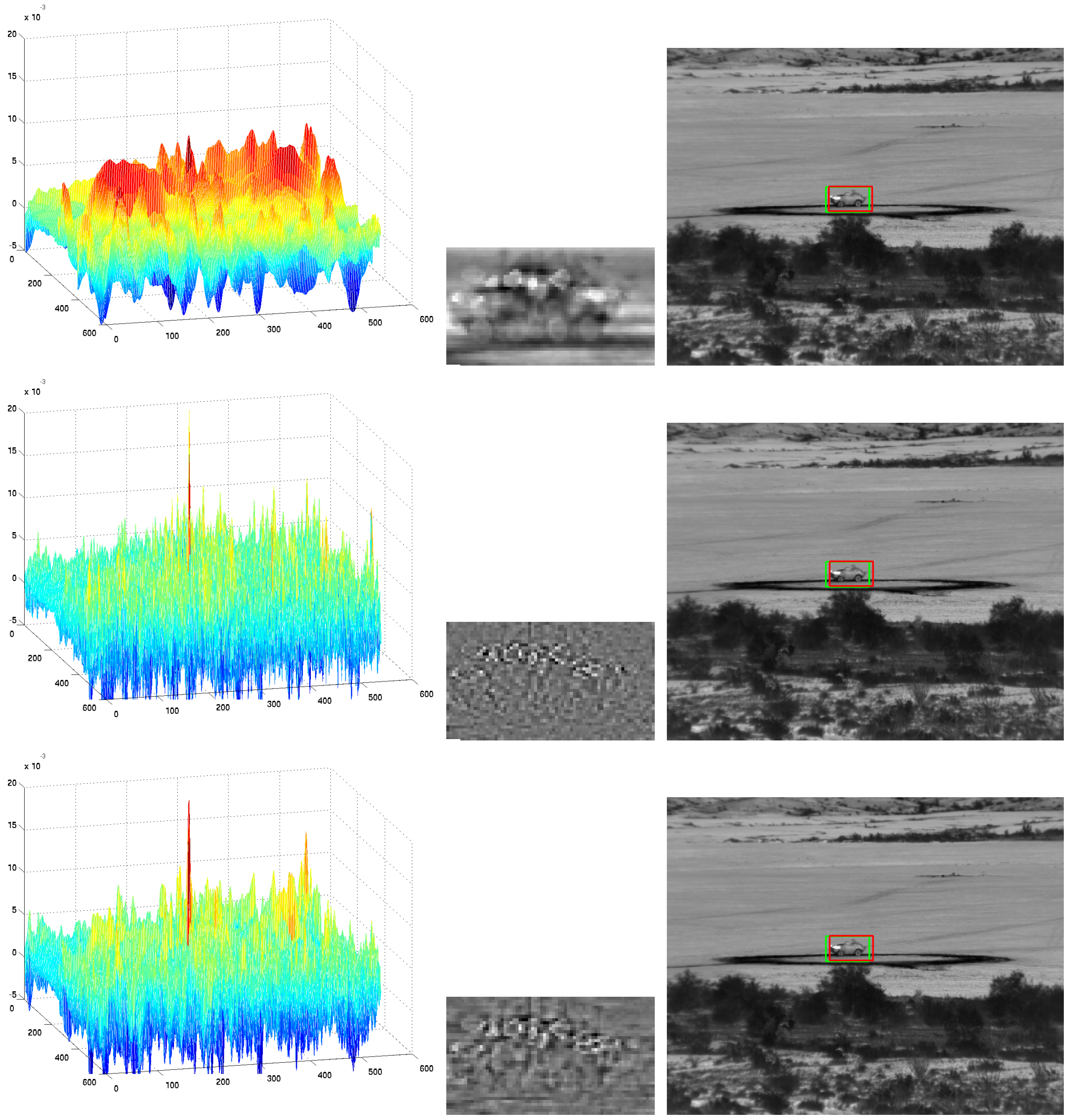


Figure 7.28: The correlation output response (left) when the MMCF classifier (center) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

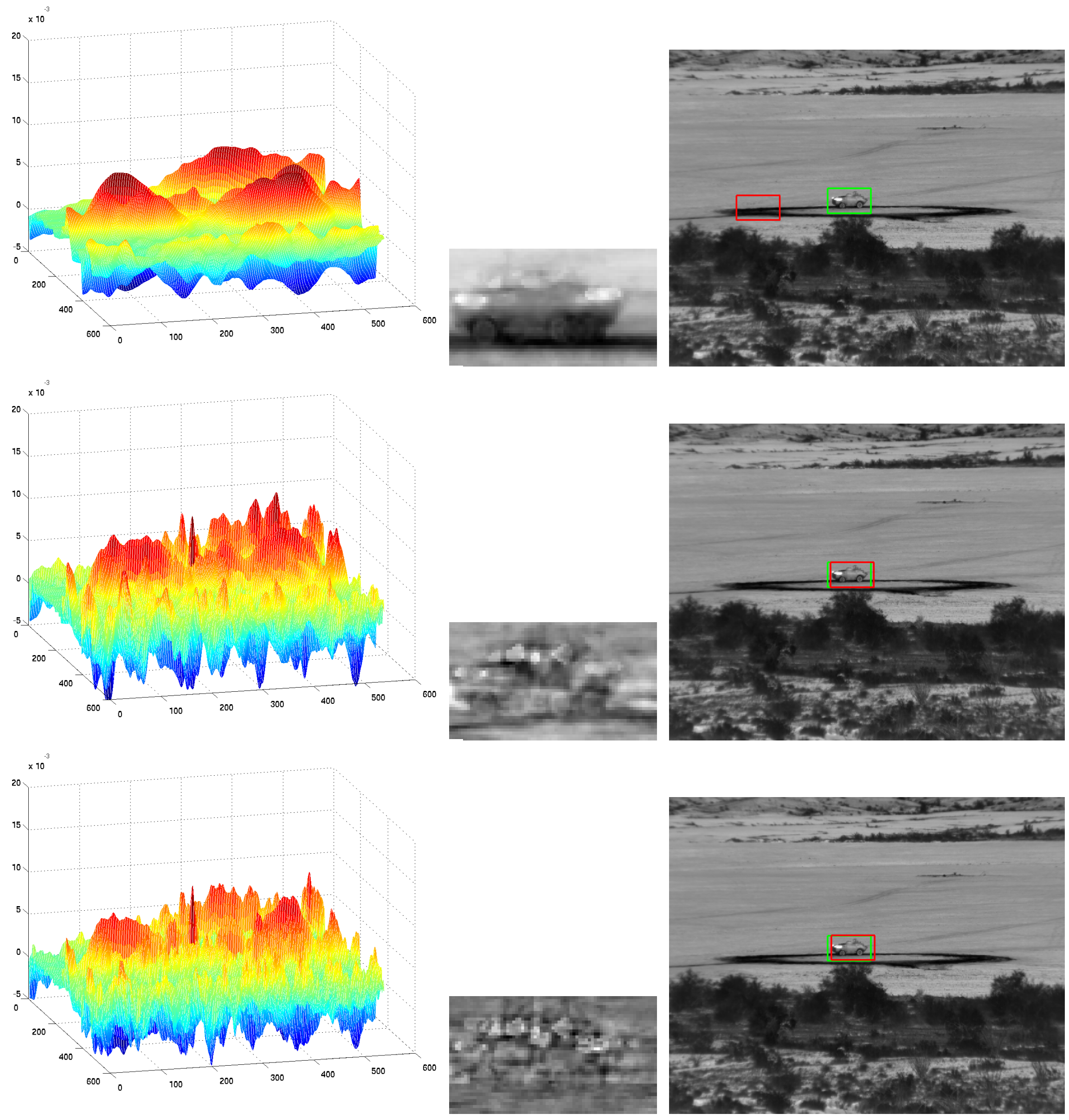


Figure 7.29: The correlation output response (left) when the SVM template (center) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

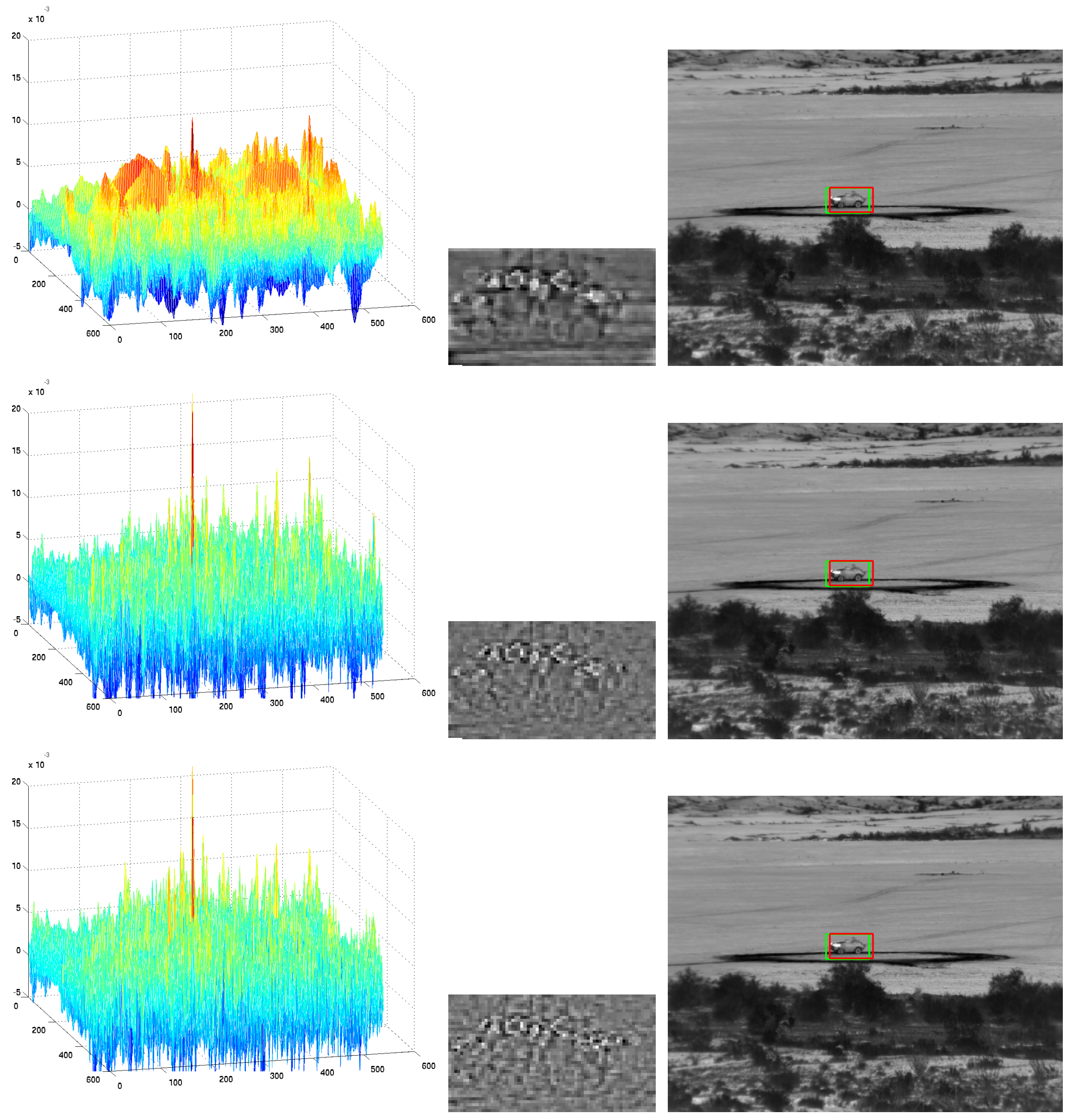


Figure 7.30: The correlation output response (left) when the CCF template (center) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

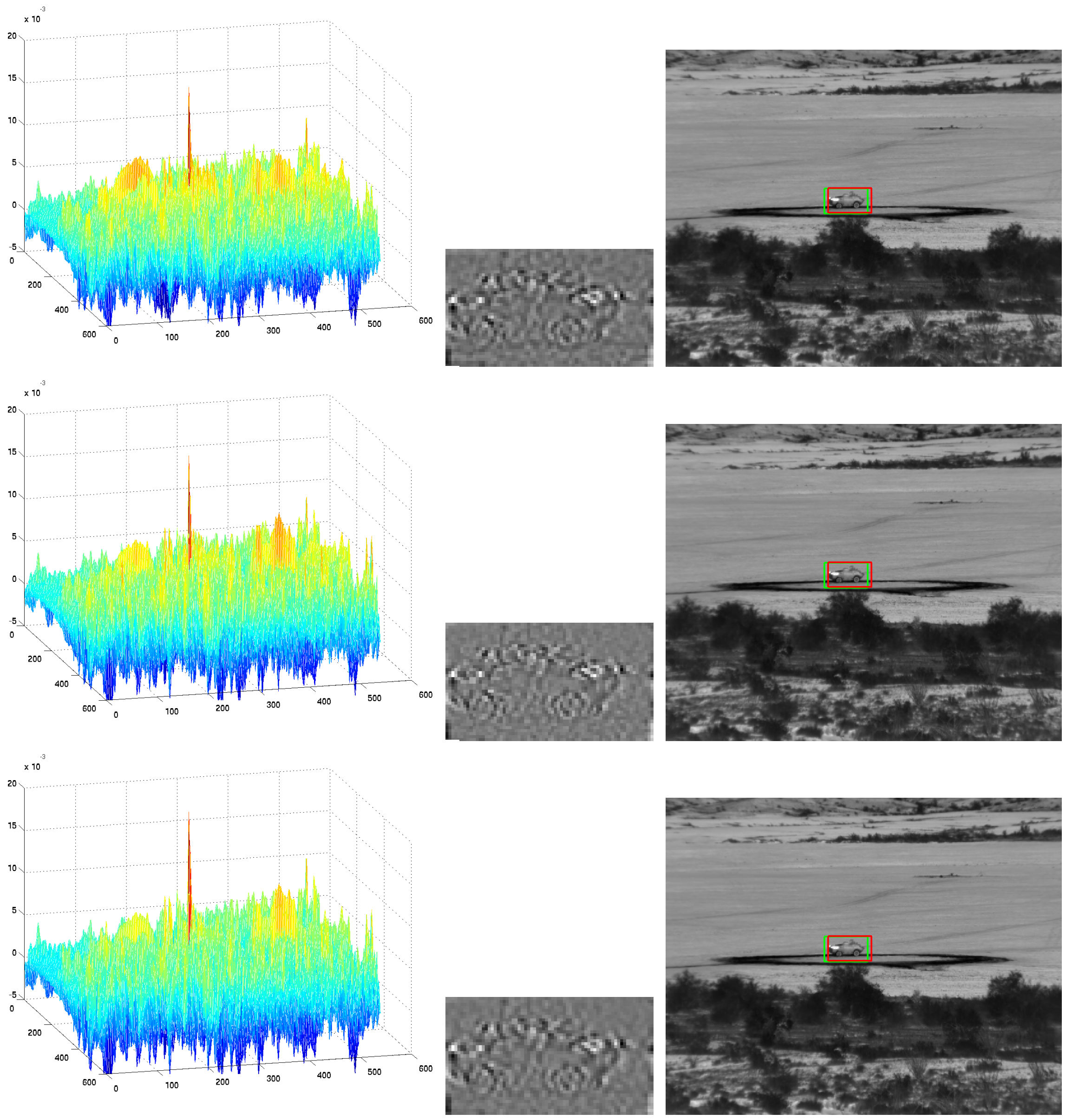


Figure 7.31: The correlation output response (left) when the mCCF template (center) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

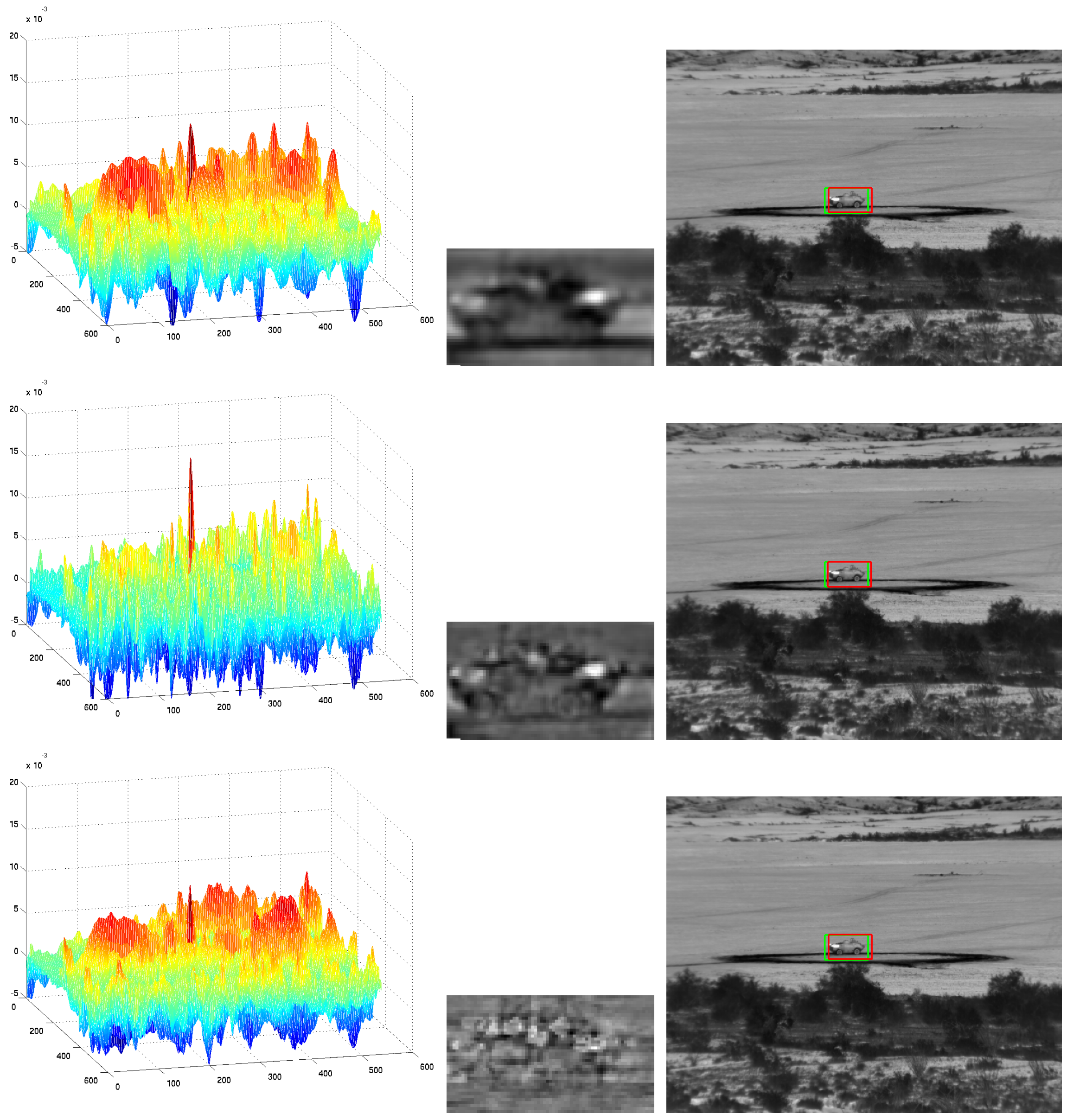


Figure 7.32: The correlation output response (left) when the UCF template (center) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

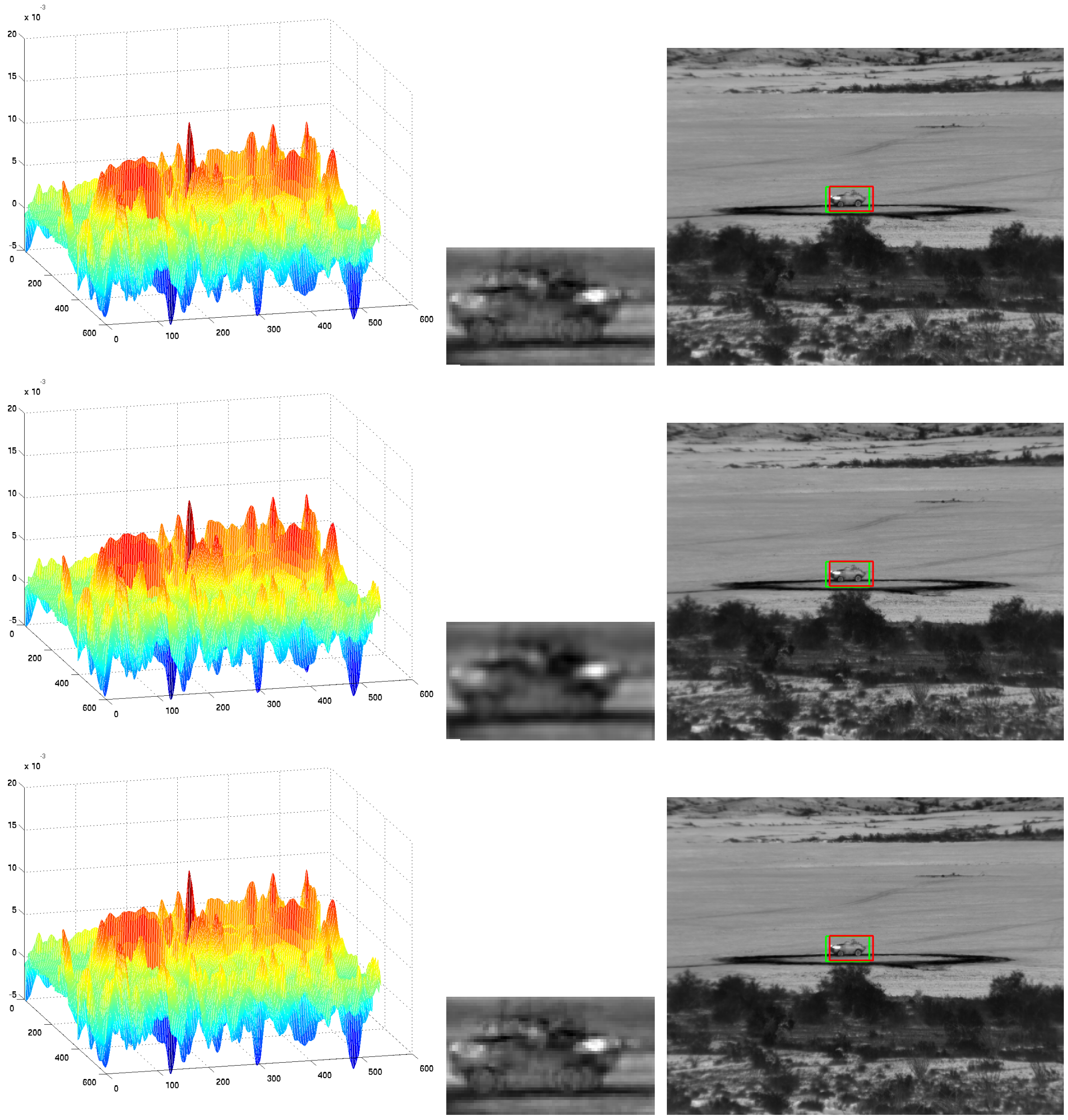


Figure 7.33: The correlation output response (left) when the EASEF template (center) is applied to a test image (right) in (top) Set 1, (middle) Set 2, and (bottom) Set 3. The test image has a green box around the ground truth target location and a red box around the maximum correlation peak value.

that this is because QMMCF was not implemented as derived because of computational limitations (see Section 5.3), and our heuristic implementation does not produces as good results, particularly in this lower resolution. However, the results show the same pattern as our previous set up where QMMCF outperforms QSVM and MMCF outperforms SVM and all the other linear filters, thus showing consistency with different ranges. In Ch. 8, we discuss ideas that may further improve performance.

Table 7.23: Filter performance (%) using gray-scaled pixels and retraining using 1500 meters range tests frames.

| | class | local | recog | $\lambda$ | $\dot{\mathbf{g}}_{\sigma^2}$ | ret |
|---|---|---|---|---|---|---|
| QMMCF | 40.3 | 68.3 | 27.7 | **0.35** | 0 | 4 |
| QSVM | 27.5 | 40.5 | 14.0 | 0 | 0 | 5 |
| TQCF | 35.7 | 59.6 | 29.6 | **0.90** | 0 | 2 |
| MMCF | 37.2 | 72.4 | 32.2 | **0.85** | **0** | 4 |
| SVM | 16.6 | 38.1 | 15.1 | 0 | 0 | 8 |
| CCF | 42.3 | 73.0 | 31.4 | **0.95** | **0** | 4 |
| mCCF | 27.3 | 62.7 | 23.9 | **0.90** | **0** | 5 |
| UCF | 27.1 | 47.6 | 24.4 | **0.85** | $\frac{1}{2}$ | 4 |
| EASEF | 17.2 | 35.4 | 15.8 | **0.70** | **1** | 3 |

Table 7.24: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the QMMCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 62 | 9 | 84 | 30 | 0 | 0 | 2 | 13 |
| 2 | 1 | 35 | 63 | 4 | 0 | 15 | 1 | 81 |
| 3 | 0 | 0 | 200 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 136 | 47 | 0 | 0 | 0 | 17 |
| 5 | 41 | 10 | 44 | 49 | 23 | 0 | 0 | 33 |
| 6 | 2 | 6 | 41 | 32 | 0 | 68 | 9 | 42 |
| 7 | 5 | 4 | 68 | 10 | 0 | 3 | 67 | 43 |
| 8 | 2 | 0 | 34 | 21 | 0 | 0 | 0 | 143 |

Table 7.25: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the QSVM.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 184 | 1 | 0 | 6 | 0 | 0 | 0 | 9 |
| 2 | 107 | 0 | 0 | 0 | 0 | 0 | 0 | 93 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200 |
| 5 | 7 | 13 | 9 | 112 | 56 | 0 | 0 | 3 |
| 6 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 189 |
| 7 | 1 | 7 | 0 | 1 | 4 | 0 | 7 | 180 |
| 8 | 0 | 0 | 0 | 17 | 1 | 0 | 0 | 182 |

Table 7.26: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the TQCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 79 | 0 | 80 | 1 | 0 | 10 | 30 | 0 |
| 2 | 3 | 67 | 104 | 1 | 0 | 18 | 7 | 0 |
| 3 | 197 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 80 | 0 | 0 | 96 | 0 | 11 | 13 | 0 |
| 5 | 14 | 9 | 20 | 1 | 0 | 11 | 145 | 0 |
| 6 | 22 | 0 | 7 | 5 | 0 | 124 | 42 | 0 |
| 7 | 13 | 0 | 0 | 13 | 0 | 33 | 140 | 1 |
| 8 | 8 | 0 | 49 | 0 | 0 | 38 | 43 | 62 |

Table 7.27: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the MMCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 69 | 57 | 0 | 42 | 25 | 3 | 3 | 1 |
| 2 | 14 | 112 | 0 | 29 | 41 | 5 | 2 | 0 |
| 3 | 0 | 8 | 57 | 51 | 10 | 0 | 0 | 74 |
| 4 | 0 | 6 | 15 | 113 | 1 | 20 | 6 | 39 |
| 5 | 5 | 43 | 4 | 77 | 51 | 13 | 3 | 4 |
| 6 | 4 | 18 | 5 | 49 | 13 | 105 | 2 | 4 |
| 7 | 16 | 23 | 9 | 50 | 6 | 40 | 21 | 35 |
| 8 | 1 | 23 | 12 | 77 | 10 | 9 | 0 | 68 |

Table 7.28: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the SVM.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | 0 | 1 | 0 | 191 | 4 | 1 |
| 2 | 1 | 2 | 0 | 0 | 0 | 197 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 200 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 200 | 0 | 0 |
| 5 | 0 | 14 | 0 | 58 | 0 | 100 | 28 | 0 |
| 6 | 0 | 0 | 0 | 16 | 0 | 176 | 0 | 8 |
| 7 | 0 | 0 | 0 | 0 | 0 | 140 | 40 | 20 |
| 8 | 0 | 2 | 0 | 18 | 0 | 124 | 9 | 47 |

Table 7.29: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the CCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 55 | 52 | 1 | 33 | 40 | 0 | 9 | 10 |
| 2 | 25 | 101 | 6 | 22 | 41 | 1 | 0 | 4 |
| 3 | 5 | 6 | 159 | 14 | 0 | 10 | 0 | 6 |
| 4 | 6 | 3 | 45 | 99 | 5 | 18 | 8 | 16 |
| 5 | 10 | 42 | 3 | 63 | 47 | 12 | 7 | 16 |
| 6 | 6 | 7 | 16 | 23 | 7 | 82 | 6 | 53 |
| 7 | 4 | 13 | 5 | 37 | 10 | 17 | 52 | 62 |
| 8 | 4 | 20 | 35 | 40 | 1 | 7 | 12 | 81 |

Table 7.30: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the mCCF.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 70 | 30 | 1 | 52 | 14 | 5 | 1 | 27 |
| 2 | 51 | 62 | 3 | 22 | 45 | 2 | 0 | 15 |
| 3 | 55 | 4 | 13 | 12 | 55 | 12 | 39 | 10 |
| 4 | 19 | 18 | 27 | 60 | 38 | 12 | 18 | 8 |
| 5 | 39 | 7 | 15 | 11 | 98 | 3 | 1 | 26 |
| 6 | 12 | 65 | 1 | 6 | 12 | 58 | 8 | 38 |
| 7 | 33 | 49 | 11 | 29 | 27 | 0 | 41 | 10 |
| 8 | 30 | 30 | 1 | 57 | 13 | 25 | 9 | 35 |

Table 7.31: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the UCF.

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 124 | 11 | 0 | 39 | 0 | 18 | 0 | 8 |
| 2 | 27 | 79 | 0 | 66 | 0 | 19 | 0 | 9 |
| 3 | 38 | 0 | 8 | 136 | 0 | 1 | 0 | 16 |
| 4 | 22 | 3 | 5 | 111 | 0 | 3 | 0 | 56 |
| 5 | 71 | 29 | 0 | 56 | 0 | 0 | 0 | 39 |
| 6 | 26 | 0 | 0 | 68 | 0 | 11 | 0 | 95 |
| 7 | 30 | 1 | 17 | 14 | 0 | 0 | 1 | 137 |
| 8 | 60 | 0 | 0 | 1 | 0 | 45 | 0 | 94 |

Table 7.32: Confusion matrices using gray-scaled pixels and retraining using 1500 meters range tests frames for the EASEF.

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 72 | 31 | 0 | 0 | 0 | 0 | 0 | 97 |
| 2 | 1 | 56 | 0 | 14 | 0 | 0 | 0 | 129 |
| 3 | 176 | 0 | 4 | 2 | 0 | 0 | 0 | 18 |
| 4 | 13 | 5 | 15 | 9 | 0 | 0 | 0 | 158 |
| 5 | 52 | 61 | 0 | 56 | 13 | 0 | 0 | 20 |
| 6 | 51 | 7 | 52 | 17 | 15 | 0 | 0 | 58 |
| 7 | 124 | 1 | 0 | 0 | 0 | 0 | 0 | 75 |
| 8 | 30 | 4 | 9 | 24 | 11 | 0 | 0 | 122 |

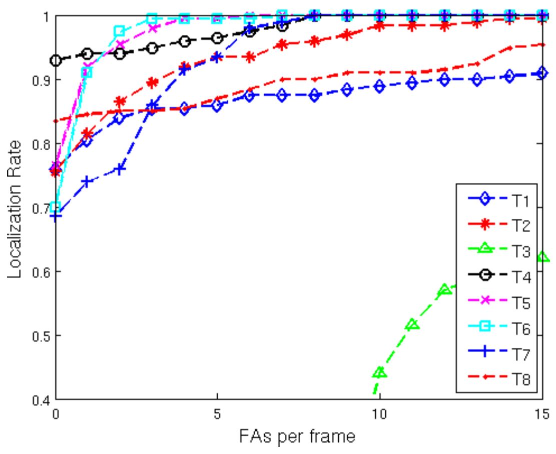


Figure 7.34: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the QMMCF.

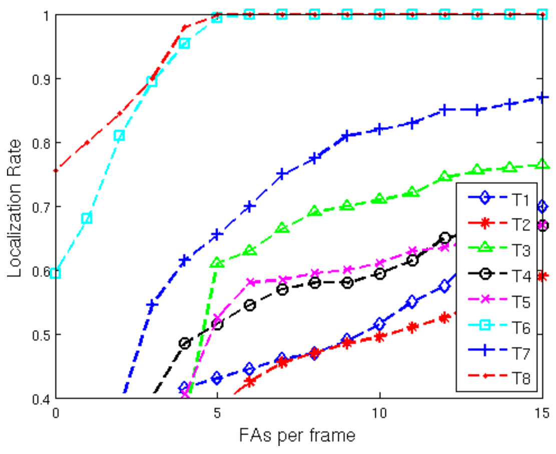


Figure 7.35: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the QSVM.

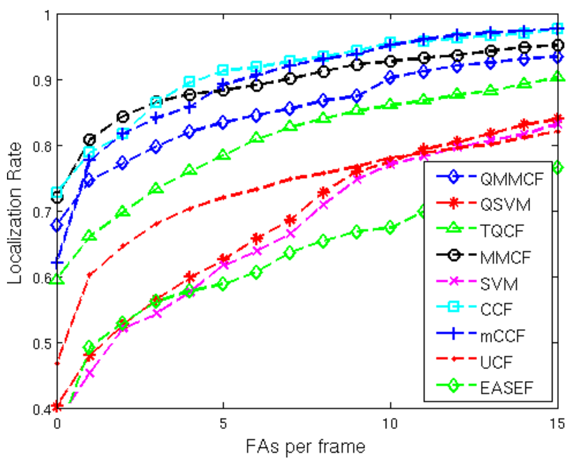


Figure 7.43: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames.

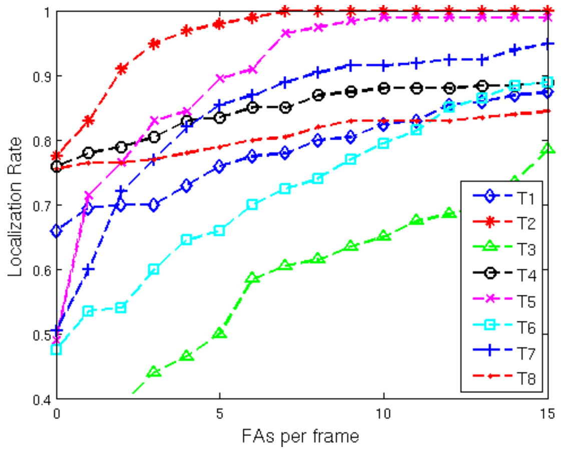


Figure 7.36: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the TQCF.

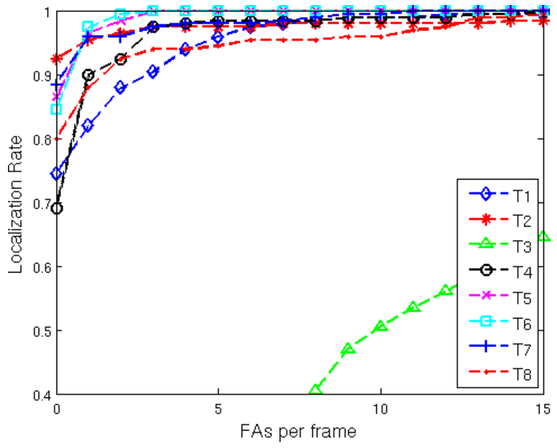


Figure 7.37: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the MMCF.

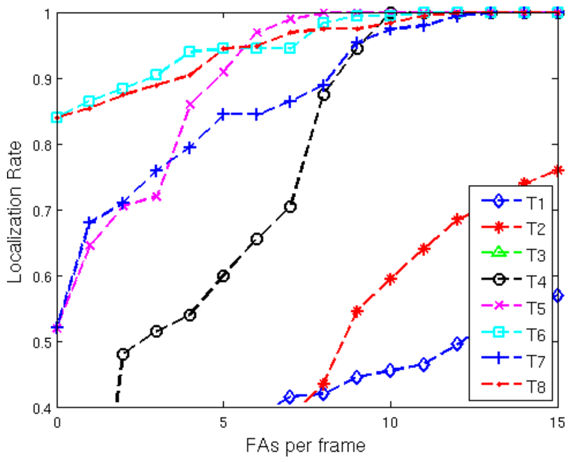


Figure 7.38: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the SVM.

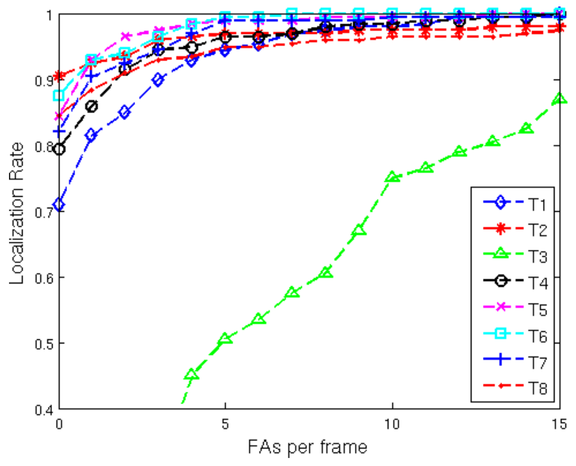


Figure 7.39: Localization rate as a function of the number of FAs per frame using gray-scaled pixel for and retraining using 1500 meters range tests frames for the CCF.

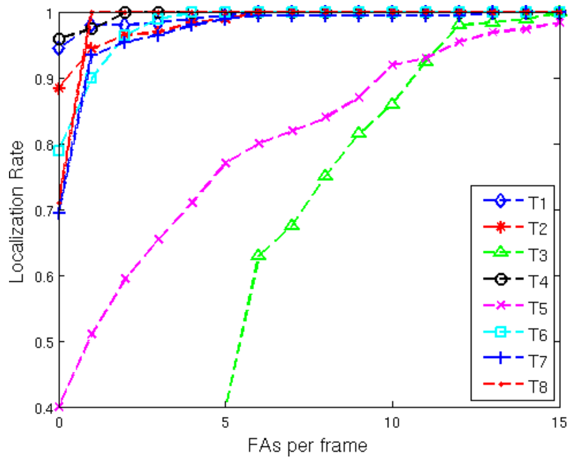


Figure 7.40: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the mCCF.

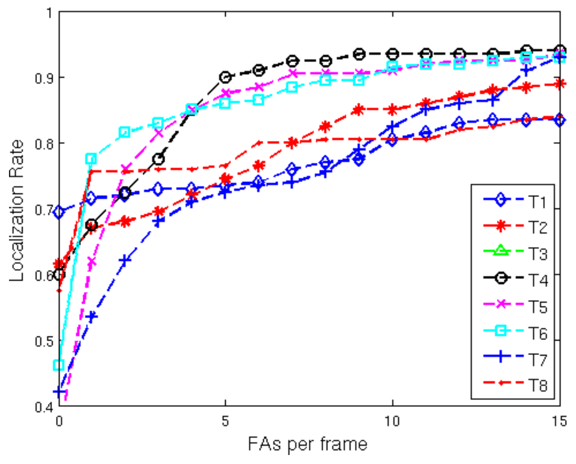


Figure 7.41: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the UCF.

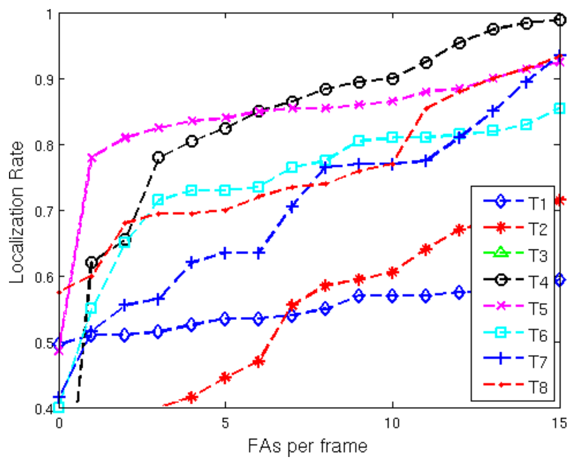


Figure 7.42: Localization rate as a function of the number of FAs per frame using gray-scaled pixel and retraining using 1500 meters range tests frames for the EASEF.

## 7.6 Summary

In this chapter we explained in detail the dataset used in our experiments. We discussed the pre- and post-processing stages, explained our method for selecting parameters, gave more details on the results presented in previous chapters, and presented our conclusions. Our experiments show that in our dataset, our linear and quadratic designs outperformed state-of-the-art linear and quadratic CFs and SVMs, respectively, using gray-scaled pixel values and HOG features.

CHAPTER 8

# CONCLUSIONS

We have presented in this thesis the linear and quadratic Maximum Margin Correlation Filter (MMCF) which combines the design principles of SVMs and CFs. To this end, we reviewed CFs, and introduced the UCF and CCF that take into account the criteria used in state-of-the-art linear CFs. We introduced the QMMCF and we showed the relation between QCFs and QSVMs. We adapted our MMCF filter design to include vector features. We tested our designs with gray-scaled pixels and with HOG features. In all of our experiments our linear designs outperformed all other linear CFs and linear SVM, and our quadratic designs outperformed other quadratic CFs and QSVMs. In this chapter, we outline our contributions, conclusions, and future work.

## 8.1 CONTRIBUTIONS

The main contributions of this thesis are as follows:

- Improved existing CFs. We presented three generalized linear CFs that encompass all the state-of-the-art linear CFs, the Unconstrained Correlation Filter (UCF), the Constrained Correlation Filter (CCF), and the modified CCF (mCCF). We presented a method to enhanced the performance of quadratic CFs both in computational speed and performance, and showed experimental results. We presented a design that improves recognition performance when using CFs in a sequence of images, i.e., in video, by using the correlation output with a tracker. We provided experimental results showing our algorithms are more efficient and yield improved performance on our dataset.

- Designed a new algorithm that combines the capabilities of SVMs and our generalized CFs to yield the Maximum Margin Correlation Filter (MMCF). We investigated the relationship between localization and classification, and show that the criteria used to achieved localization and generalization are conflicting, i.e., one is optimized at the expense of the other, and we showed how to achieve an optimal tradeoff between these criteria. We showed that MMCF is an extension to both SVM and CFs by deriving MMCF first starting with an SVM and then with a CF, thus we bridged these two important research fields. We provided experimental results showing that MMCF outperforms both linear SVMs and linear CFs on our dataset.

- Designed the Quadratic MMCF (QMMCF) by extending the maximum margin principles to quadratic CFs (QCFs). QMMCF is better able to exploit the second-order statistics of the data resulting in superior performance. This improved performance comes at the cost of added computation during the testing. We showed the relationship between quadratic correlation filters and Kernel-SVMs. We provided experimental results showing that QMMCF outperforms quadratic CFs and quadratic SVMs (also known as second order Kernel SVMs) on our dataset.

- Extended the capabilities of CFs and therefore of MMCF to include vector features. CFs usually used scalar features, i.e., gray-scaled pixels. We adapted our algorithms to include non-scalar features, e.g., features such as Histogram of Oriented Gradients (HOG) which have recently gained much popularity. We provided experimental results showing that MMCF and QMMCF maintain superior performance over other state-of-the-art linear and quadratic algorithms, respectively, when using HOG features on our dataset.

## 8.2 CONCLUSIONS

From our experimental results, we draw the following conclusions:

- QMMCF is the recommended filter to use when ample computational resources are available for testing. Our experiments showed that QMMCF usually outperforms all the other filters.

- TQCF is the recommended filter to use when computational resources prevents the use of QMMCF *and* when gray-scaled pixels are used as features.

- UCF is the recommended filter to use when only true-class images are available. However, obtaining false-class images is trivial (e.g., through retraining or the Internet), and therefore this is not a advantage of UCF. UCF is also the recommended filter when online training or adaptive training is required and computational resources are limited.

- mCCF is the recommended filter to use when computational resources are limited *and* there are only a few training images with no outliers. These are the conditions when CCF performs best, and since our experiments show that mCCF usually outperforms CCF, mCCF is the chosen for these scenarios.

- MMCF is the recommended filter to use when computational resources are limited *and* there are many training images with possible outliers. MMCF is designed to handle outliers and ignores correctly classified data that is far from the decision boundary.

- The ASM criterion does not significantly improve the results and can be ignored. In our experiments, setting $\sigma = 1$ (i.e., ignoring the ASM criterion) usually gave the best results (on a few filters setting $\sigma = 0.99$ or $\sigma = 0.999$ gave an insignificant improvement of less than $1\%$). The MACH filter does benefit from varying $\sigma$, but in our experiments UCF with $\sigma = 1$ always outperformed the MACH filter, and there is no computational advantage or any other reason to use the MACH filter over UCF.

- A tradeoff between the MSE criterion (in particular the ACE criterion which is the MSE criterion when $\mathbf{g} = \mathbf{0}$) and ONV criterion significantly improves performances over just ignoring one of the two criterion.

- Retraining (cross-correlating the template with training frames, adding the false positives as false-class training images, and retraining the template) significantly improves performance most of the time.

## 8.3 Future Work

In this section several methods to further improve performance are listed:

- Felzenswalb et al., [24] proposed a parts-based model Latent SVM which presents a few ideas that can also be used in this work. In their work the training images from the same class are autonomously clustered into different groups, and a classifier is designed for each group. Further, each training image is broken into six rectangular parts covering high energy regions, and a classifier is designed for each part. In our work we only designed one filter per class. We expect improved performance by designing multiple filters per class in a similar way to Felzenswalb et al.

- In our experiments, we did not use a tracker because we wanted to compute the performance of the unaided CFs. Using a simple tracker has been shown to improve the overall performance over using CFs alone [57]. We proposed combining our proposed designs with a tracker to improve recognition performance.

- The vehicles used in our experiments do not change much in scale, i.e., the distance from the vehicles to the camera does not significantly change. When scale is an issue, it is popular to used a pyramid approach, i.e., train a filter for different scales. However, there are infinite in-between scales and designing a filter for every possible scale is thus impossible. In this scenario, we proposed using the MACE-Mellin Radial Harmonic (MACE-MRH) filter [31].

This filter offers a controlled scaled response for one training image. The smaller the scale, the better the response. We propose that instead of using one training image, we use a filter as the input to the MACE-MRH. Thus, the output is filtered with some controlled scale response.

- In this thesis, we derived the QMMCF but due to computational limitations we could not compute matrix $\mathbf{T}$ and had to implement a modified version. We propose finding a method to transform the images by the inverse of $\mathbf{T}$ without actually computing matrix $\mathbf{T}$. This would implement QMMCF as it was derived, which may improve performance.

- In this work we focus our attention on the MSE, ASM, and ONV criteria. We proposed comparing other methods discussed in Chapter 2 that have been used in designing CFs such as minimizing the variance of the correlation peaks, maximizing the average correlation peak intensity, and using the largest(s) eigenvalue(s) of the $\mathbf{D}$ matrix (the average power spectral density of the training images).

- Linear CFs have been used for action recognition [5, 63] sometimes using Spatio-temporal Regularity Flow (SPREF) [2] features. Each pixel is replaced by a 3-D feature vector that represents the direction along which the intensity changes the least. In addition, the Clifford Fourier transform [23] is used in order to compute Fourier transforms of the videos with vector features. We proposed adapting our MMCF and QMMCF designs in a similar way to use them for action recognition.

APPENDIX A

# MINIMIZING A QUADRATIC SUBJECT TO LINEAR CONSTRAINTS

The goal is to minimize a quadratic subject so some linear constraints, i.e.,

$$\begin{aligned} \min_{\mathbf{h}} \quad & \mathbf{h}^\dagger \mathbf{T} \mathbf{h} \\ s.t. \quad & \mathbf{X}^\dagger \mathbf{h} = \mathbf{u}, \end{aligned} \tag{A.1}$$

where $\mathbf{T}$ is assumed to be a Hermitian (symmetric if $\mathbf{T}$ is real) matrix. We use Lagrange multipliers, take the gradient, set it equal to zero and solve for $\mathbf{h}$, i.e.,

$$\mathcal{L}(\mathbf{h}, \Gamma) = \mathbf{h}^\dagger \mathbf{T} \mathbf{h} - 2\Gamma^\dagger (\mathbf{X}^\dagger \mathbf{h} - \mathbf{u}). \tag{A.2}$$

Taking the gradient gives,

$$\frac{d\mathcal{L}(\mathbf{h}, \Gamma)}{d\mathbf{h}} = 2\mathbf{T}\mathbf{h} - 2\mathbf{X}\Gamma = 0, \tag{A.3}$$

and solving for $\mathbf{h}$ gives,

$$\mathbf{h} = \mathbf{T}^{-1}\mathbf{X}\Gamma. \tag{A.4}$$

Substituting $\mathbf{h}$ into the original constraint in Eq. A.1, the Lagrange multiplier vector is computed as follows,

$$\begin{aligned} \mathbf{X}^{\dagger}\mathbf{h} &= \mathbf{u} \\ \mathbf{X}^{\dagger}(\mathbf{T}^{-1}\mathbf{X}\Gamma) &= \mathbf{u}, \end{aligned} \tag{A.5}$$

and solving for $\Gamma$ gives,

$$\Gamma = (\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}. \tag{A.6}$$

Finally substituting $\Gamma$ in Eq. A.6 back into $\mathbf{h}$ in Eq. A.4 gives the solution,

$$\begin{aligned} \mathbf{h} &= \mathbf{T}^{-1}\mathbf{X}\Gamma \\ &= \mathbf{T}^{-1}\mathbf{X}(\mathbf{X}^{\dagger}\mathbf{T}^{-1}\mathbf{X})^{-1}\mathbf{u}. \end{aligned} \tag{A.7}$$

APPENDIX **B**

# MINIMIZING A RATIO OF QUADRATIC TERMS

The goal is to find the $\mathbf{h}$ that maximizes the Rayleigh Quotient (RQ),

$$J(\mathbf{h}) = \frac{\mathbf{h}^\dagger \mathbf{A}\mathbf{h}}{\mathbf{h}^\dagger \mathbf{B}\mathbf{h}}, \tag{B.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are assumed to be Hermitian (symmetric if they are real) matrices, and $\mathbf{h}^\dagger \mathbf{B}\mathbf{h} \neq 0$ (or equivalently, $\mathbf{B}$ is a non-singular matrix). This is accomplished by first taking the gradient and setting it equal to zero as follows,

$$\frac{\partial J(\mathbf{h})}{\partial \mathbf{h}} = \frac{2\mathbf{A}\mathbf{h}(\mathbf{h}^\dagger \mathbf{B}\mathbf{h}) - 2\mathbf{B}\mathbf{h}(\mathbf{h}^\dagger \mathbf{A}\mathbf{h})}{(\mathbf{h}^\dagger \mathbf{B}\mathbf{h})^2} = 0. \tag{B.2}$$

Rearranging terms gives,

$$\mathbf{A}\mathbf{h}(\mathbf{h}^\dagger \mathbf{B}\mathbf{h}) = \mathbf{B}\mathbf{h}(\mathbf{h}^\dagger \mathbf{A}\mathbf{h}), \tag{B.3}$$

diving both sides by the scalar $\mathbf{h}^\dagger \mathbf{B}\mathbf{h} \neq 0$ gives,

$$\begin{aligned} \mathbf{A}\mathbf{h} &= \mathbf{B}\mathbf{h}\frac{\mathbf{h}^\dagger \mathbf{A}\mathbf{h}}{\mathbf{h}^\dagger \mathbf{B}\mathbf{h}} \\ &= \mathbf{B}\mathbf{h}J(\mathbf{h}), \end{aligned} \tag{B.4}$$

and premultiplying by $\mathbf{B}^{-1}$ gives,

$$\begin{aligned} \mathbf{B}^{-1}\mathbf{A}\mathbf{h} &= \mathbf{h}J(\mathbf{h}) \\ &= \lambda\mathbf{h}, \end{aligned} \tag{B.5}$$

where $J(\mathbf{h}) = \lambda$. Eq. B.5 is an eigenvalue problem. Since $J(\mathbf{h}) = \lambda$ represents the eigenvalues of $\mathbf{B}^{-1}\mathbf{A}$, then $J(\mathbf{h})$ is maximized by the eigenvector $\mathbf{h}$ corresponding to the largest eigenvalue of $\mathbf{B}^{-1}\mathbf{A}$.

APPENDIX C

# DISTANCE FROM A POINT TO A HYPERPLANE

The goal is to find the distance from a point to a hyperplane. The vector $\mathbf{h}$ is orthogonal to the hyperplane $\mathbf{h}^\dagger\mathbf{x} + b = 0$, or equivalently, the vector $\mathbf{h}$ is orthogonal to any vector $\mathbf{v} = \mathbf{x}_A - \mathbf{x}_B$, where $\mathbf{x}_A$ and $\mathbf{x}_B$ are two different points on the hyperplane, i.e., $\mathbf{h}^\dagger\mathbf{x}_A + b = 0$ and $\mathbf{h}^\dagger\mathbf{x}_B + b = 0$ for $\mathbf{x}_A \neq \mathbf{x}_B$. This is easily verified using the fact that the inner product of two orthogonal vectors is zero, i.e.,

$$\begin{aligned} \mathbf{h}^\dagger\mathbf{v} &= \mathbf{h}^\dagger(\mathbf{x}_A - \mathbf{x}_B) \\ &= \mathbf{h}^\dagger\mathbf{x}_A - \mathbf{h}^\dagger\mathbf{x}_B \\ &= -b - (-b) = 0. \end{aligned} \tag{C.1}$$

Figure C.1 shows that

$$\mathbf{x}_i = \mathbf{x}_\perp + d\frac{\mathbf{h}}{|\mathbf{h}|}, \tag{C.2}$$

where $d$ is the perpendicular distance to a hyperplane from the point $\mathbf{x}_\perp$ to the point $\mathbf{x}_i$. Multiplying both sides by $\mathbf{h}^\dagger$ gives

$$\begin{aligned} \mathbf{h}^\dagger\mathbf{x}_i &= \mathbf{h}^\dagger\mathbf{x}_\perp + d\frac{|\mathbf{h}|^2}{|\mathbf{h}|} \\ &= -b + d|\mathbf{h}|. \end{aligned} \tag{C.3}$$

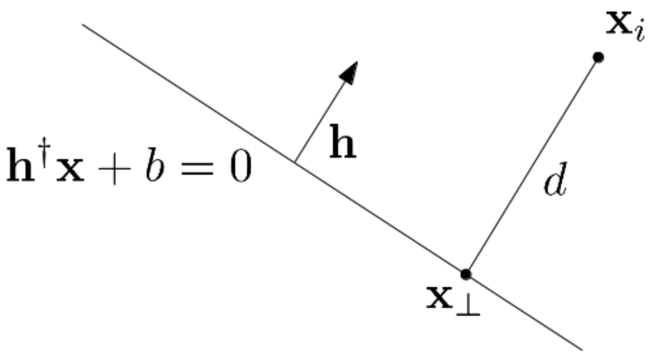


Figure C.1: The perpendicular distance from a point $\mathbf{x}_n$ to the hyperplane $\mathbf{h}^\dagger \mathbf{x} + b = 0$

Thus, the perpendicular distance from a given point $\mathbf{x}_i$ to the hyperplane $\mathbf{h}^\dagger \mathbf{x} + b = 0$ is given by

$$d = \frac{\mathbf{h}^\dagger \mathbf{x}_i + b}{|\mathbf{h}|}. \tag{C.4}$$

APPENDIX D

# QUADRATIC CORRELATION MATRIX

The QCF output for the $i$th training feature vector, $\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i$, can be expressed as follows,

$$\begin{aligned}\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i &= \mathbf{x}_i^T[\mathbf{q}_1,...,\mathbf{q}_d]\mathbf{x}_i\\ &= [\mathbf{x}_i^T\mathbf{q}_1,...,\mathbf{x}_i^T\mathbf{q}_d]\mathbf{x}_i,\end{aligned} \tag{D.1}$$

where $\mathbf{q}_1$ to $\mathbf{q}_d$ are the column vectors of $d \times d$ matrix $\mathbf{Q}$. If $\mathbf{x}_i$ represents a 1-D feature vector (instead of a vectorized 2-D feature image), the correlation output can be computed as follows (the first index of $\mathbf{q}_1$ is represented by $q_1[0]$),

$$\begin{aligned}g_i[n] &= (q_1[0]x_i[n+0] + q_1[1]x_i[n+1] + ... + q_1[d-1]x_i[n+d-1])\,x_i[n+0] + ...\\ &\quad (q_2[0]x_i[n+0] + q_2[1]x_i[n+1] + ... + q_2[d-1]x_i[n+d-1])\,x_i[n+1] + ...\\ &\quad \vdots\\ &\quad (q_d[0]x_i[n+0] + q_d[1]x_i[n+1] + ... + q_d[d-1]x_i[n+d-1])\,x_i[n+d-1]\\ &= \left(\sum_{k=0}^{d-1} x_i[n+k]q_1[k]\right)x_i[n+0] + ... + \left(\sum_{k=0}^{d-1} x_i[n+k]q_d[k]\right)x_i[n+d-1]\\ &= \sum_{k,l=0}^{d-1} x_i[n+k]q_{l+1}[k]x_i[n+l]\\ &= \sum_{k,l=0}^{d-1} x_i[n+k]q[l,k]x_i[n+l],\end{aligned} \tag{D.2}$$

where $q[l,k] = q_{l+1}[k]$. Note that Eq. D.2 can expressed as

$$\begin{aligned} g_i[n] &= \sum_{l=0}^{d-1} x_i[n+l] \sum_{k=0}^{d-1} x_i[n+k] q[l,k] \\ &= \sum_{l=0}^{d-1} x_i[n+l] o_i[l,n], \end{aligned} \tag{D.3}$$

where $o[l,n]$ is the correlation of $x_i[n]$ and $q[l,n]$ and can be efficiently computed in the frequency domain. That is, the correlation of $x_i[n]$ and $q[l,n]$ in the frequency domain is,

$$\begin{aligned} O_i(l,\omega) &= \sum_{n=0}^{d-1} o_i[l,n] e^{-j\omega n} \\ & \sum_{n=0}^{d-1} \left( \sum_{k=0}^{d-1} x_i[n+k] q[l,k] \right) e^{-j\omega n}. \end{aligned} \tag{D.4}$$

Substituting $p = n + k$ so that $n = p - k$ gives the following,

$$\begin{aligned} O_i(l,\omega) &= \sum_{p=0+k}^{d-1+k} \left( \sum_{k=0}^{d-1} x_i[p] q[l,k] \right) e^{-j\omega(p-k)} \\ &= \left( \sum_{p=0}^{d-1} x_i[p] e^{-j\omega p} \right) \left( \sum_{k=0}^{d-1} q[l,k] e^{j\omega k} \right) \\ &= X_i(\omega) Q^*(l,\omega). \end{aligned} \tag{D.5}$$

Eq. D.3 is not a correlation. Expressing the frequency domain of $g_i[n]$ in Eq. D.3 as combination of the frequency domains of $x_i$ and $q_l$ is not trivial. $g_i[n]$ can be expressed as

$$\begin{aligned}
g_i[n] &= \sum_{l=0}^{d-1} x_i[n+l]o_i[l,n] \\
&= \sum_{l=0}^{d-1} x_i^*[n+l]o_i[l,n] \\
&= \sum_{l=0}^{d-1} \left(\frac{1}{2\pi}\int_v X_i(v)e^{jvn}e^{jvl}dv\right)^* \left(\frac{1}{2\pi}\int_f O_i(l,f)e^{jfn}df\right) \\
&= \frac{1}{4\pi}\int_v\int_f X_i^*(v)\left(\sum_{l=0}^{d-1} O_i(l,f)e^{-jvl}\right)e^{-jvn}e^{jfn}dvdf \\
&= \frac{1}{4\pi}\int_v\int_f X_i^*(v)X_i(\omega)\left(\sum_{l=0}^{d-1} Q^*(l,\omega)e^{-jvl}\right)e^{-jvn}e^{jfn}dvdf \\
&= \frac{1}{4\pi}\int_v\int_f X_i^*(v)X_i(f)\mathcal{Q}^*(v,f)e^{-jvn}e^{jfn}dvdf, \qquad \text{(D.6)}
\end{aligned}$$

where

$$\mathcal{Q}^*(v,f) = \sum_{l=0}^{d-1} Q^*(l,\omega)e^{-jvl} \qquad \text{(D.7)}$$

is the Fourier transform with respect to $l$ of the Fourier transform of $q[l,n]$ with respect to $n$, and $x_i$ is assumed to be real ($x_i^*$ is used to simplify the notation). $G_i(\omega)$ can be expressed as

$$\begin{aligned}
G_i(\omega) &= \sum_{n=0}^{d-1} g_i[n]e^{-j\omega n} \\
&= \sum_{n=0}^{d-1}\left(\frac{1}{4\pi}\int_v\int_f X_i^*(v)X_i(f)\mathcal{Q}^*(v,f)e^{-jvn}e^{jfn}dvdf\right)e^{-j\omega n} \\
&= \frac{1}{4\pi}\int_v\int_f X_i^*(v)X_i(f)\mathcal{Q}^*(v,f)\left(\sum_{n=0}^{d-1} e^{-jvn}e^{jfn}e^{-j\omega n}\right)dvdf \\
&= \frac{1}{4\pi}\int_v\int_f X_i^*(v)X_i(f)\mathcal{Q}^*(v,f)\delta(f-v-\omega)dvdf \\
&= \frac{1}{2\pi}\int_v X_i^*(v)X(v+\omega)\mathcal{Q}^*(v,v+\omega)dv, \qquad \text{(D.8)}
\end{aligned}$$

which is not a product of Fourier transforms. Therefore, using the frequency domain does *not* provide a computational advantage. For this reason, QMMCF is derived in the space domain.

The quadratic correlation output in Eq. D.2 can be written as,

$$\mathbf{g}_i = \begin{bmatrix} \sum_{l,k=1}^{d} x_i[l]x_i[k]q_l[k] \\ \sum_{l,k=1}^{d} x_i[1+l]x_i[1+k]q_l[k] \\ \vdots \\ \sum_{l,k=1}^{d} x_i[d-1+l]x_i[d-1+k]q_l[k] \end{bmatrix} = \begin{bmatrix} \sum_{l,k=1}^{d} x_i[l]x_i[k]q_l[k] \\ \sum_{l,k=1}^{d} x_i^{(1\downarrow)}[l]x_i^{(1\downarrow)}[k]q_l[k] \\ \vdots \\ \sum_{l,k=1}^{d} x_i^{(d-1\downarrow)}[l]x_i^{(d-1\downarrow)}[k]q_l[k] \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{l=1}^{d} x_i[l]\mathbf{q}_l^T\mathbf{x}_i \\ \sum_{l=1}^{d} x_i^{(1\downarrow)}[l]\mathbf{q}_l^T\mathbf{x}_i^{(1\downarrow)} \\ \vdots \\ \sum_{l=1}^{d} x_i^{(d-1\downarrow)}[l]\mathbf{q}_l^T\mathbf{x}_i^{(d-1\downarrow)} \end{bmatrix} = \begin{bmatrix} \sum_{l=1}^{d} \mathbf{q}_l^T(\pi_l\mathbf{x}_i) \\ \sum_{l=1}^{d} \mathbf{q}_l^T(\pi_l\mathbf{x}_i^{(1\downarrow)}) \\ \vdots \\ \sum_{l=1}^{d} \mathbf{q}_l^T(\pi_l\mathbf{x}_i^{(d-1\downarrow)}) \end{bmatrix}, \qquad \text{(D.9)}$$

where $\mathbf{g}_i$ is a vector of length $d$, $\mathbf{x}_i^{(r\downarrow)}$ represents the vector $\mathbf{x}_i$ shifted by $r$ pixels, and $\pi_l\mathbf{x}_i^{(r\downarrow)}$ represents the shifted vector multiplied by its $l$th value. Eq. D.2 can be expressed using matrix notation as follows,

$$\mathbf{g}_i = \begin{bmatrix} \sum_{l=1}^{d} \mathbf{q}_l^T(\pi_l\mathbf{x}_i) \\ \sum_{l=1}^{d} \mathbf{q}_l^T(\pi_l\mathbf{x}_i^{(1\downarrow)}) \\ \vdots \\ \sum_{l=1}^{d} \mathbf{q}_l^T(\pi_l\mathbf{x}_i^{(d-1\downarrow)}) \end{bmatrix} = \begin{bmatrix} \pi_1\mathbf{x}_i^T & \pi_2\mathbf{x}_i^T & \dots & \pi_d\mathbf{x}_i^T \\ \pi_1\mathbf{x}_i^{(1\downarrow)T} & \pi_2\mathbf{x}_i^{(1\downarrow)T} & \dots & \pi_d\mathbf{x}_i^{(1\downarrow)T} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_1\mathbf{x}_i^{(d-1\downarrow)T} & \pi_2\mathbf{x}_i^{(d-1\downarrow)T} & \dots & \pi_d\mathbf{x}_i^{(d-1\downarrow)T} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_d \end{bmatrix},$$

$$= \mathbf{X}_{\mathbf{C}i}^T\mathbf{h}, \qquad \text{(D.10)}$$

where

$$\mathbf{X}_{\mathbf{C}i} = \begin{bmatrix} \pi_1\mathbf{x}_i & \pi_1\mathbf{x}_i^{(1\downarrow)} & \dots & \pi_1\mathbf{x}_i^{(d-1\downarrow)} \\ \pi_2\mathbf{x}_i & \pi_2\mathbf{x}_i^{(1\downarrow)} & \dots & \pi_2\mathbf{x}_i^{(d-1\downarrow)} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_d\mathbf{x}_i & \pi_d\mathbf{x}_i^{(1\downarrow)} & \dots & \pi_d\mathbf{x}_i^{(d-1\downarrow)} \end{bmatrix} \qquad \text{(D.11)}$$

is a $d^2 \times d$ matrix and

$$\mathbf{h} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_d \end{bmatrix} \tag{D.12}$$

is a vector of length $d^2$. The shifts in Eq. D.11 could be regular shifts or circular shifts. In previous chapters, we implicitly used circular shifts by not zero extending the images in the training phase. However, if regular shifts are desired, then when $\mathbf{x}_i$ is shifted by $r$, there are $r$ more zeros in $\mathbf{x}_i^{(r\downarrow)}$ than in $\mathbf{x}_i$, and therefore $\mathbf{X}_{\mathbf{C}i}$ can be simplified as

$$\mathbf{X}_{\mathbf{C}i} = \begin{bmatrix} \pi_1\mathbf{x}_i & \pi_1\mathbf{x}_i^{(1\downarrow)} & ... & \pi_1\mathbf{x}_i^{(d-1\downarrow)} \\ \pi_2\mathbf{x}_i & \pi_2\mathbf{x}_i^{(1\downarrow)} & ... & 0 \\ \vdots & \vdots & & \vdots \\ \pi_d\mathbf{x}_i & 0 & ... & 0 \end{bmatrix}. \tag{D.13}$$

In our experiments, $\mathbf{x}_i$ represents a 2-D feature image. In this case, the quadratic correlation output when $x$ is a 2-D $R \times C$ feature image is

$$g_i[n,m] = \sum_{k=0}^{R-1}\sum_{l=0}^{R-1}\sum_{u=0}^{C-1}\sum_{v=0}^{C-1} x_i[n+l,m+u]x_i[n+k,m+v]q_{l+1,u+1}[k,v], \tag{D.14}$$

where the columns of $\mathbf{Q}$ are each rearrange as 2-D $R \times C$ arrays. The values of $\mathbf{g}_i$ can be expressed as follows,

$$
\mathbf{g}_i = \begin{bmatrix}
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l,u]x_i[k,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1}\sum_{u,v=0}^{C} x_i[l+1,u]x_i[k+1,v]q_{l,u}[k,v] \\
\vdots \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l+R-1,u]x_i[k+R-1,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l,u+1]x_i[k,v+1]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l+1,u+1]x_i[k+1,v+1]q_{l,u}[k,v] \\
\vdots \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l+R-1,u+1]x_i[k+R-1,v+1]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l,u+2]x_i[k,v+2]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l+1,u+2]x_i[k+1,v+2]q_{l,u}[k,v] \\
\vdots \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l+R-1,u+C-1]x_i[k+R-1,v+C-1]q_{l,u}[k,v]
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i[l,u]x_i[k,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(1\downarrow)}[l,u]x_i^{(1\downarrow)}[k,v]q_{l,u}[k,v] \\
\vdots \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(R-1\downarrow)}[l,u]x_i^{(R-1\downarrow)}[k,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(\overrightarrow{1})}[l,u]x_i^{(\overrightarrow{1})}[k,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(1\downarrow,\overrightarrow{1})}[l,u]x_i^{(1\downarrow,\overrightarrow{1})}[k,v]q_{l,u}[k,v] \\
\vdots \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(R-1\downarrow,\overrightarrow{1})}[l,u]x_i^{(R-1\downarrow,\overrightarrow{1})}[k,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(\overrightarrow{2})}[l,u]x_i^{(\overrightarrow{2})}[k,v]q_{l,u}[k,v] \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(1\downarrow,\overrightarrow{2})}[l,u]x_i^{(1\downarrow,\overrightarrow{2})}[k,v]q_{l,u}[k,v] \\
\vdots \\
\sum_{l,k=0}^{R-1}\sum_{u,v=0}^{C-1} x_i^{(R-1\downarrow,\overrightarrow{C-1})}[l,u]x_i^{(R-1\downarrow,\overrightarrow{C-1})}[k,v]q_{l,u}[k,v]
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(1\downarrow)} \\
\vdots \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(R-1\downarrow)} \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(\overrightarrow{1})} \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(1\downarrow,\overrightarrow{1})} \\
\vdots \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{1})} \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(\overrightarrow{2})} \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(1\downarrow,\overrightarrow{2})} \\
\vdots \\
\sum_{l=1}^{R}\sum_{u=0}^{C-1} \mathbf{q}_{uR+l}^{T}\pi_{uR+l}\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})}
\end{bmatrix}
= \begin{bmatrix}
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(1\downarrow)} \\
\vdots \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(R-1\downarrow)} \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(\overrightarrow{1})} \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(1\downarrow,\overrightarrow{1})} \\
\vdots \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{1})} \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(\overrightarrow{2})} \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(1\downarrow,\overrightarrow{2})} \\
\vdots \\
\sum_{n=1}^{d} \mathbf{q}_n^{T}\pi_n\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})}
\end{bmatrix}, \qquad \text{(D.15)}
$$

where $\mathbf{x}_i^{(r\downarrow)(c\rightarrow)}$ represents the vectorized 2-D $R \times C$ image shifted by $r$ pixels down and $c$ pixels to the right, and $\pi_n\mathbf{x}_i^{(r\downarrow)(c\rightarrow)}$ represents the shifted vector multiplied by its $n$th value. Eq. D.14 can be expressed in matrix notation as follows,

$$\begin{aligned}\mathbf{g}_i = \begin{bmatrix} \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(1\downarrow)} \\ \vdots \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(R-1\downarrow)} \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(\overrightarrow{1})} \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(1\downarrow,\overrightarrow{1})} \\ \vdots \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{1})} \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(\overrightarrow{2})} \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(1\downarrow,\overrightarrow{2})} \\ \vdots \\ \sum_{n=1}^{d} \mathbf{q}_n^T \pi_n \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \end{bmatrix} &= \begin{bmatrix} \pi_1 \mathbf{x}_i^T & ... & \pi_d \mathbf{x}_i^T \\ \pi_1 \mathbf{x}_i^{(1\downarrow)T} & ... & \pi_d \mathbf{x}_i^{(1\downarrow)T} \\ \vdots & & \vdots \\ \pi_1 \mathbf{x}_i^{(R-1\downarrow)T} & ... & \pi_d \mathbf{x}_i^{(R-1\downarrow)T} \\ \pi_1 \mathbf{x}_i^{(\overrightarrow{1})T} & ... & \pi_d \mathbf{x}_i^{(\overrightarrow{1})T} \\ \pi_1 \mathbf{x}_i^{(1\downarrow,\overrightarrow{1})T} & ... & \pi_d \mathbf{x}_i^{(1\downarrow,\overrightarrow{1})T} \\ \vdots & & \vdots \\ \pi_1 \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{1})T} & ... & \pi_d \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{1})T} \\ \pi_1 \mathbf{x}_i^{(\overrightarrow{2})T} & ... & \pi_d \mathbf{x}_i^{(\overrightarrow{2})T} \\ \pi_1 \mathbf{x}_i^{(1\downarrow,\overrightarrow{2})T} & ... & \pi_d \mathbf{x}_i^{(1\downarrow,\overrightarrow{2})T} \\ \vdots & & \vdots \\ \pi_1 \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})T} & ... & \pi_d \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})T} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_d \end{bmatrix} \\ &= \mathbf{X}_{\mathbf{C}i}^T \mathbf{h}, \end{aligned} \tag{D.16}$$

where

$$\mathbf{X}_{\mathbf{C}i} = \begin{bmatrix} \pi_1 \mathbf{x}_i & ... & \pi_1 \mathbf{x}_i^{(R-1\downarrow)} & \pi_1 \mathbf{x}_i^{(\overrightarrow{1})T} & \pi_1 \mathbf{x}_i^{(1\downarrow,\overrightarrow{1})T} & ... & \pi_1 \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\ \pi_2 \mathbf{x}_i & ... & \pi_2 \mathbf{x}_i^{(R-1\downarrow)} & \pi_2 \mathbf{x}_i^{(\overrightarrow{1})T} & \pi_2 \mathbf{x}_i^{(1\downarrow,\overrightarrow{1})T} & & \pi_2 \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \pi_d \mathbf{x}_i & ... & \pi_d \mathbf{x}_i^{(R-1\downarrow)} & \pi_d \mathbf{x}_i^{(\overrightarrow{1})T} & \pi_d \mathbf{x}_i^{(1\downarrow,\overrightarrow{1})T} & ... & \pi_d \mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \end{bmatrix} \tag{D.17}$$

is a $d^2 \times d$ matrix, $\mathbf{x}_i^{(r\downarrow)(c\rightarrow)}$ represents the vectorized 2-D $R \times C$ feature image shifted by $r$ pixels down and $c$ pixels to the right. The vector that starts at the $n, m$ entry of this $\mathbf{X}_{\mathbf{C}i}^T$ matrix is

$$\pi_n \mathbf{x}_i^{\left(m\%R\downarrow, \overrightarrow{\lfloor \frac{m}{R} \rfloor - 1}\right)}, \tag{D.18}$$

where $\%$ represents the modulus (or remainder) operator, and $\lfloor \cdot \rfloor$ represents the floor operator. The shifts in Eq. D.17 can be regular shifts or circular shifts. If they are regular shifts, then when $\mathbf{x}_i$ is shifted down by $r$, there are $r$ more zeros in $\mathbf{x}_i^{(r\downarrow)}$ than in $\mathbf{x}_i$, and the same applies to shifts to the right, and therefore $\mathbf{X}_{\mathbf{C}i}$ can be simplified as

$$\mathbf{X}_{\mathbf{C}i} = \begin{bmatrix} \pi_1\mathbf{x}_i & \ldots & \pi_1\mathbf{x}_i^{(R-1\downarrow)} & \ldots & \pi_1\mathbf{x}_i^{(\overrightarrow{C-1})} & \ldots & \pi_1\mathbf{x}_i^{(R-1\downarrow,\overrightarrow{C-1})} \\ \pi_2\mathbf{x}_i & \ldots & 0 & \ldots & \pi_2\mathbf{x}_i^{(\overrightarrow{C-1})} & \ldots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ \pi_C\mathbf{x}_i & \ldots & 0 & & \pi_C\mathbf{x}_i^{(\overrightarrow{C-1})} & \ldots & 0 \\ \pi_{C+1}\mathbf{x}_i & \ldots & \pi_{C+1}\mathbf{x}_i^{(R-1\downarrow)} & \ldots & 0 & \ldots & 0 \\ \pi_{C+2}\mathbf{x}_i & \ldots & 0 & \ldots & 0 & \ldots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ \pi_{2C}\mathbf{x}_i & \ldots & 0 & \ldots & 0 & \ldots & 0 \\ \pi_{2C+1}\mathbf{x}_i & \ldots & \pi_{2C+1}\mathbf{x}_i^{(R-1\downarrow)} & \ldots & 0 & \ldots & 0 \\ \pi_{2C+2}\mathbf{x}_i & \ldots & 0 & & 0 & \ldots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ \pi_d\mathbf{x}_i & \ldots & 0 & \ldots & 0 & \ldots & 0 \end{bmatrix}. \tag{D.19}$$

Note that $\mathbf{X}_{\mathbf{C}i}$ is an upper triangular matrix formed by upper triangular matrices.

The QCF output for the $i$th training feature vector, $\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i$, can be written as a dot product as follows,

$$\begin{aligned} \mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i &= \mathbf{x}_i^T[\mathbf{q}_1, ..., \mathbf{q}_d]\mathbf{x}_i \\ &= \mathbf{h}^T\mathbf{y}_i, \end{aligned} \tag{D.20}$$

where

$$\mathbf{h} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_d \end{bmatrix}, \quad \mathbf{y}_i = \begin{bmatrix} \pi_1\mathbf{x}_i \\ \pi_2\mathbf{x}_i \\ \vdots \\ \pi_d\mathbf{x}_i \end{bmatrix}, \tag{D.21}$$

where $\pi_n\mathbf{x}_i$ represents the $\mathbf{x}_i$ multiplied by its $n$th value, and $\mathbf{q}_1$ to $\mathbf{q}_d$ are the column vectors of matrix $\mathbf{Q}$. Note that the vectors $\mathbf{y}_i$ and $\mathbf{h}$ are of dimension $d^2$, and vector $\mathbf{x}_i$ is of dimension $d$.

The quadratic correlation output is of dimension $d$; however, the correlation between $\mathbf{h}$ and $\mathbf{y}_i$ is of dimension $d^2$. Therefore, the quadratic correlation output is not just $\mathbf{c} = \mathbf{h} \otimes \mathbf{y}_i$. In fact, the $d^2 \times d^2$ correlation matrix $\mathbf{Y}_{\mathbf{C}i}$ such that $\mathbf{c} = \mathbf{h} \otimes \mathbf{y}_i = \mathbf{Y}_{\mathbf{C}i}^T \mathbf{h}$ can be shown to be,

$$\mathbf{Y}_{\mathbf{C}i} = \left[ \begin{array}{cccc} \mathbf{y}_i & \mathbf{y}_i^{(1\downarrow)} & ... & \mathbf{y}_i^{(d^2-1)\downarrow)} \end{array} \right], \tag{D.22}$$

which is different than the $d^2 \times d$ matrix

$$\begin{aligned} \mathbf{X}_{\mathbf{C}i} &= \left[ \begin{array}{cccc} \pi_1 \mathbf{x}_i & \pi_1 \mathbf{x}_i^{(1\downarrow)} & ... & \pi_1 \mathbf{x}_i^{(d-1\downarrow)} \\ \pi_2 \mathbf{x}_i & \pi_2 \mathbf{x}_i^{(1\downarrow)} & ... & \pi_2 \mathbf{x}_i^{(d-1\downarrow)} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_d \mathbf{x}_i & \pi_d \mathbf{x}_i^{(1\downarrow)} & ... & \pi_d \mathbf{x}_i^{(d-1\downarrow)} \end{array} \right] \\ &= \left[ \begin{array}{cccc} \mathbf{y}_i & \mathbf{y}_i^{(d\downarrow)} & ... & \mathbf{y}_i^{(d(d-1)\downarrow)} \end{array} \right] \end{aligned} \tag{D.23}$$

(note that $\mathbf{y}_i$ is shifted by a product of $d$ in $\mathbf{X}_{\mathbf{C}i}$) used in Chapter 5.

# BIBLIOGRAPHY

[1] K. Al-Mashouq, B. V. K. Vijaya Kumar, and M. Alkanhal. Analysis of signal-to-noise ratio of polynomial correlation filters. In *Proc. SPIE*, 1999.

[2] O. Alatas, P. Yan, and M. Shah. Spatio-temporal regularity flow (SPREF): its estimation and applications. *IEEE Trans. Circuits and Systems for Video Technology*, 17(5):584–589, 2007.

[3] M. Alkanhal and B. V. K. Vijaya Kumar. Polynomial distance classifier correlation filter for pattern recognition. *Applied Optics*, 42(23):4688–4708, 2003.

[4] M. Alkanhal, B. V. K. Vijaya Kumar, and A. Mahalanobis. Improving the false alarm capabilities of the maximum average correlation height correlation filter. *Opt. Eng.*, 39(5):1133–1141, 2000.

[5] Anwaar-ul-Haq, Iqbal Gondal, and Manzur Murshed. Action recognition using spatio-temporal distance classifier correlation filter. In *Int'l Conf. on Digital Image Computing*, 2011.

[6] A. Ashraf, S. Lucey, and T. Chen. Reinterpreting the applicaiton of gabor filters as a manipulation of the margin in linear support vector machines. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(7):1335–1341, 2010.

[7] P. Banerjee, J. Chandra, and A. Datta. Feature based optimal trade-off parameter selection of frequency domain correlation filter for real time face authentication. In *Proc. Int'l Conf. on Communication, Computing & Security*, 2011.

[8] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. Wiley, New York, 2001.

[9] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] V. N. Boddeti, F. Su, and B. V. K. Vijaya Kumar. A biometric key-binding and template protection framework using correlation filters. In *Lecture Notes on Computer Science*, 2009.

[11] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[12] D. S. Bolme, B. A. Draper, and J. R. Beveridge. Average of synthetic exact filters. In *IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[13] D. S. Bolme, Y. M. Lui, B. A. Draper, and J. R. Beveridge. Simple real-time human detection using a single correlation filter. In *IEEE Int'l Workshop on Performance Evaluation of Tracking and Surveillance*, 2010.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. Fifth Annual Workshop on Computational Learning Theory*, 1992.

[15] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[16] D. Casasent, G. Ravichandran, and S. Bollapraggada. Gaussian minimum average correlation energy correlation filters. *Applied Optics*, 30(35):5176–5181, 1991.

[17] Military Sensing Information Analysis Center. www.sensiac.org.

[18] O. Chapelle and B. Scholkopf. Incorporating invariances in non-linear support vector machines. *Advances in Neural Information Processing Systems*, 1(1):609–616, 2002.

[19] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[20] Germund Dahlquist and Ake Bjorck. *Numerical Methods in Scientific Computing: Volume 1*. Society for Industrial and Applied Mathematics, 2008.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[22] D. Decoste and B. Scholkopf. Training invariant support vector machines. *Machine Learning*, 46(1):161–190, 2002.

[23] J. Ebling and G. Scheuermann. Clifford Fourier transform on vector fields. *IEEE Trans. Visualization and Computer Graphics*, 11(4):469–479, 2005.

[24] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[25] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Cascade object detection with deformable part models, Aug 2012.

[26] J. Figue and P. Refregier. Optimality of trade-off filters. *Applied Optics*, 32(11):1933–1935, 1993.

[27] K. Fukunaga and W. L. G. Koontz. Representation of random processes using the finite karhunen-loève expansion. *IEEE Trans. Information and Control*, 16(1):85–101, 1970.

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

[29] Y. N. Hsu and H. H. Arsenault. Optical character recognition using circular harmonic expansion. *Applied Optics*, 21(22):4016–4019, 1982.

[30] W. Karush. *Minima of functions of several variables with inequalities as side constraints*. Master's thesis, University of Chicago, 1939.

[31] R. A. Kerekes and B. V. K. Vijaya Kumar. Correlation filters with controlled scale response. *IEEE Trans. Image Processing*, 15(7):1794–1802, 2006.

[32] R. A. Kerekes and B. V. K. Vijaya Kumar. Multiple target detection in video using quadratic multi-frame correlation filtering. In *Proc. SPIE*, 2008.

[33] R. A. Kerekes and B. V. K. Vijaya Kumar. Selecting a composite correlation filter design: a survey and comparative study. *Opt. Eng.*, 47(6):1–18, 2008.

[34] R. A. Kerekes and B. V. K. Vijaya Kumar. Enhanced video-based target detection using multi-frame correlation filtering. *IEEE Trans. Aerospace and Electronic Systems*, 45(1):289–307, 2009.

[35] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proc. Berkeley Symp. on Mathematical Statistics and Probabilities*. Univ. of California Press, 1951.

[36] A. Mahalanobis, R. Muise, and S. R. Stanfill. Quadratic correlation filter design methodology for target detection and surveillance applications. *Applied Optics*, 43(27):5198–5205, 2004.

[37] A. Mahalanobis, R. Muise, S. R. Stanfill, and A. Van Nevel. Design and application of quadratic correlation filters for target detection. *Applied Optics*, 40(3):837–850, 2004.

[38] A. Mahalanobis and H. Singh. Application of correlation filters for texture recognition. *Applied Optics*, 33(11):2173–2179, 1994.

[39] A. Mahalanobis, R. Stanfill, and K. Chen. A bayesian approach to activity detection in video using multi-frame correlation filters. In *Proc. SPIE*, 2011.

[40] A. Mahalanobis and B. V. K. Vijaya Kumar. Polynomial filters for higher-order and multi-input information fusion. In *Euro American Opto-Electronic Information Processing Workshop*, 1997.

[41] A. Mahalanobis, B. V. K. Vijaya Kumar, and D. Casasent. Minimum average correlation energy filters. *Applied Optics*, 26(5):3633–3640, 1987.

[42] A. Mahalanobis, B. V. K. Vijaya Kumar, and R. Frankot. Intraclass and between-class training-image registration for correlation-filter synthesis. *Applied Optics*, 39(17):2918–2924, 2000.

[43] A. Mahalanobis, B. V. K. Vijaya Kumar, and S. R. F. Sims. Distance classifier correlation filters for distortion tolerance, discrimination and clutter rejection. In *Proc. SPIE*, 1993.

[44] A. Mahalanobis, B. V. K. Vijaya Kumar, and S. R. F. Sims. Distance classifier correlation filters for multiclass automatic recognition. *Applied Optics*, 35(17):3127–3133, 1996.

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

[46] D. Mendlovic, E. Marom, and N. Konforti. Shift and scale invariant pattern recognition using Mellin radial harmonics. *Opt. Comm.*, 67(3):172–176, 1988.

[47] R. Muise, A. Mahalanobis, R. Mohapatra, X. Li, D. Han, and W. Mikhael. Constrained quadratic correlation filters for target detection. *Applied Optics*, 43(2):304–314, 2004.

[48] A. Nevel and A. Mahalanobis. Comparative study of maximum average correlation height filter variants using ladar imagery. In *Proc. SPIE*, 2003.

[49] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time signal processing*. Prentice Hall, 2009.

[50] A. V. Oppenheim, A. S. Willsky, and S. Hamid. *Signals and Systems*. Prentice Hall, 1997.

[51] E. Osuna, R. Freund, and F. Girosit. Training support vector machines: an application to face detection. In *IEEE Conf. Computer Vision and Pattern Recognition*, 1997.

[52] K. B. Petersen and M. S. Pedersen. The matrix cookbook, Oct 2008.

[53] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods–Support Vector Learning*, 208(14):98–112, 1998.

[54] G. Ravichandran and D. Casasent. Minimum noise and correlation energy (MINACE) optical correlation filter. *Applied Optics*, 31(11):1823–1833, 1992.

[55] Ph. Réfrégier. Filter design for optical pattern recognition: multicriteria optimization approach. *Opt. Lett.*, 15(15):854–856, 1990.

[56] Ph. Réfrégier and J. Figue. Optimal trade-off filters for pattern recognition and their comparison with the wiener approach. *Optical Computing and Processing*, 1(3):245–266, 1991.

[57] A. Rodriguez, J. Panza, and B. V. K. Vijaya Kumar. Automatic recognition of multiple targets with varying velocities using quadratic correlation filters and kalman filters. In *IEEE Radar*, 2010.

[58] A. Rodriguez, J. Panza, and B. V. K. Vijaya Kumar. Selecting a background for the training images of a correlation filter: a comparative study. In *Proc. SPIE*, 2010.

[59] A. Rodriguez and B. V. K. Vijaya Kumar. Automatic multi-target recognition from two classes using quadratic correlation filters. In *Proc. SPIE*, 2010.

[60] A. Rodriguez and B. V. K. Vijaya Kumar. Automatic target recognition of multiple targets from two classes with varying velocities using correlation filters. In *IEEE Int'l Conf. of Image Processing*, 2010.

[61] A. Rodriguez and B. V. K. Vijaya Kumar. Segmentation-free ocular detection and recognition. In *Proc. SPIE*, 2011.

[62] Andres Rodriguez, Vishnu Naresh Boddeti, B. V. K. Vijaya Kumar, and Abhijit Mahalanobis. Maximum margin correlation filter: A new approach for localization and classification. *IEEE Trans. Image Processing*, 2012. (submitted).

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

[64] M. Savvides, J. Heo, J. Thornton, P. Hennings, C. Xie, K. Venkataramani, R. A. Kerekes, M. Beattie, and B. V. K. Vijaya Kumar. Biometric identification using advanced correlation

filter methods. In *Springer-Verlag Lecture Notes in Computer Science, Ambient Intelligence*, 2005.

[65] M. Savvides, K. Venkataramani, and B. V. K. Vijaya Kumar. Incremental updating of advanced correlation filters for biometric authentication systems. In *IEEE Proc. Int'l Conf. on Multimedia Expo*, 2003.

[66] M. Savvides and B. V. K. Vijaka Kumar. Illumination normalization using logarithm transforms for face authentication. In *Proc. Int'l Conf. on Advances in Pattern Recognition*, 2003.

[67] M. Savvides and B. V. K. Vijaya Kumar. Efficient design of advanced correlation filters for robust distortion-tolerant face recognition. In *IEEE Conf. Advanced Video and Signal Based Surveillance*, 2003.

[68] M. Savvides, B. V. K. Vijaya Kumar, and P. Khosla. Face verification using correlation filters. In *IEEE Workshop on Automatic Identification Advanced Technologies*, 2002.

[69] G. F. Schils and D. W. Sweeney. Rotationally invariant correlation filtering. *J. Opt. Soc. Am. A*, 2(9):1411–1418, 1985.

[70] B. Scholkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In *Proc. Int'l Conf. on Artificial Neural Networks*, 1996.

[71] R. K. Shenoy. *Object detection and classification in SAR images using MINACE correlation filters*. Master's thesis, Carnegie Mellon University, Pittsburgh, April 1995.

[72] P.K. Shivaswamy and T. Jebara. Relative margin machines. In *Proc. Advances in Neural Information Processing Systems*, 2008.

[73] S. R. F. Sims and A. Mahalanobis. Performance evaluation of quadratic correlation filters for target detection and discrimination in infrared imagery. *Opt. Eng.*, 43(8):1705–1711, 2004.

[74] R. Singh. *Advanced correlation filters for multi-class synthetic aperture radar detection and classification*. Master's thesis, Carnegie Mellon University, Pittsburgh, May 2002.

[75] R. Singh and B. V. K. Vijaya Kumar. Performance of the extended maximum average correlation height (EMACH) filter and the polynomial distance classifier correlation filter (PDCCF) for multiclass SAR detection and classification. In *Proc. SPIE*, 2002.

[76] E. Tajahuerce, A. Moya, J. Garcia, and C. Ferreira. Real filter based on Mellin radial harmonics for scale-invariant pattern recognition. *Applied Optics*, 33(14):3086–3093, 1994.

[77] J. Thornton. *Iris matching under deformation and occlusion*. PhD thesis, Carnegie Mellon University, Pittsburgh, April 2007.

[78] J. Thornton, M. Savvides, and B. V. K. Vijaya Kumar. Linear shift-invariant maximum margin svm correlation filter. In *Proc. Intelligent Sensors, Sensor Networks and Information Processing Conf.*, 2005.

[79] J. Thornton, M. Savvides, and B. V. K. Vijaya Kumar. A Bayesian approach to deformed pattern matching of iris images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(4):596–606, 2007.

[80] A. Van Nevel and A. Mahalanobis. Comparative study of maximum average correlation height filter variants using ladar imagery. *Opt. Eng.*, 42(2):541–550, 2004.

[81] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[82] B. V. K. Vijaya Kumar. Minimum-variance synthetic discriminant functions. *J. Opt. Soc. Am. A*, 3(10):1579–1584, 1986.

[83] B. V. K. Vijaya Kumar and M. Alkanhal. Eigen-extended maximum average correlation height (EEMACH) filters for automatic target recognition. In *Proc. SPIE*, 2001.

[84] B. V. K. Vijaya Kumar, D. W. Carlson, and A. Mahalanobis. Optimal trade-off synthetic discriminant function filters for arbitrary devices. *Opt. Lett.*, 19(19):1556–1558, 1994.

[85] B. V. K. Vijaya Kumar and L. Hassebrook. Performance measures for correlation filters. *Applied Optics*, 29(20):2997–3006, 1990.

[86] B. V. K. Vijaya Kumar and A. Mahalanobis. Alternate interpretation for minimum variance synthetic discriminant functions. *Applied Optics*, 25(15):2484–2485, 1986.

[87] B. V. K. Vijaya Kumar, A. Mahalanobis, and R. D. Juday. *Correlation Pattern Recognition*. Cambridge Univ. Press, 2005.

[88] B. V. K. Vijaya Kumar, A. Mahalanobis, S. Songs, S. Sims, and J. Epperson. Minimum squared error synthetic discriminant function. *Opt. Eng.*, 31(5):915–922, 1992.

[89] B. V. K. Vijaya Kumar, A. Mahalanobis, and A. Takessian. Optimal tradeoff circular harmonic function correlation filter methods providing controlled in-plane rotation response. *IEEE Trans. Image Processing*, 9(6):1025–1034, 2000.

[90] B. V. K. Vijaya Kumar and T. Ng. Multiple circular-harmonic-function correlation filter providing specified response to in-plane rotation. *Applied Optics*, 11(11):1871–1878, 1996.

[91] B. Walls and A. Mahalanobis. Performance of the MACH filter and DCCF algorithms in the presence of data compression. In *Proc. SPIE*, 1999.

[92] R. Wu and H. Stark. Rotation-invariant pattern recognition using a vector reference. *Applied Optics*, 23(6):838–840, 1984.