Asian Journal of Physics Vol 11, No 2 (2002) 1-12

Invited Paper

Similarity suppression algorithm for designing pattern discrimination filters

David R. Selviah and Epaminondas Stamos
Department of Electronic and Electrical Engineering
University College London, Torrington Place, London, WC1E 7JE, UK
TEL: +44 20 7679 3056, FAX: +44 20 7388 9325
Email: d.selviah@ee.ucl.ac.uk

The similarity suppression (SS) algorithm is introduced which designs inner product correlator filters for optical pattern recognition and discrimination. The algorithm which mutually orthogonalises two sets of patterns is compared to the matrix design technique for Linear Combination Filters (LCF) and is shown to converge to a similar set of filters. Computer simulations for discrimination of a set of 32, 16x16 binary, bipolar patterns demonstrate that the designed filters can tolerate 7 dB more noise than matched filters. for the same performance and the dynamic range required is reduced by 25 dB making them suitable for use in optical correlators. Moreover, filters designed after only 2 iterations were found to have much better discrimination ability than LCFs in the important case of high noise and low SNR. Their outer products were also found to be lower.

Key words: pattern recognition, correlators, inner product filters, filter banks, noise robustness, orthogonal, optical, algorithm.

1 Introduction

Optical correlators are particularly useful for fast, two dimensional image recognition when the input image is entered into the correlator system directly from the real world, preserving all of the amplitude detail, noise and phase information of the original. There has been a great deal of work on the development of filters, for use in optical correlators, which can recognise the presence of any one of several patterns in the input, for example, SDF filters[1-4] and optimal trade-off filters[5-7]. Input patterns are invariably covered in noise (either additive, multiplicative or non-overlapping[8]) and noise is added by the optical system itself due to laser speckle, dust and vibration. The aim of this paper is to develop an algorithm to strongly distinguish between known patterns in the presence of noise in a limited dynamic range optical correlator, by designing a set of filters which give the maximum difference between their outputs. Each pattern to be distinguished has its own discrimination filter. The filters are designed in the space domain, so they are suitable for use in space domain inner product correlators[9]. Of course, our filter design can be used in a Fourier domain correlator[10] but, in general, the Fourier Transform of our space domain filter will have complex elements. It is difficult to implement programmable, fully complex filters which can give any complex value on the unit disc in the complex plane[11-14], although programmable filters exist which can access part of the complex plane[6]. The filters we design in this paper have pixel values along a line through the origin in the complex plane, as a particular case of a complex filter. The pixels can, therefore, take both positive and negative real values between +1 and -1 including zero, without loss of generality.

Matched filters which are the complex conjugates of the spectrum of the original patterns, are optimal for detecting signals in white, Gaussian noise[15]. However, if the task at hand is instead to distinguish between a number of patterns, a different filter is required. As a solution to this problem, Caulfield and Maloney[16] proposed the design of Linear Combination Filters (LCFs). Each of these filters would give unit output when correlated with one of the input patterns and zero with all of the others, which means that they would be orthogonal. Caulfield and Maloney[16] required a set of simultaneous equations to be solved to calculate an array of coefficients, which could then be used to combine the

training patterns to set the off-diagonal elements in a vector inner product matrix to zero. Iterative techniques exist to find inverse matrices and to diagonalise matrices[17]. In this paper we describe an iterative technique that is derived from pattern similarity considerations and gives a deeper insight into the effect that the iterative process has on the patterns. In essence, our algorithm iteratively diagonalises a matrix, it gradually reduces off-diagonal elements by reducing the similarities between the input patterns. On-diagonal elements are maintained at a constant level. If the iterations are halted before convergence our technique preserves the features and character of each individual pattern, having suppressed the largest off-diagonal inner products most. We will show that the filters produced after a few iterations only, perform better than the fully converged filters and considerably better than the filters produced with Caulfield's and Maloney's method. Moreover, the mathematical formulation of much of our algorithm in terms of image operations is particularly suitable for optical implementation.

2 Similarity suppression algorithm

This section contains a brief description of the similarity suppression (SS) algorithm. When designing filters for optical pattern recognition through correlation we usually face one or more of the following problems: Two or more of the known patterns are very similar, or the input patterns are noisy, or the optical system is noisy. If any one - or a combination - of these problems exists then two filters, out of a bank of recognition filters, can simultaneously give a high inner product output. The aim is then to detect which is the highest, but if the inner products differ by only a small amount this is difficult. The problem is due to the high pedestal on which the small difference is offset. If this input is too large for the system leading to saturation then the input must be scaled down but this also scales down the difference between the peaks and may bring it below the resolving limit of the system. The resolving limit is due to the noise floor of the system. If the pedestals were smaller or very small the inner product peaks could even be amplified to bring them up to just below the systems saturation limit which will *increase* the difference between the peaks to above the resolving limit of the system. So the filters have to be designed in such a way so that each of them has a very high inner product with one of the input patterns only, and very low inner products with all of the other input patterns. In this case a fair amount of noise in the input or in the system could be tolerated.

The SS algorithm uses a set of training patterns to create a set of discrimination filters. Each of these filters is derived and corresponds to one of the initial training patterns. It is designed in such a way so that it has a constant high inner product with the training pattern from which it was derived and low inner products with all of the other training patterns. The SS algorithm creates the filters by suppressing the similarities between the initial training patterns. The algorithm is iterative and at each iteration a small amount of each of the training patterns is subtracted from the filters and in that way the similarities between them are suppressed. After each iteration the inner product between each of the filters and each one's corresponding training pattern is normalised to a constant high value.

Throughout this paper we are going to use the following notation: We denote patterns as vectors $\mathbf{s}_j = [s_{j1}, s_{j2}, \ldots, s_{jN}]^T$ of length N, where $\mathbf{s}_j$ is the j^{th} pattern of M patterns and N is the number of pixels in each image. M is the size of the training set and subsequently the filter set. We refer to $\mathbf{g}_j^{(i)}$ as being the new pattern vector for the j^{th} filter after the i^{th} application of the algorithm to all of the known training patterns. The inner product between two patterns will be denoted by $\mathbf{g}.\mathbf{s}$ and is equal to $\Sigma^N_{k=1}\, \mathbf{g}_k.\mathbf{s}_k$ or in vector notation $\mathbf{g}^T.\mathbf{s}$. We refer to the inner product of a pattern with itself as the auto-inner product and the inner product of a pattern and another pattern as a cross-inner product.

The SS algorithm can be described by the following equations

$$\widetilde{\mathbf{g}}_j^{(i)} = \mathbf{g}_j^{(i-1)} - \beta \sum_{\substack{k=1 \\ k \neq j}}^{M} \left\{ \mathbf{g}_j^{(i-1)}.\mathbf{s}_k \right\}^2 \mathbf{s}_k \qquad (1)$$

$$\mathbf{g}_j^{(i-1)} = \widetilde{\mathbf{g}}_j^{(i)} \frac{\mathbf{g}_j^{(i-1)}.\mathbf{s}_j}{\widetilde{\mathbf{g}}_j^{(i)}.\mathbf{s}_j} \qquad (2)$$

The algorithm, when i has run to completion, creates a set of filters $\mathbf{g}_j$, $j = 1 \ldots M$, each of which is mutually orthogonal to all of the training patterns, $\mathbf{s}$, except for one from which it originated. The inner product between each of these filters $\mathbf{g}_j$ and the corresponding training pattern $\mathbf{s}_j$ is equal to a constant, high, predetermined value, P. P is usually equal to the auto-inner products of the training patterns. The inner product between the filter $\mathbf{g}_j$ and any other training pattern $\mathbf{s}_k$, $j \neq k$ is equal to 0. As we said earlier, the final filters are created by suppressing the similarities between the training patterns $\mathbf{s}$. This operation is performed by the first algorithmic equation (1), by subtracting all except for one of the training patterns from each of the filters at each iteration. The second algorithmic equation (2) normalises the inner product between the filter and the training pattern from which it was derived. A convergence parameter β is used in the first algorithmic equation (1). This parameter is a number smaller than one and it diminishes the effect of the subtractions on the filters. It controls the convergence speed and the stability of the algorithm. We can also notice that the inner product in equation (1) is squared. The algorithm works without this square as well, but by using it we increase the convergence speed particularly in the first few iterations. We have also noticed from our computer simulations that when the inner product is squared the algorithm usually converges to a slightly better solution. This can be explained if we consider that by squaring we emphasise the higher inner products and, hence, strongly subtract the patterns that are most similar to the filter.

2.1 The relationship between the SS cross orthogonalisation algorithm and Caulfield's and Maloney's Linear Combination Filters

The SS cross orthogonalisation algorithm is described by equation (1) which we rewrite here:

$$\mathbf{g}_j^{(i)} = \mathbf{g}_j^{(i-1)} - \beta \sum_{\substack{k=1 \\ k \neq j}}^{M} \left\{ \mathbf{g}_j^{(i-1)} . \mathbf{s}_k \right\} \mathbf{s}_k \qquad (3)$$

Notice that we have omitted the square in equation (1). There is also a normalisation step (equation (2)), which is not necessary for our analysis here and it is omitted for the sake of simplicity. Let us consider what happens to an individual filter throughout the training. At each iteration, all of the training images are subtracted from it, each with a different weight. After all of the iterations a total amount of each of the training images has been subtracted from it. This total amount is equal to the sum of all of the individual weights which were used for the subtraction of each training image during the training. The same thing happens to all of the filters. Therefore, they can be given by the following equations:

$$\begin{aligned} C_{11}\mathbf{s}_1 + C_{12}\mathbf{s}_2 + \mathrm{K} + C_{1M}\mathbf{s}_M &= \mathbf{g}_1 \\ C_{11}\mathbf{s}_1 + C_{12}\mathbf{s}_2 + \mathrm{K} + C_{1M}\mathbf{s}_M &= \mathbf{g}_2 \\ \mathrm{M}\,\mathrm{M} \quad \mathrm{M}\,\mathrm{M} \qquad\quad \mathrm{M}\;\mathrm{M} \quad \mathrm{M} & \\ C_{M1}\mathbf{s}_1 + C_{M2}\mathbf{s}_2 + \mathrm{K} + C_{MM}\mathbf{s}_M &= \mathbf{g}_M \end{aligned} \qquad (4)$$

where each of the coefficients C_{11}, C_{12}, …, C_{MM} is equal to the sum of all of the individual weights that were used for the subtraction of each of the training images during the training. These coefficients are all negative, except for C_{jj}, $\forall j$ which are equal to zero. It is obvious that if the coefficients C_{11}, C_{12}, …, C_{MM} can be calculated then the final filters can be created without the need for an iterative procedure. The aims of the SS algorithm can be expressed by the following set of equations:

$$\begin{aligned} \mathbf{s}_i . \mathbf{g}_j &= 1 \qquad \text{if } i = j \\ \mathbf{s}_i . \mathbf{g}_j &= 0 \qquad \text{if } i \neq j, \forall i, j \end{aligned} \qquad (5)$$

The previous set of equations are written in a matrix form as follows:

$$\mathbf{S}\mathbf{G}^{T} = \mathbf{I} \qquad (6)$$

where **S** is a Mx1 vector whose elements are the training patterns **s**, **G** is a Mx1 vector whose elements are the filters **g** and **I** is an MxM identity matrix. Equation (4) can be written in a matrix form as follows:

$$\mathbf{C}\mathbf{S} = \mathbf{G} \qquad (7)$$

where **C** is an MxM matrix each element of which is the coefficient C_{ij}. From equations (6) and (7) we can calculate the values of the coefficients C_{ij}:

$$\mathbf{C} = \left(\mathbf{R}^{-1}\right)^{T} \qquad (8)$$

where **R** is an MxM matrix ($\mathbf{R} = \mathbf{S}\mathbf{S}^{\mathrm{T}}$), each element of which, r_{ij}, is equal to the inner product between the training patterns $\mathbf{s}_i$ and $\mathbf{s}_j$. So, the coefficients C_{ij} can be calculated from equation (8) as long as matrix **R** can be inverted. In that case the final filters can be calculated directly from equation (7) by substituting the coefficients matrix **C** from equation (8)

$$\mathbf{G} = \left(\mathbf{R}^{-1}\right)^{T}\mathbf{S} \qquad (9)$$

Caulfield and Maloney[16] calculated their Linear Combination Filters in two steps. The first step was to calculate the vector inner product matrix, **R**, of the input patterns. This matrix had each of its elements r_{ij} equal to the inner product between the training patterns $\mathbf{s}_i$ and $\mathbf{s}_j$.

In the second step they formed linear combinations of the responses $r_{ij}s_j$. Using these linear combinations, the final response when testing pattern s_i for its identity to s_k would be

$$F_{ik} = r_{ik} + \sum_{l \neq k} C_{kl} r_{il} \qquad (10)$$

They imposed the constraint that F_{ik} had to be zero if $i \neq k$ and nonzero if $i = k$, i.e.,

$$F_{ik} = F_{kk}\delta_{ik} \qquad (11)$$

Equations (10) and (11) were formulated as matrix equations[2,18] leading to the general synthetic discriminant function SDF described by the following equation

$$\mathbf{R}\mathbf{a}_i = \mathbf{d}_i \qquad (12)$$
$$\Rightarrow \mathbf{a}_i = \mathbf{R}^{-1}\mathbf{d}_i$$

Equation (8) is in essence the same as equation (12). We have calculated the coefficients for the M filters, while equation (12) calculates the coefficients for a single filter. In addition, the matrix with the

desired correlation values in our case is the identity matrix, while equation (11) is more general as the diagonal elements need not be the same. The vector-inner product matrix **R** is transposed as well as inverted in our equation (8) because we have defined the coefficient vector for each of the filters as a 1xM vector while in equation (12), $\mathbf{a}_i$ is a Mx1 vector. So in effect we see that the SS cross orthogonalisation algorithm should finally converge to the solution which is obtained using general synthetic discriminant functions or Caulfield's and Maloney's method. The main difference between our algorithm and the two methods, is that our algorithm is iterative. The first question that automatically arises is whether the SS algorithm converges to exactly the same solution as the other two methods. We provide an answer to this question in the next section using computer simulations. The SS algorithm described by equation (1) has the subtraction weight squared. This square does not affect the previous result, as it can be included in the coefficients C_{ij} without any change in the subsequent analysis.

3 Convergence Simulations

This section presents the performance of the algorithm during the iterative training phase. In order to assess the efficacy of the algorithm it is necessary to choose and to devise appropriate performance measures. These are introduced below, followed by a detailed description of the simulation parameters and results.

3.1 Performance Measures

We define two performance metrics:

3.1.1 Cross-inner product matrix

A matrix $\boldsymbol{R}$ which we call the cross-inner product matrix was calculated at each iteration. The R_{ij} element of the cross-inner product matrix was equal to the value of the inner product of the patterns $\mathbf{g}_i$ and $\mathbf{s}_j$. Before the first iteration, when the **g** patterns are identical to the **s** patterns, matrix $\boldsymbol{R}$ is *the vector (auto) inner product matrix* of the input patterns as defined by Kumar[19].

3.1.2 Global Energy

A term which we will call the total energy of the system was defined as

$$TE = \frac{1}{M^2 P} \sum_{i=1}^{M} \sum_{j=1}^{M} \left| \mathbf{g}_i . \mathbf{s}_j \right| \qquad (13)$$

In other words the total energy of the system is equal to the normalised sum of the modulus of all of the elements of the cross-inner product matrix. The total energy is a measure of the height of all of the cross inner products. As the algorithm converges, we expect the total energy to decrease.

3.2 Binary, bipolar patterns

Our aim when conducting these simulations was to see how much the cross-inner products were reduced, how many iterations it took for these reductions to take place and how the final cross-inner product values and the convergence speed were affected by the convergence parameter β. The first training set of patterns to be recognised, consisted of 32, 16x16 binary, bipolar patterns denoted by $\mathbf{s}_i$, i=1,…,32. Eight patterns in the training set were chosen to have random elements. Those were patterns numbers 1, 7, 10, 11, 17, 20, 21 and 22 as they appeared in the set. The other patterns were similar to one of patterns 1, 11, or 22, differing by 7, 14, 28, 56 or 112 pixels. Table 1 shows the order of the patterns in

the training set, the similar patterns and by how many pixels they differ. The patterns that are similar to one another were created by copying the initial pattern and then randomly changing the desired number of pixels.

Pattern No.:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Similar to:	-	1	1	1	1	1	-	22	22	-	-	11	11	11	11	11
Differing by:	-	7	14	28	56	112	-	7	14	-	-	7	14	28	56	112

Pattern No.:	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Similar to:	-	11	11	-	-	-	22	22	22	22	22	1	11	1	22	1
Differing by:	-	7	14	-	-	-	7	14	28	56	112	7	7	7	14	14

Table 1 Number of pixels differing in the training set.

The SS algorithm, described in equations (1) and (2), was tested with several different values of the convergence parameter β. Here we present some representative results for 6 different values of β, which are $\beta = 0.01$, 0.1, 0.5, 1, 4 and 6. For most of the β values the algorithm converged to a sufficiently stable solution within the first 1500 iterations.

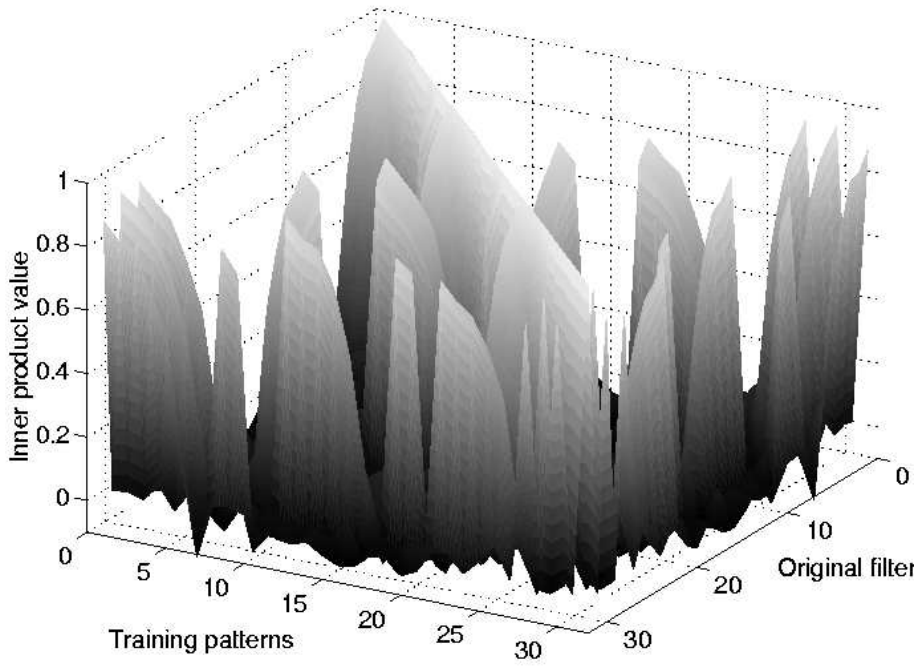


Figure 1 Cross-inner product matrix before the training.
The graph is shown with the x and y axes reversed for clarity.

In figure 1 one can see the three dimensional graph of the initial state of the cross-inner product matrix. The palest shading shows the highest peaks. The large values on the diagonal represent the auto-inner products. All of the other peaks, some of which are large (but no bigger than the auto-inner products) represent the cross-inner products and those are the ones that we want to decrease.

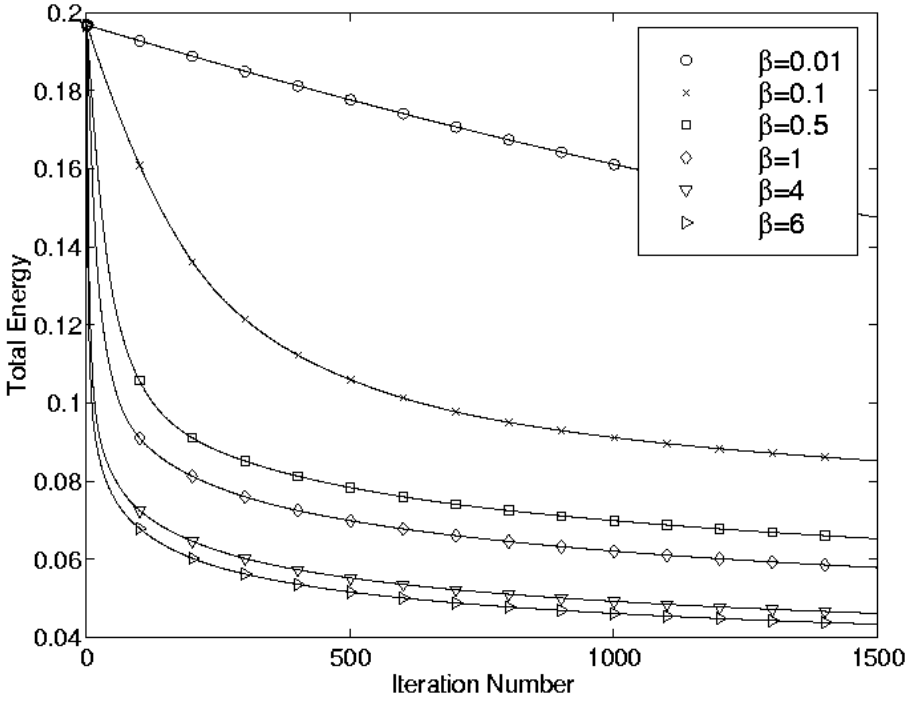


Figure 2 Total energy index.

The total energy index is plotted in figure 2 as a function of iteration number. For most of the β values, the total energy reduces rapidly, which means that the cross-inner products decrease. In addition, for most of the β values the total energy decreases exponentially. The decrease is very fast initially and slows down later. The three dimensional graph of the cross-inner product matrix after training with $\beta = 6$ is shown in figure 3. It is very easy to see, by comparing figures 3 and 1, that the SS algorithm has been very successful at suppressing all of the cross-inner products.

3.3 Peak-to-Correlation Energy of the correlations between the training patterns and the trained and untrained filters

The algorithm forces the inner products to decrease but it does not place any constraints on the outer products. In figure 4 we can see the intensity profile in the correlation plane for two correlations. Subfigure 4 (a) depicts the auto-correlation of pattern 1. Subfigure 4 (b) depicts the correlation between pattern 1 and filter 1, which was obtained after 1500 iterations with $\beta = 6$. We can see that the auto-correlation of pattern 1 has a sharp peak (as expected) and a PCE equal to 0.53. When using the filter corresponding to pattern 1, the inner product has remained stable, but the outer products have increased a lot and the PCE is now only 0.027.

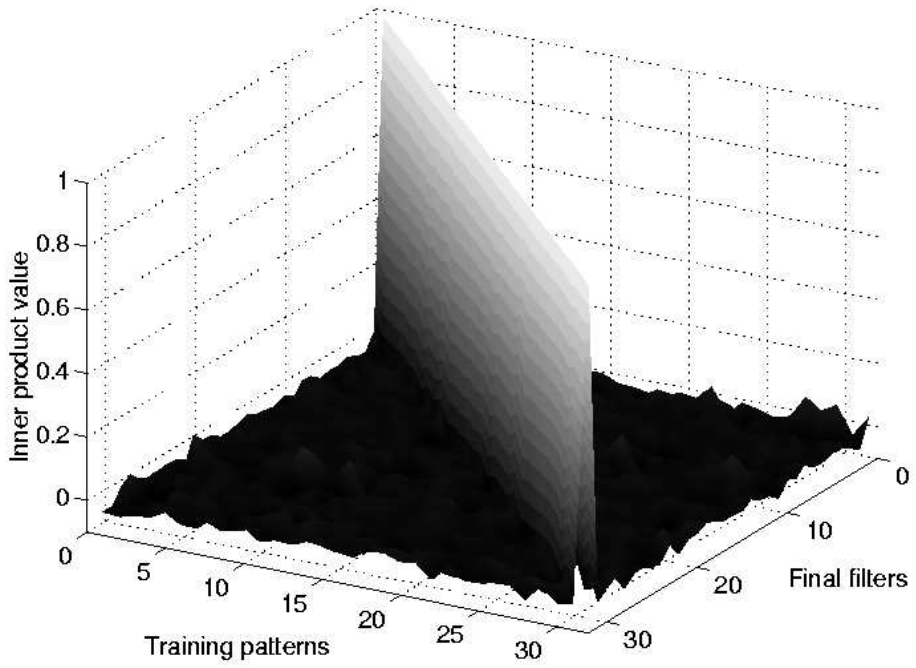
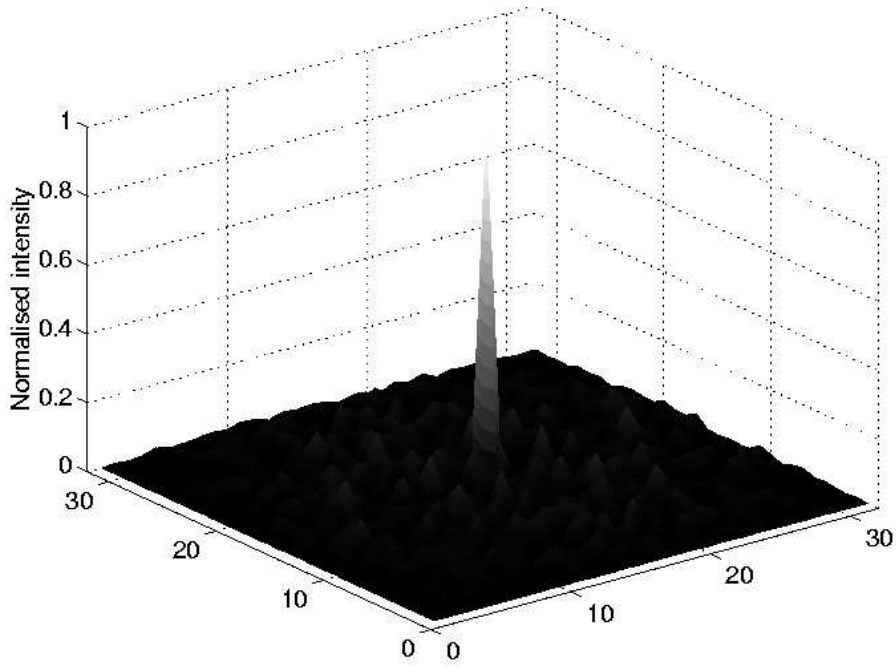


Figure 3 Cross-inner product matrix after 1500 iterations for a convergence factor of $\beta = 6$
The graph is shown with the x and y axes reversed.

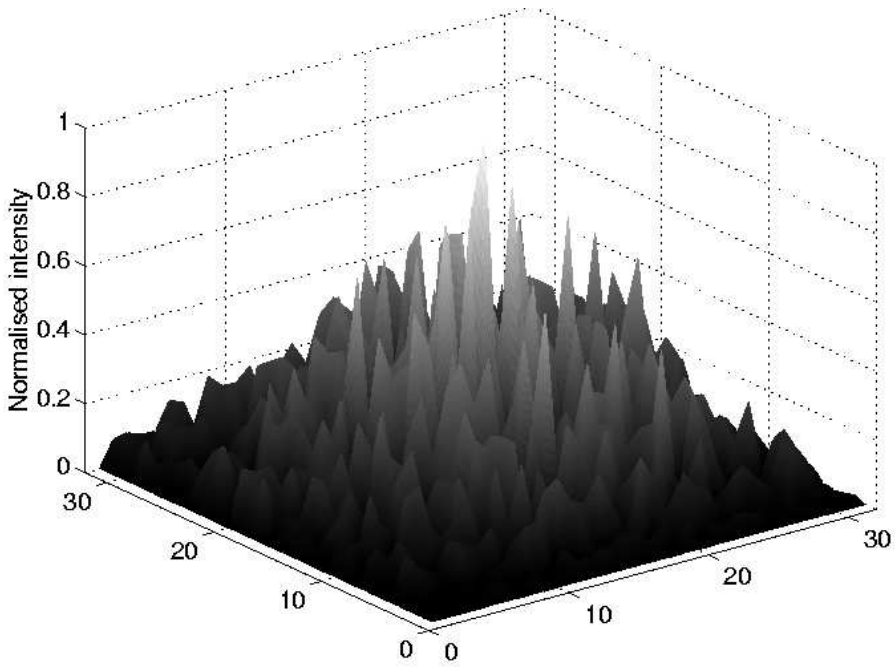
Another interesting example can be seen in figure 5 which shows the correlations between pattern 7 and pattern 1 (in subfigure 5 (a)) and pattern 7 and filter 1 (in subfigure 5 (b)). In the correlation between patterns 7 and 1 there is no correlation peak and the outer products are all low, because the two patterns are very different. However, when using filter 1, the central point of the correlation plane, i.e. the inner product, may still have a very low value, but the outer products have increased dramatically and one of them is about 80% of the auto-correlation peak value, P.

To conclude, the algorithm reduces the inner products, but does not put any constraints on the outer products, so they increase. Not all of the correlations have their outer products increased by the same amount. The biggest increase in the outer products occurs in correlations of filters that were derived from patterns which were similar to others at the beginning of the training. The auto-correlation's outer products increase a lot because of the normalisation step. A filter is changed during the training but that makes it different from the pattern from which it was derived, as well as from other patterns. When it is normalised so that it's auto-inner product reaches the desired level, it's magnitude increases and as a result, all of the outer products of the correlation increase.

This increase in the outer products in the correlations plays no role in electronic systems, but is a major drawback in optical systems, unless the input images are always centred. In the case when the location of the object in the input scene is not known precisely, the increased outer products would make an optical system a lot less useful, because a high outer product could be mistaken for a correlation peak.

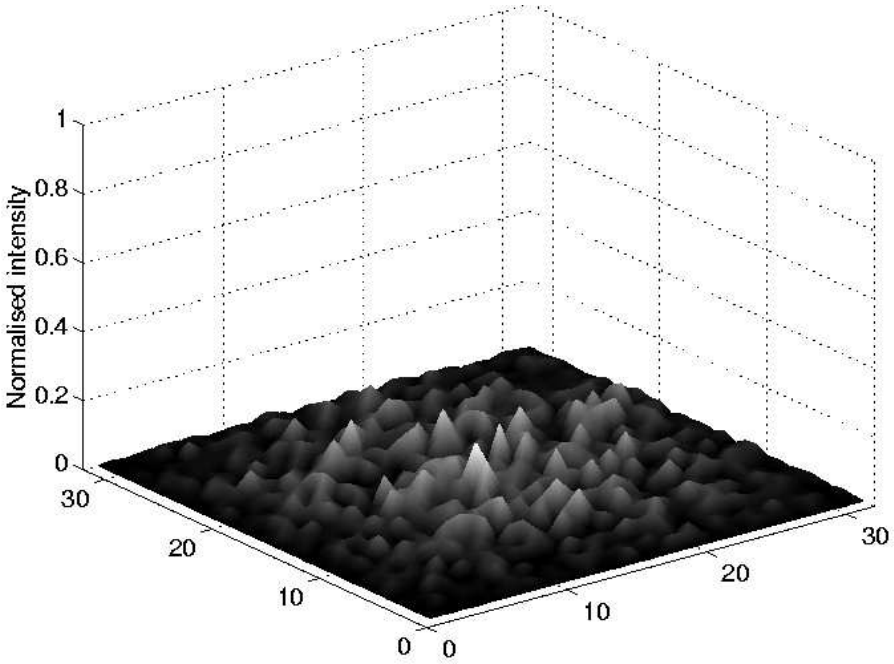


(a) $\mathbf{s}_1 \otimes \mathbf{s}_1$, PCE=0.53

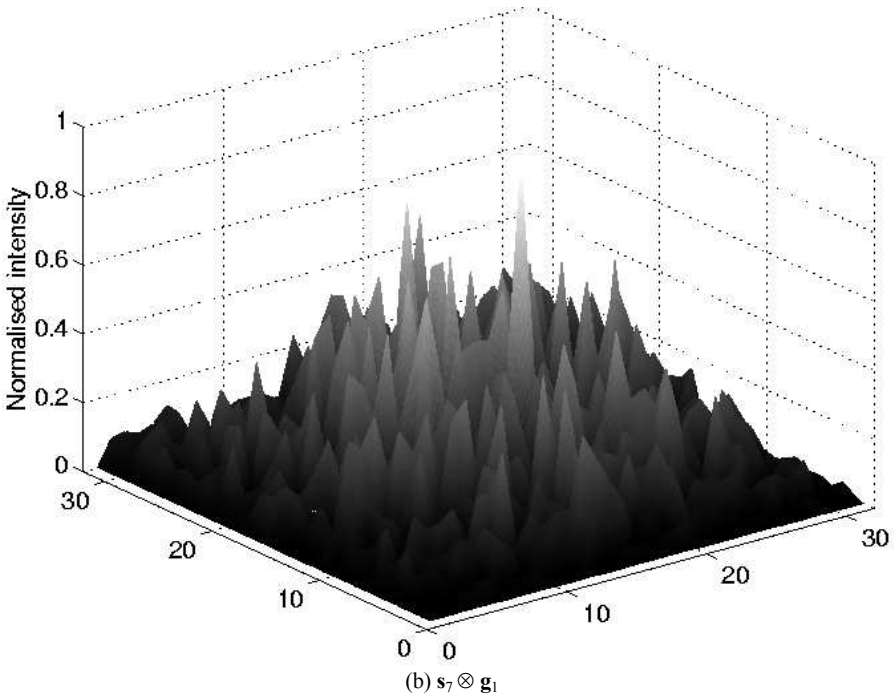


(b) $\mathbf{s}_1 \otimes \mathbf{g}_1$, PCE=0.0.027

Figure 4 Correlation plane intensity for auto-correlation of pattern 1
and correlation between pattern 1 and filter 1



(a) $\mathbf{s}_7 \otimes \mathbf{s}_1$



(b) $\mathbf{s}_7 \otimes \mathbf{g}_1$

Figure 5 Correlation plane intensity for correlations between pattern 7 and pattern 1 and between pattern 7 and filter 1

4 Probability of discrimination and dynamic range

Optical inner product correlator pattern recognition systems suffer from the limited dynamic range inherent in optics. The SS algorithm minimises the cross-inner products and holds the auto-inner products constant so that they differ by a larger amount. The dynamic range required by a detector at the output inner-product plane of an optical system is reduced and less sensitive equipment is needed.

In most cases the pattern that needs to be recognised will contain an amount of noise, where we are using the word ``noise'' in a broad sense indicating additive or multiplicative noise or distortion, rotation or a proportion of another pattern. It is important to see how much noise can be tolerated before the pattern becomes unrecognisable, and how much the required dynamic range is, for each noise level. We conducted simulations with analogue additive noise. The dynamic range requirements for correct discrimination and the probability of discrimination were calculated for different levels of noise. The noise added to the patterns was normally distributed with a zero mean. The input signal to noise ratio (SNR) varied from 20 to −10 dB. The results shown in this section were obtained using the filters which were calculated with $\beta = 6$. The method for calculating the probability of discrimination, was to calculate all of the inner products between an input pattern and all of the filters and then to choose the highest of them. For correct discrimination, the highest inner product had to be the one with the filter which corresponded to the input pattern. The experiment was repeated for all of the training patterns for 5000 different samples of noise for each different noise level. We did not use a fixed threshold because the SS algorithm addresses the problem of discrimination between patterns and not detection.

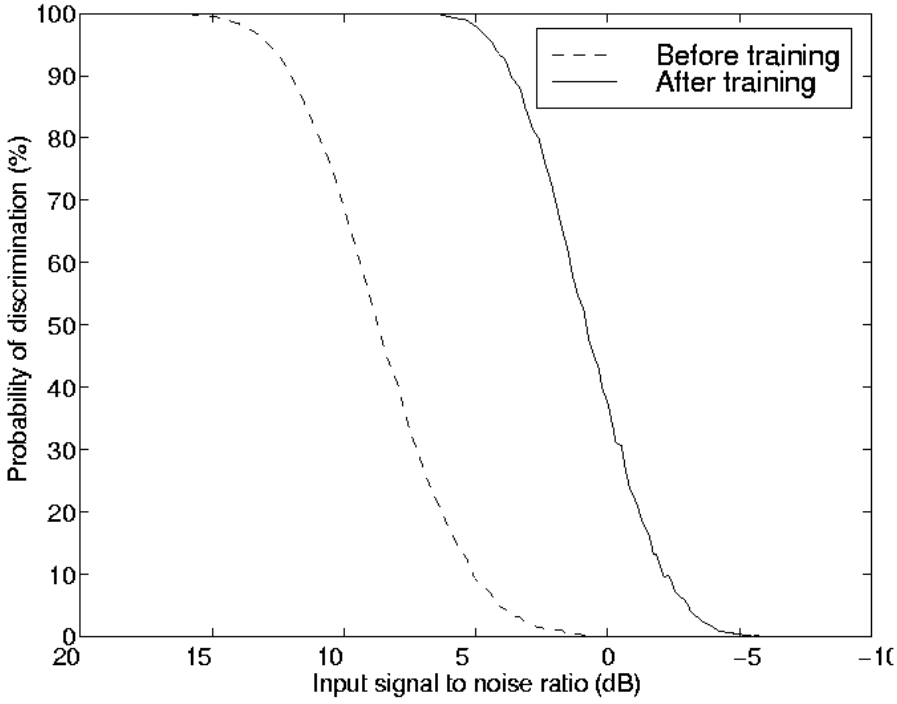


Figure 6 Probability of discrimination versus input signal to noise ratio

The resulting curves for the probability of discrimination before and after training are shown in figure 6. We can see, in that figure, that there is a significant increase in the probability of discrimination after the training. For example, with an input signal to noise ratio of 3 dB the probability of discrimination is 2% before the training and it increases to 83% after the training. Also the probability of discrimination falls to 50% at an input signal to noise ratio of 8.9 dB before the training and 0.8 dB after the training. The curve after the training is almost a shifted version of the curve before the training, although it is slightly steeper. This means that the same pattern discrimination behaviour versus SNR can be achieved but we can tolerate 7 dB more noise.

The dynamic range of the recognition system was defined to be the ratio of the difference between the auto-inner product and the maximum cross-inner product, to the corresponding auto-inner product, in decibels. This can be written as:

$$dynamic\ range = \max \forall i \left(-20 \log_{10} \left\{ \frac{\mathbf{s}_i . \mathbf{g}_i - \max \forall j \left(\mathbf{s}_i . \mathbf{g}_j \right)}{\mathbf{s}_i . \mathbf{g}_i} \right\} \right), \qquad (14)$$

$$i = 1 \Lambda\ M,\ j = 1 \Lambda\ M,\ j \neq i$$

This definition assumes that the system has some form of automatic gain control which, for example, scales the maximum auto-inner product to a constant near the top of the dynamic range.

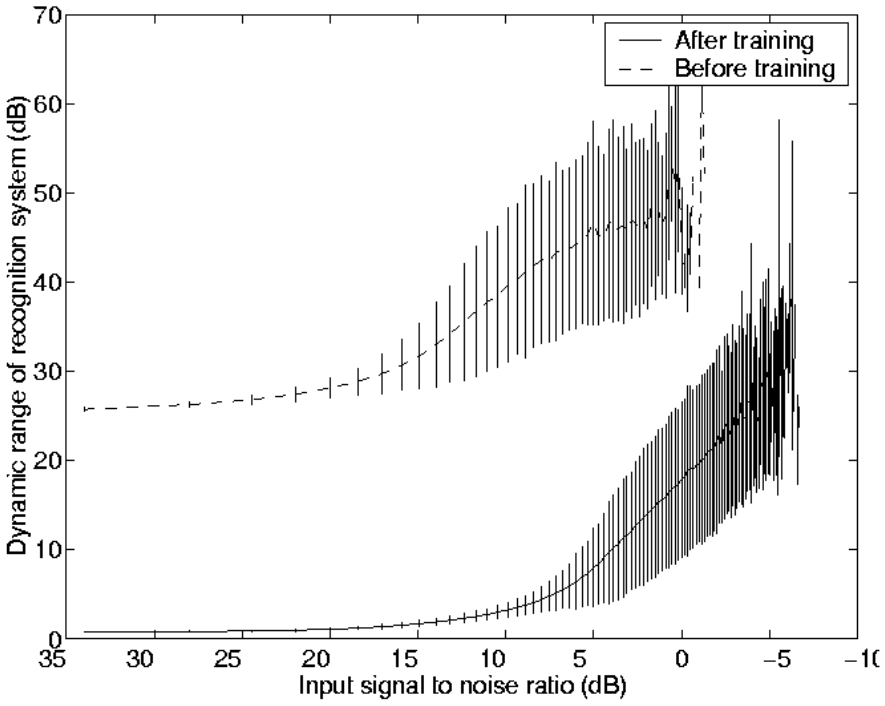


Figure 7 Dynamic range of the recognition system as a function of the signal to noise ratio. The error bars show the standard deviation for 5000 measurements.

The resulting plot of the dynamic range versus the signal to noise ratio before and after the training is shown in figure 7. The error bars in figure 7 indicate the standard deviation of the calculated dynamic range values for 5000 measurements. We can see from figure 7 that there is a very large reduction in the dynamic range required for correct discrimination, of the order of 25 dB, after the training. The amount of reduction lessens for higher amounts of noise. The error bars increase as the noise is increased due to the random nature of the noise. The worse case after the training is better than the best case before the training, for the same amount of additive noise, because the error bars do not meet. The curve before the training does not extend to higher noise levels because, from figure 6, when the probability of discrimination drops to zero it is not meaningful to plot the dynamic range. From graph 7 we can also see that if an optical system has a dynamic range of 30 dB this means that, before training, patterns can be recognised having an input SNR of 15 dB upwards whereas, after training, the dynamic range of the system does not limit discrimination.

5 Comparison between the filters produced with the SS algorithm and the linear combination filters

In section 2.1 we compared the SS algorithm to the method proposed by Caulfield and Maloney[16] for designing linear combination filters and we concluded that the SS algorithm converges to the same solution as the one provided from Caulfield's and Maloney's method. In this section we use that method to create filters which are mutually orthogonal to the binary, bipolar patterns in our first training set and compare them to the filters that were created with the SS algorithm. We calculated the cross-inner product matrix between the input patterns and the set of filters created with the matrix method. The three dimensional graph of this matrix can be seen in figure 8. This can be compared to figure 3 which shows the cross-inner product matrix between the input patterns and the filters that were created using the SS algorithm.

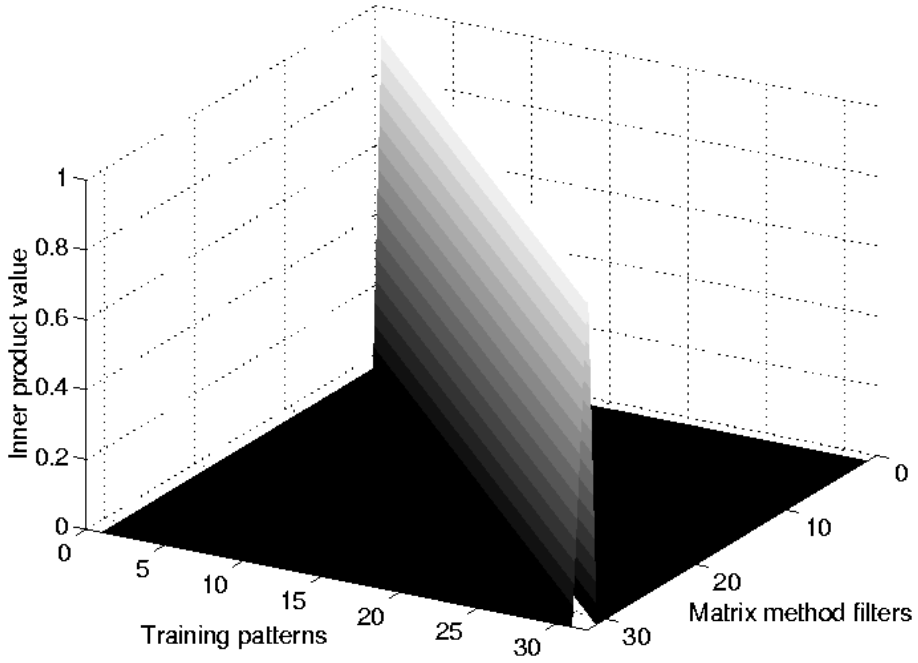
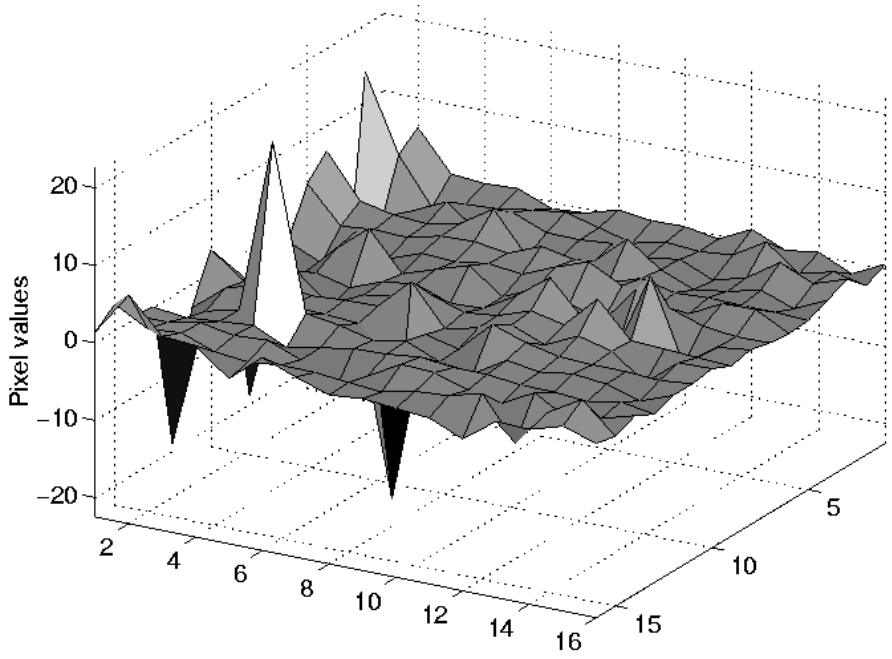
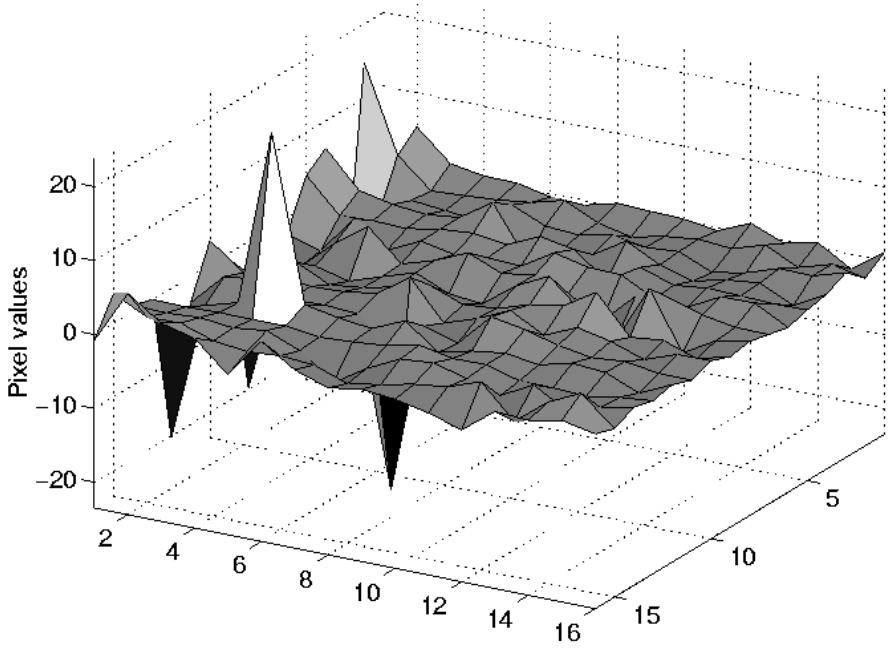


Figure 8 Cross-inner product matrix between the input patterns and the filters created using equation 9. The graph is shown with the x and y axes reversed for clarity.



(a) Filter created with the SS algorithm



(b) Filter created with the matrix method

Figure 9 Pixel values of the two versions of filter 2

We can see from figure 8 that the filters created using equation (9) are completely cross-orthogonal to the input patterns as was expected. The filters that were created using the SS algorithm are almost (figure 3) - but not completely - orthogonal and the SS algorithm may converge to the same solution if it is allowed to run for more iterations. To investigate further we looked at the actual filters. An example is shown in figure 9, which shows the two versions of filter 2. Subfigure (a) depicts filter 2 created by the SS algorithm and subfigure (b) depicts filter 2 created with the matrix method. The two figures are very similar, although not identical.

One might argue at this stage that there is no point in using the SS algorithm to create the filters since they can be obtained with fewer calculations, and, therefore, faster from equation (9). However, since the filters obtained with the two different methods are not identical, we decided to test the tolerance to input noise and the dynamic range that would be required by an optical system for correct discrimination, when using the second set of filters (the ones calculated with the matrix method). We conducted the same simulations as in the previous section.

The resulting curves for the probability of discrimination using the two filter sets are shown in figure 10. Also in the same graph there is a third curve which shows the probability of discrimination before the training. We can see, in that figure, that there is a significant increase in the probability of discrimination after the training whichever of the two sets of filters we use. However, the filters obtained with the SS algorithm are slightly more tolerant to noise.

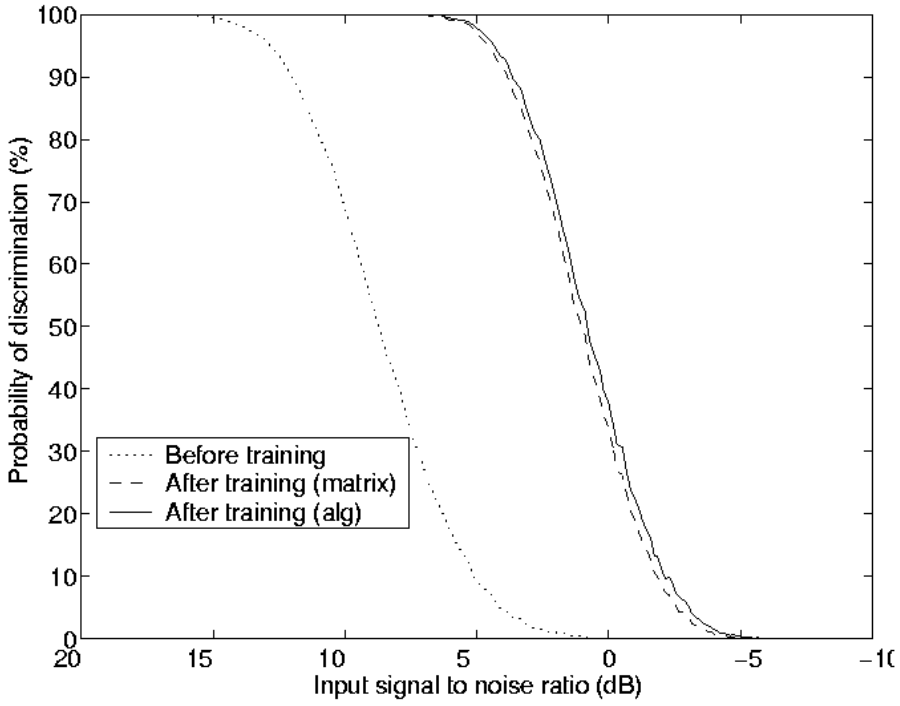


Figure 10 Probability of discrimination versus input signal to noise ratio

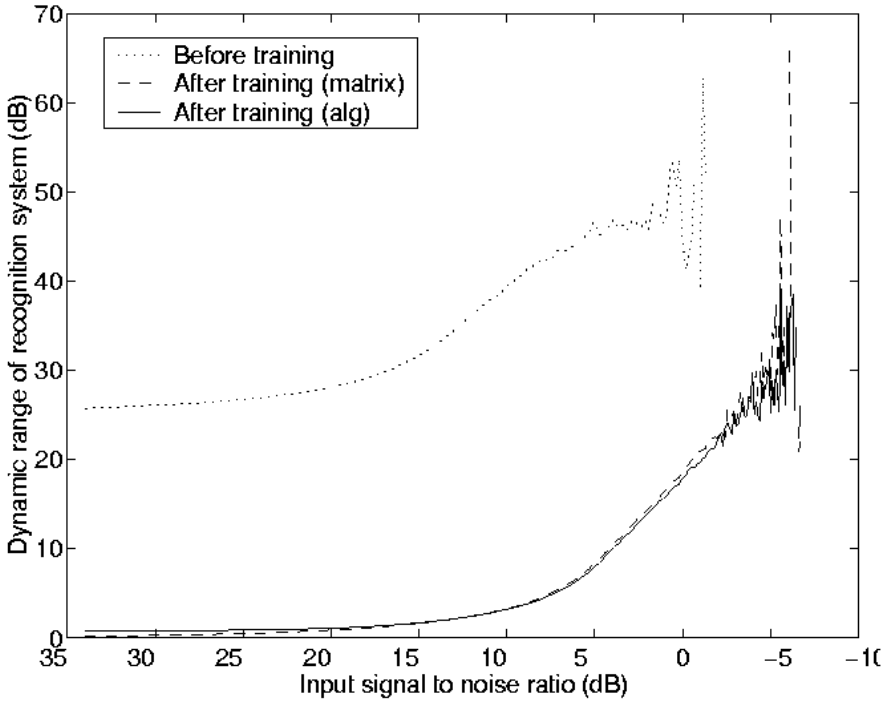


Figure 11 Dynamic range of the recognition system as a function of the signal to noise ratio.

The plot of the dynamic range versus the signal to noise ratio, again using both filter sets, is shown in figure 11. As with the probability of discrimination graph, the dynamic range graph shows us that the filters obtained with the SS algorithm are slightly more (maximum difference between two sets of filters ~ 1 dB) noise tolerant. Obviously for small amounts of noise the filters obtained with the matrix method (equation 9) yield better dynamic range results because they are completely orthogonal to the input patterns. How can these results be explained? It may be that completely cross-orthogonalising the filters to the patterns is not the best solution after all. Maybe the matrix method results in some kind of over-fitting to the training data which makes the final filters less able to generalise and, therefore, less tolerant to input noise.

6 Optimisation of number of iterations for the similarity suppression algorithm

The results shown in the previous section motivated us to investigate the noise tolerance of the various sets of filters obtained when using the SS algorithm and allowing it to run for different numbers of iterations. To do that we used the SS algorithm to train the filters for the binary, bipolar patterns in our first training set and during the training, after each iteration, we calculated the probability of discrimination and the dynamic range required for correct discrimination with the newly produced set of filters. Each time the same amount of random noise was added to the input. As before the noise was analogue, normally distributed, with zero mean and with constant variance equal to unity.

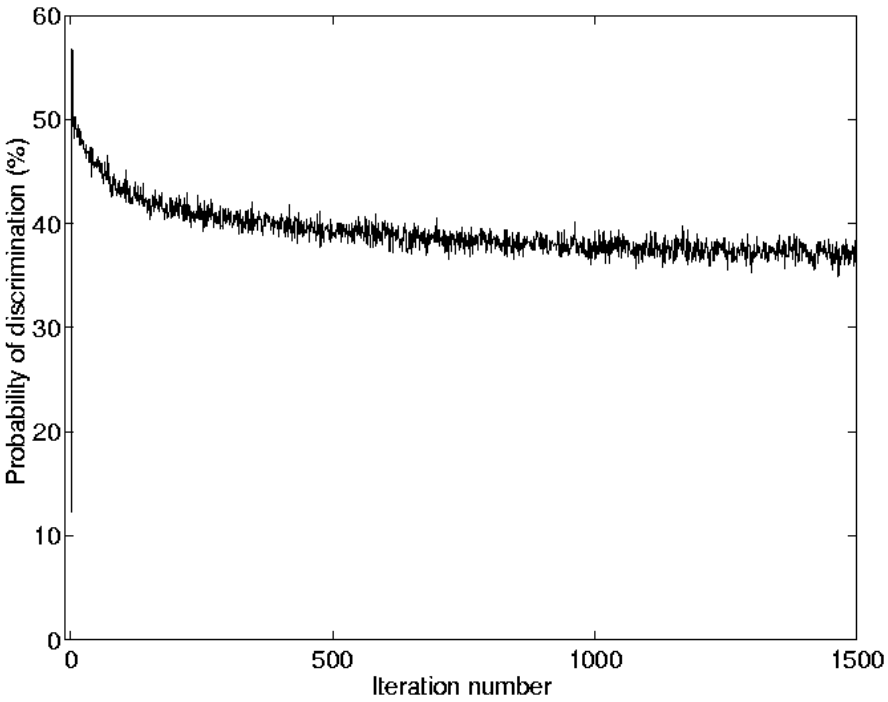


Figure 12 Probability of discrimination versus number of iterations.

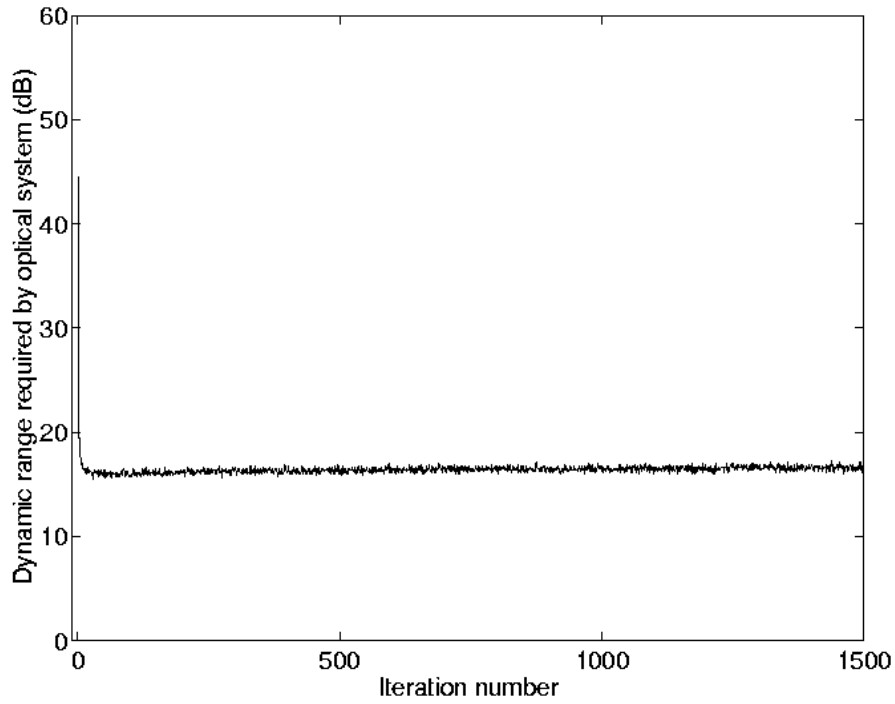


Figure 13 Dynamic range of the recognition system versus number of iterations.

The probability of discrimination versus iteration number is shown in figure 12. We can see that there is a sharp increase of the probability of discrimination in the first iterations and then the probability of discrimination decreases, until it finally converges to a relatively constant level. The dynamic range required by the optical system for correct discrimination versus iteration number is shown in figure 13. As we can see the required dynamic range decreases very quickly and after the first few iterations it converges to a constant level.

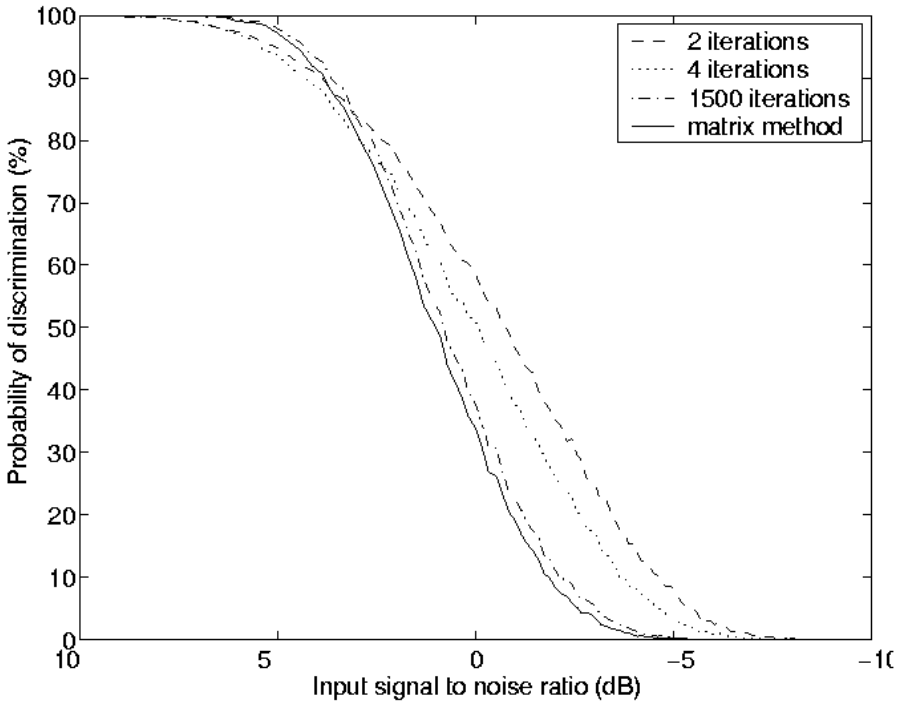


Figure 14 Probability of discrimination as a function of the signal to noise ratio

We then calculated the probability of discrimination and the dynamic range required by the optical system for correct discrimination using the filters obtained after the first few iterations. The corresponding graphs for the probability of discrimination can be seen in figure 14. As we can see from the probability of discrimination curves, the filters produced after only 2 or 4 iterations perform slightly worse for a higher signal to noise ratio but as the SNR worsens, these filters perform better than the ones obtained after the algorithm has converged completely (after around 1500 iterations) and better than the ones which are calculated using the matrix method of equation (9). In figure 15 we have plotted the difference between the probability of discrimination when using the filters produced after 2 iterations of the algorithm and when using the filters produced after 1500 iterations. The other curve in the same graph is the difference between the probability of discrimination when using the filters produced after 2 iterations and the filters produced with the matrix method.

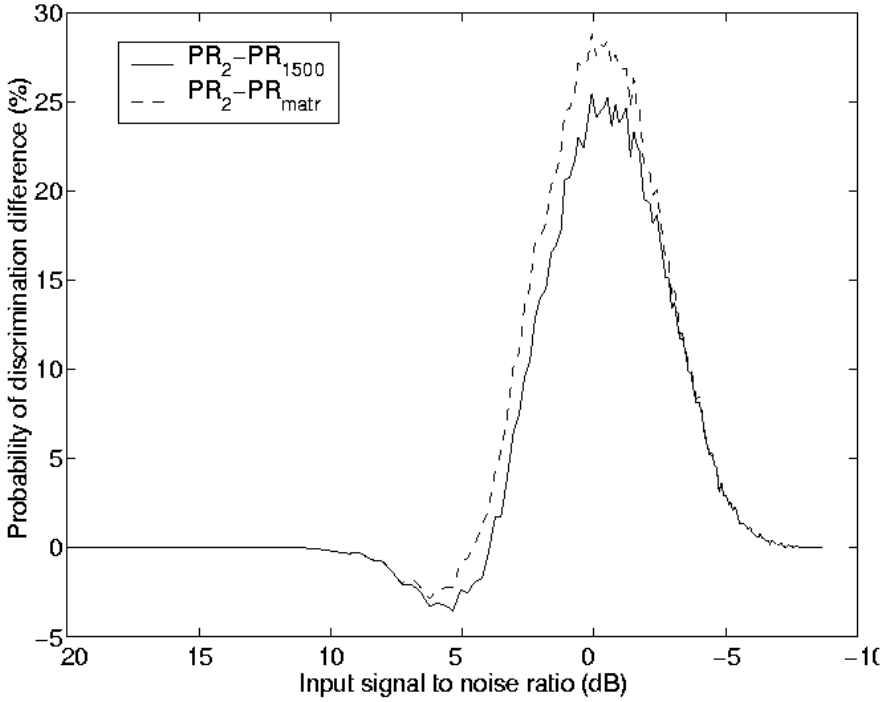


Figure 15 Probability of discrimination difference as a function of the signal to noise ratio

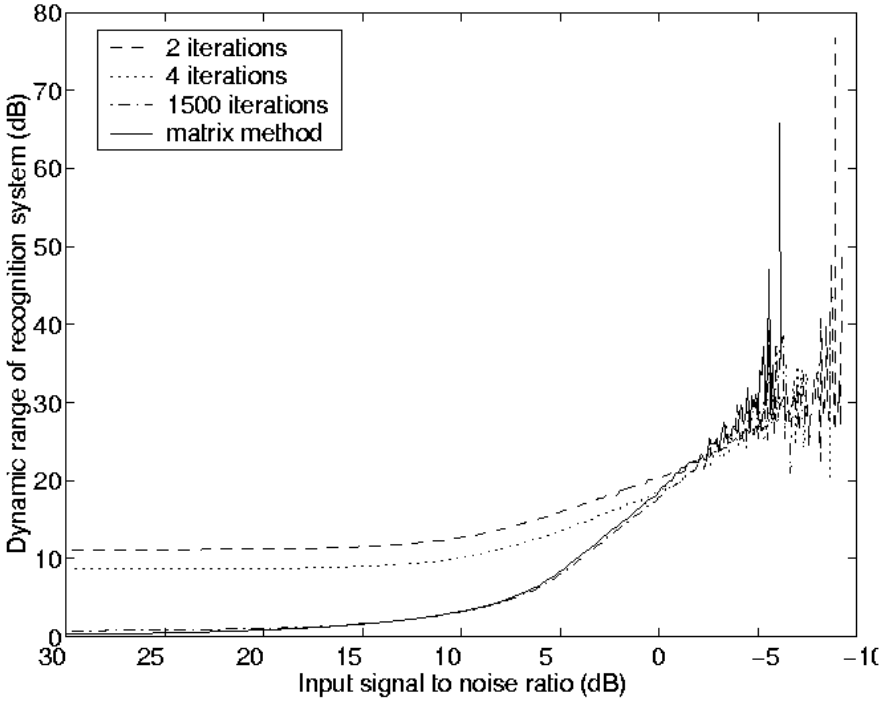


Figure 16 Dynamic range of the recognition system as a function of the signal to noise ratio

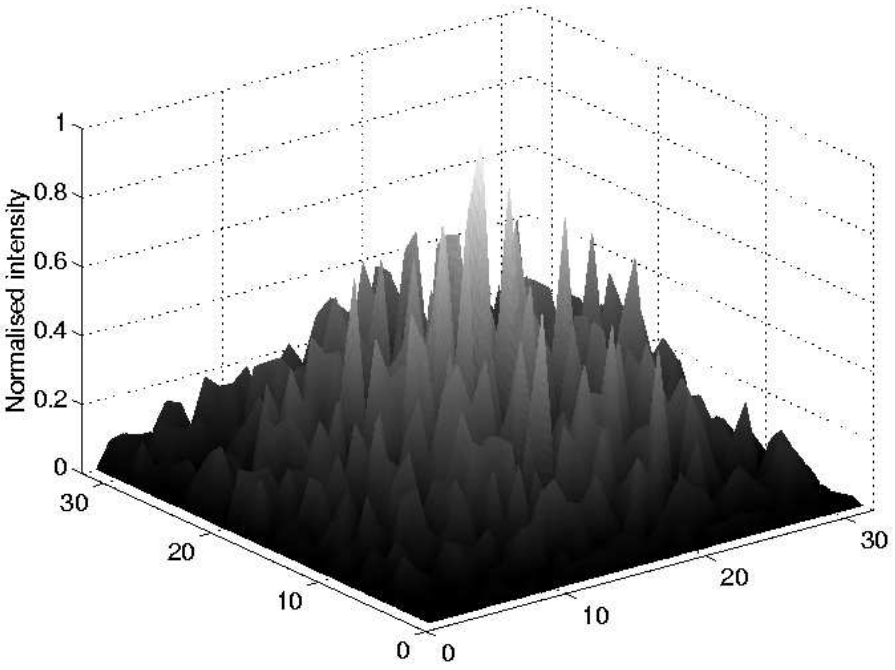
We can see in figure 15 that the largest benefit, 29%, in using the filters produced after two iterations is with a SNR of about 0 dB. When the SNR is about 6 dB it is better to use the filters produced after 1500 iterations.

The dynamic range curves comparing the performance of the filters after 2, 4 and 1500 iterations with the performance of the filters calculated with the matrix method, are shown in figure 16 and are what one would have predicted based on the knowledge gained from the probability of discrimination curves. The filters which are obtained with the matrix method give the lowest required dynamic range for high SNR since they are orthogonal to the input patterns. However, as the SNR decreases the curves meet and at very high noise levels the filters obtained after only 2 or 4 iterations perform slightly better.

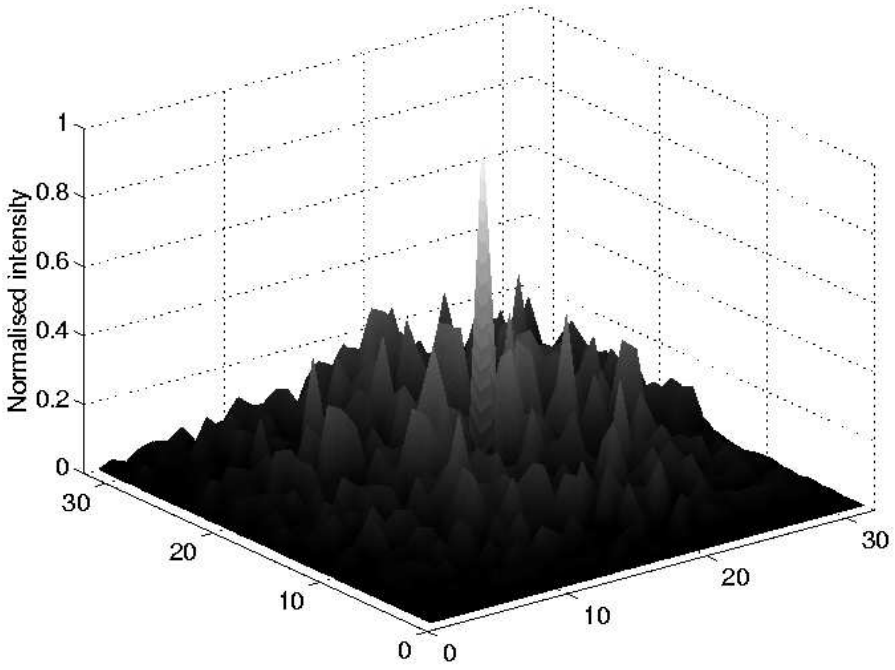
Before we discuss the trade-off between probability of discrimination and dynamic range, we are going to investigate the height of the outer products of the correlations when the 2 iteration filters are used. In figure 17 we can see the correlations between the first training pattern $\mathbf{s}_1$ and the corresponding filter $\mathbf{g}_1$. Subfigure (a) shows the correlation of $\mathbf{s}_1$ with the filter, $\mathbf{g}_1$, obtained after 1500 iterations and subfigure (b) shows the correlation of $\mathbf{s}_1$ with the filter, $\mathbf{g}_1$, obtained after 2 iterations. We can see in figure 17 that the outer products are lower when the filter obtained after 2 iterations is used. In fact, none of the outer products is now higher than 50% of the correlation peak compared to more than 70% of the correlation peak with the filter obtained after 1500 iterations. This is a very important improvement because now we can not only correctly recognise the pattern, but also locate it in the input scene if its exact location is not known. In addition, we can see that the correlation peak is sharp. This is important when more than one target exist in the input scene, in which case the two or more peaks will be distinguishable even if one is near the other.

The reduction of the outer products is even more prominent in figure 18, which shows the correlation between the seventh training pattern, $\mathbf{s}_7$ and the first filter, $\mathbf{g}_1$, obtained after 1500 iterations (subfigure 18 (a)) and after 2 iterations (subfigure 18 (b)). None of the outer products is higher than 50% of the auto-correlation peak value, when the 2 iteration filters are used, while with the 1500 iteration filters there were outer products which were as high as 80% of the auto-correlation peak value. This reduction of the outer products allows us to use the filters obtained with the SS algorithm after 2 iterations to recognise or discriminate input patterns. The filters allow recognition even when the exact location of the object in the input scene is not known, at least when no noise is present in the input.

So from the two graphs, the one for the probability of discrimination (figure 12) and the one for the dynamic range (figure 13), we can see that there is a trade-off between probability of discrimination and dynamic range. If the dynamic range of the system is absolutely critical, then one can choose to use the filters which are completely cross-orthogonal to the input images, thus minimising the required dynamic range at the expense of probability of discrimination at higher noise levels. In the opposite case when one wants to maximise the probability of discrimination, then the filters obtained after only 2 iterations give the best results of all. Another consideration is the type and amount of noise present. If the main type of noise present is system noise then dynamic range is critical and the filters created with the matrix method may be the best choice. If on the other hand, there is a lot of input noise and not a lot of system noise then one can sacrifice dynamic range for a higher tolerance to input noise which is provided by the filters produced after only 2 iterations. In addition, our final decision of which filter to use must also take account of the height of the outer products. When the exact location of the object in the input scene is not known, it is better to use the filters produced with the SS algorithm after 2 iterations, even if the dynamic range required by the recognition system is higher.

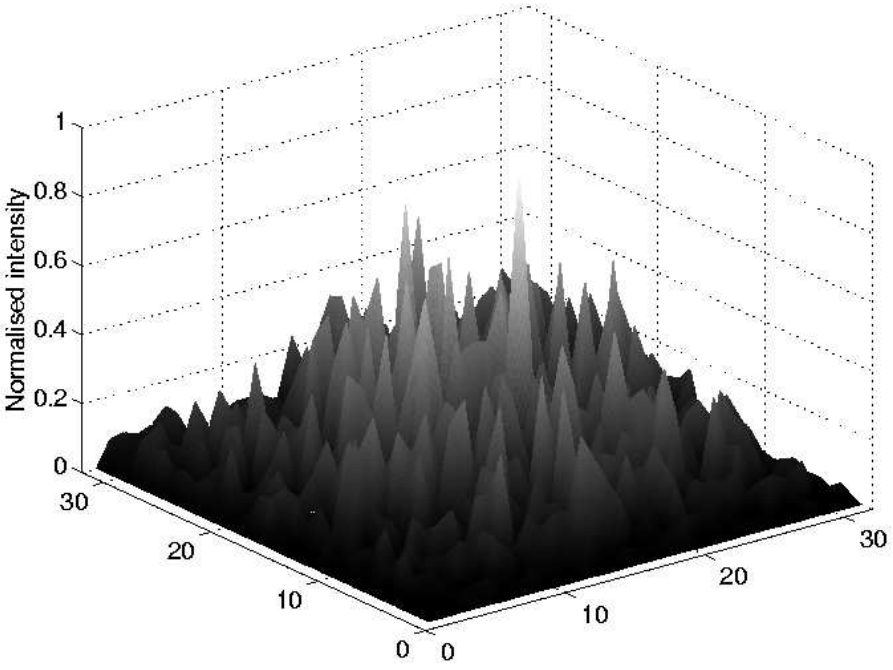


(a) $\mathbf{s}_1 \otimes \mathbf{g}_1$, 1500 iterations, PCE=0.027

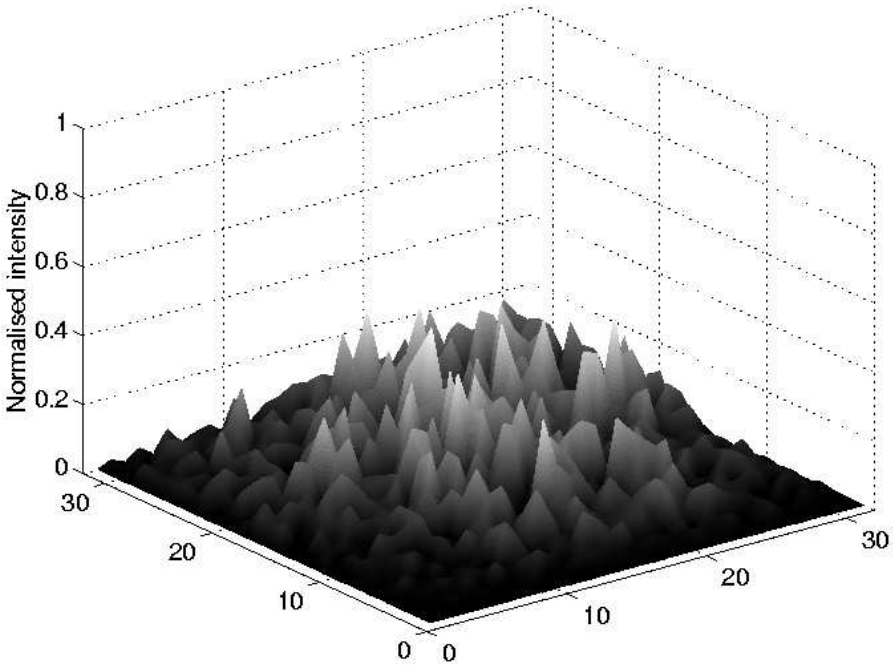


(b) $\mathbf{s}_1 \otimes \mathbf{g}_1$, 2 iterations, PCE=0.1

Figure 17 Correlation plane intensity for correlation between pattern 1 and filter 1 after 1500 and after 2 iterations



(a) $\mathbf{s}_7 \otimes \mathbf{g}_1$, 1500 iterations



(b) $\mathbf{s}_7 \otimes \mathbf{g}_1$, 2 iterations

Figure 18 Correlation plane intensity for correlations between pattern 7 and filter 1 after 1500 and after 2 iterations

7 Conclusions

We have described the development of our similarity suppression algorithm for calculating the filters for inner product pattern recognition correlators. The algorithm de-correlates the training patterns to make a set of filters that have smaller inner products with the input patterns. The algorithm is unbiased as to the order of presentation of the training patterns. We have shown through computer simulations using binary, bipolar patterns that the algorithm gives a higher probability of discrimination. The filters showed a high tolerance to additive input noise even when the input had a low SNR. The dynamic range required by the detector at the output of the recognition system is reduced by 25 dB (compared to matched filters) after the use of the algorithm, which is important for optical systems, which usually have a poor dynamic range. So this algorithm will enable optical correlators to recognise patterns having a larger number of pixels than matched filters alone. Moreover, the algorithm is formulated in terms of image operations and so has a form that is itself amenable to optical implementation. We have proved the mathematical equivalence between the SS algorithm and the matrix method proposed by Caulfield and Maloney[16] for designing mutually orthogonal filters. In addition, we have shown using computer simulations that the filters produced after only two iterations of the SS algorithm show better discrimination ability in the presence of additive input white noise than the fully converged filters and the filters produced using Caulfield's method. In addition, they produce a sharp correlation peak and low sidelobes which allow the input patterns to be recognised even when their location at the input scene is not known. These two benefits come at the cost of higher dynamic range required by the recognition system.

Acknowledgments

We would like to acknowledge valuable discussions with Prof. John Midwinter. We thank EPSRC, DTI and Mr I. Stamos for their financial support.

References

1 Bahri, Z. and Kumar, B. V. K. Vijaya*, J. Opt. Soc. Am.,* A 5(4) (1988) 562-571
2 Casasent, D., *Applied Optics*, 23 (1984) 1620-1627
3 Kumar, B. V. K. Vijaya, *J. Opt. Soc. Am.*, A 3 (1986) 1579-1584
4 Ravichandran, G. and Casasent, D., *Applied Optics*, 31(11) (1992) 1823-1833
5 Figué, J. and Réfrégier, P., *Applied Optics*, 32(11) (1993) 1933-1935
6 Laude, V. and Réfrégier, P., *Applied Optics*, 33 (20) (1994) 4465-4471
7 Réfrégier, P., *Optics Letters*, 15 (15) (1990) 854-856
8 Javidi, B, Réfrégier, P. and Willett, P., *Optics Letters*, 18 (19) (1993) 1660-1662
9 Weaver, C. and Goodman, J., *Applied Optics*, 5 (7) (1966) 1248-1249
10 Vander Lugt, A., *IEEE Trans. Inf. Theory*, IT10 (1964) 139-145
11 Gregory, D. A., Kirsch, J. A. and Tam, E. C., *Applied Optics*, 31 (1992) 163-165
12 Juday, R. D., *Applied Optics*, 28 (1989) 4865-4869
13 Kilpatrick, R. E., Gilby, J. H., Day, S. E. and Selviah, D. R., *Optical Pattern Recognition IX SPIE* Vol. 3386 (1998) 70-77
14 Gardner, M. C., Kilpatrick, R. E., Day, S. E., Renton, R. E. and Selviah, D. R., *J. Opt. A: Pure Appl. Opt*. 1 (1999) 299-303
15 Van Trees, H. L*., Detection, Estimation and Modulation Theory: Part 1*, (Wiley, New York) 1968
16 Caulfield, H. J. and Maloney, W. T., *Applied Optics*, 8 (11) (1969) 2354-2356
17 Press W. H., Flannery, B. P., Tenkolsky, S. A. and Vetterling, W. T., *Numerical Recipes The Art of Scientific Computing*, (Cambridge University Press), 1988
18 Casasent, D., Kumar, B. V. K. Vijaya and Sharma, V*., Proc. Soc. Photo-Opt. Instrum. Eng*. 360 (1982)
19 Kumar, B. V. K. Vijaya, Bahri, Z. and Mahalanobis, A., *Applied Optics*, 27 (2) (1988) 409-413