DESIGN OF COMPOSITE FILTER REALIZABLE ON PRACTICAL SPATIAL LIGHT MODULATORS

FINAL REPORT
Contract No. NAG 9-710, Basic

Submitted to:
Lyndon B. Johnson Space Center
National Aeronautics and Space Administration
Houston, Texas

By
Dr. P. k. Rajan
and
R. Ramakrishnan

Technical Report #MCTR-0794-01

Center for Manufacturing Research
and
Technology Utilization

Cookeville, Tennessee 38505

DESIGN OF A COMPOSITE FILTER REALIZABLE ON PRACTICAL SPATIAL LIGHT MODULATORS

Final Report
Contract No. NAG 9-710, Basic

Submitted to:

Lyndon B. Johnson Space Center
National Aeronautics and Space Administration
Houston, Texas

By
Dr. P.K. Rajan
and
R. Ramakrishnan

Center for Manufacturing Research
and Technology Utilization
Tennessee Technological University
Cookeville, TN 38505

July 1994

ABSTRACT

Hybrid optical correlator systems use two Spatial Light Modulators (SLMs), one at the input plane and the other at the filter plane. Currently available SLMs such as the Deformable Mirror Device (DMD) and Liquid Crystal Television (LCTV) SLMs exhibit arbitrarily constrained operating characteristics. The pattern recognition filters designed with the assumption that the SLMs have ideal operating characteristic may not behave as expected when implemented on the DMD or LCTV SLMs. Therefore it is necessary to incorporate the SLM constraints in the design of the filters.

In this report, an iterative method is developed for the design of an unconstrained Minimum Average Correlation Energy (MACE) filter. Then using this algorithm a new approach for the design of a SLM constrained distortion invariant filter in the presence of input SLM is developed. Two different optimization algorithms are used to maximize the objective function during filter synthesis, one based on the simplex method and the other based on the Hooke and Jeeves method. Also, the simulated annealing based filter design algorithm proposed by Khan and Rajan is refined and improved.

The performance of the filter is evaluated in terms of its recognition/discrimination capabilities using computer simulations and the results are compared with a simulated annealing optimization based MACE filter. The filters are designed for different LCTV SLM's operating characteristics and the correlation responses are compared. The distortion tolerance and the false class image discrimination qualities of the filter are comparable to those of the simulated annealing based filter but the new filter design takes about 1/6 of the computer time taken by the simulated annealing filter design.

ACKNOWLEDGMENTS

The authors would like to express their appreciation to Dr. Richard D. Juday of NASA Johnson Space Center, Houston for providing the LCTV SLM characteristics and other filter design software.

The authors would like to thank the Lyndon B. Johnson Space Center, National Aeronautical and Space Administration (NASA), Houston, Texas for funding the research contract No. NAG 9-710. The authors would also like to thank the Center for Manufacturing Research and Technology Utilization of Tennessee Technological University and the Department of Electrical Engineering, Tennessee Technological university for supporting this research.

This report was extracted with minor revisions from the thesis for M.S. degree in Electrical Engineering submitted by Ramakrishnan Ramachandran to the Graduate School of Tennessee Technological University, comprising the research carried out under the supervision of Dr. P. K. Rajan.

TABLE OF CONTENTS

Page

Chapter

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1
INTRODUCTION

There has been considerable growth of interest in problems of pattern recognition and image processing for the past two decades. Applications of pattern recognition and image processing include character recognition, target detection, medical diagnosis, remote sensing, speech recognition, fingerprint identification, archaeology, missile guidance, and automatic inspection. Due to its inherent parallel operation and high speed, optical pattern recognition (OPR) is a preferred method for real-time applications such as missile guidance, target tracking, and aerospace missions. Hence considerable work is being carried out in developing a practical optical correlator system for pattern recognition applications.

1.1 Optical Correlator

The possibility of implementing two-dimensional Fourier transform using optical lenses [1] encouraged the use of correlation-based techniques for pattern recognition. The research in optical pattern recognition was triggered by the optical realization of complex valued matched filter by Vander Lugt in 1964 [2]. Vander Lugt's method is illustrated in Figure 1.1. When a coherent parallel beam of light from lens L1 passes through a transparency P1 of a scene, the light becomes amplitude modulated with that scene. The resulting beam is focused on plane P2 by lens L2 producing a spatial Fourier transform of the original scene.

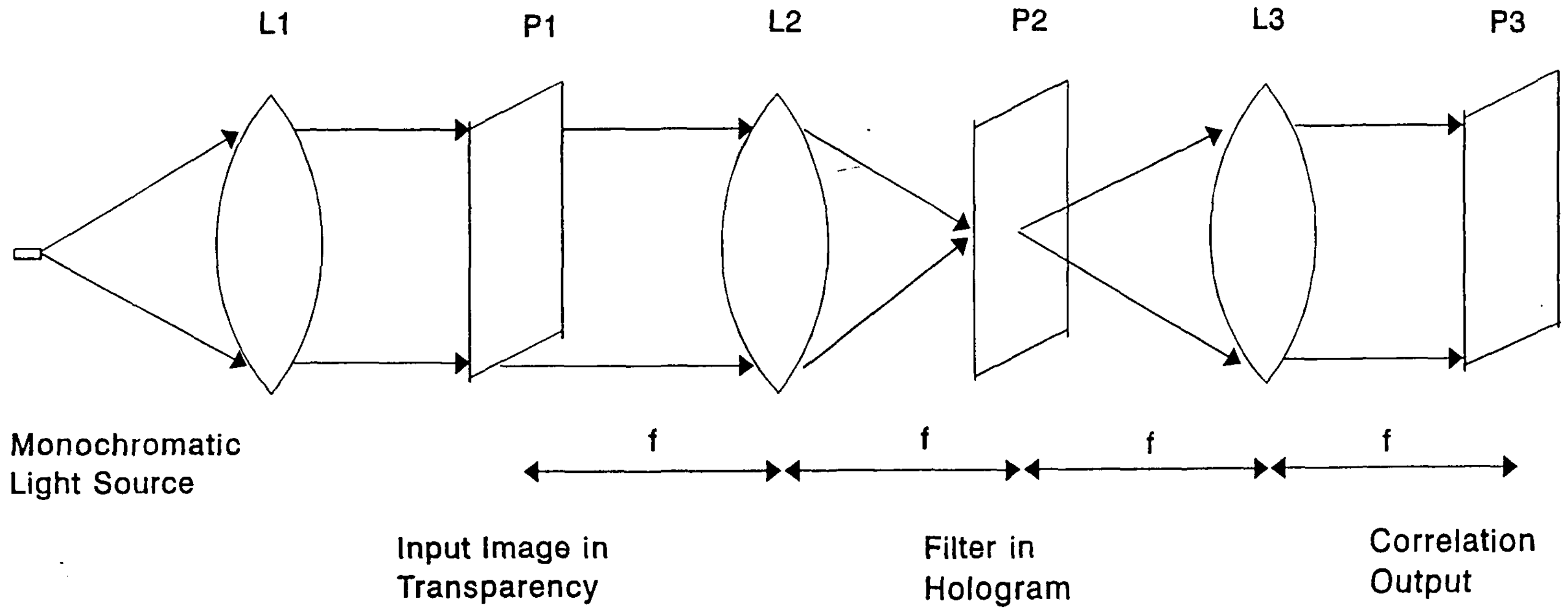

Figure 1.1. VanderLugt's Correlator

The output at plane P2 is the product of the Fourier transformed input image and the matched spatial filter (MSF). The lens L3 then takes the inverse Fourier transform. The light intensity at the correlator output plane P3 can be used to determine the presence of the reference scene in the input scene.

Optical correlators as described above have certain difficulties for real-time practical applications. They are :

1. To process the signal at the correlator plane, certain electronic systems are required.
2. Purely optical systems can be designed for a specific application; however, they cannot be used where a great level of flexibility is required.

Correlators can be made more practicable by using a hybrid system shown in Figure 1.2. Hybrid systems are made by combining electronic technology with optical systems. In the hybrid systems fast switching of filters is possible and also the output correlation plane can be processed for decision making.

In the hybrid system shown in Figure 1.2, L1 is the collimating lens that gives rise to a uniform light distribution, t(x,y). The object of interest, i(x,y), is imaged by the camera. The camera output is used by the Spatial Light Modulator (SLM) driver to generate the drive signal for the input SLM. The input SLM (SLM1) modulates the incident light and the resulting amplitude value at SLM1 output is given by

$$s_1(x,y) = i(x,y) \cdot t(x,y). \tag{1.1}$$

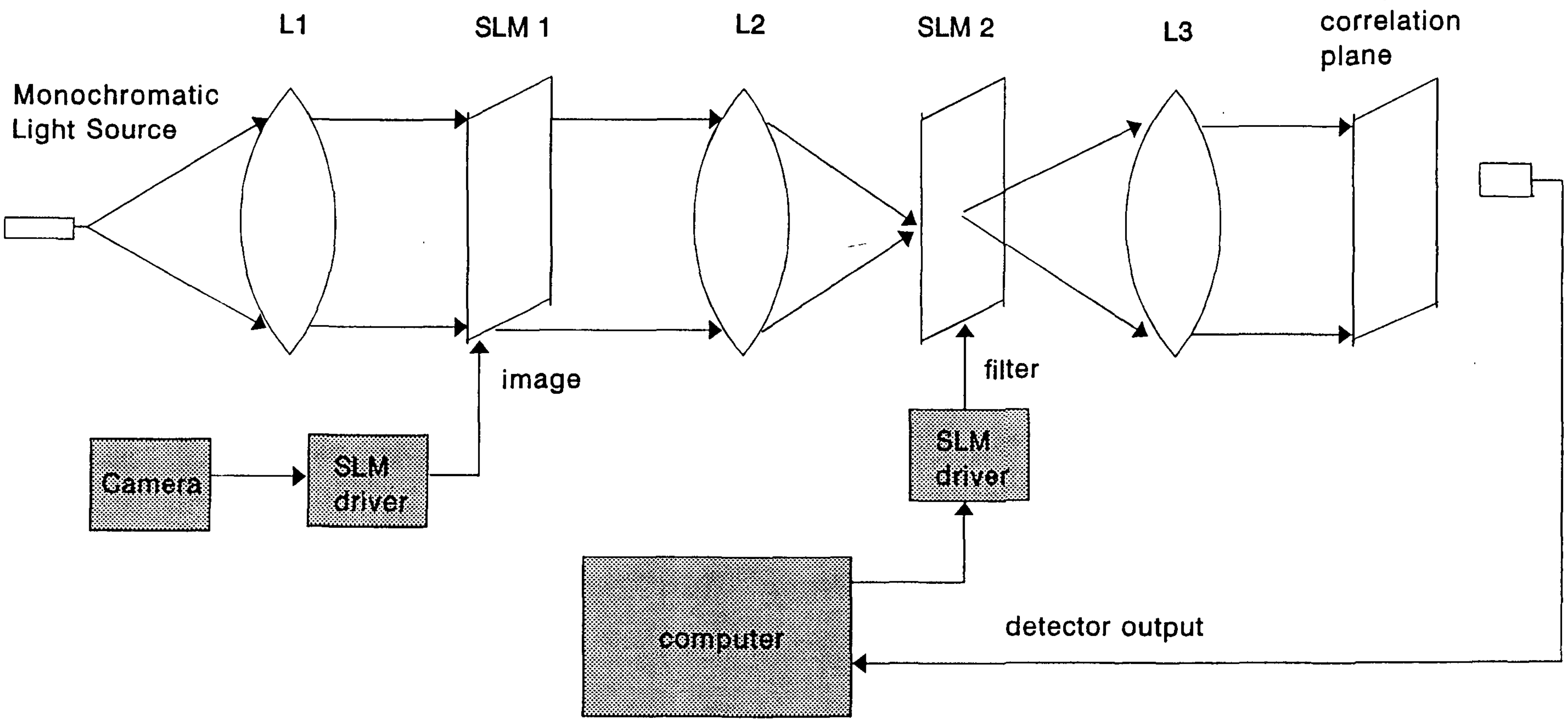

Figure 1.2. A typical Hybrid Correlator System

The lens L2 produces the Fourier transform of s_1(x,y) at the location of SLM2 , denoted by

$$S_1(u,v) = \mathcal{F}\{s_1(x,y)\} = \iint s_1(x,y) \cdot e^{-j2\pi(ux+vy)} dxdy. \tag{1.2}$$

To implement correlation in optical Fourier domain, the computer synthesized frequency domain complex valued filter H(u,v) is implemented on SLM2. If the filter is a matched spatial filter (MSF), then the filter is the conjugate of the Fourier transform of the reference image, given as

$$H(u,v) = S_2^\star(u,v) \tag{1.3}$$

where the superscript $\star$ denotes the complex conjugate operation and

$$S_2(u,v) = \mathcal{F}\{s_2(x,y)\} \tag{1.4}$$

where $s_2(x,y)$ is the reference image. The output of SLM2 is

$$\begin{aligned} C_{12}(u,v) &= S_1(u,v) \cdot H(u,v) && (1.5) \\ &= S_1(u,v) \cdot S_2^\star(u,v). && (1.6) \end{aligned}$$

The lens L3 takes the inverse Fourier transform and the correlation output is available at the correlation plane. It is given as

$$\begin{aligned} c_{12}(x,y) &= \mathcal{F}^{-1}\{C_{12}(u,v)\} && (1.7) \\ &= \iint S_1(u,v)H(u,v)e^{j2\pi(ux+vy)}dudv && (1.8) \\ &= \iint s_1(x',y')s_2^\star(x'-x,y'-y)dx'dy' && (1.9) \end{aligned}$$

$$= \quad s_1(x,y) \otimes s_2(x,y) \tag{1.10}$$

where $\otimes$ denotes the correlation operation.

It can be shown that for two images s_1 and s_2 the correlation has the following property [3]

$$[c_{12}(x,y)]^2 \leq c_{11}(0,0) \cdot c_{22}(0,0) \tag{1.11}$$

where c_{ii} is the auto-correlation value at the origin for the i^{th} image, given as

$$c_{ii}(0,0) = ||s_i||^2 = \iint |s_i(x,y)|^2 dxdy. \tag{1.12}$$

If the input image s_1 is the same as the reference image s_2 then

$$|c_{11}(x,y)| \leq c_{11}(0,0). \tag{1.13}$$

So, the correlation value at the origin can be used to make a decision whether the input image is identical to the reference image.

1.2 Need for the Study

Vander Lugt's correlator used an MSF in the filter plane. These MSFs yield the highest possible Signal-to-Noise Ratio (SNR) when detecting a known image in the presence of additive white noise. Generally it is expected that a correlation filter possesses the following characteristics:

1. Produce a large correlation peak sharp enough for easy identification of the true class (wanted) image at the input.

2. Produce a small correlation peak for a false class (unwanted) image when present at the input.
3. Produce a high SNR at the correlation plane.
4. Produce good light efficiency.
5. Be tolerant to the geometric distortions (different perspective views) of the same object. The filter is normally called "distortion tolerant," if it possesses the above quality.
6. Be implementable on the commercially available SLMs.

Unfortunately, no single filter can provide all the above characteristics. Even though the MSFs give the highest SNR when detecting a known image in white noise they do not produce sharp correlation peaks and they are light inefficient [4]. Also the MSFs are very sensitive to geometric distortions in the input images. Further, MSFs are complex valued in nature. Currently available SLMs, viz., Magneto-Optic SLM (MOSLM), Deformable Mirror Device (DMD), Liquid Crystal Light Valve (LCLV), and Liquid Crystal Television (LCTV) cannot accommodate a fully complex valued filter.

To overcome the limitations of MSF, several new filter synthesis techniques have been proposed. The light efficiency (Horner efficiency) is improved by the Phase-Only (POF) filters [5]. The distortion tolerance is improved by the composite filters known as Synthetic Discriminant Function (SDF) filters [6]. They are discussed next.

1.2.1 Composite Filters

The SDF is synthesized from a linear combination of input training images with weights adjusted to satisfy the specified correlation values at the origin [7]. The SDF has two major problems: (1) The SDFs control only the output at the origin. When the peak is not at the origin this leads to sidelobe problems. (2) The SDF filter is generally complex valued and cannot be implemented on the currently available SLMs.

To alleviate the shortcomings of SDF, several variations to SDFs have been proposed. Among them, the Minimum Average Correlation Energy (MACE) filter [8] and Minimum Noise and Correlation Energy (MINACE) filter [9] have received considerable attention. The MACE filter tries to minimize the average correlation plane energy while maintaining the specified value at the origin in the correlation plane and thus produces a sharp correlation peak. The MINACE filter reduces the correlation plane energies resulting from both noise and training images and offers better noise tolerance than MACE filter. Both of the above filter designs use training images which are sufficiently representative of all the expected geometric distortions, to produce distortion invariant filters. The details of the filter designs are given in Chapter 2.

The MACE and the MINACE filters are generally complex valued in nature. As mentioned earlier, currently available SLMs cannot implement a fully complex filter. Each SLM has a different constraint. The constraints are imposed on the filter in the Fourier domain. The filter designed assuming ideal operating conditions may not behave as expected when implemented on the constrained SLMs. So it is necessary to take SLM constraints into account when designing a filter. Some single image SLM constrained filters are discussed next.

1.2.2 SLM Constrained Single Image Filters

Recently, experimenters have shown interest in the use of inexpensive LCTV SLM's for input and filter planes in optical correlators. This SLM has a single control signal which affects both phase and amplitude together at each pixel point in the SLM [10]. This type of operating characteristic is called "arbitrarily constrained " or "amplitude and phase cross coupled." Juday [11] has developed a set of necessary conditions for optimizing an optical correlation filter realizable with coupled phase and amplitude SLM . Farn and Goodman [12] proposed a technique for the design of an optimal filter (in the sense of maximizing intensity) for arbitrarily constrained devices. Vijayakumar, Juday, and Rajan [13] proposed the design of Saturated Filters (SF's). The SF's optimize SNR if the SLM exhibits annular response operating curve. Saturated filter design also considered the detector noise at the correlation plane. Juday [14] proposed a unified approach to synthesize optimal realizable filters for various metrics, viz., Intensity, SNR, Peak to Correlation Energy (PCE), and Peak to Total Energy (PTE). An important outcome of the above approach is that optimal performance can be obtained for any SLM limitation by using the Minimum Euclidean Distance Mapping (MED) between the optimal filter (unconstrained) and the achievable filter (SLM constrained). The optimal mapping minimizes the mean squared error between the optimal filter and the achievable filter responses.

All the filter designs proposed above consider only a single image as reference for filter synthesis. Since these filter designs do not consider the possible image distortions while constructing the filter, the filters are sensitive to geometric distortions in the input image and do not produce acceptable performance. To design distortion invariant filters for constrained SLMs, several techniques have been proposed by researchers. They are discussed next.

1.2.3 SLM Constrained Composite Filters

Jared and Ennis [15] have presented the design of Binary Phase-Only Filter (BPOF) by including the filter modulation (due to SLM) in the synthesis process.

Casasent and Rozzi [16] have shown that the performance of Phase-only and Binary Phase-only SDFs is not satisfactory for pattern recognition. Balendra and Rajan [17]-[18] have developed real-valued MACE (RMACE) and real-valued MVSDF-MACE filters. Mahalanobis and Song have also developed the real-valued MACE filter [19]. These filters are suitable for implementation on a filter SLM whose operating characteristic is not complex. Rajan and Muttiah [20]-[21] designed saturated MACE filters implementable on SLMs with origin-centered, finite contrast annular response regions. Commercially available SLM's such as DMD and LCTV exhibit arbitrarily constrained operating characteristics.

Carlson and Vijayakumar [22] extended Jared and Ennis's [15] relaxation algorithm for coupled SLMs. They introduced a projection function for mapping unconstrained MACE filter on the SLM operating curve to get a realizable filter. The designed filter maximizes the Fisher Ratio [23] for two-class (True, False) Pattern recognition. Their results for MOSLM, DMD, and LCTV SLMs reveal that the filter behaviour for non-training images is not satisfactory. They did not consider the effect of input SLM and noise in the input images. There is no guarantee that the relaxation-based iterative technique will always converge.

Khan and Rajan [24]-[25] used a simulated annealing-based optimization technique [26] for minimizing the average correlation plane energy and the deviation of the obtained correlation peak value at origin from the specified value. Simulated annealing is a form of stochastic optimization, and is generally used for problems involving a large number of variables. The methods discussed above for the design of filters for arbitrarily constrained SLMs use a mapping technique to map an unrealizable filter to an SLM operating curve during the optimization process. But Khan and Rajan's method uses optimization on the realizable filter and does not involve mapping during the iteration and this reduces the computation time. Also in the simulated annealing optimization algorithm, it is possible to reduce the probability of being trapped in local minima while searching for the global minimum. The simulation results show that the filter performance is satisfactory. The correlation response of the filter for true class images shows a sharp correlation peak and low correlation plane energy. This helps in easy detection of the peak which is needed for decision-making. But Khan's method has a few drawbacks. They are:

1. The algorithm at the start assumes a value at the origin of the correlation plane and tries to achieve it for all the correlation planes. As the value is chosen arbitrarily, the specified value may not be the best value for the selected SLM.
2. The design also assumes the correlation value at the origin to be real. This restriction is not necessary for the filter design.
3. The design did not consider the presence of an input SLM.
4. Simulated annealing-based optimization technique [26] is computationally intensive.

1.3 Objectives

In order to overcome the shortcomings mentioned above in the design of composite filters for arbitrarily constrained SLMs, the main objective of the research reported in this report was to develop a technique for the design of input and filter SLM constrained composite filters for pattern recognition and study its performance for various LCTV SLM's operating characteristics. The LCTV SLM exhibits different operating characteristics such as amplitude-only, phase-mostly, high-contrast, and highly-coupled. These characteristics were provided by Dr. Richard Juday of NASA Johnson Space center.

Since the MACE-SDF produces a sharp correlation peak with minimum correlation plane energy, the MACE-SDF formulation was used during the design of the SLM constrained filters. Three different optimization techniques were used. They are: (1) simplex method [27], (2) Hooke and Jeeves method [28], and (3) simulated annealing method [26]. The sum of peak to correlation energy (PCE) for all the correlation planes belonging to the training images was chosen as the objective function to be maximized using either Simplex or Hooke and Jeeves method. Juday's [14] Minimum Euclidean Distance (MED) concept was used for mapping an unconstrained filter on an SLM operating curve during the filter design. Finally a relaxation based [15] technique was used to get equal correlation peak values for all the training images. Khan and Rajan's [24] simulated annealing-based MACE filter design was refined and used to compare with the new filter design technique based on simplex or Hooke and Jeeves method.

1.4 Outline of the report

The report is organized as follows. In Chapter 2, various unconstrained and SLM constrained filter designs are reviewed. Chapter 2 also discusses the characteristics of currently available SLMs. Chapter 3 discusses the design of a composite filter for arbitrarily constrained SLMs at both the input and the filter planes. Chapter 4 presents the simulation tests carried out to study the filter performance for distortion in the input images. The performance of filters designed using simplex method and Hooke and Jeeves method are compared with simulated annealing based filter design. Chapter 5 presents the conclusions of this research and recommendations for future work.

CHAPTER 2
PATTERN RECOGNITION VIA OPR FILTERS

In this chapter, a review of correlation-based filter design techniques that have been proposed in literature is presented. The correlation-based filters can be broadly classified as Unconstrained filters and SLM constrained filters. The filters designed without considering the SLM's limitation are termed as unconstrained filters. The SLM constrained filters are designed by incorporating the SLM's limitations in the design process.

2.1 Unconstrained Filters

Unconstrained filters are designed with the assumption that the SLMs have infinite contrast ratio and can accommodate any complex valued filter. In this section some filter designs are discussed which do not take into account the SLM's limitations. Throughout this report the images as well as the filters are expressed in discrete domain as either vector or scalar functions. The vectors are represented by the symbol ($\vec{\ }$) and scalar quantities are denoted by lowercase letters. Uppercase symbols also refer to the frequency plane terms and lowercase to represent spatial domain quantities.

2.1.1 The Matched Filter (MF)

An optical correlation filter matched to the reference image s(x,y) should have a transfer function proportional to the complex conjugate of the image spectrum.

The MF is given as

$$H(u,v) = k\frac{S^{\star}(u,v)}{P_n(u,v)} \tag{2.1}$$

where $S^{\star}(u,v)$ is the complex conjugate of the image spectrum, $P_n(u,v)$ is the power spectral density of the noise in the test image, and k is an arbitrary constant. In the case where noise spectral density can be assumed constant, the filter transfer function and its impulse response become

$$H(u,v) = k^{'}S^{\star}(u,v) \tag{2.2}$$

and

$$h(x,y) = k^{'}s^{\star}(-x,-y). \tag{2.3}$$

The matched filter maximizes the signal-to-noise ratio (SNR) when the noise is stationary and additive [29]. The noise spectral density can often be considered uniform (white noise). However, in pattern recognition applications, false class images (unwanted) are considered as noise. Clearly, this noise can, in general, be highly correlated to the image; in this case the MF described in Equation 2.1 is not necessarily optimum [29]. The major shortcoming of MF is its sensitivity to distortions of the reference images, poor light efficiency, and the complex valued nature of the filter.

Various techniques have been proposed to get distortion invariant filters for OPR. The most well-known is the Synthetic Discriminant Function (SDF) [6] and its variants. The design of SDF is discussed next.

2.1.2 Conventional SDF

The SDF is constructed as a filter that is matched to a linear combination of different images in the training set that cover the possible range of distortions. In the following a derivation of SDF as proposed in [6] is outlined.

Let $\vec{s}_i = [s_i(1) \quad s_i(2) \quad \ldots s_i(d)]^T$ represent the i^{th} training image in the set of N_t training images each with d pixels. The vector $\vec{S}_i = [S_i(1) \quad S_i(2) \quad \ldots S_i(d)]^T$ denotes the DFT sequence of $\vec{s}_i$. Let $S = [\vec{S}_1 \quad \vec{S}_2 \quad \ldots \vec{S}_{N_t}]$ be a matrix of $d \times N_t$ formed by DFT vectors of training images. The SDF filter, $\vec{H}_{SDF}$ is designed to satisfy the constraint given as

$$S^\dagger \vec{H}_{SDF} = \vec{c}, \tag{2.4}$$

where $\vec{c} = [c_1(0) \quad c_2(0) \quad \ldots c_{N_t}(0)]^T$ is the correlation output vector and the superscript † denotes the conjugate transpose operation. c_i is the correlation output at the origin when the image $\vec{s}_i$ is placed at the input of correlator. Normally c_i is assumed to be 1 for true class and 0 for false class images.

The SDF filter is given as

$$\vec{H}_{SDF} = S\vec{a}. \tag{2.5}$$

The unknown weight vector, $\vec{a} = [a_1 \quad a_2 \quad \ldots a_{N_t}]^T$ can be determined by substituting Equation (2.5) in Equation (2.4),

$$\vec{a} = (S^\dagger S)^{-1}\vec{c}. \tag{2.6}$$

By substituting Equation (2.6) in Equation (2.5), the SDF filter is given as

$$\vec{H}_{SDF} = S(S^{\dagger}S)^{-1}\vec{c}. \tag{2.7}$$

The SDF designed as above has two major drawbacks. They are :

1. The SDF controls only the output at the origin. When the peak is not at the origin, it is difficult to locate the peak value in the presence of noise.
2. The SDF is generally complex valued and cannot be implemented on currently available SLMs.

When the input noise is white, the SDF is the best one (in the sense of minimizing the output variance) [30]. But in reality noise is not always white.

To overcome the shortcomings of SDF, different variants of SDF have been proposed. Vijayakumar [30] proposed Minimum Variance Synthetic Discriminant Function (MVSDF) which minimized the output variance due to colored noise in the input. Similar to SDF, the MVSDF also controls the response at only one point in the output correlation plane. Thus large sidelobes in the correlation may degrade its performance. Mahalanobis et al. [8] proposed the Minimum Average Correlation Energy (MACE) filter which produces sharp correlation peak at the origin while minimizing the average correlation plane energy. The details of the design for MACE filter is given next.

2.1.3 The MACE Filter

Let $\vec{s}_i$ represent the i^{th} training image in the set of N_t training images and $\vec{S}_i$ denote the DFT sequence of $\vec{s}_i$. Let S be a matrix of $d \times N_t$ formed by DFT vectors of N_t training images, given as

$$S = [\vec{S}_1 \quad \vec{S}_2 \quad \ldots \vec{S}_{N_t}]. \tag{2.8}$$

The vector $\vec{h}$ represents the MACE filter in space domain and its DFT sequence is $\vec{H}$. The average correlation plane energy for all the N_t images is given by

$$E_{ave} = \frac{1}{N_t}\sum_{i=1}^{N_t}\left[\frac{1}{d}\sum_{k=1}^{d}\left\{\left|\vec{H}(k)\right|^2\left|\vec{S}_i(k)\right|^2\right\}\right]. \tag{2.9}$$

The above equation can be written in matrix-vector form as

$$E_{ave} = \vec{H}^{\dagger} D \vec{H} \tag{2.10}$$

where the superscript † denotes the conjugate transpose operation. D is a diagonal matrix of size $d \times d$ and its diagonal elements are given by

$$D(k,k) = \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{\left|\vec{S}_i(k)\right|^2}{d}. \tag{2.11}$$

The objective of the MACE filter is to minimize the average correlation plane energy while maintaining the user-specified value at the correlation plane origin for each training image. The user-specified constraint at the origin is given as

$$c_i(0) = \frac{1}{d}\sum_{k=1}^{d}\vec{S}_i^{*}(k)\vec{H}(k) = b_i \tag{2.12}$$

for all $i = 1, 2, ..., N_t$ training set images. By using the Lagrange Multiplier method, the constrained minimization problem is solved and the MACE filter is obtained as

$$\vec{H}_{MACE} = D^{-1}S(S^{\dagger}D^{-1}S)^{-1}\vec{b}. \tag{2.13}$$

To get the relationship between the MACE filter and the conventional SDF, the MACE filter is written as

$$\begin{aligned} \vec{H}_{MACE} &= D^{-1/2}D^{-1/2}S(S^{\dagger}D^{-1/2}D^{-1/2}S)^{-1}\vec{b} \\ &= D^{-1/2}\bar{S}(\bar{S}^{\dagger}\bar{S})^{-1}\vec{b} \\ &= D^{-1/2}\bar{\vec{H}}_{SDF}. \end{aligned} \tag{2.14}$$

So, the MACE filter can be viewed as the conventional SDF for the preprocessed images. The preprocessor, $D^{-1/2}$, forces the average (over all the training images) power spectrum of the training images to become white. The MACE filter is optimal for target recognition in the presence of noise for which $D^{-1/2}$ is the whitening filter. Since the MACE filter is synthesized using training images of possible distortions, this filter can be used for distortion invariant pattern recognition. However, the filter is not realizable on the available SLMs. In Chapter 3, a method is proposed to design a MACE filter so that it is realizable on arbitrarily constrained SLMs.

2.2 SLM Constrained Filters

The filter designs discussed in Section 2.1 did not consider the SLM limitations. So, these filters are not always practically realizable. To overcome this difficulty various filter design techniques which take into account the SLM limitations have been proposed. These are discussed next.

2.2.1 Spatial Light Modulators (SLMs)

The implementation of OPR correlator requires two SLMs, one at the input plane and the other at the filter plane. Several SLMs are developed for this purpose. They are : Magneto-Optic SLM (MOSLM), Liquid Crystal Light Valve (LCLV), Deformable Mirror Device (DMD), and Liquid Crystal Television (LCTV). The SLMs encode the phase and amplitude information of an image or a filter onto an incident beam of coherent light as a function of space. Figure 2.1 shows some operating characteristics of SLMs. None of the above SLMs have ideal characteristics. The LCTV and DMD SLMs exhibit phase-mostly and arbitrarily constrained operating characteristics. MOSLM is capable of producing binary phase only (BPO) and ternary phase only (TPO) characteristics.

2.2.2 Filter Designs for Binary and Ternary Characteristics

The MOSLM is a binary or ternary modulator and it can take two values +1 and -1 or three values +1, 0 and -1. Psaltis et al. [31] proposed a Binary Phase-Only Filter (BPOF) which is implementable on MOSLM. Generally BPOF is expressed as

$$\begin{aligned} H_{BPOF}(u,v) &= \mathrm{sgn}\left[S_R(u,v)\cos\beta + S_I(u,v)\sin\beta\right] \qquad (2.15) \\ \mathrm{sgn[x]} &= \begin{cases} +1 & \text{if x} \geq 0 \\ -1 & \text{otherwise} \end{cases} \end{aligned}$$

where β ranges from $-\pi/2$ to $\pi/2$ and $S_R(u,v)$ and $S_I(u,v)$ are real and imaginary parts of the image spectrum, respectively. The threshold angle β needs to be searched in the range $[-\pi/2,\pi/2]$ to find the optimum choice.

To improve the SNR of BPOF, Vijayakumar and Bahri [32] suggested a 3-level filter taking on values -1, 0, +1 and this filter is called a Ternary Phase Amplitude Filter (TPAF). This filter is implementable on a MOSLM. Dickey et al. [33] proposed complex ternary matched filter (CTMF).

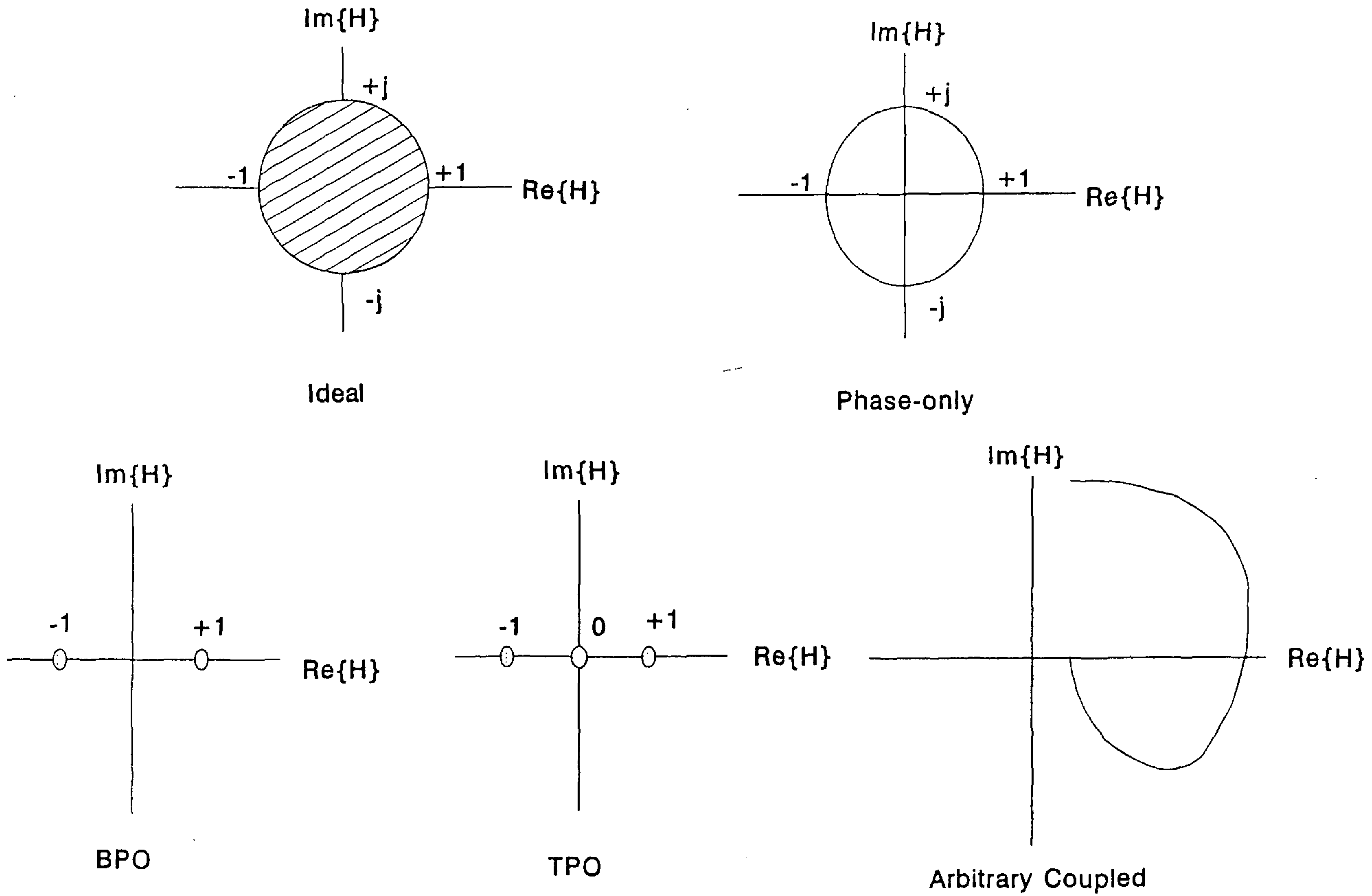

Figure 2.1. Operating Characteristics of SLMs

The CTMF can be implemented using two MOSLMs operating in a 3-level mode. The CTMF filter yields output SNR within 1dB of the SNR provided by the MF.

All the filters discussed above, namely BPOF, TPAF, and CTMF were synthesized with one training image and so they are not distortion invariant. Hendrix and Vijayakumar [34] proposed an iterative algorithm for synthesizing BPOF composite filters. Recently, Downie [35] reported that binary composite filters show fairly low levels of discrimination ability and SNR and he proposed ternary SDFs (TSDFs) with the purpose of increasing the overall performance over BSDFs.

2.2.3 Filter Designs for Real, Phase-Only, and Annular Characteristics

Balendra and Rajan [17] developed real-valued MACE, real-valued MVSDF-MACE, and real-valued space domain MACE filters. Mahalanobis and Song [19] have also developed the design of real MACE. These filters are suitable for practical implementation on SLMs whose operating characteristics are real. In some operating modes, the LCTV SLM exhibits nearly real characteristic.

Since the Phase-Only Filter (POF) produces sharper correlation peak and higher light efficiency than MF, some researchers worked on the development of Phase-Only SDFs. Horner and Gianino [36] modified the conventional SDF to phase-only SDF by setting the magnitude of the filter to unity. In this approach, the original SDF requirements may no longer be met. Jared and Ennis [15] included the filter modulation in the filter synthesis process itself. They used a relaxation algorithm to synthesize BPOF-SDF and POF-SDF. Casasent and Rozzi [16] showed with the help of simulations that the performance of BPOF-SDF and POF-SDF are in general unacceptable.

Vijayakumar, Juday, and Rajan [13] proposed the design of saturated filters (SFs) which optimized the SNR for finite contrast, annular response SLMs. The POFs and BPOFs are special cases of SF for phase-only and real SLMs, respectively. Rajan and Muttiah [20] designed saturated MACE filters for SLMs having origin-centered, finite contrast, annular response regions. Commercially available SLMs do not exhibit this type of characteristic.

Recently Roberge and Shang designed a phase-only composite filter using LCTV SLM [37]. They used a relaxation-based iterative technique to synthesize the filter and implemented the filter on the phase-mostly operating characteristic of LCTV. The correlation results show low sidelobes and sharp correlation peak. The distortion tolerance of the filter is not given. They assumed the phase-only characteristic for filter SLM but in reality the LCTV SLM exhibits phase-mostly operating characteristic.

2.2.4 Filter Designs for Arbitrarily Constrained SLMs

The LCTV and DMD SLMs exhibit arbitrarily constrained operating characteristics. These devices are also called arbitrarily constrained or cross-coupled devices. In the following sections different techniques for filter synthesis suitable for implementation using cross-coupled SLMs are discussed.

2.2.4.1 Farn and Goodman's approach. Farn and Goodman [12] proposed a fast algorithm to design a filter which maximized the correlation intensity at the output origin. A brief description of the algorithm as given in [12] is presented below. Since the realizable region (SLM's constraint) is assumed continuous, the analysis for the filter design is presented in the continuous domain.

The correlation intensity at the output origin is expressed as

$$I = \left| \int_{-\infty}^{+\infty} S(u)H(u)du \right|^2 \tag{2.16}$$

where $S(u) = |S(u)| \exp(j\theta_s(u))$ and $H(u) = |H(u)| \exp(j\theta_H(u))$. The correlation value at the origin is

$$c = \int_B S(u)H(u)du = |c| \exp(j\alpha) \tag{2.17}$$

where B denotes the passband of the filter. The objective of the design is to maximize $|c|$ with respect to an SLM realizability constraint given as

$$H(u) \ \epsilon \ \Omega \ \text{ for all } \ u \tag{2.18}$$

where Ω is the region of realizability. The maximum correlation value (magnitude) is dependent on α, the phase of the correlation output at the origin. So

$$|c(\alpha)| = \int_B \mathrm{Re}\left[S(u)H(u)\mathrm{exp}(-j\alpha)\right] du \tag{2.19}$$

and

$$\begin{aligned} max\,|c(\alpha)| &= max \left[\int_B \mathrm{Re}\left[H(u)S(u)\mathrm{exp}(-j\alpha)\right] du \right] \\ &= \int_B |S(u)|\, max \left\{ |H(u)| \cos\left[\theta_H(u) + \theta_S(u) - \alpha\right] \right\} du. \end{aligned} \tag{2.20}$$

The above equation explains that to determine the best α which maximizes the intensity at the origin, the α is to be searched in the range $[0, 2\pi)$ for each signal frequency. Also, the term H(u) is variable. So, for each signal frequency, H(u) is varied on the realizable curve and α is varied in the range $[0, 2\pi]$. This is a computationally intensive operation.

To speed up this process Farn and Goodman suggested a projection function G given as

$$\begin{aligned} G(\phi) &= max\left\{\mathrm{Re}\left[H(u)\exp(-j\phi)\right]\right\} \\ &= max\left\{|H(u)|\cos[\theta_H(u) - \phi]\right\}. \end{aligned} \tag{2.21}$$

Since the region Ω is known a priori, the projection function G is calculated ahead of time. The projection function is illustrated in Figure 2.2. Now, the maximum value at the origin can be expressed as

$$max\,|c(\alpha)| = \int_B |S(u)|\, G[\alpha - \theta_S(u)]du. \tag{2.22}$$

From the above equation it is understood that the optimization is simply a search over α.

2.2.4.2 Juday's approach. In this section, the analysis is presented in the continuous domain because the SLM's realizable region is assumed to be continuous. Juday [11] used calculus of variation to derive the necessary conditions under which the filter produces the maximum correlation intensity. Let the signal be $s(t)$ and its Fourier transform be $S(f)$. Then $S(f) = A(f)\exp[j\phi(f)]$. The coupled SLMs are normally controlled by a single variable called drive V. The filter is $H(f) = \mu[V(f)]\exp\{jv[V(f)]\}$, where $\mu[.]$ and $v[.]$ are SLM's magnitude and phase values for the given drive value V.

The value at the origin of the correlation plane is

$$c(0) = \int_{-\infty}^{+\infty} A(f)\exp[j\phi(f)]\mu[V(f)]\exp\{jv[V(f)]\}df. \tag{2.23}$$

Juday derived a necessary condition for the maximum value of c(0) as

$$\phi(f) + a[V(f)] = \theta, \text{ a constant,} \tag{2.24}$$

where a[.] is called the augmented phase (Refer Figure 2.3). The augmented phase is given as:

$$v[V(f)] + \text{Arg}\{[\frac{\partial\mu(V)}{\partial V} + j\mu(V)\frac{\partial v[V]}{\partial V}]\} = a(V). \tag{2.25}$$

The augmented phase is the analog of the phase produced in a phase-only SLM, including additionally the amplitude variation of the cross-coupled SLM.

The main outcome of the above result is that for maximum intensity at the correlation plane origin, the necessary condition as defined above should be met.

2.2.4.3 Carlson and Vijayakumar's approach. Carlson and Vijayakumar [22] designed a MACE-SDF for implementation on an arbitrarily constrained SLM. They used the projection function as suggested by Farn and Goodman for mapping an unrealizable filter on the SLM operating curve. They used a relaxation algorithm [15] for updating the real weights of the MACE-SDF. Their simulation results show that the filter performance for non-training images is not satisfactory.

2.2.4.4 Khan and Rajan's approach. Khan and Rajan [24] used a simulated annealing optimization technique to design a composite MACE filter for an arbitrarily constrained filter SLM. They used the following objective function which was minimized by optimization.

$$E(H) = \vec{H}^{\dagger}D\vec{H} + \sum_{i=1}^{N_t} k_{1i}[\text{Re}\{S_i^{\dagger}\vec{H}\} - u_i]^2 + k_2\sum_{i=1}^{N_t}[\text{Im}\{S_i^{\dagger}\vec{H}\}]^2. \tag{2.26}$$

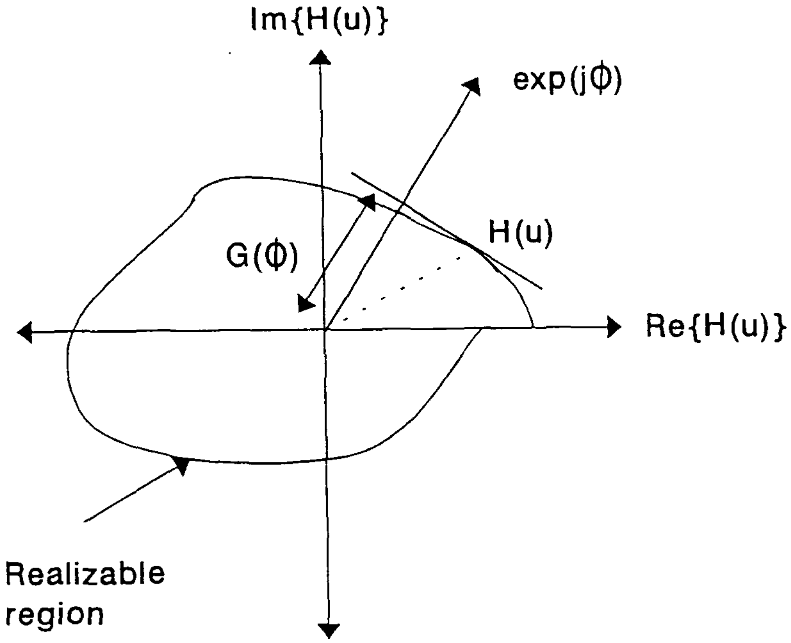

Figure 2.2. The Projection Function [12]

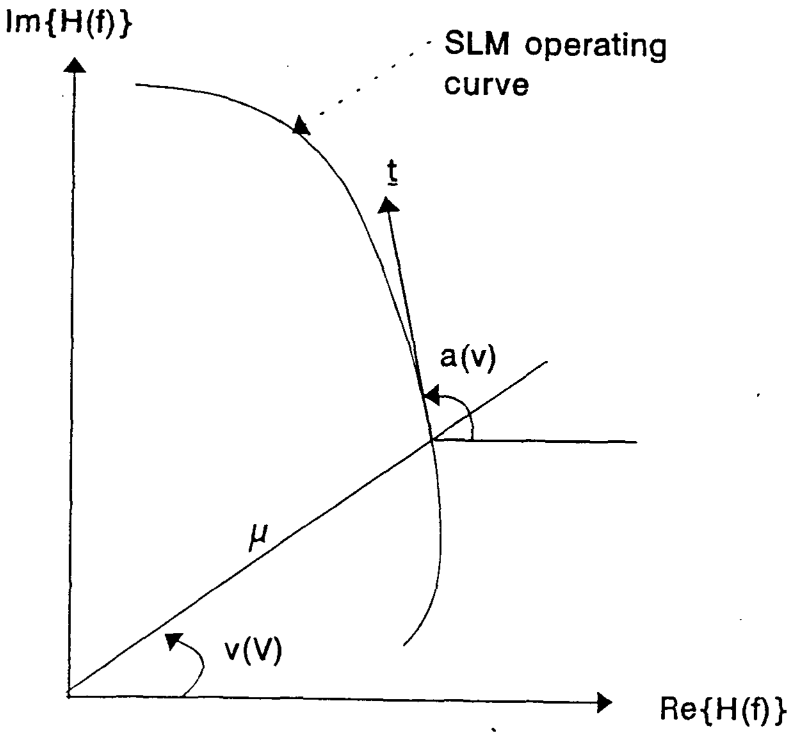

Figure 2.3. Physical Interpretation of an Augmented Phase [11]

The distortion tolerance and the discrimination abilities of the filter are satisfactory. The correlation response of the filter for true class image is sharp at the origin and the correlation plane energy is low. This helps in the easy detection of the peak value in the correlation plane for decision making.

However, the simulated annealing based optimization is computation intensive. Also, the arbitrary values selected for the u_is may not be the best ones for a given SLM. A refinement of the algorithm is proposed in the next chapter.

2.2.4.5 Juday's optimal realizable filters. Juday [14] proposed a unified approach for designing optimal realizable filters for maximizing the metrics of Intensity, SNR, PCE, and PTE. He showed that optimal performance can be obtained for any specified SLM limitation by using the Minimum Euclidean Distance (MED) method of mapping between the optimal filter (unconstrained) and a filter that is achievable with the SLM. The MED mapping minimizes the mean squared error between the optimal filter and the achievable filter responses.

Juday considered the region of realizable complex filter values as the union of its boundary and its interior because different constraints apply to filter values in each. Let the signal be $S(f) = A(f)\exp[j\phi(f)]$, the interior of the filter be $H(f) = M(f)\exp[j\theta(f)]$, and the filter on the boundary be $H(f) = \mu[V(f)]\exp\{jv[V(f)]\}$, where V is a unidimensional parameter which specifies the position along the boundary. The correlation value at the origin is $c(0) = B\exp(j\beta) = \sum_k H_k S_k$.

The following metrics are optimized:

$$\begin{aligned} Intensity &= B^2 && (2.27) \\ SNR &= \frac{B^2}{\sigma_d^2 + \sum_k P_{nk} M_k^2} && (2.28) \\ PCE &= \frac{B^2}{\sum_k A_k^2 M_k^2} && (2.29) \end{aligned}$$

$$PTE = \frac{B^2}{\sigma_d^2 + \sum_k M_k^2(P_{nk} + A_k^2)} \tag{2.30}$$

where σ_d^2 is the variance of the correlation plane detector noise, and P_{nk} is the power spectral density of the additive input noise. The filter H is said to be optimal with respect to a metric if the metric is stationary with respect to the filter value at each frequency, H_k given as

$$\frac{\partial\{Metric\}}{\partial H_k} = 0. \tag{2.31}$$

The optimal filters for SNR and PCE are given below

$$\begin{aligned}
H_{SNR}^{+} &= G_{SNR}\frac{A_m}{P_{nm}}exp[j(\beta - \phi_m)], && (2.32)\\
H_{PCE}^{+} &= G_{PCE}\frac{1}{A_m}exp[j(\beta - \phi_m)], && (2.33)\\
\text{where } G_{SNR} &= \frac{\sigma_d^2 + \sum_k P_{nk}M_k^2}{B}, && (2.34)\\
G_{PCE} &= \frac{\sum_k A_k^2 M_k^2}{B}. && (2.35)
\end{aligned}$$

The design of optimal realizable filters involves a two-dimensional search for the parameters G and β.

The approach proposed by Juday considers only a single image as the reference for filter synthesis. Since the filter design does not consider the possible distortions in the input image while constructing the filter, the designed filters are not distortion invariant. A procedure to design filters that are distortion invariant will be presented in the next chapter.

2.3 Summary

Chapter 2 covered the design details of unconstrained filters and SLM constrained filters. In the unconstrained filters, the concepts of matched filter, SDF filter, and MACE filter were covered. Since the area of interest for this report is on the design of SLM constrained filters for an arbitrarily constrained SLM, Farn and Goodman's method of optimal filter design for intensity maximization, Juday's necessary condition for intensity maximization, and Juday's optimal realizable filter concepts were covered in some detail.

CHAPTER 3
DESIGN OF MACE FILTERS REALIZABLE ON ARBITRARILY CONSTRAINED SLMS

In Section 2.2.4, different techniques for the design of filters realizable on SLMs exhibiting arbitrarily constrained operating characteristics were discussed. Among these, only two methods deal with the design of composite filters for arbitrarily coupled SLMs. One method is due to Carlson and Vijayakumar [22] and the other is due to Khan and Rajan [24]. The drawbacks of the above two methods were also discussed in Chapter 2. In this chapter, some refinements to the simulated annealing method are first presented. Then a new faster algorithm for the design of composite MACE filters for arbitrarily coupled SLMs is presented. In both the methods, the presence of an input SLM is assumed.

3.1 Improved SLM-MACE Design using Simulated Annealing Optimization

In the simulated annealing-based method proposed by Khan and Rajan [24], the peak value constraint is included in the objective function using the penalty function method and the search of the filter values that minimize the objective function is carried out by the simulated annealing technique. The following objective function was minimized using the optimization.

$$E(H) = \vec{H}^{\dagger} D \vec{H} + \sum_{i=1}^{N_t} k_{1i}[Re\{S_i^{\dagger}\vec{H}\} - u_i]^2 + k_2 \sum_{i=1}^{N_t}[Im\{S_i^{\dagger}\vec{H}\}]^2 \qquad (3.1)$$

where k_{1i}'s and k_2 are positive constants greater than unity, ideally infinity and u_i's are user-specified values at origin. As may be seen, the specified value at the origin is $(u_i + j0)$, purely real. However, from the recognition point of view, only intensity is observed and hence phase value is of no concern. It is enough to constrain the magnitude leaving arbitrary phase. This additional degree of freedom is likely to yield a lower objective function resulting in a better filter. Hence in this report the modified objective function given below is used. The function to be minimized is

$$E(H) = \vec{H}^{\dagger} D \vec{H} + \sum_{i=1}^{N_t} k_{1i} \cdot [|S_i^{\dagger} \vec{H}| - u_i]^2. \tag{3.2}$$

The algorithm for the minimization of the above objective function will be similar to that given by Khan [25] and is briefly outlined next. The filter pixel values are the variables to be determined in this optimization. So, the objective function to be minimized is E(V_i), i=1,2,...,d, where V_i's represent the function variables to be determined.

The optimization is started by randomly perturbing the variable V_i. Khan considered the phase values of the filter pixels as V_i's. In an arbitrarily constrained SLM operating curve, a single phase value of SLM may have more than one amplitude value. So, for a given phase value, the amplitude value may not be unique. In the modified method, instead of the phase perturbation, the drive to the SLM corresponding to each pixel value of the filter is perturbed. The energy change ΔE caused by the random perturbation ΔV_i for the i^{th} variable is calculated as $\Delta E = E_{new} - E_{old}$, where $E_{new} = E(V_i^{old} + \Delta V_i)$ and $E_{old} = E(V_i^{old})$. If $\Delta E > 0$, then the perturbation is accepted based on the acceptance probability

$$P(\Delta E) = \frac{1}{1 + exp(\frac{\Delta E}{T})}, \tag{3.3}$$

where T is the temperature parameter. If $\Delta E < 0$ then the perturbation is accepted unconditionally. The technique of accepting higher energy values in this optimization avoids the possibility of system being trapped in local minima.

In the above method, the initial temperature T is chosen to be one-half the initial value of the objective function. The temperature parameter T is lowered exponentially as $T_n = (0.95)^n T_o$, where T_n is the temperature at the n^{th} stage and T_o is the initial temperature. If the desired number of accepted perturbations is not achieved at three successive temperatures, the system is considered to have reached its lowest energy state, and the optimization is concluded.

A Fortran program implementing the above algorithm is given in Appendix C. Performance of the filters designed using this program is given in Chapter 4.

3.2 An Iterative Method for the Design of Realizable MACE Filters

As pointed out earlier, the simulated annealing-based optimization takes the SLM constraints and peak value constraints simultaneously and minimizes the energy in a single step. However this results in a time-consuming process. An alternative approach will be to perform the design in two steps: (i) design a realizable MACE filter with equal peak values and (ii) map the designed filter on an SLM. Perform the two steps repeatedly so that the final filter is an SLM realizable optimal filter. The conventional MACE filter design is achieved using the formula

$$\vec{H}_{MACE} = D^{-1}S(S^{\dagger}D^{-1}S)^{-1}\vec{c}. \tag{3.4}$$

However this filter as stated earlier is not guaranteed to be implementable on a given SLM. In the following an iterative method which will yield a filter implementable on an SLM is developed. First the iterative method for an unconstrained

MACE filter is developed and then it is modified for the design of a constrained MACE filter.

3.2.1 Design of an Unconstrained MACE Filter

Let $\vec{s}_i = [s_i(1) \quad s_i(2) \quad \dots s_i(d)]^T$ represent the i^{th} training image in the set of N_t training images each with d pixels. All the training images are affected by the input SLM's operating limitations. Normally, the SLM's operating characteristic is available to the filter designer as a table of pairs of values representing phase and magnitude. The gray scale value for each pixel of the training image is transformed to the corresponding input SLM's phase and magnitude. The transformed training images are then used in the synthesis of the MACE filter. The 2-D DFT of the transformed image $\vec{\check{s}}_i$ is expressed in vector form as $\vec{\check{S}}_i = [\check{S}_i(1) \quad \check{S}_i(2) \quad \dots \check{S}_i(d)]^T$. Let S be a matrix of size $d \times N_t$ formed by DFT vectors of N_t training images, given as

$$S = [\vec{\check{S}}_1 \quad \vec{\check{S}}_2 \quad \dots \check{S}_{N_t}]. \tag{3.5}$$

Then the MACE filter is given as

$$\vec{H}_{MACE} = D^{-1}S(S^{\dagger}D^{-1}S)^{-1}\vec{c}, \tag{3.6}$$

where D is a diagonal matrix of size $d \times d$, the diagonal elements of which are given by

$$D(k,k) = \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{|\vec{\check{S}}_i(k)|^2}{d}, \tag{3.7}$$

and $\vec{c} = [c_1(0) \quad c_2(0) \quad \ldots c_{N_t}(0)]^T$ is the output correlation vector. $c_i(0)$ is the desired correlation value at the origin when the i^{th} image is passed through $\vec{H}_{MACE}$.

From Equation (2.14), the MACE filter is given as

$$\begin{aligned} \vec{H}_{MACE} &= D^{-1/2}\vec{H}_{SDF} \\ &= \bar{S}\vec{a} \end{aligned} \tag{3.8}$$

where $\bar{S} = D^{-1/2}S$. The MACE filter is nothing but the conventional SDF for prewhitened images. The unknown weight vector $\vec{a} = [a_1 \quad a_2 \quad \ldots a_{N_t}]^T$ is obtained by using a relaxation algorithm similar to the one suggested by [15] as discussed below.

3.2.1.1 Relaxation Algorithm to Obtain the Weight Vector

The steps involved in this algorithm are given below:

1. Let k=0; Select a set of initial values for the elements of $\vec{a^o}$.
2. Construct the MACE filter

$$\vec{H}^k_{MACE} = \bar{S}\vec{a}^k, \tag{3.9}$$

where the superscript k represents the k^{th} step in the iteration.

3. Calculate the correlation value at the origin for all the training images. The correlation vector is $\vec{m} = [m_1 \quad m_2 \quad \ldots m_{N_t}]^T$, where m_i is the correlation value at the origin when the i^{th} image is correlated with $\vec{H}_{MACE}$. The correlation vector is formed as

$$\vec{m}^k = \bar{S}^\dagger \vec{H}^k_{MACE}. \tag{3.10}$$

4. Update the weight vector $\vec{a}^k$. The individual elements of $\vec{a}^k$ are updated as given below:

$$a_i^{k+1} = a_i^k + \alpha \cdot \left(\frac{c_i}{c_1} - \frac{\left|m_i^k\right|}{\left|m_1^k\right|} \right), \quad i = 1, 2, \ldots N_t, \tag{3.11}$$

where c_i is the desired value at the correlation plane origin for the i^{th} training image. Normally, for true class images c_i' s are chosen to be 1. α is a user selected damping parameter chosen to ensure the convergence of the algorithm.

5. Check whether $\vec{a}^{k+1}$ is approximately equal to a^k. If so stop; otherwise increment k by 1. Go to Step 2.

The $\vec{H}_{MACE}$ obtained at the end of the iteration is a complex valued filter and cannot therefore be accommodated on constrained SLMs.

3.2.2 Design of SLM Constrained MACE Filter

The unconstrained MACE filter obtained in the previous section is used as the start value for the constrained filter design. It may be easily verified that if $\vec{H}_{MACE}$ is an optimal filter (in the sense of maximizing the peak to correlation energy ratio (PCER)), then filter $G\vec{H}_{MACE}e^{j\beta}$ will also be optimal because $e^{j\beta}$ does not affect the peak magnitude and G does not affect PCER.

Following Juday's method of optimal filter expression [14] , the MACE filter is expressed as

$$\vec{H}_{MACE}^{+} = G \cdot \vec{H}_{MACE} \cdot e^{j\beta} \tag{3.12}$$

and G is assumed to be real whose range is unknown and β is limited to $[0,2\pi)$. The superscript + denotes an optimal but unrealizable filter. To convert $\vec{H}^{+}_{MACE}$ into a realizable filter $\vec{H}^{\circ}$, one may use minimum euclidean distance (MED) method suggested by Juday [14]. However, different values of G and β will yield different H° and each will have different performance functions. To obtain the best possible filter, one has to carry out a two-dimensional search on the G-β space such that the objective function given below is maximized.

$$\Phi = \sum_{i=1}^{N_t} \frac{\left|\vec{S}_i^{\dagger} \vec{H}^{\circ}_{MACE}\right|^2}{\vec{H}^{\circ\dagger} D_i \vec{H}^{\circ}} \tag{3.13}$$

where $\vec{H}^{\circ}$ is the realizable MACE filter obtained after mapping $\vec{H}^{+}$ on the SLM operating curve by MED method suggested by Juday [14]. To perform the G and β search efficiently, one may use direct search methods or gradient based methods. In this report direct search methods are used. The reasons for selecting the direct search methods are discussed next.

3.2.2.1 Direct Search Methods

Many gradient-based algorithms exist in literature for multivariable optimization. Many multivariable problems in engineering have objective functions that are mathematically complex or are based on tabular data. For these optimization problems, finding the partial derivatives required to calculate the gradient is often impossible. For these problems, it is necessary to use a search algorithm that does not depend on calculation of the gradient. One type of technique that satisfies this requirement is the pattern search method. "Pattern search methods are known for their simplicity and are popular in the optimization field because their performance is often better than methods that rely on gradient calculation or gradient approximation. Pattern search algorithms have been demonstrated to be far superior to gradient methods when used on merit surfaces that have sharply defined ridges as a result of the imposed constraints [38]." Two methods are popular in pattern search techniques because of their ease of implementation and fast convergence; they are : (1) The Simplex method, and (2) Hooke and Jeeves method. In this report, both these methods were independently used for performing a two variable search to find the best G and β, for the filter such that the objective function is maximized.

3.2.2.1.1 Simplex method of optimization. It is to be noted here that the simplex method in pattern search is different from Dantzig's simplex method in linear programming problems. The Simplex method was proposed by Nelder and Mead in 1965 [27]. The simplex is an n-dimensional, closed geometric figure in space that has straight line edges intersecting at n+1 vertices. In two dimensions this figure would be a triangle. In three dimension it would be a tetrahedron. Since the optimization of the objective function (sum of PCERs) involves only two unknowns, G and β, the simplex is a triangle.

The idea of the method is to compare the values of the function at the (n+1) vertices of the simplex and move the simplex towards the optimum point during the iterative process. The movement of simplex is achieved by the application of three basic operations *reflection, expansion*, and *contraction.* The above mentioned simplex method gives the best values for G and β denoted as $G^\diamond$ and $\beta^\diamond$, respectively.

3.2.2.1.2 Hooke and Jeeves method of optimization. The algorithm [28] proceeds as follows. First, a base point is chosen with exploration step sizes. Next, an exploration is performed with a given increment along each of the independent variable directions. A new temporary base point is established whenever there is a functional improvement. Once this exploration is complete, a new base point is established and a "pattern move " takes place. This pattern move consists of an extrapolation along a line between a new base point and the previous base point. Once the new temporary base point has been found, an exploration about this point is instituted to see if a better base point can be found. Using this optimization technique, the $G^\diamond$ and $\beta^\diamond$ values are found.

3.2.3 Relaxation Algorithm to Satisfy SDF Constraint at the Origin

The objective function chosen for the filter design does not guarantee equal correlation values at the origin for all training images. So, an iterative technique is used to satisfy the above requirement. The steps in the algorithm are given below:

1. Let k=0. Use $G^\diamond$ and $\beta^\diamond$ values obtained after Simplex or Hooke and Jeeves optimization.
2. Construct the filter with $G^\diamond$ and $\beta^\diamond$ as

$$\vec{H}^k = G^\diamond \cdot (\bar{S}\vec{a}^k) \cdot e^{j\beta^\diamond}. \tag{3.14}$$

3. Use the MED method of mapping to map the unrealizable filter $\vec{H}^k$ on to the SLM operating curve. The mapped filter is $\vec{H}^\circ$.
4. Update the weight vector $\vec{a}$. The elements of $\vec{a}$ are updated as given below:

$$a_i^{k+1} = a_i^k + \alpha \cdot \left(\frac{c_i}{c_1} - \frac{|m_i^k|}{|m_1^k|} \right) \tag{3.15}$$

where $\vec{m}^k = \bar{S}^\dagger \vec{H}^{\circ k}$ and k represents the k^{th} iteration step.

5. The iteration is repeated from Step 2, until the elements of vector $\vec{a}$ are $a_i^{k+1} \approx a_i^k$.

The simplex or Hooke and Jeeves method of optimization and the relaxation algorithm are repeated until no changes in $\vec{a}$ occur during the relaxation algorithm. After the above iteration, the filter $\vec{H}^{\circ}$ is the realizable MACE filter.

3.3 Summary

This chapter discussed a modified simulated annealing optimization-based MACE filter design. Then it presented an iterative method without requiring a matrix inverse for the design of an unconstrained MACE filter. Following this, this chapter presented the steps involved in the design of an unconstrained MACE filter in the presence of an input SLM, the formulation of the objective function, the simplex and Hooke and Jeeves methods of optimization to find the values $G^{\diamond}$ and $\beta^{\diamond}$ which maximized the objective function, and the relaxation method of iterative technique to get equal values at the correlation plane origin for all the training images. Performance of the filters designed using the above methods with respect to real world SLMs in pattern recognition problems is presented in the next chapter.

CHAPTER 4
PERFORMANCE EVALUATION OF THE REALIZABLE MACE FILTER

In Chapter 3, a new approach for the design of a realizable MACE filter was discussed. An algorithm for its construction was also presented. Khan and Rajan's [24] simulated annealing-based MACE filter design was modified and the updated algorithm was also discussed. In this chapter the distortion tolerance and the discrimination ability properties of the designed filter are studied using computer simulations.

4.1 Simulation

The realizable MACE filter was designed on a computer (SUN SPARC station 2) using three different algorithms (simplex method, Hooke and Jeeves method, and simulated annealing-based optimization) described in Chapter 3. The filter was synthesized using seven training images of space shuttle (s128.15n, s128.10n, s128.5n, s128.0, s128.5, s128.10, s128.15) where s128.15n represents the shuttle image rotated by 15 degrees in the clockwise (negative) direction and s128.15 represents 15 degrees anti-clockwise rotated image. A space shuttle image is shown in Figure 4.1. These images are of size 128×128 pixels and were obtained by padding a 32×32 image with zeros. The above seven images are considered to belong to the true class and represent possible distortions in the distortion range.

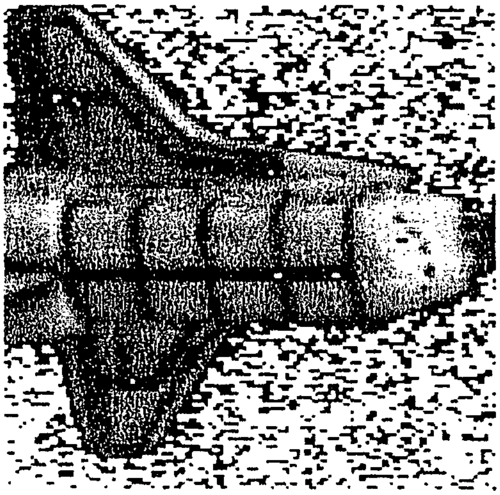

Figure 4.1. The Space Shuttle Image

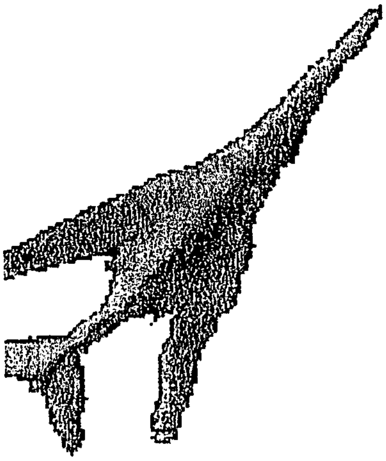

Figure 4.2. The Airplane Image

The distortion considered here are in-plane rotations. Airplane images were used to test the discrimination ability of the filters for false class images. A typical airplane image is shown in Figure 4.2. To estimate the filter's ability to recognize the out-of-plane rotated images, a test with truck images was performed and the results are also presented in this chapter. The database used to test the recognition and distortion tolerance of the filters consisted of thirty-one images in each of the shuttle class and the airplane class. These images were obtained by rotating an image in each class from minus fifteen degrees to plus fifteen degrees in steps of one degree. The shuttle and airplane image databases are shown in Figure 4.3 and Figure 4.4, respectively.

LCTV SLMs are available which exhibit different operating characteristics, namely, phase-mostly, high-contrast, highly-coupled, and amplitude-only. The magnitude and phase relationship of the LCTV SLM characteristics are shown in Figures 4.5 through 4.8.

4.1.1 Unconstrained MACE Filter in the Presence of an Input SLM

An unconstrained MACE filter was designed using the relaxation algorithm discussed in Section 3.1.1. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the space shuttle class were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.1. Figure 4.9 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The maximum magnfor shuttle

images are represented with a □ and airplane images are represented with ×.

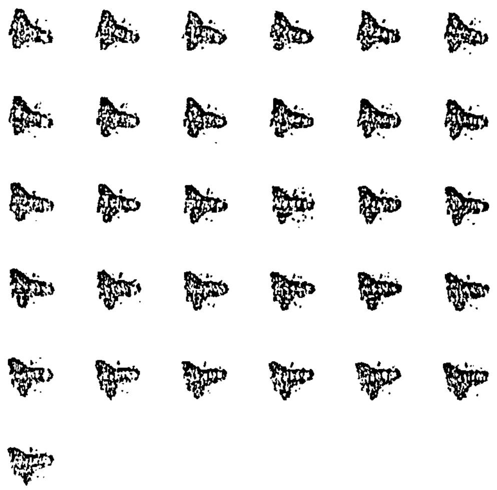

Figure 4.3. The Shuttle Image Database

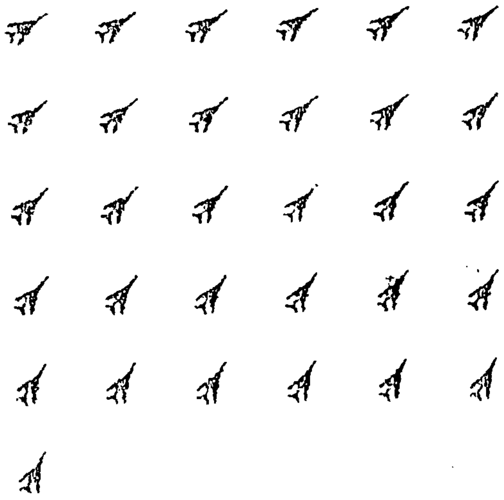

Figure 4.4. The Airplanc Imagc Database

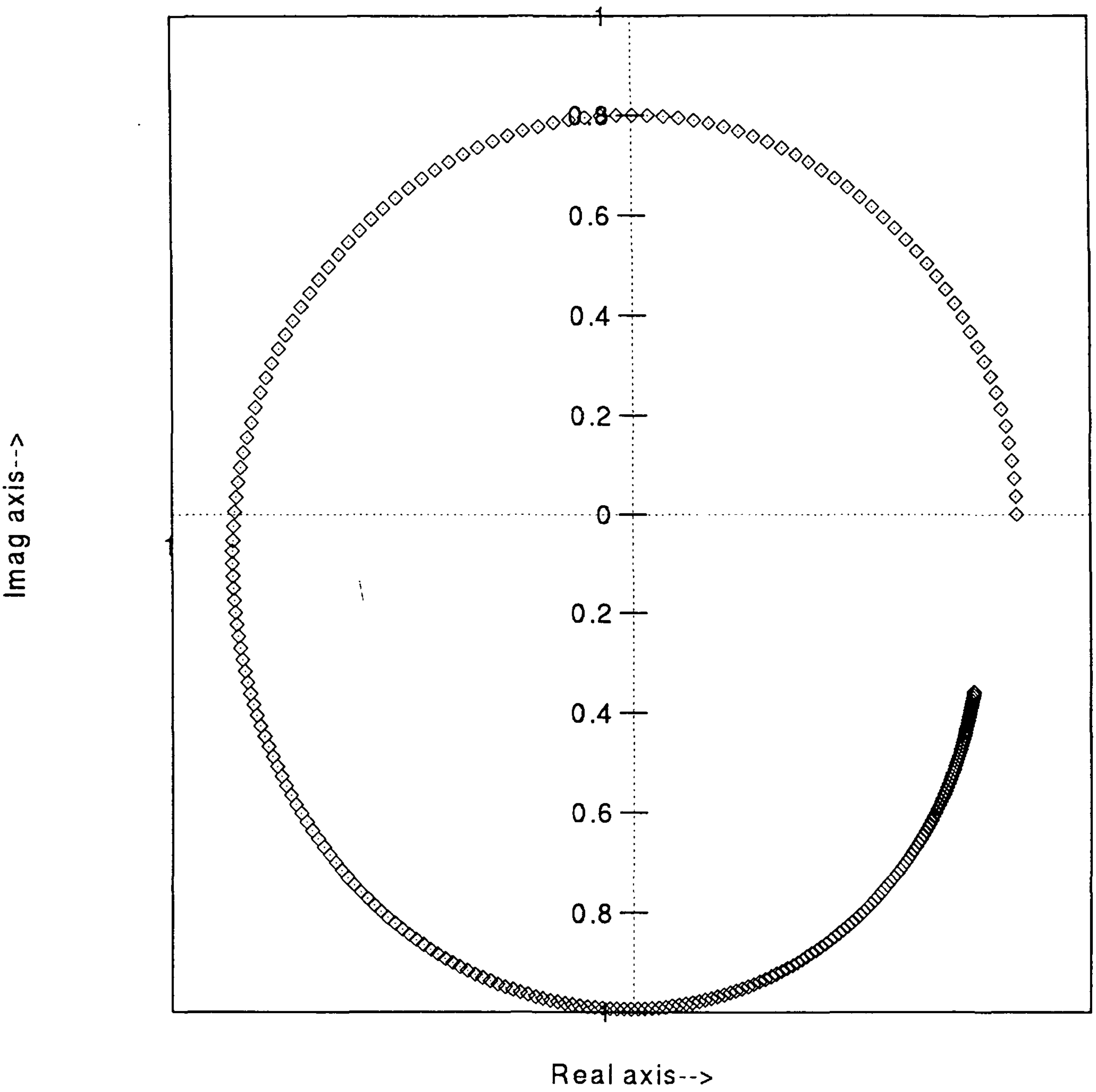

Figure 4.5. Phase-Mostly Characteristic

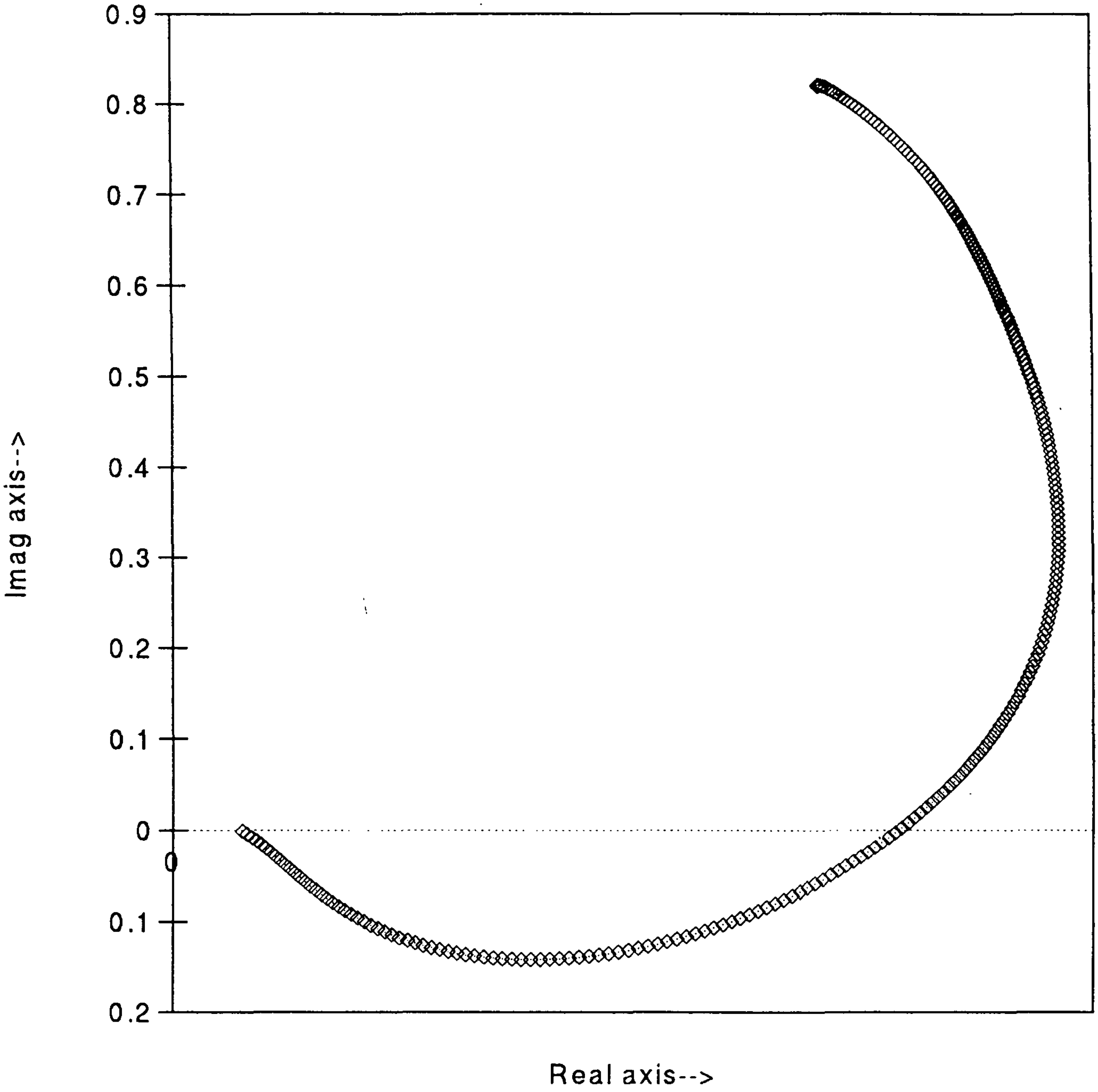

Figure 4.6. High-Contrast Characteristic

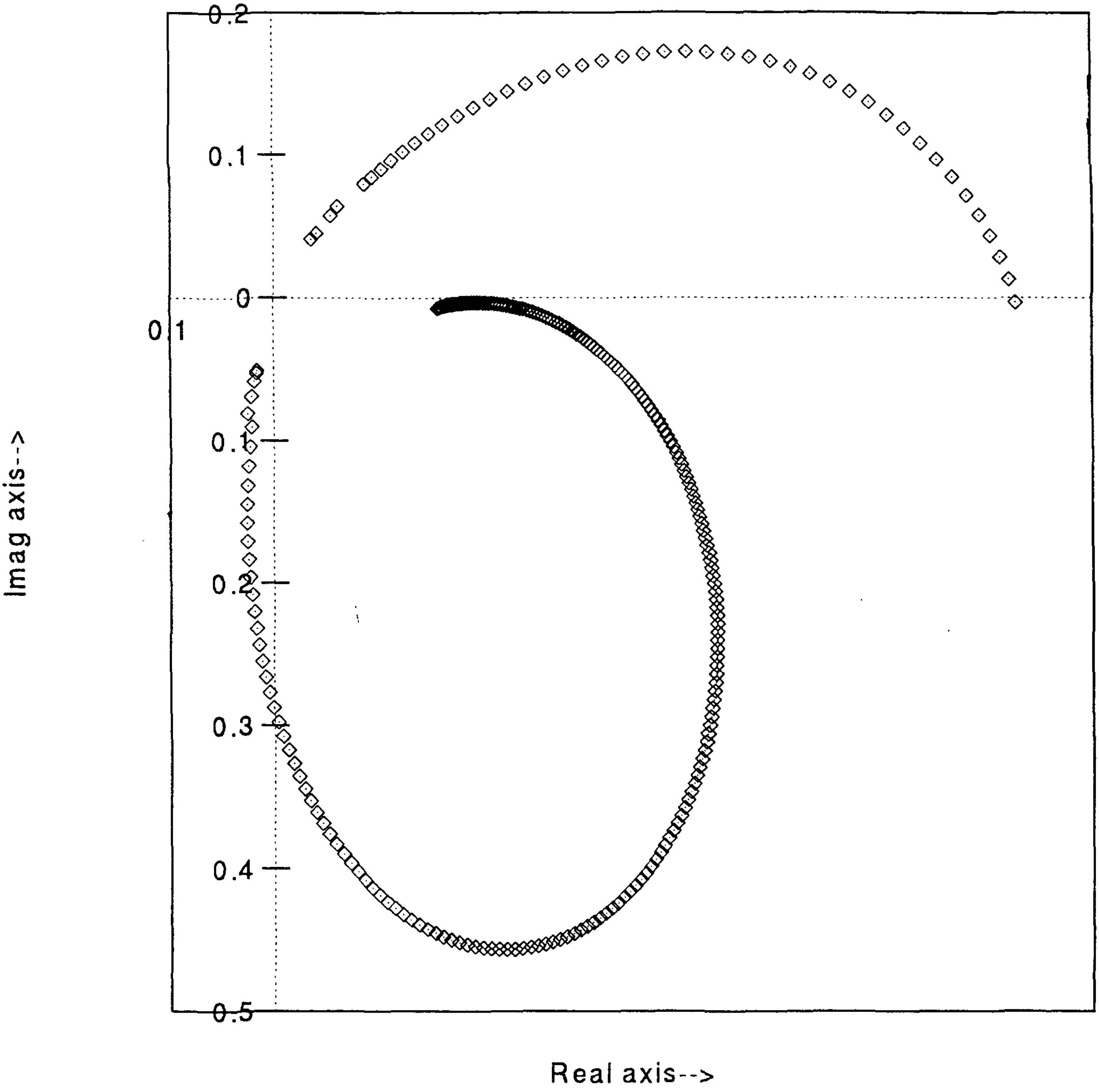

Figure 4.7. Highly-Coupled Characteristic

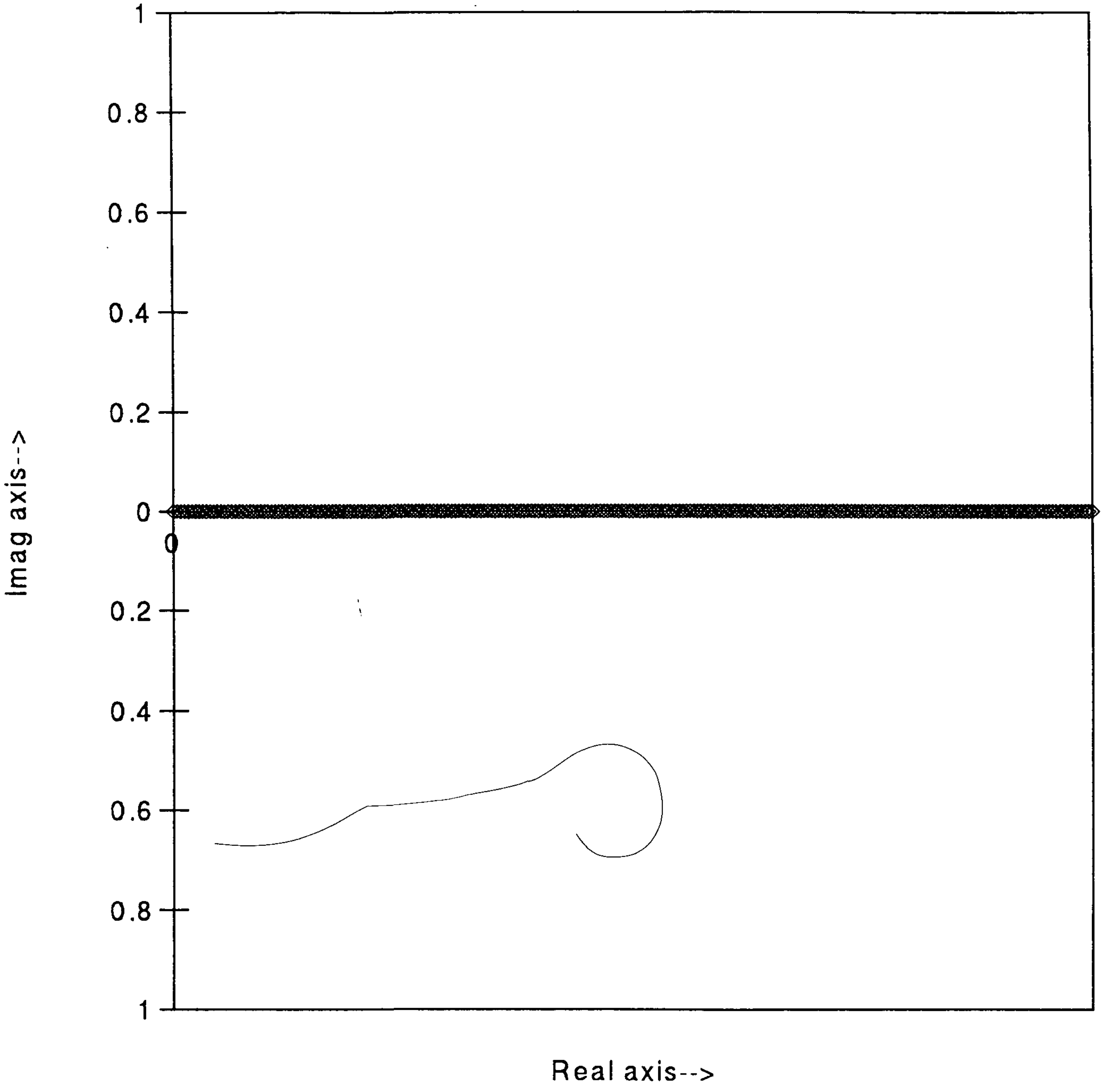

Figure 4.8. Amplitude-Only Characteristic

Table 4.1. Correlation Plane Statistics for an Unconstrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	22419.0	22419.0	22419.0	22418.9	22419.1	22419.0	22419.0
Energy	119688.0	118438.0	117873.0	177389.0	117569.0	119228.0	117924.0
PCE	4199.3	4243.6	4338.3	2833.3	4275.0	4215.5	4262.1
PSR	5.65	5.65	4.85	4.37	4.44	5.57	5.07

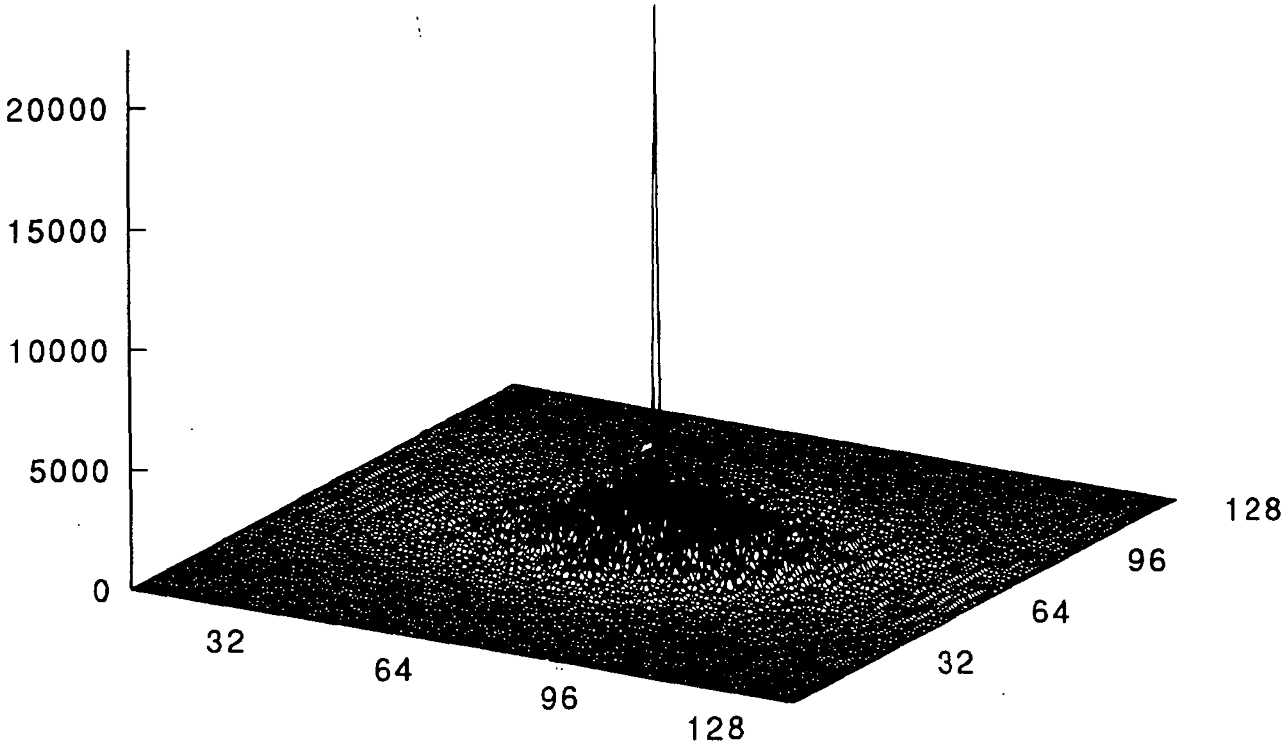

Figure 4.9. The Output Correlation Plane for the Unconstrained MACE Filter

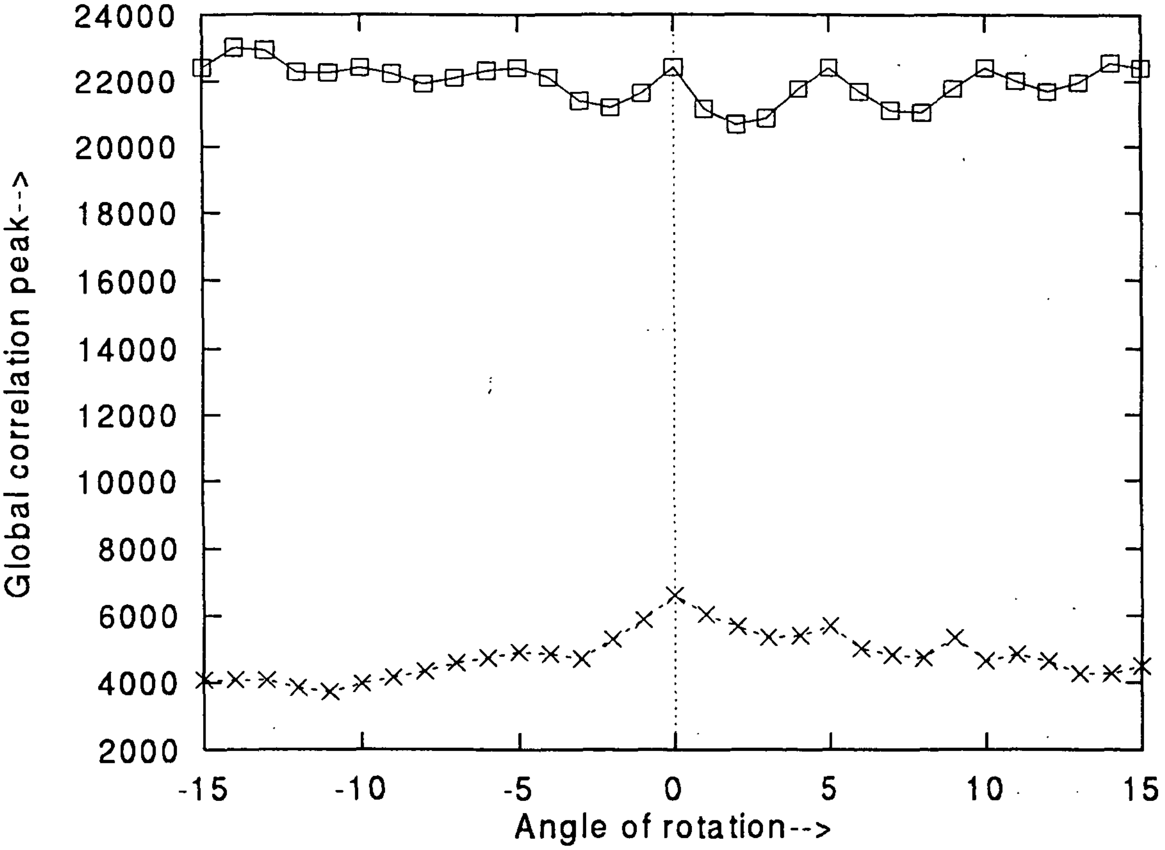

Figure 4.10. Distortion Test Results for the Unconstrained MACE Filter

4.2 Performance of a Simplex-based MACE Filter

The algorithm for the design of an SLM-constrained MACE filter using the simplex method of optimization was discussed in Chapter 3. The computer simulation results are presented below.

4.2.1 Constrained MACE Filter for Phase-Mostly Operating Curve

An unconstrained MACE filter was designed using the relaxation algorithm as discussed in Section 3.1.1. Then using the simplex method of optimization, the objective function was maximized for the Phase-Mostly filter SLM characteristic shown in Figure. 4.5. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the true class (space shuttle) were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.2. Figure 4.11 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two-dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.12. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.2. Correlation Statistics for Phase-Mostly SLM Constrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	12777.6	12828.7	12799.0	12775.3	12771.7	12781.3	12756.8
Energy	154694.0	156246.0	157471.0	176920.0	158033.0	157081.0	156541.0
PCE	1055.4	1053.3	1040.2	922.4	1032.1	1039.9	1039.5
PSR	2.21	2.83	2.47	1.88	2.54	2.76	2.11

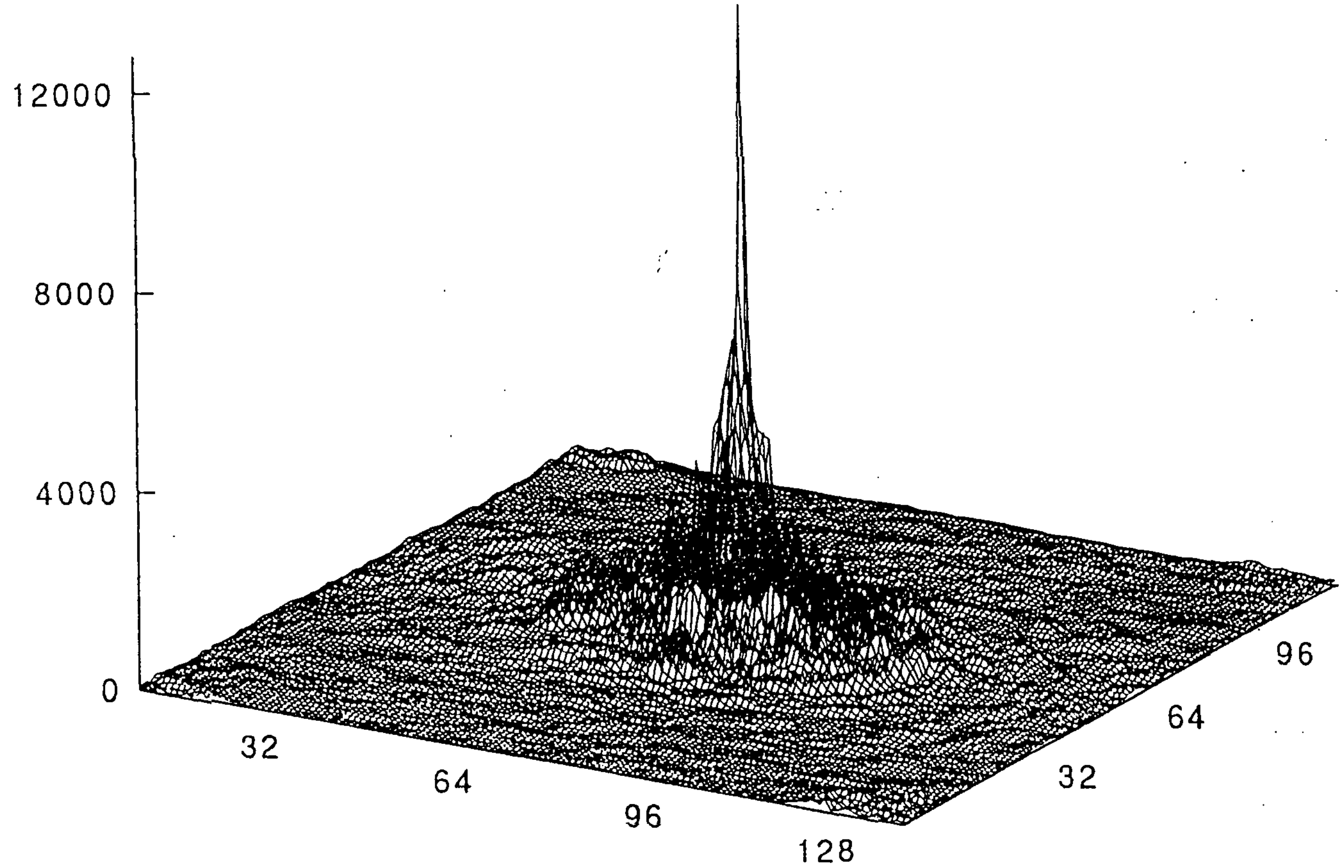

Figure 4.11. The Output Correlation for Phase-Mostly SLM Constrained MACE Filter

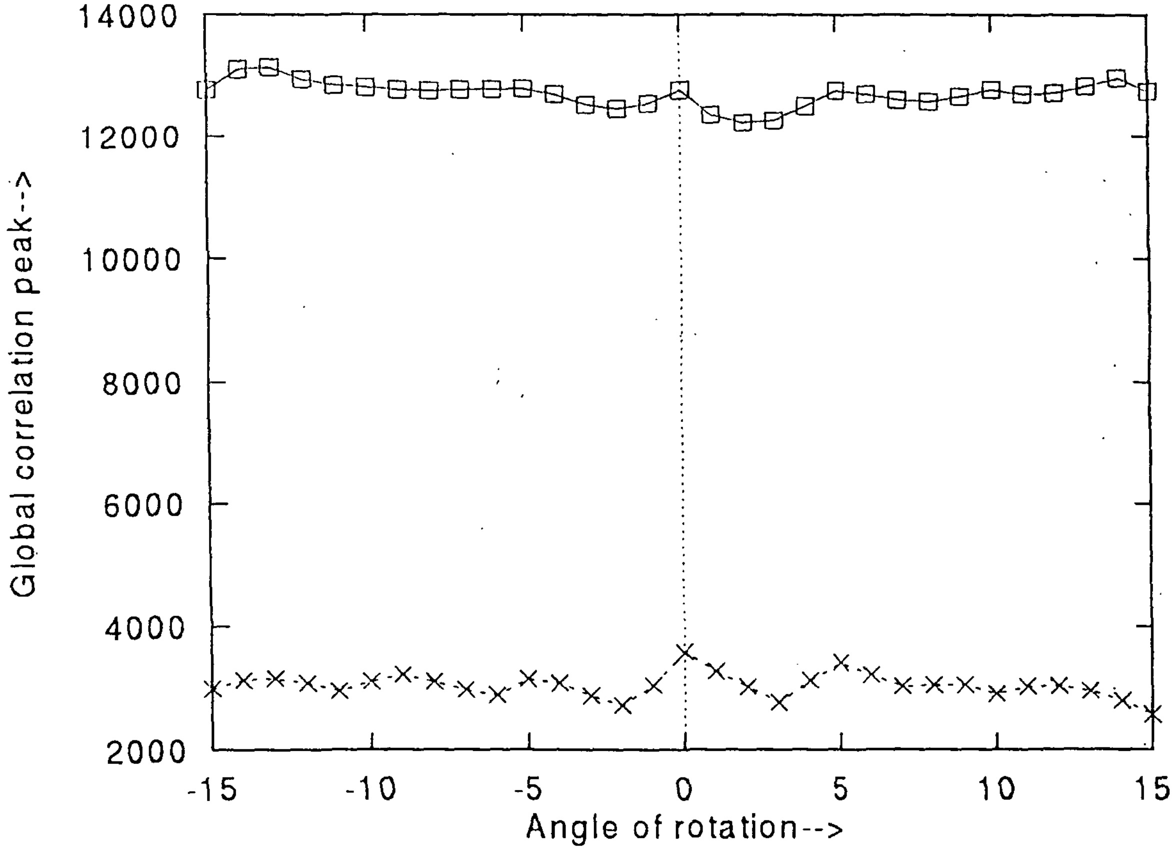

Figure 4.12. Distortion Test Results for Phase-Mostly SLM Constrained MACE Filter

4.2.2 Constrained MACE Filter for High-Contrast Operating Curve

An unconstrained MACE filter was designed using the relaxation algorithm as discussed in Section 3.1.1. Then using the simplex method of optimization, the objective function was maximized for the High-Contrast filter SLM characteristic shown in Figure 4.6. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the true class (space shuttle) were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.3. Figure 4.13 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in the shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.14. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.3. Correlation Statistics for High-Contrast SLM Constrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	3261.8	3259.9	3261.1	3256.1	3262.9	3266.1	3267.5
Energy	10326.2	10495.6	10666.4	16833.1	11379.9	11247.2	11035.1
PCE	1030.3	1012.5	997.0	629.8	935.5	948.5	967.5
PSR	2.46	2.37	3.25	2.57	2.67	2.66	2.86

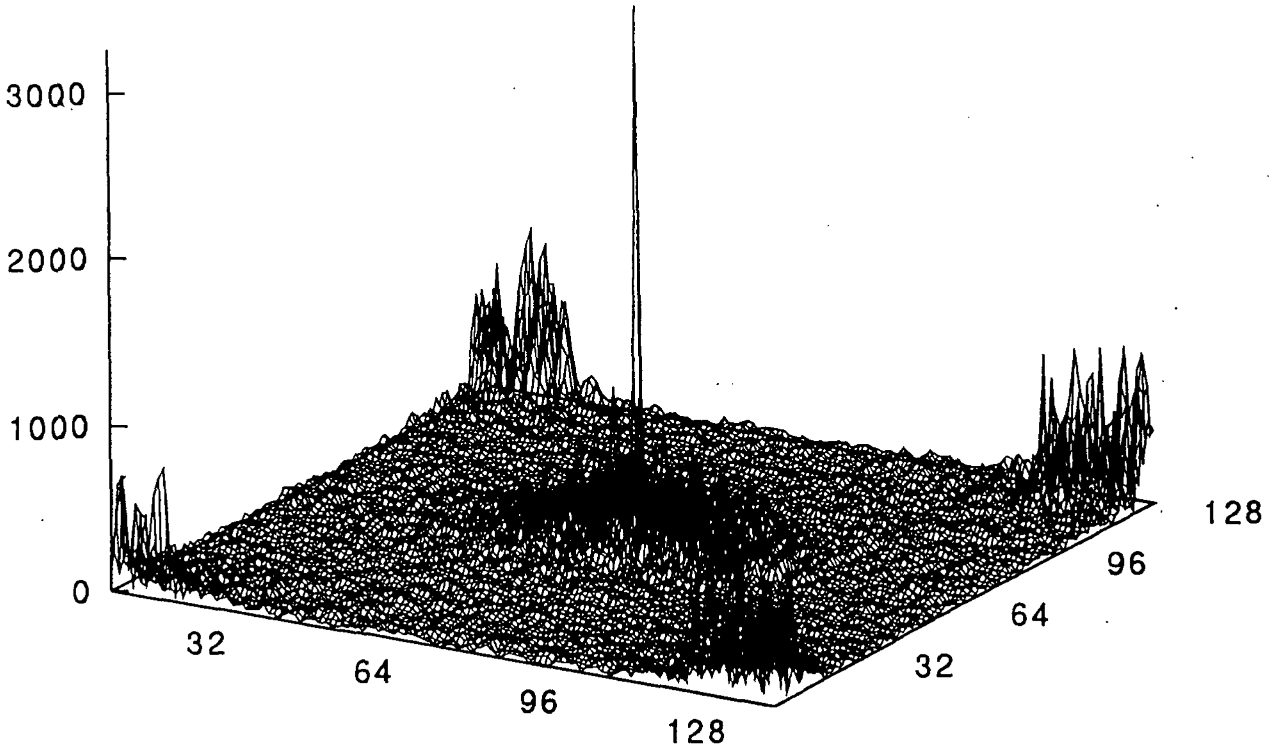

Figure 4.13. The Output Correlation for the High-Contrast SLM Constrained MACE Filter

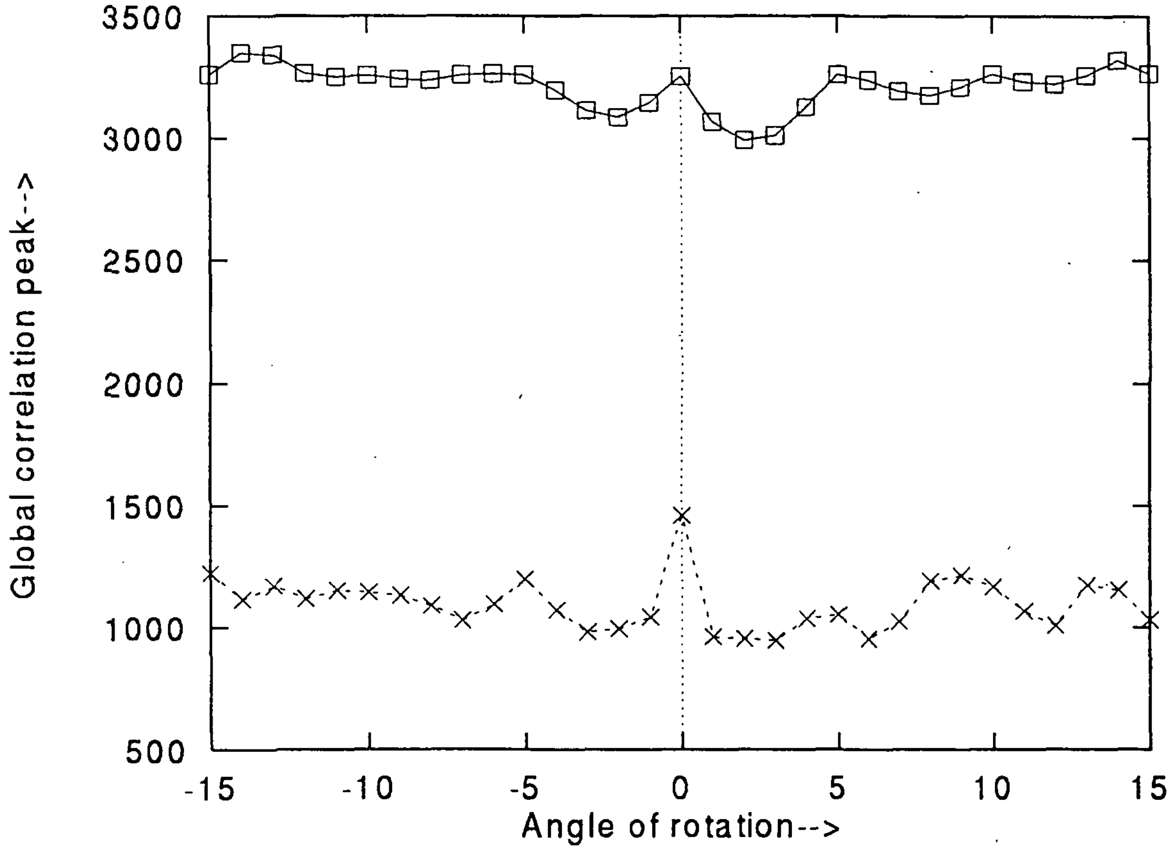

Figure 4.14. Distortion Test Results for the High-Contrast SLM Constrained MACE Filter

4.2.3 Constrained MACE Filter for Highly-Coupled Operating Curve

The filter was designed for the Highly-Coupled LCTV operating curve shown in Figure 4.7. As done before, the input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the true class (space shuttle) were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.4. Figure 4.15 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two-dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in the shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.16. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.4. Correlation Statistics for Highly-Coupled SLM Constrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	2612.2	2614.8	2614.16	2616.7	2610.2	2613.0	2606.2
Energy	4966.5	5036.6	5012.8	8113.0	5431.19	5431.6	5323.5
PCE	1373.8	1357.5	1363.2	843.9	1254.5	1257.1	1275.9
PSR	3.6	3.11	3.62	2.79	3.09	3.92	3.86

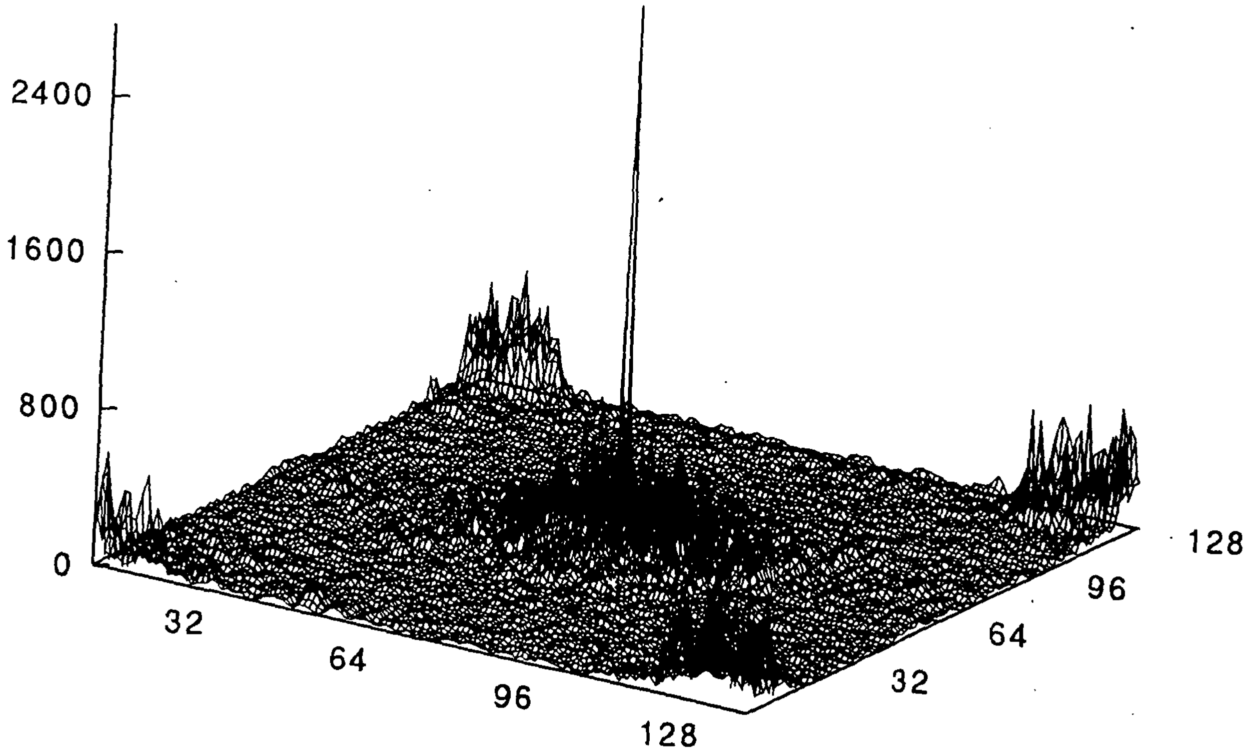

Figure 4.15. The Output Correlation for the Highly-Coupled SLM Constrained MACE Filter

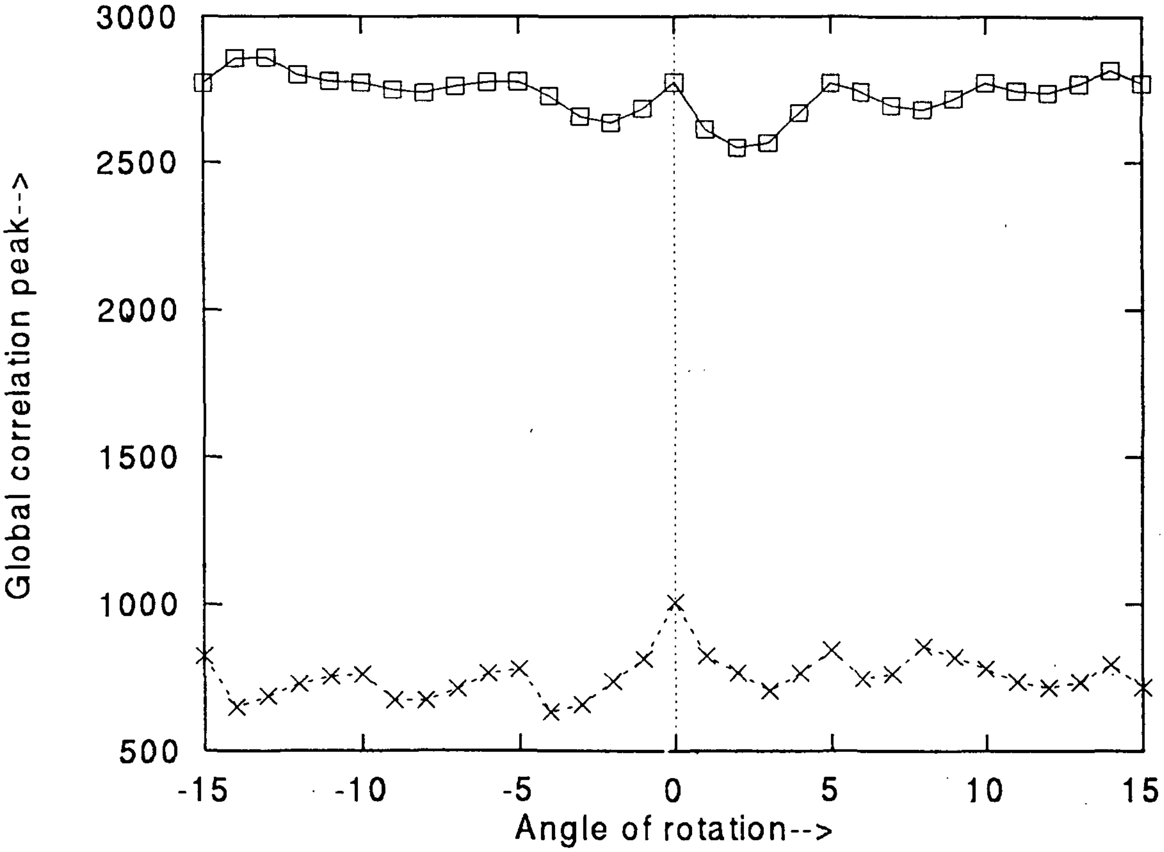

Figure 4.16. Distortion Test Results for the Highly-Coupled SLM Constrained MACE Filter

4.2.4 Filter for High-Contrast Input and Highly-Coupled Filter SLMs

The filter was designed for the Highly-Coupled LCTV operating curve (Figure 4.7) using the simplex method of optimization. The input images were constrained by the High-Contrast operating characteristic shown in Figure 4.6. Seven training images from the shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.5. Figure 4.17 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two-dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in the shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.18. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.5. Correlation Statistics for High-Contrast Input and Highly-Coupled Filter SLMs Constrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	5495.9	5494.2	5498.0	5492.4	5498.9	5507.3	5496.5
Energy	24385.9	25193.3	25820.7	37278.0	26946.8	26244.5	25703.3
PCE	1238.6	1198.2	1170.7	809.2	1122.2	1155.6	1175.4
PSR	3.06	2.93	3.79	2.95	3.15	3.35	3.84

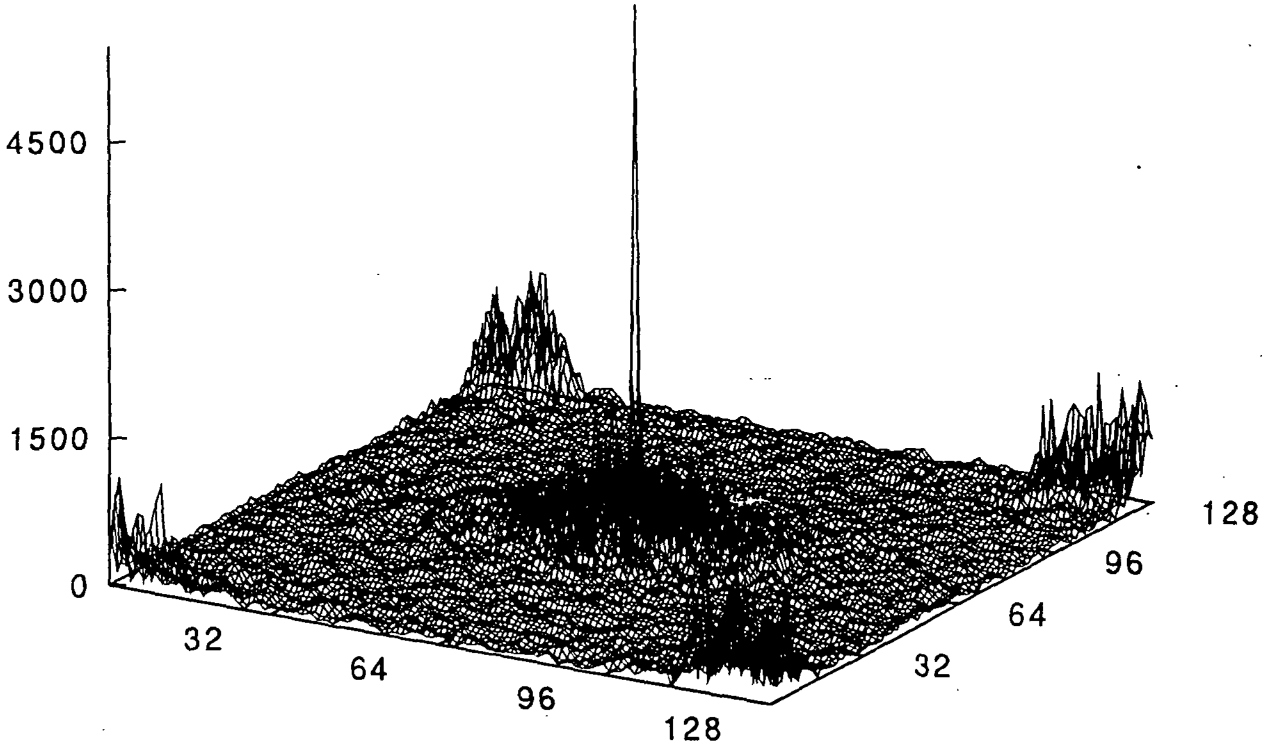

Figure 4.17. The Output Correlation for the High-Contrast Input and Highly-Coupled SLMs Constrained MACE Filter

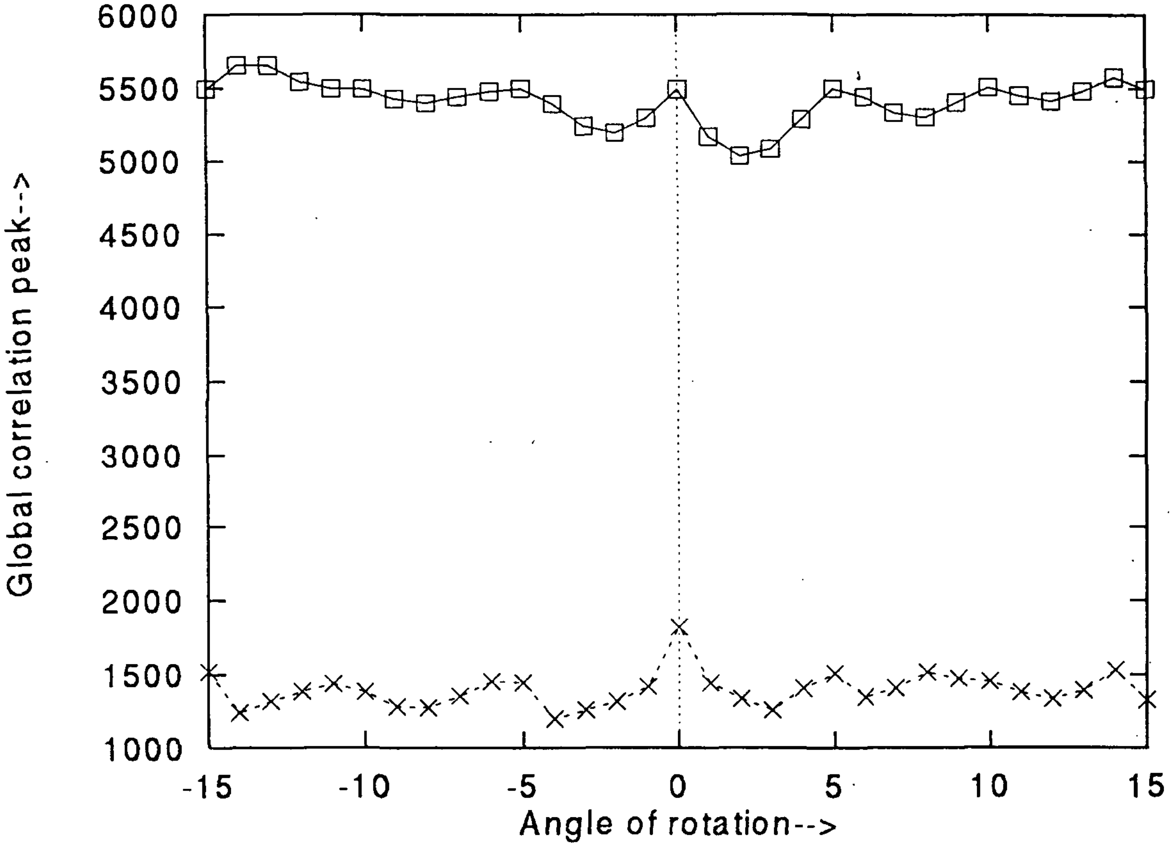

Figure 4.18. Distortion Test Results for the High-Contrast Input and Highly-Coupled SLMs Constrained MACE Filter

4.2.5 Filter for High-Contrast Input and Phase-Mostly Filter SLMs

The filter was designed for the Phase-Mostly filter SLM using the input images which were constrained by the High-Contrast operating characteristic. Seven training images from the space shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.6. Figure 4.19 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two-dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in the shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.20. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.6. Correlation Statistics for High-Contrast Input and Phase-Mostly Filter SLMs Constrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	24066.5	24095.2	24082.0	24068.5	24086.4	24081.6	24070.2
Energy	1220000	1230480	1230000	1306002	1230000	1230011	1230000
PCE	473.3	471.8	469.2	443.3	468.1	469.0	469.4
PSR	2.21	2.57	2.47	1.90	2.59	2.59	2.83

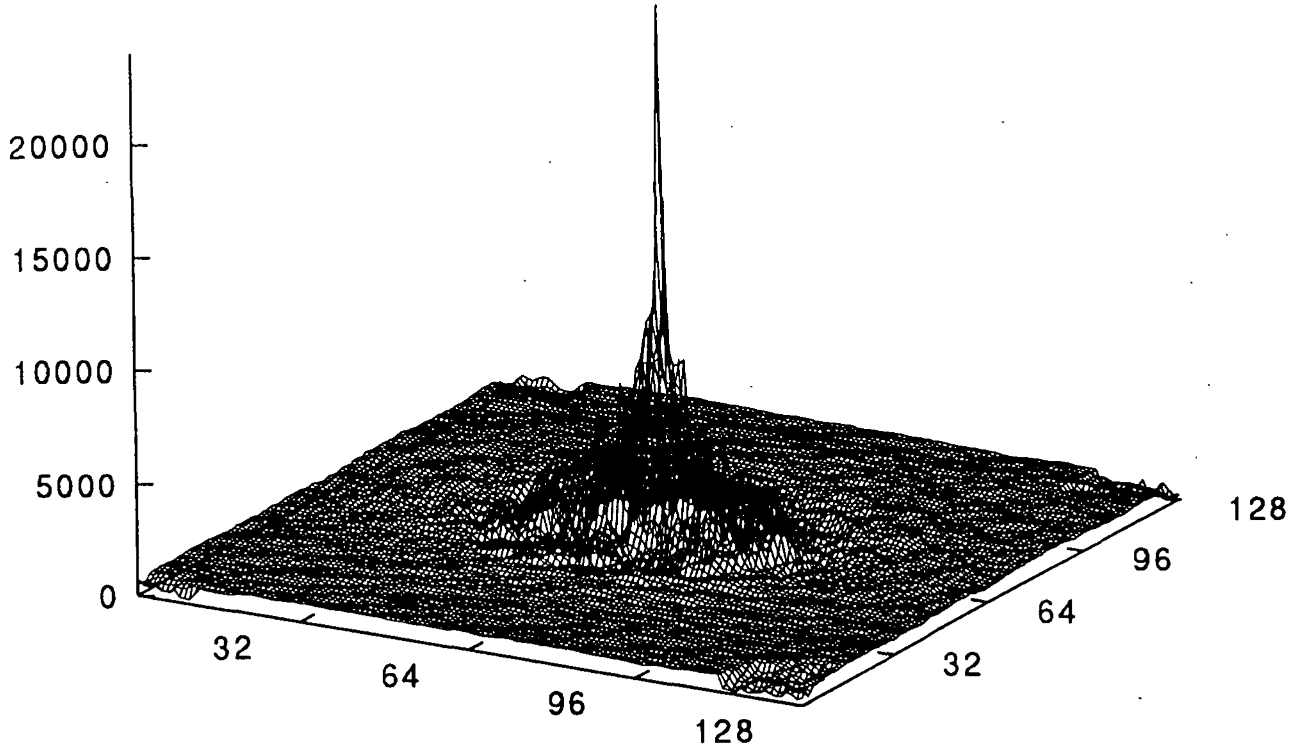

Figure 4.19. The Output Correlation for the High-Contrast Input and Phase-Mostly Filter SLMs Constrained MACE Filter

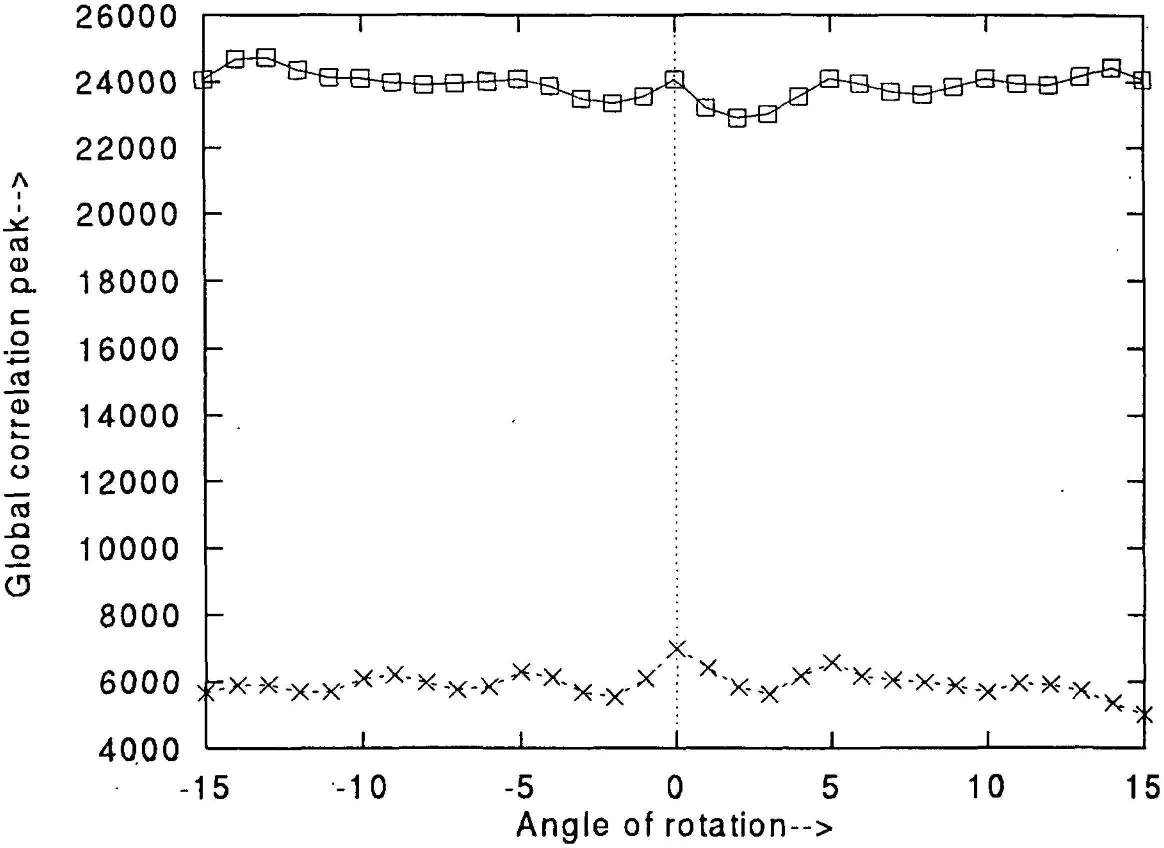

Figure 4.20. Distortion Test Results for the High-Contrast Input and Phase-Mostly Filter SLMs Constrained MACE Filter

4.2.6 Filter for Phase-Mostly Input and Phase-Mostly Filter SLMs

The filter was designed for the Phase-Mostly filter SLM using the input images which were constrained by the Phase-Mostly operating characteristic. Seven training images from the true class (space shuttle) were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.7. Figure 4.21 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two-dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in the shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.22. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.7. Correlation Statistics for Phase-Mostly Input and Phase-Mostly Filter SLMs Constrained MACE Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10
Peak	107625	107902	107726	107608	107753	107755
Energy	122241000	122256000	122268000	122388000	122278000	122277000
PCE	94.8	95.2	94.9	94.6	94.9	94.9
PSR	2.4	2.4	2.6	1.90	2.4	2.3

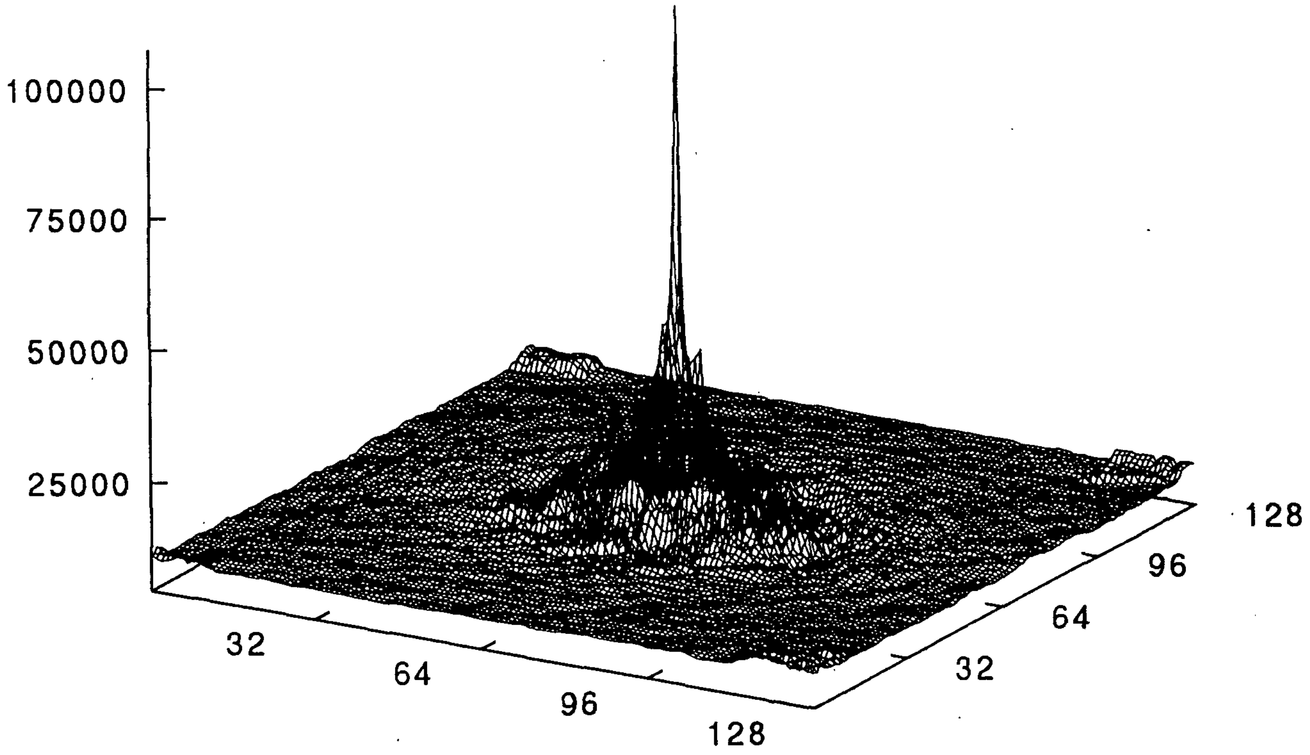

Figure 4.21. The Output Correlation for the Phase-Mostly Input and Phase-Mostly Filter SLMs Constrained MACE Filter

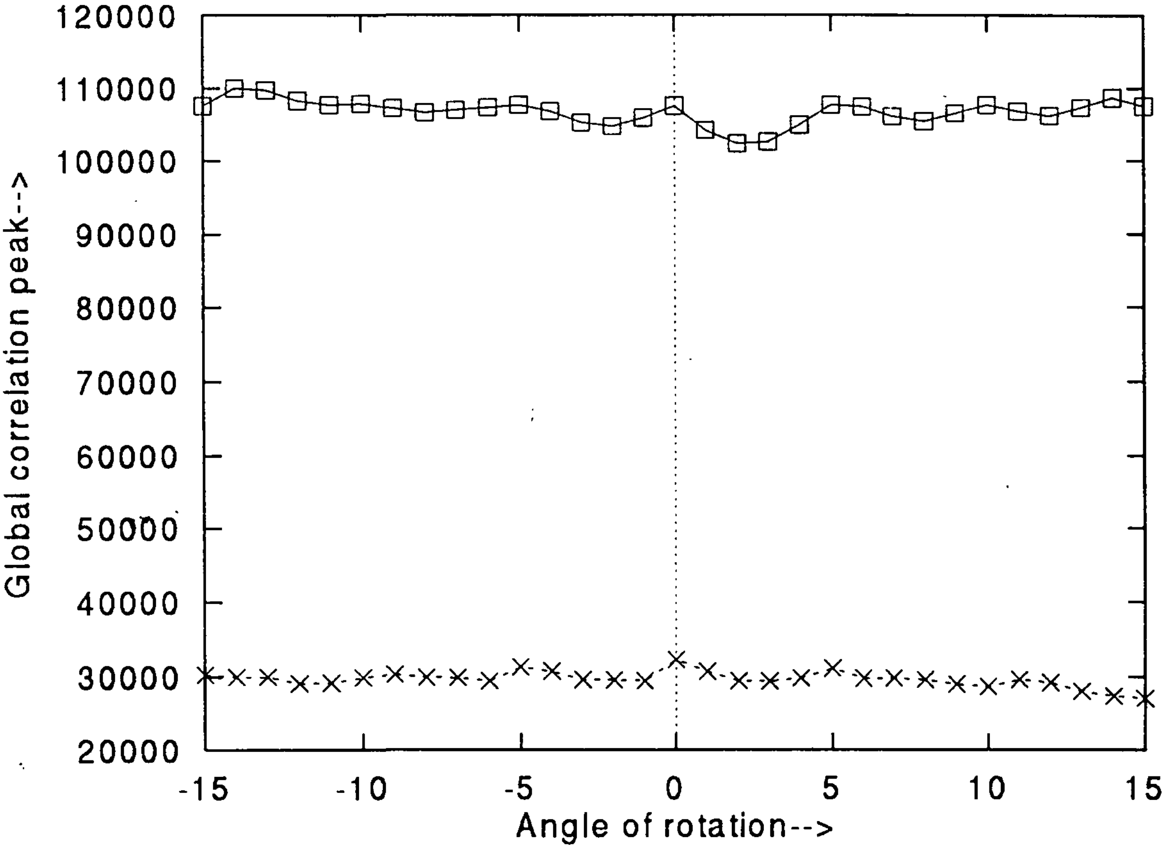

Figure 4.22. Distortion Test Results for the Phase-Mostly Input and Phase-Mostly Filter SLMs Constrained MACE Filter

4.3 Performance of Hooke and Jeeves-based MACE Filter

The algorithm for the design of a SLM-constrained MACE filter using the Hooke and Jeeves method was discussed in Chapter 3. The computer simulation results are presented below.

4.3.1 MACE Filter for Phase-Mostly SLM

The filter for the Phase-Mostly operating curve (Figure 4.5) was synthesized using the Hooke and Jeeves method of optimization. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.8.

4.3.2 MACE Filter for High-Contrast SLM

The filter for the High-contrast operating curve (Figure 4.6) was synthesized using the Hooke and Jeeves method of optimization. The input images were constrained by the Amplitude-Only operating characteristic.

Table 4.8. Correlation Statistics for Phase-Only SLM Constrained Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	12779.9	12816.1	12804.9	12788.6	12796.9	12796.9	12767.8
Energy	155561.0	157128.0	158388.0	177927.0	159044.0	158117.0	157604.0
PCE	1049.9	1045.4	1035.2	919.2	1029.7	1035.7	1034.3
PSR	2.20	2.81	2.48	1.88	2.55	2.77	3.27

Table 4.9. Correlation Statistics for High-Contrast SLM Constrained Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	2971.7	2974.2	2973.1	2960.0	2971.5	2970.7	2971.8
Energy	8563.3	8716.7	8627.2	13808.0	8984.1	9197.4	9181.8
PCE	1031.2	1014.8	1024.5	634.5	982.8	959.6	961.9
PSR	2.6	2.3	3.11	2.44	2.87	2.8	3.05

Seven training images from the shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.9.

4.3.3 MACE Filter for Highly-Coupled SLM

The filter for the Highly-Coupled operating curve (Figure 4.7) was synthesized using the Hooke and Jeeves method of optimization. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.10.

Table 4.10. Correlation Statistics for Highly-Coupled SLM Constrained Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	2846.7	2841.3	2829.0	2800.1	2838.4	2846.6	2849.1
Energy	5931.5	6082.0	6218.0	9677.1	6500.2	6336.6	6202.5
PCE	1366.2	1327.3	1287.2	810.2	1239.4	1278.8	1308.7
PSR	3.46	3.16	3.78	2.84	3.2	3.94	3.82

4.4 Performance of Simulated Annealing-based MACE Filter

The algorithm for the design of an SLM-constrained MACE filter using simulated annealing optimization was discussed in Chapter 3. For filter synthesize the value at the origin was chosen from the value obtained in the simplex based filter. The computer simulation results are presented below.

4.4.1 MACE Filter for Phase-Mostly SLM

The filter for Phase-Only operating curve (Figure 4.5) was synthesized using simulated annealing optimization. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.11.

Table 4.11. Correlation Statistics for Phase-Only SLM Constrained Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	11562.7	12921.2	13073.7	13966.0	13108.0	12811.7	11687.1
Energy	148586.0	150277.0	151689.0	171006.0	152730.0	152012.0	151603.0
PCE	899.7	1111.0	1126.8	1140.6	1125.0	1079.7	900.9
PSR	2.05	3.01	2.59	2.02	3.81	2.99	3.81

4.4.2 MACE Filter for High-Contrast SLM

The filter was designed by considering the High-Contrast operating curve for filter SLM. The input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the space shuttle database were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.12. Figure 4.23 gives a typical three-dimensional plot of the output correlation plane when the filter was correlated with a training image (s128.15). The correlation magnitude values are shown with respect to pixel numbers in the two-dimensional plane. The maximum magnitude occurs at the origin (65,65). The filter was correlated with all the images available in the shuttle and airplane databases. The correlation outputs for all the images are shown in Figure 4.24. The correlation outputs for the shuttle images are represented with a □ and the airplane images are represented with ×.

Table 4.12. Correlation Statistics for High-Contrast SLM Constrained Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	3580.5	3916.8	3880.2	4206.9	3946.7	3992.5	3694.7
Energy	14712.9	15285.9	15877.9	23099.5	16341.9	15887.9	15767.0
PCE	871.3	1003.6	948.3	766.18	953.2	1003.3	865.8
PSR	2.68	2.65	2.88	2.55	3.07	2.98	2.9

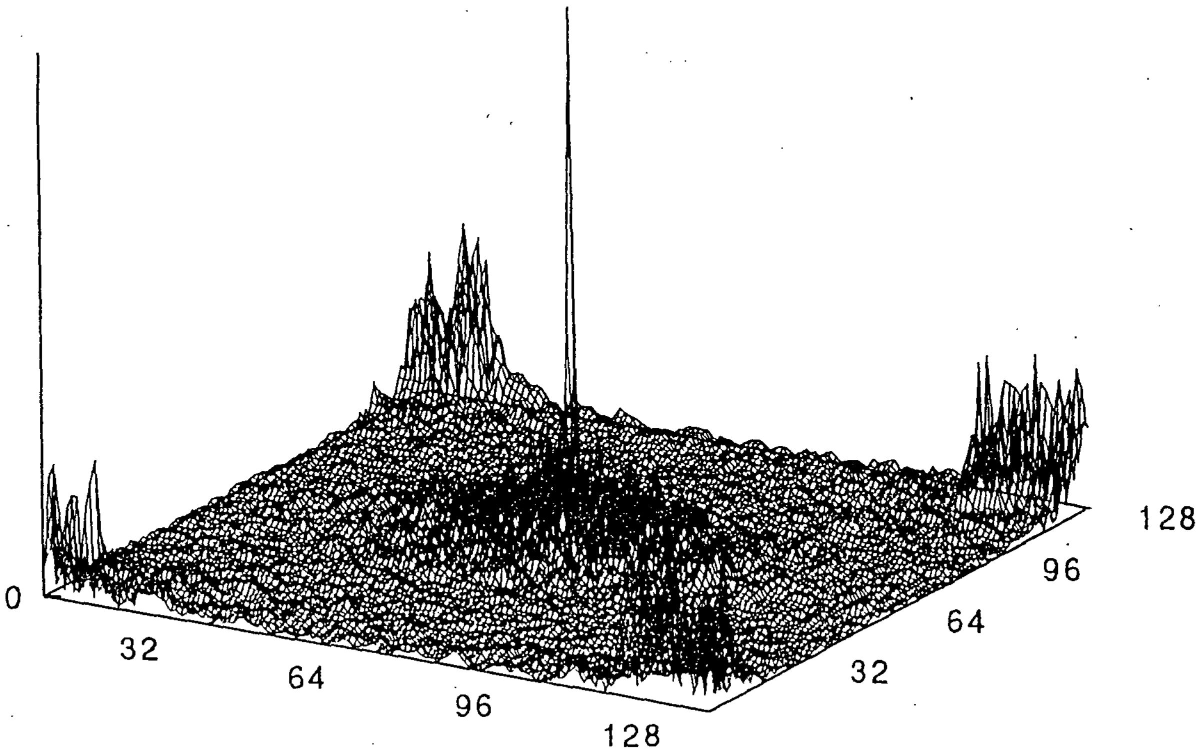

Figure 4.23. The Output Correlation for the High-Contrast SLM Constrained Filter

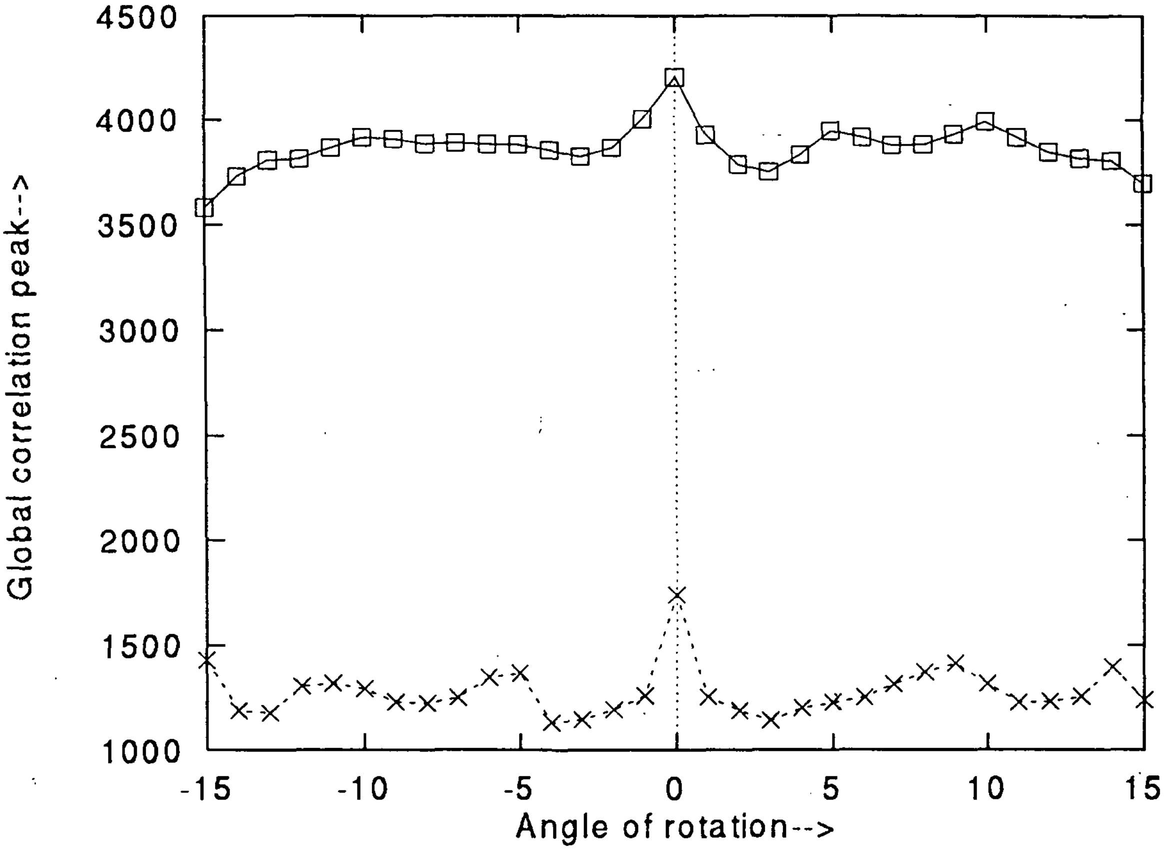

Figure 4.24. Distortion Test Results for the High-Contrast SLM Constrained Filter

4.4.3 MACE Filter for Highly-Coupled SLM

The filter was designed for the Highly-Coupled LCTV operating curve. As done before, the input images were constrained by the Amplitude-Only operating characteristic. Seven training images from the space shuttle were used for filter synthesis. The correlation plane statistics for the training images are listed in Table 4.13.

4.5 Comparison of SLM Constrained Filters

Figure 4.25 illustrates the correlation response at the origin for filters designed using various combinations of SLMs at the input and filter plane of the correlator. The filters were designed using seven training images from the space shuttle database. The simplex method was used for the filter's design. All the images from the space shuttle database were used for the correlation purpose.

Table 4.13. Correlation Statistics for Highly-Coupled SLM Constrained Filter

Image	s128.15n	s128.10n	s128.5n	s128.0	s128.5	s128.10	s128.15
Peak	3460.5	3860.6	3859.9	4173.4	3885.2	3815.6	3814.1
Energy	12436.2	12901.5	13315.5	17579.8	13740.2	13693.8	13608.0
PCE	962.9	1155.2	1118.9	990.7	1098.6	1063.2	907.4
PSR	2.76	3.45	4.12	2.49	4.19	4.12	3.81

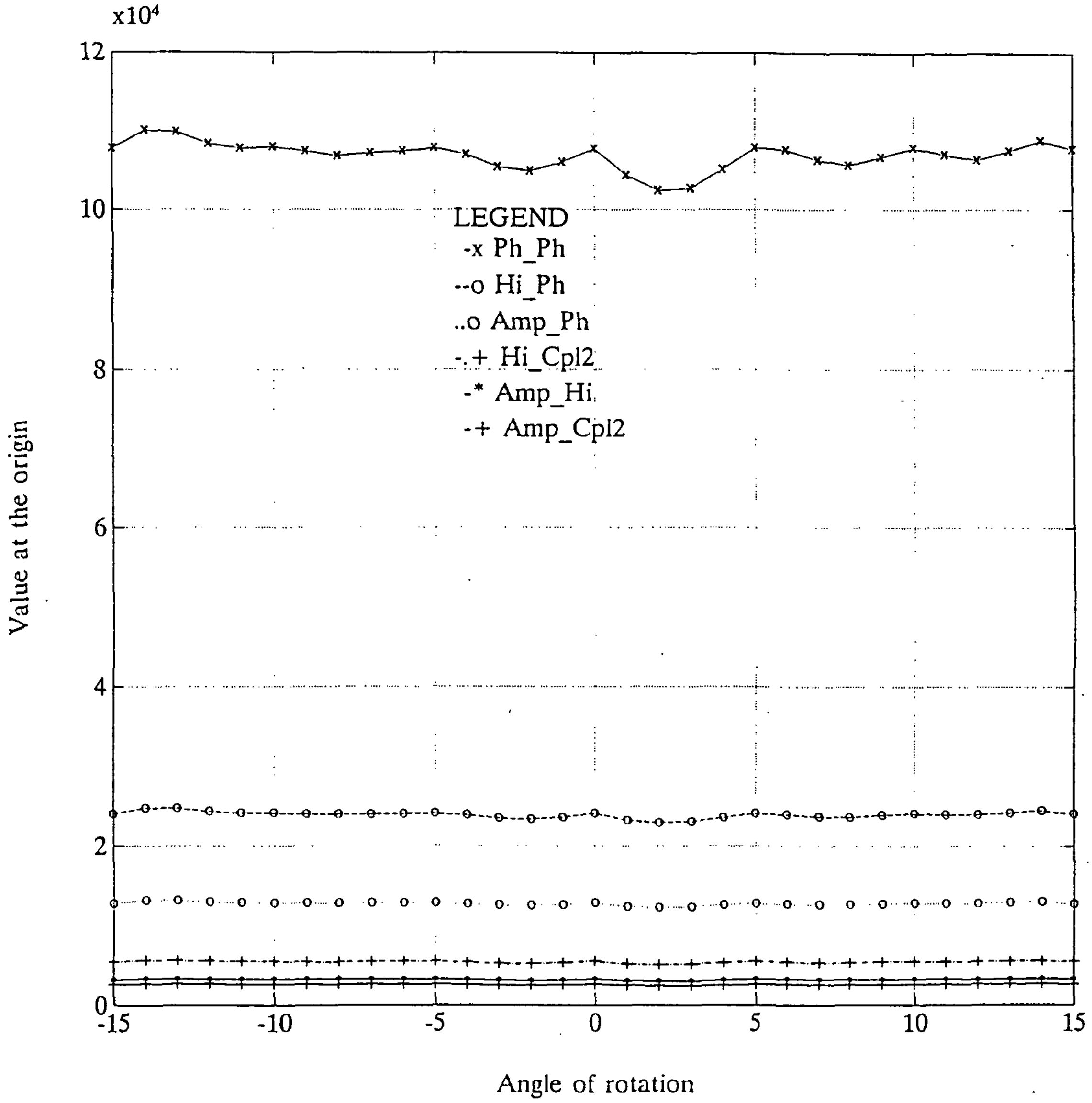

Figure 4.25. Correlation Response at the Origin for Various Filters

In the Figure 4.25, the filter names are abbreviated with SLM's names; for example, the filter name 'Amp_Cpl2' reveals that the filter was synthesized by assuming an Amplitude-Only input SLM and a Highly-Coupled filter SLM. Similarly, the name 'Hi_Ph' tells that the filter design assumed High-Contrast and Phase-Mostly SLMs at the input and filter planes respectively. The filter designed with Phase-Mostly SLMs at both input and filter planes gives the highest correlation values at the origin compared to the other SLM combinations tried in this report.

Figure 4.26 shows the Peak to correlation energy (PCE) variations with respect to the in-plane rotated shuttle images. The filter designed using an Amplitude-Only input SLM and a Highly-Coupled filter SLM gives the best PCEs compared to other filters. The Phase-Mostly SLM constrained filter in the presence of Phase-Mostly SLM gives the lowest PCEs.

In Chapter 3, three different techniques were discussed for the design of SLM constrained MACE filter. The filters designed using these techniques are now compared with Minimum Euclidean Distance (MED) mapped filter. Figure 4.27 illustrates the correlation response at the origin for High-Contrast SLM constrained filters (with Amplitude-Only input SLM) designed using four different methods. It is to be observed that the MED mapped filter was obtained by mapping an unconstrained MACE filter (but all the training images were passed through an Amplitude-Only input SLM) on to the filter SLM operating curve using MED concept. The unconstrained MACE filter was initially designed with the specified origin value of 3256.1 (the correlation value at the origin obtained for High-Contrast SLM constrained filter in the presence of Amplitude-Only input SLM).

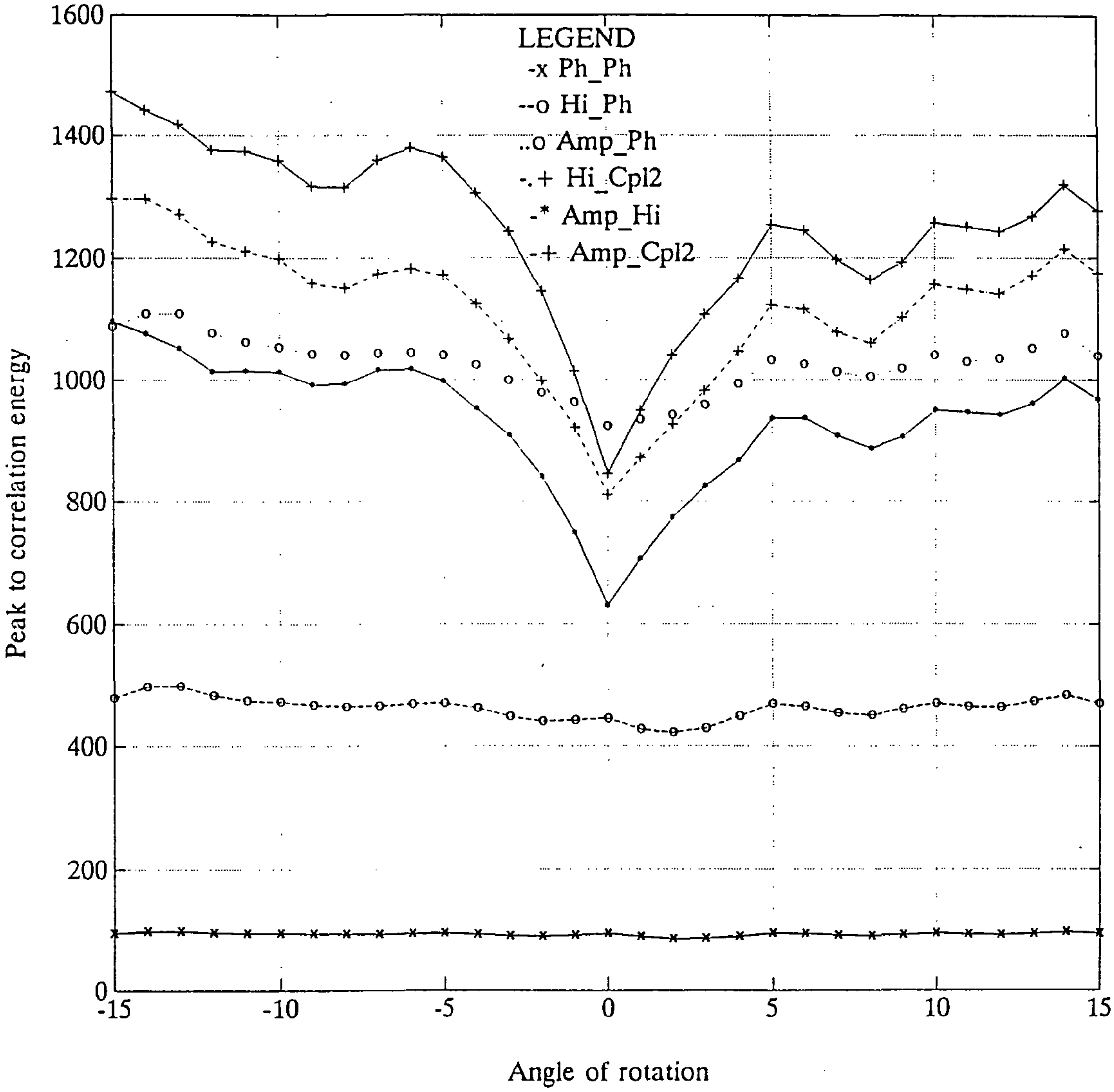

Figure 4.26. Peak to Correlation Energy Ratio for Various Filters

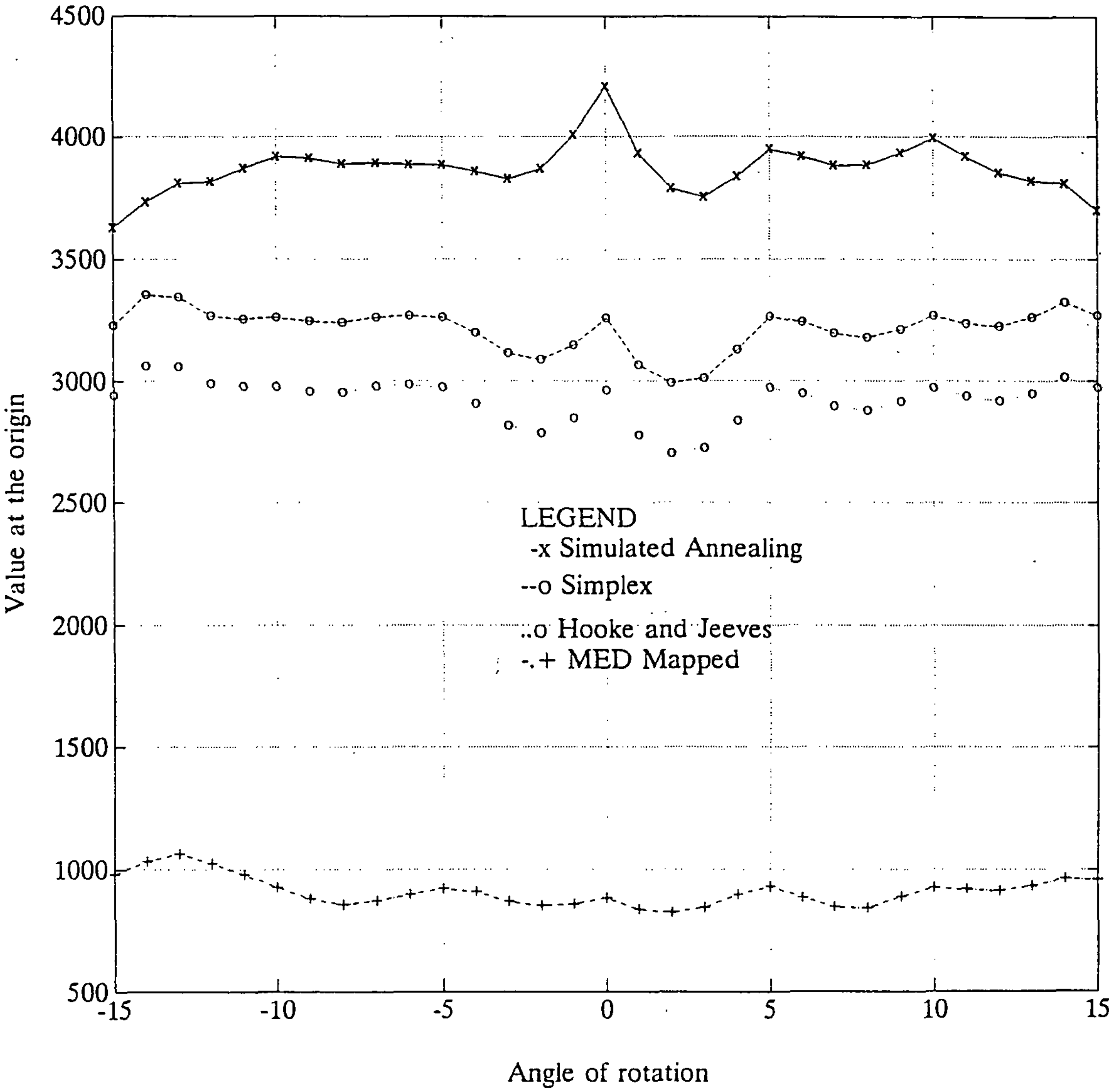

Figure 4.27. Correlation Response at the Origin for Amplitude-Only Input and High-Contrast Filter SLM Constrained Filter Designed Using Different Methods

Figure 4.28 gives the PCE ratios for a High-Contrast SLM constrained filter in the presence of Amplitude-Only input SLM. Four methods were used for the filter design. The simulated annealing, simplex, and Hooke and Jeeves methods give very similar performance.

The filter design techniques based on the simplex method and the Hooke and Jeeves method use a relaxation-based iterative technique to get equal values at the correlation plane origin for all the training images. Figure 4.29 illustrates the correlation response for training images when the images were correlated with a High-Contrast SLM constrained filter (in the presence of Amplitude-Only input SLM) designed with and without the relaxation technique. Figure 4.30 shows the PCE ratios for a filter designed with and without the relaxation technique.

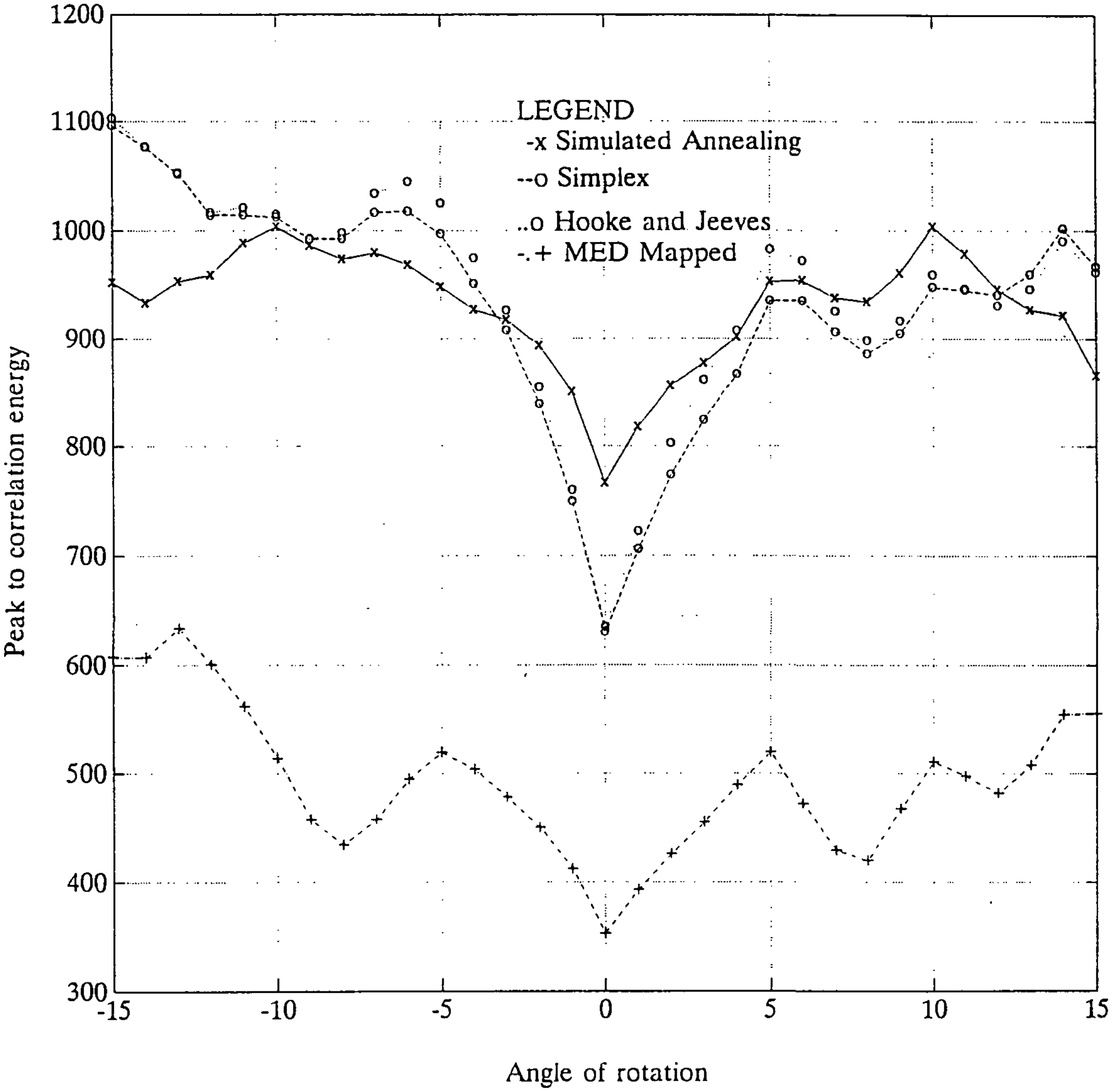

Figure 4.28. Peak to Correlation Energy Ratio for Amplitude-Only Input and High-Contrast Filter SLM Constrained Filter Designed Using Different Methods

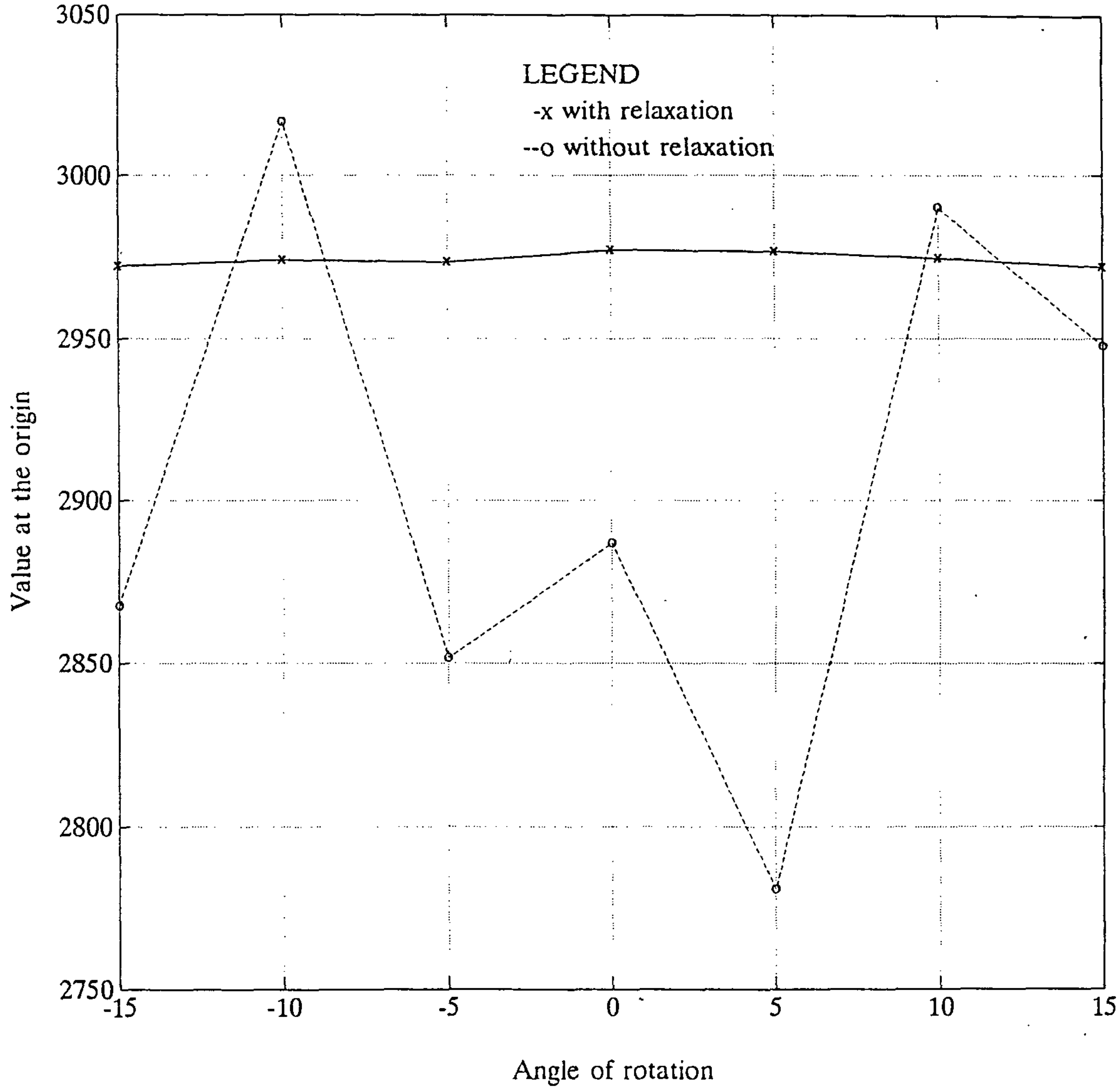

Figure 4.29. Correlation Response at the Origin for a MED Mapped Filter with and without Relaxation Algorithm Applied During Filter Design

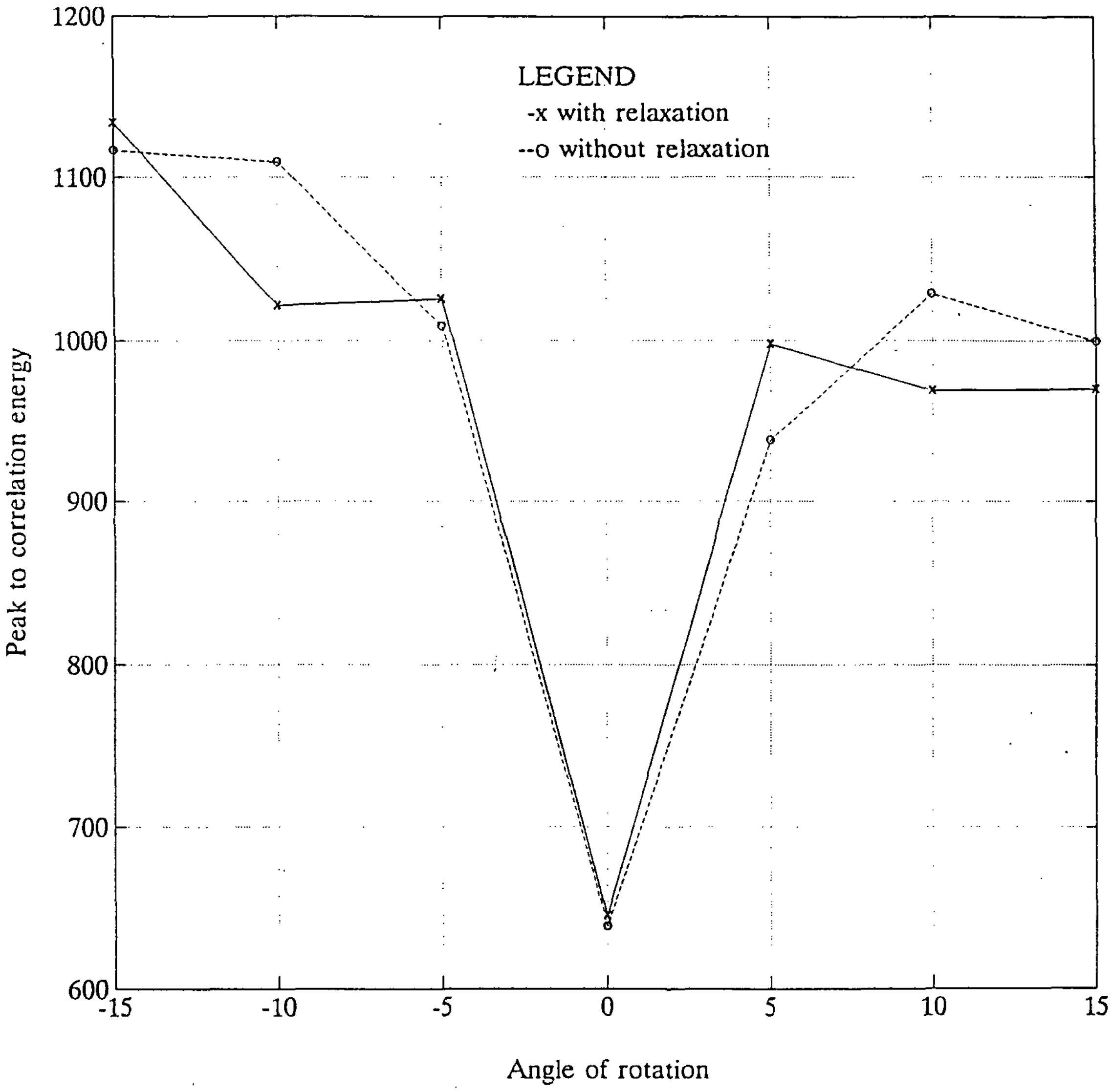

Figure 4.30. Peak to Correlation Energy for a MED Mapped Filter with and without Relaxation Algorithm Applied During Filter Design

4.6 Noise Analysis

The LCTV SLM's operating curves were used for the design of simplex based MACE filters. The performance of these filters were studied when the input image was corrupted by zero mean Gaussian noise.

A space shuttle image was added with zero mean, Gaussian noise to obtain images with different signal-to-noise ratios (SNRs). Figures 4.31 through 4.34 show images with 30dB, 20dB, 10dB, and 0dB SNRs, respectively. These images were used to evaluate the filters designed with different input and filter SLMs operating characteristics. The correlation plane statistics of various filters, for images shown in Figures 4.31, 4.32, 4.33, and 4.34, are given in tables 4.14, 4.15, 4.16, and 4.17, respectively. The Figure 4.35 gives a typical three-dimensional plot of the output correlation plane when the Phase-mostly SLM constrained filter was correlated with the shuttle image (Figure 4.34). The filter names are abbreviated with SLM's names as done in the previous section. Figure 4.36 shows average values at the origin when shuttle images with different SNRs are correlated with a High-Contrast SLM constrained filter (Amplitude-Only input SLM is assumed). It is to be noted here that ten different seed values are used for the generation of noise for each image (for a particular SNR) and the average correlation value is used for the plot. Figure 4.37 illustrates the standard deviation of the value at the origin.

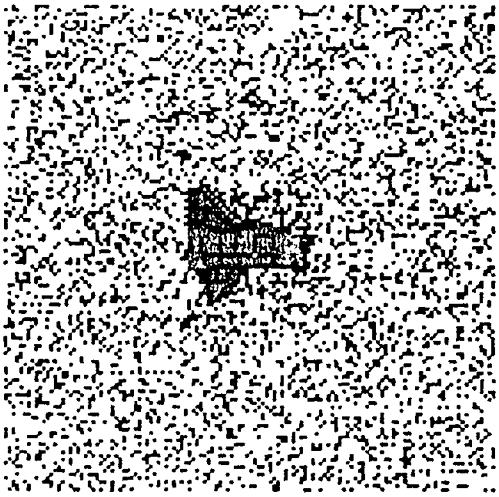

Figure 4.31. The shuttle image with 30dB SNR

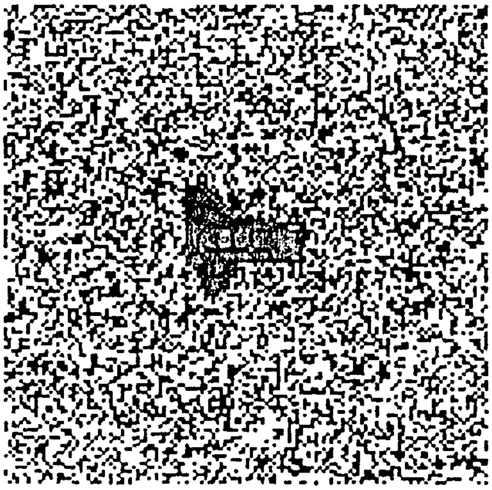

Figure 4.32. The shuttle image with 20dB SNR

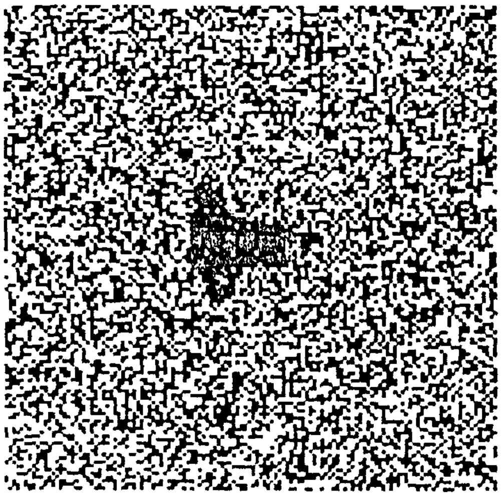

Figure 4.33. The shuttle image with 10dB SNR

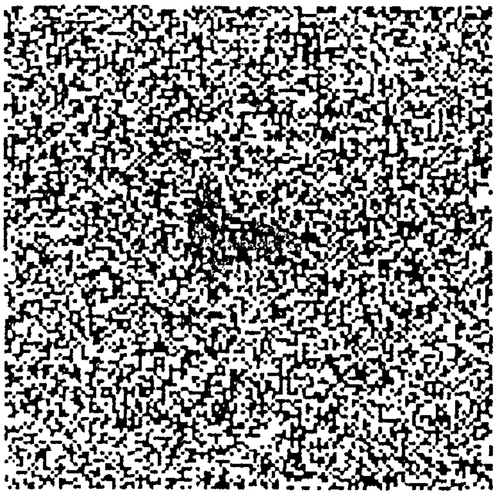

Figure 4.34. The Shuttle image with 0dB SNR

Table 4.14. Correlation Statistics for Space shuttle image with 30dB SNR

Filter	amp-cpl2	amp-hi	amp-ph	hi-cpl2	hi-ph	ph-ph
Peak	2599.9	3233.7	12723.7	5462.7	23988.5	107072
Energy	8138.2	16903.7	177043	37338.0	1307350	122385000
PCE	830.5	618.6	914.4	799.2	440.2	93.67
PSR	2.7	2.52	1.87	2.87	1.89	1.89

Table 4.15. Correlation Statistics for Space shuttle image with 20dB SNR

Filter	amp-cpl2	amp-hi	amp-ph	hi-cpl2	hi-ph	ph-ph
Peak	2577.7	3181.3	12569.3	5376.6	23581.7	104680.1
Energy	8315.8	19760.8	185701	41157.6	1399940	122199000
PCE	799.0	512.2	850.7	702.4	397.2	89.7
PSR	2.53	2.44	1.86	2.99	1.85	1.83

Table 4.16. Correlation Statistics for Space shuttle image with 10dB SNR

Filter	amp-cpl2	amp-hi	amp-ph	hi-cpl2	hi-ph	ph-ph
Peak	2487.0	2959.8	11965.7	5048.7	22213.2	95817.9
Energy	10073.6	47189.2	268011	79517.9	1763370	122122000
PCE	614.0	185.6	534.2	320.5	279.8	75.2
PSR	2.1	2.1	1.8	2.6	1.74	1.7

Table 4.17. Correlation Statistics for Space shuttle image with 0dB SNR

Filter	amp-cpl2	amp-hi	amp-ph	hi-cpl2	hi-ph	ph-ph
Peak	1377.6	2031.1	9629.4	3738.8	16920.2	60899.0
Energy	164576	323821	1091050	540092	4761660	125763000
PCE	11.53	12.73	84.9	25.88	60.1	29.48
PSR	1.13	1.1	1.42	1.1	1.37	1.32

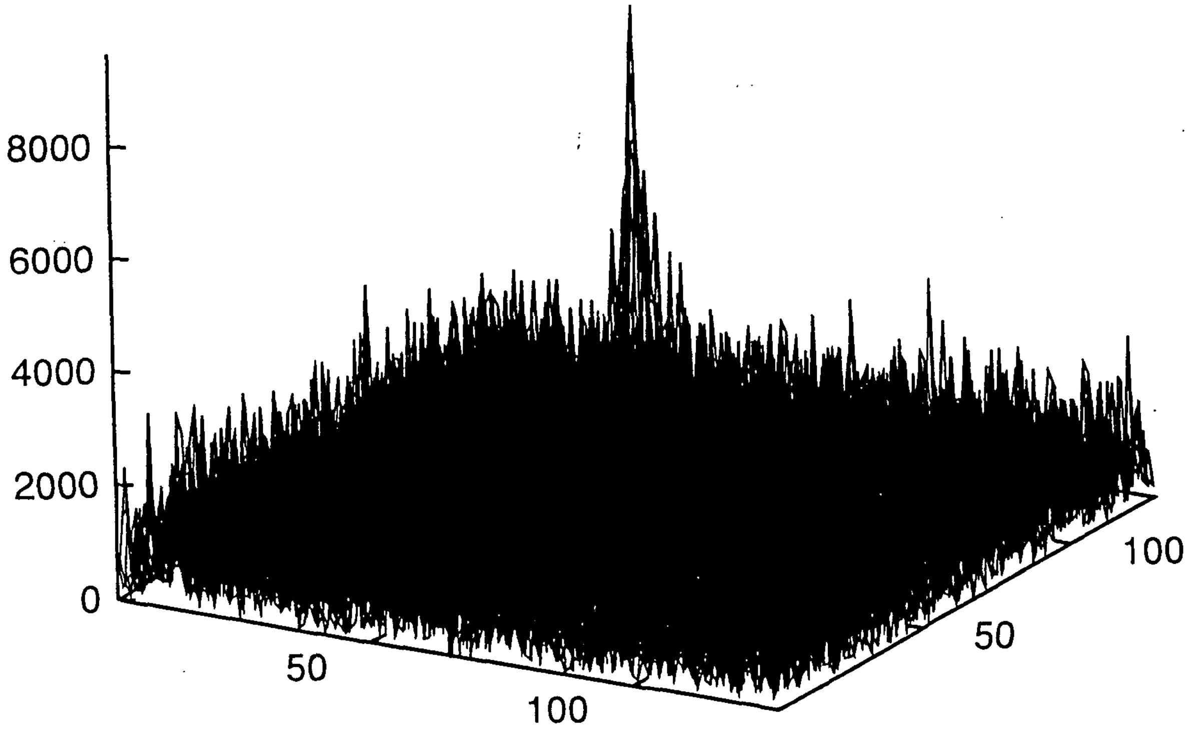

Figure 4.35. The Output Correlation for the shuttle image with 0dB SNR when correlated with amp-ph filter

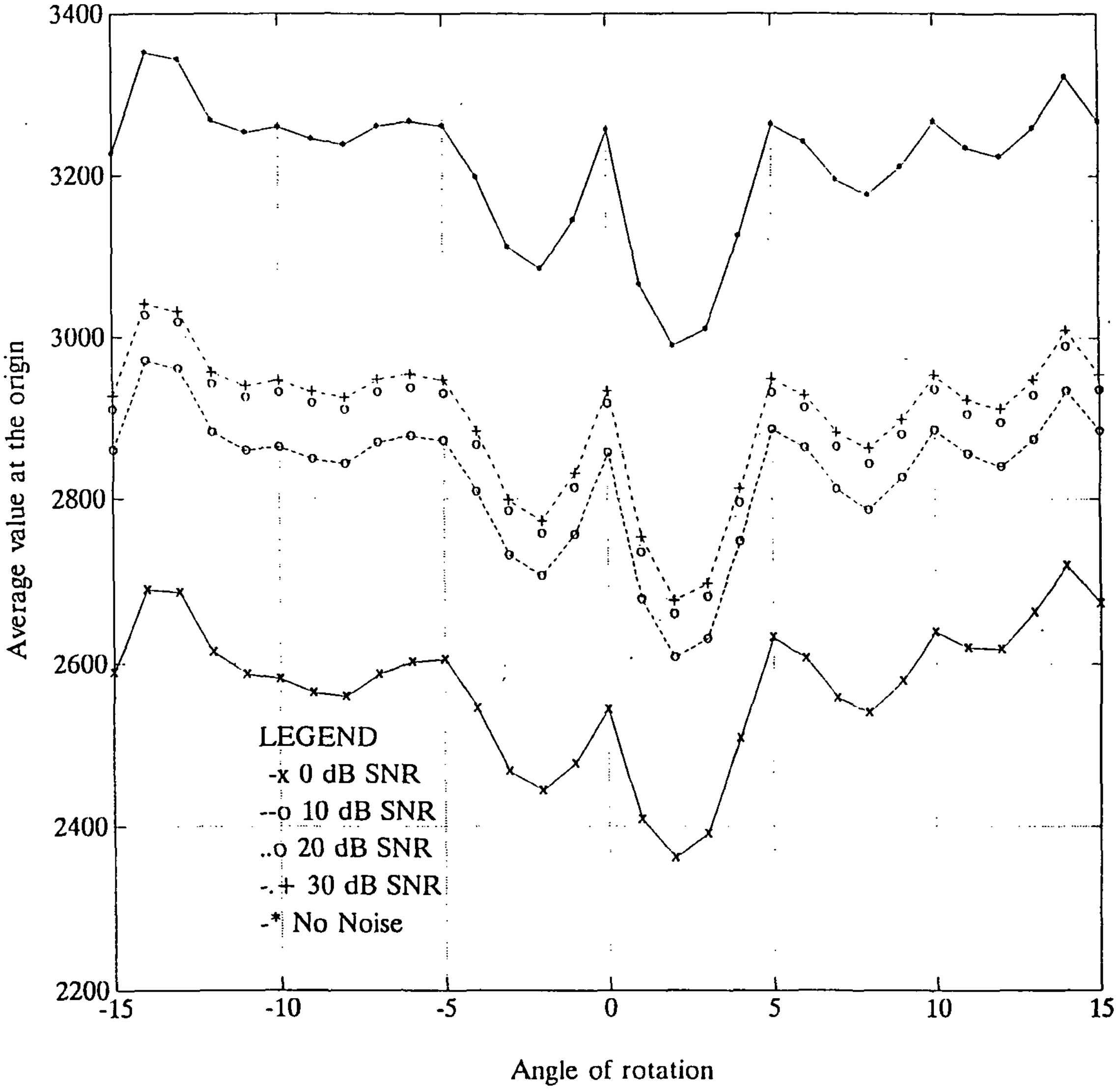

Figure 4.36. Average Value at the Origin for Amplitude-Only Input and High-Contrast Filter SLM Constrained Filter when Correlated with a Shuttle image with Different SNRs

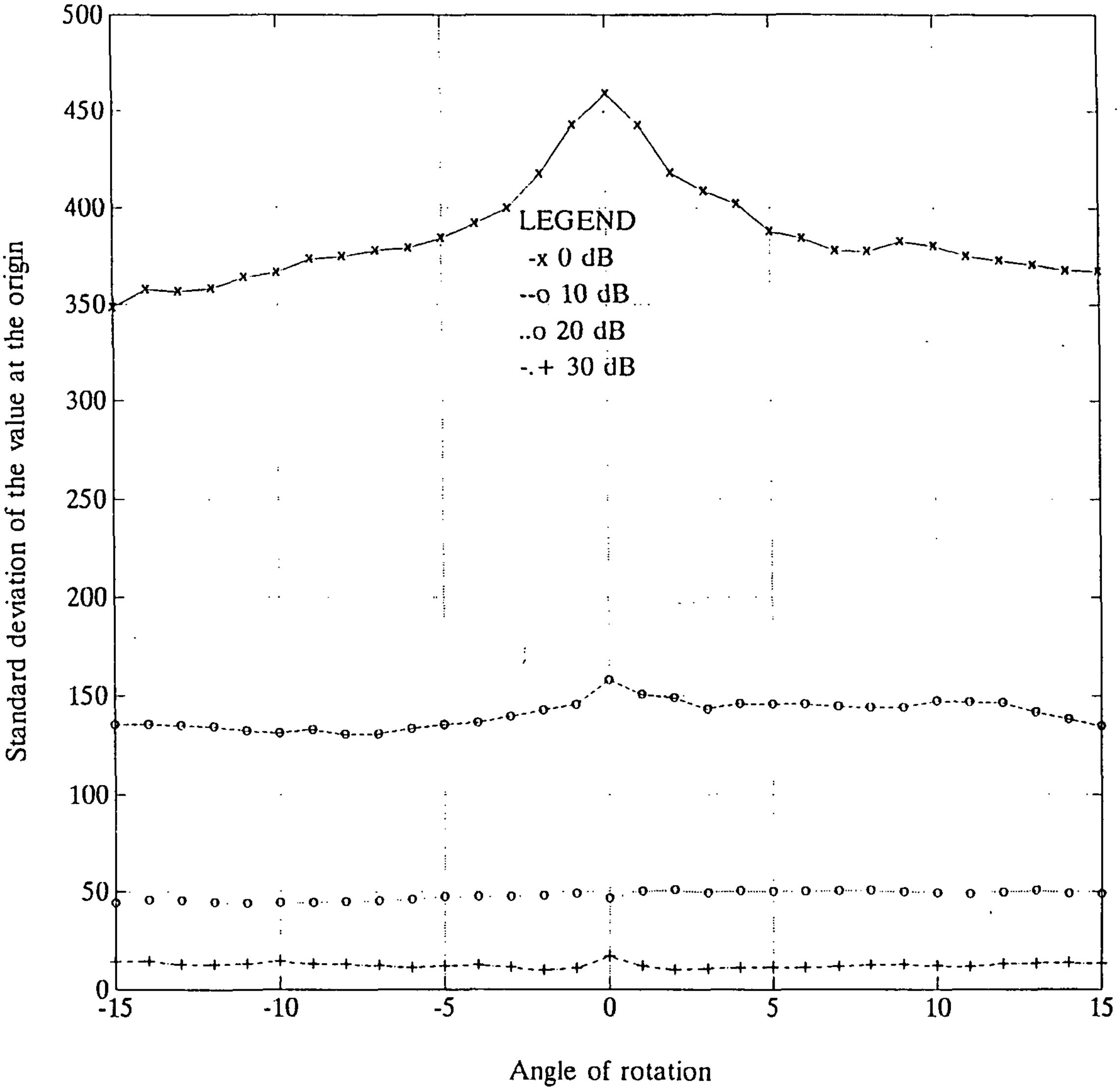

Figure 4.37. Standard Deviation of the Value at the Origin for Amplitude-Only Input and High-Contrast Filter SLM Constrained Filter when Correlated with a Shuttle image with Different SNRs

4.7 The Simplex-based MACE Filter's Performance for Out-of-Plane Rotated Images

The Phase-Mostly SLM constrained MACE filter (Amplitude-Only SLM in the input plane) was synthesized using the simplex method. Seven out-of-plane rotated truck images (Figure 4.38) of size 64 × 64 were used for the study. Initially, all the seven images were converted to 128 × 128 by appending them with zeros. Then six images were used as the training images for the filter. A typical three-dimensional plot of the output correlation plane is given in Figure 4.39, when the filter was correlated with a non-training image. The correlation plane statistics for all seven images are listed in Table 4.18. The seventh column of Table 4.18, gives the correlation statistics for non-training image.

Table 4.18. Correlation Statistics for Out-of-Plane Rotated Truck Images

Image	image1	image2	image3	image4	image5	image6	image7
Peak	74475.8	74513.5	74637.9	74613.4	74562.8	74311.9	44359.2
Energy	19102900	19354300	19390000	19598300	18724900	19589900	18862100
PCE	290.4	286.9	287.2	284.1	296.9	281.9	104.3
PSR	2.8	2.7	1.58	2.8	2.4	2.9	1.7

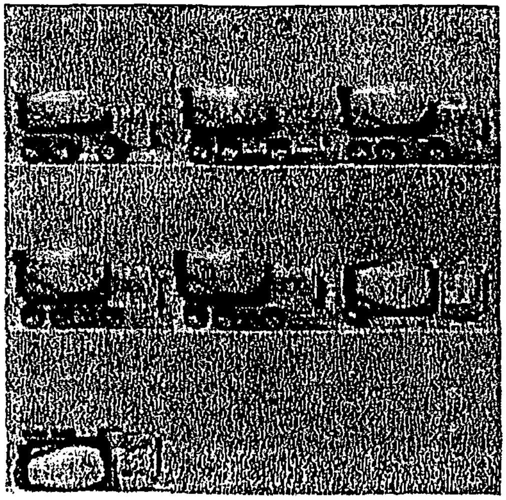

Figure 4.38. Out-of-Plane Rotated Truck Images

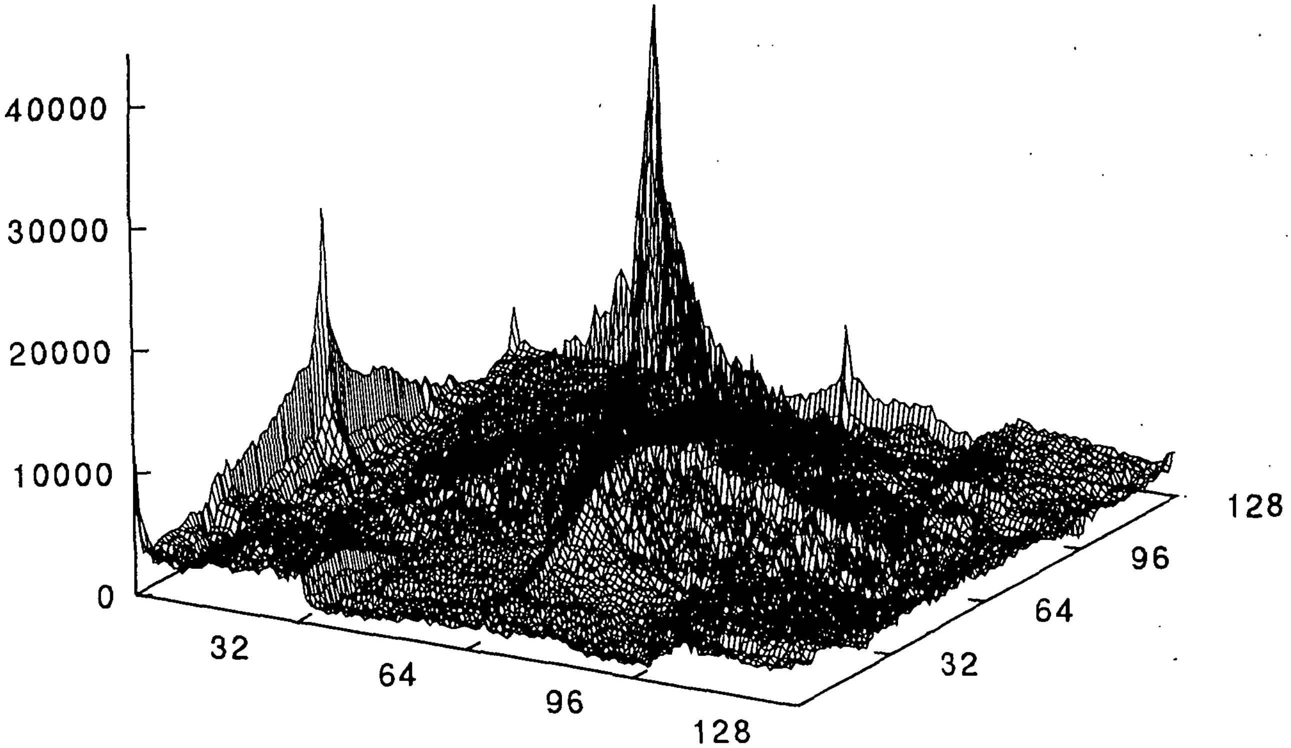

Figure 4.39. The Output Correlation for a Truck Image when Correlated with the Filter

4.8 A Study on Computation Time

In this report, three different methods were used to design a realizable MACE filter. They are: (1) simplex method, (2) Hooke and Jeeves method, and (3) simulated annealing-based optimization. The filters were designed for different characteristics of LCTV SLM. A SUN SPARC station 2, a unix-based workstation running on a 40.2 Mhz CPU clock frequency, was used for the simulation study. For a filter with seven training images of size 128×128 pixels, the simplex method-based filter synthesis takes about 10 minutes, Hooke and Jeeves method takes about 24 minutes, and the simulated annealing-based method takes about 65 minutes. For all the above three methods, Amplitude-only and Phase-mostly characteristics were assumed for SLMs at input and filter planes, respectively.

4.9 Summary

This section presented the simulation of SLM-constrained MACE filter for various input and filter SLM characteristics. It was found that the distortion tolerance and discrimination ability of the filter was satisfactory. The constrained filter for Phase-mostly SLM in the presence of Amplitude-Only SLM gave the highest peak value at the origin but the filter with Phase-mostly SLMs at both filter and input planes gave the lowest Peak to correlation energy (PCE) in the correlation plane. The simplex-based MACE filter's performance was compared with MACE filters designed using Hooke and Jeeves method and simulated annealing-based optimization. The filters designed with the simplex method generally gave better PCEs than MACE filters based on other two methods. But the simulated annealing-based filters gave better Peak to Side lobe (PSR) ratios. The simplex-based filter design took less computer time compared to the other two methods.

CHAPTER 5
SUMMARY AND RECOMMENDATIONS

The research work leading to this report involved the design of a realizable, distortion-invariant filter which can be used in an optical pattern recognition system which uses two SLMs, one at the input plane and the other at the filter plane. This filter can be implemented on available spatial light modulators (SLMs) which exhibit arbitrary operating characteristics.

5.1 Discussion

Three different techniques were discussed in Chapter 3 for the design of a realizable MACE filter. They are: (1) simplex method, (2) Hooke and Jeeves method, and (3) simulated annealing-based optimization. The filters based on Simplex and Hooke and Jeeves methods maximized the sum of peak to correlation energies for all the training images. The filter based on simulated annealing optimization minimized the average correlation energy while maintaining the specified value at the correlation plane origin. The performance of the filter was evaluated for different LCTV operating characteristics. The simulation results were presented in Chapter 4.

The filter for Phase-mostly SLM at the filter plane gave maximum correlation intensity at the origin in the presence of Amplitude-Only SLM at the input plane. But, this filter gave poor PCEs in the presence of Phase-mostly SLM at the input plane.

Generally the simplex method-based filter design gives better PCE's compared to the other two methods. But the SLM-constrained MACE filters, designed using simulated annealing-based optimization, give better peak to sidelobe ratio (PSR) compared to filters designed using simplex and Hooke and Jeeves methods. The simplex method-based filter synthesis takes about 10 minutes of computer time, Hooke and Jeeves method takes about 24 minutes, while the modified simulated annealing-based filter design takes about 65 minutes.

5.2 Recommendations

The computation time can be reduced if the objective function is modified to include the conditions for getting equal correlation peaks at the origin for all the training images. It is expected that the modified objective function will reduce the number of iterations during the filter design. This report involved the design of SLM-constrained filters for maximizing the sum of PCEs. The above design can be modified to maximize the SNR, PTE, or Intensity. While the metrics namely Intensity, SNR, PCE, and PTE are important measures for the filter, they do not directly describe the recognition/detection performance of a correlation filter. Recently, Vijayakumar et al. [39] used two more direct performance measures, the probability of detection (P_D) and the probability of false alarm (P_{FA}), for comparing matched spatial filter with phase-only filter. These performance measures may be used for evaluating the SLM constrained MACE filters. Juday and Rajan [10] reported that some adjustment is possible on the LCTV SLM operating curve. So research should continue to select the optimum LCTV curve which will maximize a given metric.

Synthetic estimation filters (SEFs) are used to estimate the orientation of an object. Embar and Rajan [40] designed SEF's using a minimum average correlation energy concept. They reported that the SEF based on MINACE filter gives accurate estimation of the angle of rotation of an object. The approach adopted in this report work can be used for the design of realizable synthetic estimation filters.

REFERENCES

[1] Goodman, J. W., Introduction to Fourier Optics, McGraw-Hill, New York, 1968.

[2] Vander Lugt, A., "Signal Detection by Complex Spatial Filtering," IEEE Trans. on Inf. Theory, IT-10, 1064, 1964, pp. 139-145.

[3] Papoulis, A., Probability, Random Variables, and Stochastic Processes, McGraw-Hill Publishing Company, Second Edition, New York, 1984.

[4] Horner, J.L., "Light Utilization in Optical Correlators," Applied Optics, Vol. 21, No. 24, Dec. 1982, pp. 4511-4514.

[5] Horner, J.L., and Gianini, P.D., "Phase-Only Matched Filtering," Applied Optics, Vol. 23, March 1984, pp. 812-816.

[6] Hester, C.F., and Casasent, D., "Multivariant Technique for Multiclass Pattern Recognition," Applied Optics, Vol. 19, June 1980, pp. 1758-1761.

[7] Casasent, D., "Unified Synthetic Discriminant Function Computational Formulation," Applied Optics, Vol. 23, No. 10, May 1984, pp. 1620-1627.

[8] Mahalanobis, A., Vijayakumar, B.V.K., and Casasent, D., " Minimum Average Correlation Energy Filters," Applied Optics, Vol. 26, Sep. 1987, pp. 6633-6640.

[9] Ravichandran, G., and Casasent, D., "Minimum Noise and Correlation Energy Optical Correlation Filter," Applied Optics, Vol. 31, No. 11, April 1992, pp. 1823-1833.

[10] Juday, R., and Rajan, P.K., "Selection of LCTV Operating Curves for Input and Filter," SPIE, Vol. 1701, 1992, pp. 78-82.

[11] Juday, R., "Correlation with Spatial Light Modulator having Phase and Amplitude Cross Coupling," Applied Optics, Vol. 28, No. 22, Nov. 1989, pp. 4865-4869.

[12] Farn, M. W., and Goodman, J. W., "Optimal Maximum Correlation Filter for Arbitrarily Constrained Devices," Applied Optics, Vol. 28, No. 15, Aug. 1989, pp. 3362-3366.

[13] Vijayakumar, B. V. K., Juday, R. D., and Rajan, P. K., "Saturated Filters," J. Opt. Soc. Am. A9, 1992, pp.405-412.

[14] Juday, R. D., "Optimal Realizable Filters and the Minimum Euclidean Distance Principle," Applied Optics, Vol. 32, Sep. 1993, pp. 5100-5111.

[15] Jared, D.A., and Ennis, D.J., "Inclusion of Filter Modulation in Synthetic discriminant Function Construction," Applied Optics, Vol. 28, Jan. 1989, pp. 232-239.

[16] Casasent, D., and Rozzi, W., "Computer Generated and Phase-Only Synthetic Discriminant Function Correlation Filters," Applied Optics, Vol. 25, Oct. 1986, pp. 3767-3772.

[17] Balendra, A., and Rajan, P.K., "Real-valued Composite Filters for Optical Pattern Recognition," SPIE, Vol. 1772, 1992, pp. 30-37.

[18] Balendra, A., Real-valued Composite Filters for Correlation-based Optical Pattern Recognition, M.S. Thesis, Department of Electrical Engineering, Tennessee Technological University, Dec. 1991.

[19] Mahalanobis, A., and Song, S., "Theoretical Framework for the Design of Purely Real Synthetic Discriminant Function Type Correlation Filters," Applied Optics, Vol. 31, No. 35, Dec. 1992, pp. 7450-7456.

[20] Rajan, P.K., and Muttiah, F.S., "A Study on Optimal Pattern Recognition Filters Implementable on Spatial Light Modulators with Annular Response Regions," Proc. the 26th Southeastern Symposium on System Theory, Athens, Ohio, March 20-22, 1994, pp. 480-484.

[21] Muttiah, F. S., A Study on Optimal Pattern Recognition Filters Implemen table on Spatial Light Modulators with Annular Response Regions, M.S. Thesis, Department of Electrical Engineering, Tennessee Technological University, Dec. 1993.

[22] Carlson, D. W., and Vijayakumar, B. V. K., "Synthetic Discriminant Function for Implementation on Arbitrarily Constrained Devices," SPIE, Vol. 1772, 1992, pp. 11-20.

[23] Vijayakumar, B.V.K., and Hassebrook, L., "Performance Measures for Correlation Filters," Applied Optics, Vol. 29, No. 20, July 1990, pp. 2997-3006.

[24] Khan, A. J., and Rajan, P.K., "Design of SLM-Constrained MACE Filters using Simulated Annealing Optimization," SPIE, Vol. 1959, April 1993, pp. 284-292.

[25] Khan, A. J., Design of SLM Constrained MACE Filters for Correlation-Based Optical Pattern Recognition, M.S. Thesis, Department of Electrical Engineering, Tennessee Technological University, Dec. 1992.

[26] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing," Science, Vol. 220, May 1983, pp. 671-680.

[27] Nelder, J.A., and Mead, R., "A Simple Method for Function Minimization," The Comp. Journal, 7, 1965, pp. 308-313.

[28] Hooke, R., and Jeeves, T.A., "Direct Search Solution of Numerical and Statistical Problems," Journal of the Association for Computing Machinery, Vol. 8, 1961, pp. 212-229.

[29] Thomas, J.B., An Introduction to Statiscal Communication Theory, John Wiley and Sons, Inc., New York, 1969.

[30] VijayaKumar, B.V.K., "Minimum-Variance Synthetic Discriminant Functions," J. Opt. Soc. AM. A, Vol. 3, No. 10, Oct. 1986, pp. 1579-1584.

[31] Psaltis, D., Paek, E.G., and Venkatesh, S.S., "Optical Image Correlation with a Binary Spatial Light Modulator," Opt. Eng., Vol. 23, Nov. 1984, pp. 698-704.

[32] Vijayakumar, B.V.K., and Bahri, Z.,"Efficient Algorithm for Designing Ternary Valued Filter Yielding Maximum Signal-to-Noise Ratio," Applied Optics, Vol. 28, May 1989, pp. 1919-1925.

[33] Dickey, F.M., Vijayakumar, B.V.K., Romero, L.A., and Connelly, J.M., "Complex Ternary Matched Filters Yielding High Signal-to-Noise Ratios, " Opt. Eng., Vol. 29, Sep. 1990, pp. 994-1001.

[34] Hendrix, C.D., and Vijayakumar, B.V.K., " A Comparative Evaluation of Binary Phase-Only Synthetic Discriminant Function Filter, " SPIE, Vol. 2237, 1994, pp. 103-113.

[35] Downie, J.D., "Case Study of Binary and Ternary SDFs with Similar In-Class and Out-of-Class Images, " Opt. Eng., Vol. 32, Mar. 1993, pp. 560-570.

[36] Horner, J.L., and Gianino, P.D., "Applying the Phase-Only Filter Concept to the SDF Correlation Filter," Applied Optics, Vol. 24, Mar. 1985, pp. 851-854.

[37] Roberge, D., and Shang, Y., "Optical Implementation of the Phase-Only Filter using Liquid Crystal Television," SPIE, Vol. 2237, 1994, pp. 197-203.

[38] Shoup, T.E., and Mistree, F., Optimization Methods with Applications for Personal Computers, Prentice-Hall, Inc., New Jersey, 1987.

[39] Vijayakumar, B.V.K., Chen, C., and Brasher, J.D., " Correlation Filters Assessed in Terms of Probability of Detection and Probability of False Alarm," SPIE, Vol. 1959, April 1993, pp. 12-22.

[40] Embar, S., and Rajan, P.K., "Design of Synthetic Estimation Filters using Correlation Energy Minimization," SPIE, Vol. 2237, 1994, pp. 124-135.

APPENDICES

APPENDIX A
SOURCE LISTINGS

```
*****************************************************************
*                                                               *
* Program   : slm-mace1.f.                                      *
*                                                               *
* Purpose   : Implements SLM constrained composite MACE filter; *
*             considers both input and filter SLMs (LCTV).      *
*                                                               *
* Algorithm : Uses Simplex or Hooke and Jeeves  method for 2D   *
*                      Optimization.                            *
*             Uses Juday's method for MED mapping.              *
*                                                               *
* Author    : Ramakrishnan, R.                    Date: July'94 *
*                                                               *
* Version   : 01                                                *
*                                                               *
*****************************************************************

*     N--> size of the image
*     P--> No. of training images
*     beta --> damping factor
*     Q--> No. of discrete values in SLM operating curve
*     S--> Image Spectrum Matrix
*     SW-> Prewhitened Image Spectrum
*     in_slm(Q) -->Input SLM
*     slm(Q) -->Filter SLM
*     grain --> size of drive array (needed for mapping)
*     range --> max mag range for search in the complex plane
*     drive(grain,grain) --> Array containing premapped values


*************************************************************
*     Declarations

      integer*2 M,N,L,P,Q,i,j,u,v,tr_num,K,grain,opt
      parameter(N=128)
      parameter(P=7)
      parameter(Q=256)
      parameter(L=N*N)
      parameter(grain=127)
      character*12 file1,name,file2,in_slm_name,fil_slm_name
      character*12 res,map_name
      integer*2 iimg(N,N),count,drive(grain,grain),count1
      real mag(Q),ph(Q),c(P),alpha,ENERGY(P),a(P),Da(P),SS(L),Pi
      real PCE(P),Tot_PCE,max_PCE,err,range
      real h_plus_real, h_plus_imag,dist,this_dis
      complex cimg(N,N),cfimg(N,N),S(L,P),SW(L,P),H(L),HM(L)
      complex in_slm(Q),slm(Q),ORIGIN(P),filter(N,N),HH(N,N)
      complex gain_beta
      integer*2 HI,LO,G,FE,PS,BS
      real SIM(3,2),X(2),XH(2),XG(2),XL(2),XO(2),XR(2),XC(2)
      real F(3),FH,FL,FG,FO,FR,AL,BE,GA,step,S1,S2,XE(2),FEE
      real Y(2),B(2),PP(2),FI,FB

      common M
      M=N

*     Initializations

      tr_num=P
      alpha=0.99
      max_PCE=0.0
      Pi=3.14159
      count=0
      range=60.0

      do i=1,L
         SS(i)=0.0
      enddo

 5    format(a12)
 25   format(f22.19)


*******************************************************
*
 5000 print*, ' '
      print*, ' '
      print*, '************** MAIN MENU ****************'
      print*, ' '
      print*, ' Simplex Method of Optimization'
      print*, ' '
      print*, 'Size of the training images : ',N,' *',N
      print*, 'Grid size for Filter SLM map: ',grain,' *',grain
      print*, 'Magnitude range for search (G):',range
      print*, 'Input SLM :', in_slm_name
      print*, 'Filter SLM:', fil_slm_name
      print*, 'Final Filter:', file2
      print*, ' '
      print*, ' '
      print*, '1. Read Input SLM '
      print*, ' '
      print*, '2. Read Filter SLM'
      print*, ' '
      print*, '3. Read 2D plane to Filter SLM map'
      print*, ' '
      print*, '4. Create 2D plane to Filter SLM map (if you'
      print*, '     pressed option 3, option 4 is not needed)'
      print*, ' '
      print*, '5. Perform Filter Synthesis '
      print*, ' '
      print*, '9. Exit'
      print*, ' '
      print*, 'Select the option : (1,2,...9)-->'
      print*, ' '

      read(*,*) opt
      go to (5010,5020,5030,5040,5050,5060,5070,5080,5090),opt


*     Read the input SLM constraint file
 5010  print*, ' '
      print*, 'Enter input SLM file (eg. amponly.slm) -->'
      read(*,5) in_slm_name
      print*,'Reading ',in_slm_name
      print*,' '

      open(unit=11,file=in_slm_name,status='old')
      do i=1,Q
         read(11,25) mag(i)
      enddo
         read(11,*)
      do i=1,Q
         read(11,25) ph(i)
      enddo
      close(unit=11)
```

```
*     Convert SLM polar values to complex format
      do i=1,Q
         in_slm(i)=cmplx(mag(i)*cos(ph(i)),mag(i)*sin(ph(i)))
      enddo
      go to 5000

*     Read the filter SLM constraint file

 5020 print*, ' '
      print*, 'Enter filter SLM file (phasmost.slm) -->'
      read(*,5) fil_slm_name
      print*,'Reading ',fil_slm_name
      print*,' '

      open(unit=11,file=fil_slm_name,status='old')
      do i=1,Q
         read(11,25) mag(i)
      enddo
         read(11,*)
      do i=1,Q
         read(11,25) ph(i)
      enddo
      close(unit=11)

*     Convert SLM polar values to complex format

      do i=1,Q
         slm(i)=cmplx(mag(i)*cos(ph(i)),mag(i)*sin(ph(i)))
      enddo
      go to 5000


*     Read the map

 5030 print*, ' '
      print*, 'Enter map name to read (eg. phasmost.mpp)'
      read(*,5) map_name

      open(unit=11,file=map_name,form='unformatted',status='old')
       read(11) drive
      close(unit=11)

      go to 5000
```

```
*     Map the 2D plane on filter SLM operating curve using MED

 5040  print*, ' '
       print*, 'Performing mapping..wait'

      do k=1,grain-2
         do j=1,grain-2
            h_plus_real=to_value(k,grain,range)
            h_plus_imag=to_value(j,grain,range)
            dist = 100.0 * range
            do i=1,Q

this_dis=cabs((cmplx(h_plus_real,h_plus_imag))-slm(i))
               if (this_dis .lt. dist) then
                  dist=this_dis
                  drive(k,j)=i
               endif
            enddo
         enddo
      enddo

      print*, 'Do you want to store the map ? (say y or n)'
      read (*,5) res

      if (res .eq. 'y' .or. res .eq. 'Y') then
       print*, 'Give name to store the map (eg. xxxx.mpp) '
       read(*,5) name
       open(unit=11,file=name,form='unformatted',status='unknown')
        write(11) drive
       close(unit=11)
      endif
      go to 5000

 5060 go to 5000
 5070 go to 5000
 5080 go to 5000

*     Get name to store the filter

 5050 print*, ' '
      print*, 'Which method to use for optimization ?'
      print*, '-------------------------------------'
      print*, ' '
      print*, '  1. Simplex Method'
      print*, '  2. Hooke & Jeeves Method'
      print*, ' '
      print*, '    Option (1 or 2) ?'
      read(*,*) opt
      print*, 'Enter Name for Final Filter (eg. xxxx.fil)'
      read(*,*) file2
      print*, ' '
      print*, 'Now I am constructing MACE in the presence of'
      print*, '         input SLM '
      print*, ' '

*     Read initial a values

       print*, 'Initial a values'
       print*, '----------------'
       file1='weight'
       open(unit=11,file=file1,status='old')
         read(11,*) (a(i),i=1,tr_num)
         write(*,*) (a(i),i=1,tr_num)
       close(unit=11)

*     Read desired response at origin

      print*, 'Desired c values'
      print*, '----------------'
      file1 = 'des_peak'
      open(unit=11,file=file1,status='old')
        read(11,*) (c(i),i=1,tr_num)
        write(*,*) (c(i),i=1,tr_num)
      close(unit=11)


*     open training images list file, get fourier transform
*     and calculate average image spectrum.
*     S --> a matrix; each column is the Fourier spectrum of
*           one training image
*     SS--> Average image spectrum vector
*     SW--> Prewhitened image Spectrum
*     in_slm--> maps the image spectrum on input SLM op-curve

      print*, 'Training images'
      print*, '---------------'
      v =1
      file1='train_img'
      open(unit=11,file=file1,status='old')
 10   read(11,5,end=15)name
      print*, name

       call getint1(iimg,name)

       do i=1,N
         do j=1,N
           cimg(i,j)=in_slm(iimg(i,j)+1)
         enddo
      enddo
       call tdfft(1.0,cimg,cfimg)
       u =1
       do i=1,N
          do j=1,N
             S(u,v) = cfimg(i,j)
             SS(u) = SS(u) + (cabs(S(u,v)))**2
             u =u+1
          enddo
       enddo
       v =v+1
       go to 10
 15    close(unit=11)


*     Prewhiten each training image spectrum by spectral envelope

       do v=1,tr_num
          do u=1,L
              SW(u,v)=S(u,v)*tr_num/SS(u)
          enddo
       enddo


*     Synthesis unconstrained MACE

       print*, '---------------------------------------------'
       print*, 'Synthesizing unconstrained MACE filter ...'
       print*, ' '

       do 150 k=1,100
       do u=1,L
          H(u)=(0.0,0.0)
          do v=1,tr_num
```

```
             H(u)=H(u)+a(v)*SW(u,v)
          enddo
       enddo

*     Get value at origin

      do v=1,tr_num
       ORIGIN(v)=(0.0,0.0)
         do u=1,L
            ORIGIN(v)=ORIGIN(v)+S(u,v)*(conjg(H(u)))
         enddo
       enddo

*     Get new a(i)'s

      change=0.0
      do i=1,tr_num
         err=c(i)/c(1)-(cabs(ORIGIN(i))/cabs(ORIGIN(1)))
         Da(i)=alpha*c(1)*err
         change=change+abs(Da(i))
      enddo

      if (change .lt. 1.e-6) go to 160

      do i=1,tr_num
         a(i) = a(i)+Da(i)
      enddo

 150  continue

 160  print*, 'Converged: No. of Iterations :', k
      print*, 'values at origin :'
      write(*,*) (cabs(ORIGIN(i)),i=1,tr_num)
      print*, ' '
      print*, 'Final a(i) values :'
      write(*,*) (a(i),i=1,tr_num)
      print*, '---------------------------------------------'
      print*, ' '
      print*, ' '

c      print*, 'Do you want to store the ideal MACE ? (say y or n)'
c      read (*,5) res

c      if (res .eq. 'y' .or. res .eq. 'Y') then
```

```
c        print*, 'Give name to store the ideal MACE filter '
c        read(*,5) name
c        k=0
c        do i=1,N
c           do j=1,N
c             k=k+1
c             HH(i,j)=H(k)
c           enddo
c        enddo
c        open(unit=11,file=name,form='unformatted',status='unknown')
c         write(11) HH
c        close(unit=11)
c       endif
******************IDEAL MACE DESIGN OVER *********************

       count=0
       if (opt .eq. 2) go to 221


******************Two D search in complex plane***************

      print*, 'Synthesizing constrained MACE filter ...'

 222  print*, 'Count=',count

********************************************************
* Nelder and Mead's method of optimization             *
*                                                      *
*                                                      *
********************************************************

      print*, 'SIMPLEX METHOD OF OPTIMIZATION...'

*     FE --> No. of times the function was called

      FE=0
*     Initial Simplex coordinate (one vertex)

      SIM(1,1)=1.0
      SIM(1,2)=0.0

*     Step length
      step=5.0
```

```
*     setup first Simplex around initial point
      do i=2,3
         do j=1,2
            if (j .eq. (i-1)) then
             SIM(i,j)=SIM(1,j)+step
             go to 370
            endif
            SIM(i,j)=SIM(1,j)
 370     enddo
      enddo

*     Values for alpha, Beta, Gamma

      AL=1.0
      BE=0.5
      GA=2.0

*     Find function values at three vertex

      do i=1,3
         do j=1,2
            X(j)=SIM(i,j)
         enddo
         gain_beta=cmplx(X(1),X(2))
         call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $      slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)
         F(i)=Tot_PCE
      enddo

*     Find greatest and lowest function values and corresponding
*     points

 620  FH=-1.e20
      FL=1.e20
      do i=1,3
         if(F(i) .gt. FH) then
          FH=F(i)
          HI=i
         endif
         if(F(i) .lt. FL) then
          FL=F(i)
          LO=i
         endif
      enddo
```

```
*     Find second greatest value and point
      FG=-1.e20
      do i=1,3
         if(i .eq. HI) go to 800
         if(F(i) .gt. FG) then
          FG=F(i)
          G=i
         endif
 800  enddo

      do j=1,2
         XO(j)=0.0
         do i=1,3
            if (i .eq. HI) go to 910
            XO(j)=XO(j)+SIM(i,j)
 910     enddo
          XO(j)=XO(j)/2.0
          XH(j)=SIM(HI,j)
          XG(j)=SIM(G,j)
          XL(j)=SIM(LO,j)
       enddo

       do j=1,2
          X(j)=XO(j)
       enddo

         gain_beta=cmplx(X(1),X(2))
         call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $      slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

 1120  FO=Tot_PCE

*      Reflection follows

       do j=1,2
          XR(j)=XO(j)+AL*(XO(j)-XH(j))
          X(j)=XR(j)
       enddo

 1220  gain_beta=cmplx(X(1),X(2))
         call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $      slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)
```

```
      FR=Tot_PCE
      print*, 'Reflection: Total PCE', -Tot_PCE

*     If FR < FL EXPANSION
      if (FR .lt. FL) go to 1300

*     If FR > FL and FR > FG test FR and FH
*     otherwise replace XH by XR

      if (FR .gt. FG) go to 1600
      go to 1520

*     Expansion follows

 1300 do j=1,2
         XE(j)=GA*XR(j)+(1-GA)*XO(j)
         X(j)=XE(j)
      enddo
        gain_beta=cmplx(X(1),X(2))
        call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $     slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)


      FEE=Tot_PCE

      if (FEE .lt. FL) go to 1440
      go to 1520

 1440 do j=1,2
         SIM(HI,j)=XE(j)
      enddo

      F(HI)=FEE
      print*, 'Expansion: Total PCE ',-Tot_PCE

*     Test for convergence is at 2060
 1500 go to 2060

 1520 do j=1,2
         SIM(HI,j)=XR(j)
 1560 enddo

      F(HI)=FR
*      print*, 'Reflection:'
```

```
      go to 2060

 1600 if(FR .gt. FH) go to 1700

      do j=1,2
         XH(j)=XR(j)
      enddo

      F(HI)=FR

*     Contraction follows
 1700 do j=1,2
         XC(j)=BE*XH(j)+(1-BE)*XO(j)
         X(j)=XC(j)
      enddo

        gain_beta=cmplx(X(1),X(2))
        call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $     slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      FC=Tot_PCE

      if(FC .gt. FH) go to 1920
      do j=1,2
         SIM(HI,j)=XC(j)
      enddo

      F(HI)=FC
      print*, 'contraction: Total PCE ',-Tot_PCE
      go to 2060

*    Simplex reduction follows
 1920 do i=1,3
        do j=1,2
           SIM(i,j)=(SIM(i,j)+XL(j))/2
           X(j)=SIM(i,j)
        enddo
        gain_beta=cmplx(X(1),X(2))
        call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $     slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      F(i)=Tot_PCE
     enddo
     print*, 'Reduction:'
```

```
*     Test for convergence follows
 2060 S1=0.0
      S2=0.0
      do i=1,3
         S1=S1+F(i)
         S2=S2+F(i)*F(i)
      enddo

      SIG=S2/3.0-S1*S1/9.0

      if (SIG .lt. 1.e-10)  go to 2220
      go to 620

 2220 print*, 'Real and Imag parts in complex plane'
      print*, XL(1),XL(2)
      print*, 'No.of function calls in optimization :',FE
      print*, ' '

*************** Neld & Mead optimization over ***********



*     Now use Relaxation algorithm to stabilize the origin values

 111  write(*,*) 'Relaxation tech. to get equal values at origin'
      write(*,*) '--------------------------------------------'
*     Find out the number of times map_fil was called
      count1=0
*     Synthesize SDF

      gain_beta=cmplx(XL(1),XL(2))

      do 250 k=1,50
      do u=1,L
        H(u)=(0.0,0.0)
        do v=1,tr_num
           H(u)=H(u)+a(v)*SW(u,v)*gain_beta
        enddo
      enddo

*     Get realizable filter using MED method

      call map_fil(H,HM,slm,L,Q,grain,drive,range)
      count1=count1+1
*     Get value at origin

      do v=1,tr_num
      ORIGIN(v)=(0.0,0.0)
      ENERGY(v)=0.0
        do u=1,L
           ORIGIN(v)=ORIGIN(v)+S(u,v)*(conjg(HM(u)))
           ENERGY(v)=ENERGY(v)+((cabs(S(u,v)))**2)*(cabs(HM(u)))**2
        enddo
      enddo


      do i=1,tr_num
        PCE(i)=((cabs(ORIGIN(i)))**2)/(ENERGY(i))
      enddo

      write(*,*) 'ORIGIN:',(cabs(ORIGIN(i)),i=1,tr_num)
      write(*,*) 'ENERGY:', (ENERGY(i),i=1,tr_num)
      write(*,*) 'PCE:', (PCE(i),i=1,tr_num)

*     Get new a(i)'s

      change=0.0
      do i=1,tr_num
         err=c(i)/c(1)-(cabs(ORIGIN(i))/cabs(ORIGIN(1)))
         Da(i)=alpha*c(1)*err
         change=change+abs(Da(i))
      enddo

      write(*,*) 'Da:',(Da(i),i=1,tr_num)

      print*, 'CHANGE :', change
      write(*,*) '-----------------------------------------------'

      if (change .lt. 1.0e-2) go to 260

      do i=1,tr_num
         a(i) = a(i)+Da(i)
      enddo

 250  continue

 260  print*, 'No. of times fn. calls in relaxation :',count1
```

```
      if (count1 .eq. 1)  go to 270

      count=count+1
      if ((count .lt. 3) .and. (opt .eq. 1))go to 222
      if ((count .lt. 3) .and. (opt .eq. 2))go to 221
      go to 270


*****************Relaxation algorithm converged********************



******************Two D search in complex plane****************


 221  print*, 'Count=',count

*********************************************************
*  Hooke & Jeeve method of Optimization                 *
*                                                       *
*                                                       *
*********************************************************

*     Starting coordinate for search in complex plane
*     X1--> real part
*     X2--> Imag part
*     FE -> No of times the function is called

      print*, 'HOOKE & JEEVE OPTIMIZATION...'
      X(1)=1.0
      X(2)=0.0
      FE =0

*     Step for coordinate movement
      step=5.0

*     Initialize
      do i=1,2
         Y(i)=X(i)
         PP(i)=X(i)
         B(i)=X(i)
      enddo

      gain_beta=cmplx(X(1),X(2))
      call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      FI=-Tot_PCE
      print*, 'Total_PCE',-Tot_PCE
      write(*,*) (X(i), i=1,2)

*     Set Flag for basepoint search
      PS=0
      BS=1

*     Elplore about basepoint
      J=1
      FB=FI

 200  X(j)=Y(j)+step
      gain_beta=cmplx(X(1),X(2))
      call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      if(-Tot_PCE .gt. FI) go to 280

      X(j)=Y(j)-step
      gain_beta=cmplx(X(1),X(2))
      call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      if(-Tot_PCE .gt. FI) go to 280

      X(j)=Y(j)
      go to 290

 280  Y(j)=X(j)

 290  gain_beta=cmplx(X(1),X(2))
      call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      FI=-Tot_PCE
      print*, 'exploration step', -Tot_PCE
      write(*,*) ( X(i),i=1,2)

      if(j .eq. 2) go to 360
      j=j+1
      go to 200

*     If function is increased then pattern move
 360  if (FI .gt. FB) go to 540
*     But if exploration was about a pattern point and
*     no increase was made change base at 420
      if ((PS .eq.1) .and. (BS .eq. 0)) go to 420
*     otherwise reduce step length at 490
      go to 490

*     Change Base point
 420  do i=1,2
         PP(i)=B(i)
         Y(i)=B(i)
         X(i)=B(i)
      enddo

      gain_beta=cmplx(X(1),X(2))
      call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      BS =1
      PS=0
      FI=-Tot_PCE
      FB=-Tot_PCE
      print*, 'Base change',-Tot_PCE
      write(*,*) X(1),X(2)

      j=1
      go to 200

*     decrease the step length
 490  step=step/10.0
      print*, 'contract step length'
      if (step .lt. 1.0 e-02)go to 700
      j=1
      go to 200

*     do pattern move
 540  do i=1,2
         PP(i)=2*Y(i)-B(i)
         B(i)=Y(i)
         X(i)=PP(i)
         Y(i)=X(i)
      enddo

      gain_beta=cmplx(X(1),X(2))
      call func(gain_beta,Tot_PCE,H,L,tr_num,a,SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      FB=FI
      PS=1
      BS=0
      FI=-Tot_PCE

 580  print*, 'Pattern move', -Tot_PCE
      write(*,*)X(1),X(2)
      j=1
      go to 200

 700  print*, 'Maximum found at'

      write(*,*)PP(1),PP(2)

      print*, 'Max. Total PCE', FB
      print*, 'No. of function evaluation', FE

      XL(1)=PP(1)
      XL(2)=PP(2)
      go to 111
***********Hooke & Jeeves search is over************************


*      Print final a values

 270   print*, 'Final a values'
       print*, '---------------'
         write(*,*) (a(i),i=1,tr_num)
       u=0
       do i=1,N
          do j=1,N
             u=u+1
             filter(i,j)=HM(u)
          enddo
       enddo

       call kpint2(filter,file2)
      go to 5000

 5090 stop
      end

************************subroutines start here*******************

      subroutine  func(gain_beta,Tot_PCE,H,L,tr_num,a, SW,HM,
     $ slm,Q,ORIGIN,ENERGY,S,PCE,P,FE,grain,drive,range)

      integer*2 tr_num,L,Q,P,FE,grain,drive(grain,grain)
      real Tot_PCE,a(tr_num),ENERGY(tr_num),PCE(tr_num),range
      complex H(L),SW(L,P),gain_beta,HM(L),slm(Q),ORIGIN(P),S(L,P)

*     Synthesize SDF

      print*, '-----------------------------------------------------'
      do u=1,L
         H(u)=(0.0,0.0)
         do v=1,tr_num
            H(u)=H(u)+a(v)*SW(u,v)*gain_beta
         enddo
      enddo

*     Get realizable filter using MED method

      call map_fil(H,HM,slm,L,Q,grain,drive,range)


*     Get value at origin and correlation plane energy and PCE

      do v=1,tr_num
      ORIGIN(v)=(0.0,0.0)
      ENERGY(v)=0.0
         do u=1,L
            ORIGIN(v)=ORIGIN(v)+S(u,v)*(conjg(HM(u)))

ENERGY(v)=ENERGY(v)+((cabs(S(u,v)))**2)*(cabs(HM(u)))**2
         enddo
      enddo

      do i=1,tr_num
         PCE(i)=((cabs(ORIGIN(i)))**2)/(ENERGY(i))
      enddo
      write(*,*) 'ORIGIN:',(cabs(ORIGIN(i)),i=1,tr_num)
      write(*,*) 'ENERGY:', (ENERGY(i),i=1,tr_num)
      write(*,*) 'PCE:', (PCE(i),i=1,tr_num)

*     Calculate Total PCE

      Tot_PCE=0.0
      do i=1,tr_num
         Tot_PCE=Tot_PCE+PCE(i)
      enddo

*    -ve sign needed for optimization routine

      Tot_PCE=-Tot_PCE
      FE=FE+1

      return
      end


***************************************************************
*     Subroutine mapping

*     H-->unrealizable filter
*     HM->mapped (realizable) filter

      subroutine map_fil(H,HM,slm,L,Q,grain,drive,range)


      integer*2 L,Q,i,k,kk,grain,drive(grain,grain)
      real range
      complex H(L),HM(L),SLM(Q)

      do i=1,L
            k=nint(to_index(real(H(i)),grain,range))
            kk=nint(to_index(aimag(H(i)),grain,range))
            HM(i)=slm(drive(k,kk))
      enddo

      return
      end
```

```
**************************************************************
*
*     Function to convert array index to value

      function to_value(index,grain,range)

      integer*2 index,grain
      real range
      to_value=range*(2.0*index-grain+1)/(grain-3.0)
      return
      end


**************************************************************
*
*     Function to convert value to array index

      function  to_index(value,grain,range)

      integer*2 grain
      real range,sign,value,amag_lim

      if(value .gt. 0.0)then
        sign=1.0
      else
        sign=-1.0
      endif

      amag_lim=sign*amin1(abs(value),range)
      to_index=(amag_lim/range)*1.0*((grain-3)/2)+(grain-1)/2

      return
      end

**************************************************************
*
*     Subroutine to read an integer array


      SUBROUTINE GETINT1(ARRAY, ARRAYNAME)

      COMMON N
      INTEGER*2 ARRAY(N,N),N
      CHARACTER*12 ARRAYNAME

      OPEN(UNIT=8,FILE=ARRAYNAME,FORM='UNFORMATTED',STATUS='OLD')
            READ(8) ARRAY
      CLOSE(UNIT=8)

      RETURN
      END

**************************************************************
*
*     Subroutine to store a complex array


      SUBROUTINE KPINT2(ARRAY, ARRAYNAME)

      COMMON N
      INTEGER*2 N
      COMPLEX ARRAY(N,N)
      CHARACTER*12 ARRAYNAME


OPEN(UNIT=8,FILE=ARRAYNAME,FORM='UNFORMATTED',STATUS='UNKNOWN')

      PRINT *,' '
      PRINT *,'Saving the file '//ARRAYNAME

            WRITE(8) ARRAY

      CLOSE(UNIT=8)

      PRINT *,'File transfer completed'
      PRINT *,' '

      RETURN
      END




      SUBROUTINE TDFFT(D, ARRAYIN, ARRAYOUT)
```

```
C*****************************************************************
C*****************************************************************
C
C  A subroutine to index an array and call the one-dimensional
C  Fast Fourier Transform, SUBROUTINE FFT, for each column and then
C  each row of the array.
C
C
C*****************************************************************
C
C     Arguments:
C
C     D         > r   Direction of the transform:  1. = forward
C                                              -1. = inverse
C     ARRAYIN   > ca  The two-dimensional array to be transformed.
C     ARRAYOUT  < ca  The two-dimensional transformed array.
C
C     N         : i   Dimension of the array    [COMMON]
C
C******************************************************************


      COMMON N

      INTEGER*2  N
      INTEGER*2  R, C
      REAL D
      COMPLEX ARRAYIN(N,N), ARRAYOUT(N,N), B(1024)


      DO 60 C = 1, N
         DO 20 I = 1, N
            B(I) = ARRAYIN(C, I)
   20    CONTINUE
         CALL FFT (D, B)
         DO 40 I = 1, N
            ARRAYOUT(C, I) = B(I)
   40    CONTINUE
   60 CONTINUE
      DO 160 R = 1, N
         DO 120 I = 1, N
            B(I) = ARRAYOUT(I, R)
  120    CONTINUE
         CALL FFT (D, B)
         DO 140 I = 1, N
            ARRAYOUT(I, R) = B(I)
  140    CONTINUE
  160 CONTINUE
      RETURN
      END



      SUBROUTINE FFT(DIR, VEC)

C*******************************************************************
C*******************************************************************
C
C     A Fast Fourier Transform from Digital Signal Processing
C     by Oppenheim and Schafer.
C
C
C*******************************************************************
C
C     Arguments:
C
C     DIR     > r   The direction of the transform:  1. = forward
C                                                   -1. = inverse
C     VEC <> ca  The one dimensional transform array.
C
C
C  N       : i   The number of points in the transform (must be an
C                                  integer power of 2).
C
C                                    [COMMON]
C
C*******************************************************************

      COMMON N

      INTEGER*2 N
      INTEGER  NV2, NM1, M, MT
      REAL DIR
      COMPLEX VEC(N), U, W, T

      M = 0
    5 M = M + 1
      MT = 2**M
```

```
   IF (MT .NE. N) GO TO 5
   NV2 = N/2
   NM1 = N - 1
   J = 1
   DO 30 I = 1, NM1
      IF (I .GE. J) GO TO 10
      T = VEC(J)
      VEC(J) = VEC(I)
      VEC(I) = T
10    K = NV2
20    IF (K .GE. J) GO TO 30
      J = J - K
      K = K/2
      GO TO 20
30    J = J + K
   PI = 3.141592653589793
   DO 50 L=1, M
      LE = 2**L
      LE1 = LE/2
      U = CMPLX(1.,0.)
      W = CMPLX(COS(PI/FLOAT(LE1)), -DIR*SIN(PI/FLOAT(LE1)))
      DO 50 J = 1, LE1
         DO 40 I=J, N, LE
            IP = I + LE1
            T = VEC(IP)*U
            VEC(IP) = VEC(I) - T
40       VEC(I) = VEC(I) + T
50 U = U*W
   IF (DIR .EQ. 1.) GO TO 70
   DO 60 I = 1, N
      VEC(I) = VEC(I)/REAL(N)
60 CONTINUE
70 CONTINUE
   RETURN
   END
```

```
**************************************************************
* Program   : slm-mace2.f                                    *
*                                                            *
* Purpose   : Implements SLM constrained MACE filter;        *
*             considers both input and filter SLMs (LCTV)    *
*                                                            *
* Algorithm : Uses simulated annealing based optimization    *
*                                                            *
* First author: Khan, A.                                     *
*                                                            *
* Updated by  : Ramakrishnan, R.             Date: July' 94  *
*                                                            *
* Modifications : 1. Routines to accept input SLM            *
*                 2. Instead of Phase perturbation, now      *
*                    SLM's drive perturbation                *
*                 3. New Objective function                  *
*                 4. Initial mapping of ideal filter values  *
*                    by MED mapping instead of Equi-phase    *
*                    mapping                                 *
*                                                            *
* Version     : 02                                           *
*                                                            *
**************************************************************

C
C = Definitions of some important variable names
C
C
C N,M == NxN or MxM are the Image/Filter dimensions.
C
C L == Total no. of pixels in the Images ( i.e. = NxN).
C
C KKK == Number of training Images.
C
C iter == Maximum number of iterations permitted.
C
C ased,bsed,sed == Contain the seed values for use in the random
C No.generation function.These seed values are actually
C generated using the 'secnds' function in FORTRAN (i.e. it depends
C on the real clock time).
C
C val1 == User-specified peak-values for Class I
C
C k1 == Constant used in the Penalty functions for the constraints
```

```
C               of the objective function.
C
C XX == The adjacent columns of this array contain the real and
C   imaginary
C dmat == Array of dimension 4096. Contains the average of the
C (magnitude)**2 of all the training Image vectors.
C
C cst == Array of dimension 4096x2. Contains the SLM-contraints in
C two columns representing the phase and magitude respectively.
C
C drv_old == Array of dimension L. It contains the temporary values
C of the drive of the filter pixels during optimization.
C
C drv_new == Array of dimension L. It will contain the final drive
C values of the pixels of the optimized filter.
C
C objf == Contains the value of the objective function resulting
C from a successful perturbation of a variable during an iteration
C of the optimization process. The function 'computefn' also
C nreturns the value of the objective function in 'objf'.
C NOTE: A single iteration
C consists of going through every variable (filter pixel).
C Ofcourse, at the end of the optimization 'objf' will contain
C the finalobjective  function value.
C
C FF == Dimension 2. Used to store the temporary values of the
C objective function as a result of a perturbation.
C
C objo == Contains the value of the average correlation plane
C energy.
C nobjo == Used to store the temporary value of the corr. plane
C energy as a result of the perturbation of filter pixel drive
C value.
C
C con == Array of dimension KKKx2 .Holds the real and imaginary
C parts of the Corr peak constraints as the two entries of
C each row.
C ncon== Array of dimension KKKx2. Holds the temporary values of
C the above constraints.
C
C min_drv == array of dimension L. (see below for explanation)
C
C minobjf == Contains the least value of the objective function
C obtained during the optimization. After every iteration, the
```

```
C value in 'objf' is
C compared to 'minobjf'. The smaller of the two values is
C retained in
C 'minobjf'. The corresponding drive values of the filter pixels
C are stored in the array 'min_drv'. This is done by calling
C SUBROUTINE REC.
C
C At the end of the optimization, if the value of the objective
C function in
C 'minobjf' is smaller than the final value of the objective
C function in 'objf'
C then, the values from 'minX' are moved to 'X' and these are
C considered the
C optimal phase values of the filter.
C
C
******************************************************************

      integer*2 N,LC
      integer mm,oncem,r
      integer NN,M,L,KKK,iter,dev1
      integer sed,ased,bsed,pp,count
         integer accept,attempt
      real sed1,sed2,sed3,minobjf,thi,nobjo,nobjo1,objo1,bstobjf
      parameter(M=128)
      parameter(L=M*M)
         parameter(KKK=7)
         parameter(LC=256)
      integer*2 ipint(M,M)
      integer drv_new(L),drv_old(L),drv_min(L),drv_bst(L)
         real perc,dmat(L),FF(2),min,max,oldf
      real ipr(M,M),XX(L,20)
      real mag(LC),ph(LC)
      real val1,w,temp0,ncon1(KKK,1)
      real randnum1,randnum2,prob,temp
      complex ipc(M,M),ipf(M,M),filt(L),filt1(M,M),slm(LC)
         complex in_slm(LC)
      real objo,objf,xi,xr,txi,txr,con(KKK,2),ncon(KKK,2)
      character*12 filename1,filename2,filename3,name
      common N

      N=M
      dev1=6
      sed1=40000.0
```

```
     sed2=50000.0
     sed3=90000.0
        pp = 0
        accept = 0
        attempt = 0
        NN = 0
        oncem=0
        mm=0

 12  format(a12)
 13  format(f22.19)

C Generate the seed values for use in the random no. generation
function 'RAN'

     ased=abs(2*(int(secnds(sed1)))+1)
        sed=abs(2*(int(secnds(sed2)))+1)
     bsed=abs(2*(int(secnds(sed3)))+1)
        iter = 5000
        w=2.0

*       Get desired value at origin

     print*, 'Give desired value at origin'
        read(*,*) val1

*       Read input SLM

      print*, 'Enter input SLM file name-->'
     read(*,12) name

      open(unit=2,file=name,status='old')
      do i=1,LC
         read(2,13) mag(i)
      end do
     read(2,*)
      do i=1,LC
         read(2,13) ph(i)
      end do

      close(unit=2)

*********************************************************
* CONVERT SLM POLAR VALUES TO COMPLEX FORMAT
```

```
      do i =1,LC
       in_slm(i)=cmplx(mag(i)*cos(ph(i)),mag(i)*sin(ph(i)))
      enddo

*******************************************************************

C 'images4.na' contains the names of the training images which will
C be read one after another. The real and imaginary part of the
fourier
C transformed images are stored as columns of the array XX. The
magnitude
C squared values of the fourier transformed image pixels are
computed
C and stored in 'dmat'.

        filename1='train_img'

        open(unit=11,file=filename1,status='old')
 32     read(11,12,end=900)filename2
     print*, 'reading ...', filename2

        call getint1(ipint,filename2)

        do i=1,N
           do j=1,N
              ipc(i,j)=in_slm(ipint(i,j)+1)
           end do
        end do
        call tdfft(1.0,ipc,ipf)

        mm=mm+1
        jj=0
        do i=1,N
           do j=1,N
              jj=jj+1
              XX(jj,mm) = real(ipf(i,j))
              XX(jj,mm+1) = aimag(ipf(i,j))
              dmat(jj) = dmat(jj)+(XX(jj,mm)**2)+(XX(jj,mm+1)**2)
           end do
        end do
        mm=mm+1
        go to 32
```

```
C Take the average of all the elements of 'dmat'.

 900    do i=1,L
           dmat(i)=dmat(i)/(float(KKK*L))
        end do

C Read the SLM-constraint data file into the array 'cst'.



      print*, 'Enter filter SLM file name-->'
     read(*,12) name

      open(unit=2,file=name,status='old')
      do i=1,LC
         read(2,13) mag(i)
      end do
     read(2,*)
      do i=1,LC
         read(2,13) ph(i)
      end do

      close(unit=2)

**********************************************************
* CONVERT SLM POLAR VALUES TO COMPLEX FORMAT
      do i =1,LC
       slm(i)=cmplx(mag(i)*cos(ph(i)),mag(i)*sin(ph(i)))
      enddo

*******************************************************************

C Determine the starting points for the optimization.
C Here the starting SLM drive values for the filter pixels are
those obtained
C from the previously solved Composite MACE filter mapped on
Realizable SLM
C curve using Equilidian distance concept.
C 'GETCMP' is an unformatted-read  routine which reads the complex
filter
C into an array 'filt1'.

      print*, 'Enter ideal MACE filter name-->'
     read(*,12) name
```

```
      call getcmp(filt1,name)

      print*, 'Enter output filter name-->'
        read(*,12) filename3

* MAP THE FILTER ON SLM CURVE AND CREATE DRIVE ARRAY

     print*, 'Performing mapping ... wait'
     k=0
     do i =1,N
        do j =1,N
           max=9999999.
           k=k+1
           do r=1,LC
           min=(cabs((filt1(i,j)-slm(r))))
           if (min .lt. max) then
              max=min
              drv_old(k)=r
                endif
              enddo
           enddo
         enddo


C Evaluate the cost function with the initial guess. The routine
'camputefn'
C evaluates the objective function value at the given points of the
filter
C pixel values.
* objo --> Correlation plane Energy
* objf --> Objective function
* objo1--> Penalty function

 72   format(1x,68('-'))
      print 72
      print*, ' '
      print*, 'Initial Values'
      print*, '---------------'
      print*, 'Specified Peak Value at Correlation centre->',val1
        call computefn(L,KKK,XX,drv_old,dmat,slm,w,val1,objo,objf,
     $                 con,objo1)
     bstobjf=objf
     do i =1,L
          drv_bst(i)=drv_old(i)
        enddo

 420 minobjf=objf
     do i =1,L
          drv_min(i)=drv_old(i)
        enddo

     if(oncem.ne.0)then
           perc=((oldf-objo)/oldf)*100.0
           oldf=objo
           if(perc.le.(2.0))then
              count=count+1
           else
              count=0
           end if
      else
         oldf=objo
        end if

C The initial value of the temperature 'temp0' is got from the
starting value
C value of the objective function 'objf'.
C 'temp0' does not change during the entire run of the
optimization. The
C temperature parameter which is updated as the iterations progress
is 'temp'.
C Here some data files are also created.

     FF(1)=objf
        temp0=objf/2.0
        temp=temp0
     print*, ' '
      print*, 'Start Temperature :',temp
     print*, ' '

        if(count.ge.3)go to 500
        oncem=oncem+1

C  ****  The optimization iterations begin here.
***********************

     write(dev1,*)' PERFORMING OPTIMIZATION !! PLEASE WAIT'
     write(dev1,*)'  '
     do i=1,iter
        do j=1,L
               pp=pp+1

C Generate a random no. to determine whether a positive or
C negative perturbation be caused for each variable (filter SLM
C drive value).

               randnum1=ran(ased)
               if(randnum1.le.0.5)then
                  drv_new(j)=drv_old(j)+1
                  if((drv_new(j)).gt.LC)then
                     drv_new(j)=LC
            end if
            else
                  drv_new(j)=drv_old(j)-1
               if((drv_new(j)).lt.1)then
                     drv_new(j)=1
               end if
               end if

C The new objective function value due to the perturbation is
computed here. ThC value is stored in FF(2). The old value of the
objective function is stored iC FF(1). NOTE: Here the entire
objective function is not evaluated. Rather the
C change in the old objective function value caused by the
perturbation is
C computed.
* txr --> Real part of old realizable filter
*  xr --> Real part of new realizable filter

         txr=real(slm(drv_old(j)))
         txi=aimag(slm(drv_old(j)))
          xr=real(slm(drv_new(j)))
          xi=aimag(slm(drv_new(j)))

* Get New  Correlation Energy due to 1 pixel perturbation
* Subtract old and add new energy
* Eav = (cabs(XH))**2
* nobjo --> new energy after perturbation

               nobjo=objo-(dmat(j)*(txr**2+txi**2))+(dmat(j)*
     $               (xr**2+xi**2))
```

```
* Now calculate Correlation peak at origin ( conjg(X)H = C(0))
* ncon(ii,1) --> Real part of peak
* ncon(ii,2) --> Imag part of peak
                mm=0
                do ii=1,KKK
                   mm=mm+1
                   ncon(ii,1)=con(ii,1)+(XX(j,mm)*(xr-txr))
     $                       +(XX(j,mm+1)*(xi-txi))
                   ncon(ii,2)=con(ii,2)+(XX(j,mm)*(xi-txi))
     $                        +(XX(j,mm+1)*(txr-xr))

*  Absolute Peak value at center

                   ncon1(ii,1)=sqrt(ncon(ii,1)**2+ncon(ii,2)**2)
                   mm=mm+1
                end do

                nobjo1=0.0

* Calculate Penalty Function
                do ii=1,KKK
                  nobjo1=nobjo1 + w*abs(ncon1(ii,1)-val1)
                end do

* New Objective function (Eav + penalty fn)
                FF(2)=nobjo+nobjo1

C If the new objective function value (i.e. due to the
perturbation) is smaller
C than the current value then the perturbation is accepted, and the
objective
C function value 'objf' is updated along with all the variables
used to
C to compute it. For an accepted perturbation the counter 'accept'
is
C incremented.

                if(FF(2).lt.FF(1))then
                   FF(1)=FF(2)
                   objo=nobjo
                   do ii=1,KKK
                      con(ii,1)=ncon(ii,1)
                      con(ii,2)=ncon(ii,2)
                   end do
                   objf=FF(2)
             drv_old(j)=drv_new(j)
                   accept=accept+1
                   go to 701
                end if


C If the perturbation results in an increase in the objective
function value
C it will only be accepted with a certain probability, computed in
the variable
C 'prob' below. Again if this perturbation is accepted then the
objective
C function value 'objf' is updated along with the variables used to
compute it.

* As temperature falls prob reduces. So acceptance rate reduces

                thi=(FF(2)-FF(1))/temp
                if(thi.ge.80.0)then
                   thi=80.0
                elseif(thi.le.0.000001)then
                   thi=0.0
                end if
                prob=1.0/(1.0+exp(thi))
          if((i .eq. 1) .and.(j .eq.1))
     $         print*, 'Start Probability =',prob
                randnum2=ran(bsed)
                if(randnum2.lt.prob)then
                   FF(1)=FF(2)
                   objo=nobjo
                   do ii=1,KKK
                      con(ii,1)=ncon(ii,1)
                      con(ii,2)=ncon(ii,2)
                   end do
                   objf=FF(2)
             drv_old(j)=drv_new(j)
                   accept=accept+1
                   go to 701
                end if

 701       if(objf.lt.minobjf) then
           drv_min(j)=drv_old(j)
           minobjf =objf
```

```
         endif

*  Do perturbation for next pixel
           end do



* After perturbing all the filter pixels now proceed to ....


C ********** Updating the Temperature parameter **********

C If the no. of accepted pertubations >= 10xL then update the
temperature
C parameter 'temp', and proceed. Initialize the counter 'attempt',
'accept'.

            if(accept.ge.10*L)then
               temp=((0.90)**i)*temp0
               accept=0
               pp=0
            attempt=0
            end if

C If the no. of attempted perturbations (without acceptance) is >=
100xL then
C update the temperature parameter and make a note of this fact by
C by incrementing the counter 'attempt'.

            if(pp.ge.100*L)then
               temp=((0.90)**i)*temp0
               accept=0
               pp=0
               attempt=attempt+1
            end if


C If the above has occured thrice continously, then terminate the
optimization.
C This is the convergence test. If so then branch to 200.

            if(attempt.ge.3)then
            print*, 'Converged due to attempt=',attempt
               attempt=0
```

```
                accept=0
                go to 200
             end if

             if(temp.le.(1.0E-15))then
             print*, 'Converged due to temperature=',temp
                attempt=0
                accept=0
                go to 200
             end if
*  Go for next perturbation (one perturbation = 4096 pixel
perturbation)
 779     end do



C If the convergence criterion is not met after 'iter' iterations,
then quit
C the program. ( This so that the program does not get into an
endless loop).

     write(dev1,*)'   '
     write(dev1,*)' **************************** '
     write(dev1,*)' ERROR ERROR ERROR '
     write(dev1,*)' **************************** '
     write(dev1,*)'  '
     write(dev1,*)' No convergence after ',i,' iterations'
     go to 500

200     write(dev1,*) '                              '
     write(dev1,*)' ********************************* '
     write(dev1,*)' Converged after ',i,' iterations'
     write(dev1,*)'  '
        write(dev1,*)'RESULTS:'
        write(dev1,*)'--------'
     write(dev1,*)'  '
     print*, 'End Temperature :',temp

     call computefn(L,KKK,XX,drv_min,dmat,slm,w,val1,objo,objf,
     $                 con,objo1)

     print 72
     print*, ' '
     if (objf .lt. bstobjf) then
```

```
        bstobjf =objf
           do i =1,L
           drv_bst(i)=drv_min(i)
           drv_old(i)=drv_min(i)
        enddo
      else
           do i=1,L
           drv_old(i)=drv_bst(i)
        enddo
        end if

C The optimization is repeated with increased values of the
constraints
C penalty violation constants 'W'

        if(oncem.ne.20)then
           w=w+2.0

        print*, 'Start Values for Next Iteration'
        print*, '------------------------------'
     call computefn(L,KKK,XX,drv_old,dmat,slm,w,val1,objo,objf,
     $                 con,objo1)
           go to 420
        end if

C Using the drive values of the filter pixels from the array
'drv_bst'
C and compute the complex filter pixel values.
C Store these complex filter pixel values in the array 'filt'.


 500       print*, 'Finish due to count=',count
           do i=1,L
           filt(i)=slm(drv_bst(i))
           end do

C Arrange the filter from a vector form into a matrix form.

        jj=0
        do i=1,N
           do j=1,N
              jj=jj+1
              filt1(j,1)=filt(jj)
           end do
        end do

C  Save the final filter in a file for future correlations


     call kpint2(filt1,filename3)

C  Write the final value of the objective function and constraints.
     print*, '           '
     print*, 'Final Values'
     print*, '------------'
     call computefn(L,KKK,XX,drv_bst,dmat,slm,w,val1,objo,objf,
     $                con,objo1)

     stop
     end


***************************************************************
******
       subroutine computefn(L,KKK,XX,drive,dmat,slm,w,val1,objo,
     $    objf,con,objo1)

C
***************************************************************
***
C This routine calculates the value of the objective function, the
C value of the correlation energy plane, and value of the
constraints.
C XX is a 2 dimensional array containing the real part of the
C elements of the reference image in the first column and the
imaginary
C parts of the elements in the second column.
C L  (input)
C XX (input array) == Each of the columns correspond to the real
and imaginary
C                     part of each of the fourier trandformed
images.
C W (input) == as defined in the main program.
C val1,  (input) == as defined in the main program.
C objo,objf (output) == corr. plane energy value, and the objective
function
C                       value respectively.
C con (output array) == Real part of the constraints at the origin
```

```
of corr
C   plane.
C
C
*****************************************************************
******

      integer*2 LC
      integer L,KKK,i,j,drive(L)
      parameter(LC=256)
      real XX(L,20),w
      real objo,objf,objo1,con(KKK,2),dmat(L),con1(7,1)
      real rlf,imf,val1
      complex slm(LC)

      objo=0.0
      objo1=0.0
      objf=0.0

* con(i,1)--> Real part of correlation peak
* con(i,2)--> Imag part of correlation peak

       do i=1,KKK
         con(i,1)=0.0
         con(i,2)=0.0
      end do

*    objo --> Energy
*    objo1 --> Penalty fn
*    objf  --> Merit fn = Energy + Penalty fn

* Calculate Energy in the Correlation plane
      do j=1,L
      rlf=real(slm(drive(j)))
      imf=aimag(slm(drive(j)))
         objo=objo + (dmat(j)*(rlf**2+imf**2))

        mm=0
        do i=1,KKK
           mm=mm+1
           con(i,1)=con(i,1) + ((XX(j,mm)*rlf)+(XX(j,mm+1)*imf))
           con(i,2)=con(i,2) + ((XX(j,mm)*imf)-(XX(j,mm+1)*rlf))
           mm=mm+1
        end do
      end do

       print*, 'Energy=', objo
* Get absolute value of Peak
      do i=1,KKK
        con1(i,1)=sqrt(con(i,1)**2+con(i,2)**2)
      enddo

* Get the sum of weighted deviations from the specified value
* (Penalty fn.)
      objo1=0.0
      do i=1,KKK
         objo1=objo1 + w*abs(con1(i,1)-val1)
      end do

* Objective function (Energy + Penalty fn.)
        objf=objo + objo1

* Show results
     print*,  ' '
     print*, 'Objective function value=',objf
     print*, 'Correlation Plane Energy=',objo
     print*, 'Penalty function value  =',objo1
     print*, 'Correlation Peaks at origin are:'
     print*, '------------------------------'

     do i=1,KKK
        print*, 'Image',i, sqrt(con(i,1)**2+con(i,2)**2)
     end do
     print*, ' '

        return
     end
******************************************************************


      SUBROUTINE GETINT1(ARRAY, ARRAYNAME)

      COMMON N
      INTEGER*2 ARRAY(N,N),N
      CHARACTER*12 ARRAYNAME

      OPEN(UNIT=8,FILE=ARRAYNAME,FORM='UNFORMATTED',STATUS='OLD')
```

```
              READ(8) ARRAY

        CLOSE(UNIT=8)


        RETURN
        END

******************************************************************


        SUBROUTINE GETCMP(ARRAY, ARRAYNAME)

C*****************************************************************
*******
C*****************************************************************
*******
C
C     A routine to read a complex N by N array.  The data is stored
in
C     the 'UNFORMATTED' form to speed the file transfer.  The array
data
C     is read by starting with the upper left element of the array
C     and proceeding across to the upper right.  The rows are
C     incremented downward so that the last element in the data
file
C     is the lower right element of the array.
C
C
C*****************************************************************
*******
C
C     Arguments:
C
C        ARRAY      <     Ia  Integer array.
C        ARRAYNAME  >     ch  File name (name.ext)
C
C        N          :     i   Dimension of the array [COMMON]
C
C*****************************************************************
*******
```

```
      COMMON N
      INTEGER*2 N
      COMPLEX ARRAY(N,N)
      CHARACTER*12 ARRAYNAME


OPEN(UNIT=8,FILE=ARRAYNAME,FORM='UNFORMATTED',STATUS='UNKNOWN')

        PRINT *,' '
        PRINT *,'Reading the file '//ARRAYNAME

        READ(8) ARRAY

      CLOSE(UNIT=8)

      PRINT *,'File transfer completed.'
      PRINT *,' '

      RETURN
      END



      SUBROUTINE KPINT2(ARRAY, ARRAYNAME)

C****************************************************************
*******
C****************************************************************
*******
C
C     A routine to save an complex N by N array.  The data is
stored in
C     the 'UNFORMATTED' form to speed the file transfer.  The array
data
C     is stored by starting with the upper left element of the
array
C     and proceeding across to the upper right.  The rows are
C     incremented downward so that the last element in the data
file
C     is the lower right element of the array.
C
C
C****************************************************************
*******
```

```
C
C     Arguments:
C
C        ARRAY     >     ia  Integer array.
C        ARRAYNAME >     ch  File name (name.ext)
C
C        N         :     i   Dimension of the array [COMMON]
C
C****************************************************************
*******


      COMMON N
      INTEGER*2 N
      COMPLEX ARRAY(N,N)
      CHARACTER*12 ARRAYNAME


OPEN(UNIT=8,FILE=ARRAYNAME,FORM='UNFORMATTED',STATUS='UNKNOWN')

      PRINT *,' '
      PRINT *,'Saving the file '//ARRAYNAME

            WRITE(8) ARRAY
      CLOSE(UNIT=8)

      PRINT *,'File transfer completed'
      PRINT *,' '

      RETURN
      END



      SUBROUTINE TDFFT(D, ARRAYIN, ARRAYOUT)

C****************************************************************
*******
C****************************************************************
*******
C
C     A subroutine to index an array and call the one-dimensional
C     Fast Fourier Transform, SUBROUTINE FFT, for each column and
then
```

```
C     each row of the array.
C
C
C****************************************************************
*******
C
C     Arguments:
C
C     D         > r   Direction of the transform:  1. = forward
C                                                 -1. = inverse
C     ARRAYIN   > ca  The two-dimensional array to be transformed.
C     ARRAYOUT  < ca  The two-dimensional transformed array.
C
C     N         : i   Dimension of the array   [COMMON]
C
C****************************************************************
*******


      COMMON N

      INTEGER*2  N
      INTEGER*2  R, C
      REAL D
      COMPLEX ARRAYIN(N,N), ARRAYOUT(N,N), B(1024)


      DO 60 C = 1, N
         DO 20 I = 1, N
            B(I) = ARRAYIN(C, I)
   20    CONTINUE
         CALL FFT (D, B)
         DO 40 I = 1, N
            ARRAYOUT(C, I) = B(I)
   40    CONTINUE
   60 CONTINUE
      DO 160 R = 1, N
         DO 120 I = 1, N
            B(I) = ARRAYOUT(I, R)
  120    CONTINUE
         CALL FFT (D, B)
         DO 140 I = 1, N
            ARRAYOUT(I, R) = B(I)
  140    CONTINUE
  160 CONTINUE
      RETURN
      END




      SUBROUTINE FFT(DIR, VEC)

C*****************************************************************
*******
C*****************************************************************
*******
C
C     A Fast Fourier Transform from Digital Signal Processing
C     by Oppenheim and Schafer.
C
C
C*****************************************************************
*******
C
C     Arguments:
C
C     DIR     > r   The direction of the transform:  1. = forward
C                                                   -1. = inverse
C     VEC <> ca  The one dimensional transform array.
C
C
C     N       : i   The number of points in the transform (must be
an
C                                 integer power of 2).
C
C                                   [COMMON]
C
C*****************************************************************
*******

      COMMON N

      INTEGER*2 N
      INTEGER  NV2, NM1, M, MT
      REAL DIR
      COMPLEX VEC(N), U, W, T
```

```
   M = 0
 5 M = M + 1
   MT = 2**M
   IF (MT .NE. N) GO TO 5
   NV2 = N/2
   NM1 = N - 1
   J = 1
   DO 30 I = 1, NM1
      IF (I .GE. J) GO TO 10
      T = VEC(J)
      VEC(J) = VEC(I)
      VEC(I) = T
10    K = NV2
20    IF (K .GE. J) GO TO 30
      J = J - K
      K = K/2
      GO TO 20
30    J = J + K
   PI = 3.141592653589793
   DO 50 L=1, M
      LE = 2**L
      LE1 = LE/2
      U = CMPLX(1.,0.)
      W = CMPLX(COS(PI/FLOAT(LE1)), -DIR*SIN(PI/FLOAT(LE1)))
      DO 50 J = 1, LE1
         DO 40 I=J, N, LE
            IP = I + LE1
            T = VEC(IP)*U
            VEC(IP) = VEC(I) - T
40       VEC(I) = VEC(I) + T
50 U = U*W
   IF (DIR .EQ. 1.) GO TO 70
   DO 60 I = 1, N
      VEC(I) = VEC(I)/REAL(N)
60 CONTINUE
70 CONTINUE
   RETURN
   END
```