THESIS REPORT

Master's Degree

On Image Coding and Understanding: A Bayesian Formulation for the Problem of Template Matching Based on Coded Image Data

by E. Frantzeskakis
Advisor: J.S. Baras

M.S. 90-5

ISR
INSTITUTE FOR SYSTEMS RESEARCH

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

On Image Coding and Understanding

A Bayesian Formulation for the Problem of Template Matching based on Coded Image Data

by
Emmanuil N. Frantzeskakis

Thesis submitted to the faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements of the degree of
Masters of Science
1990

Advisory Committee :
Dr. J. S. Baras, Advisor
Dr. P. Krishnaprasad
Dr. L. Kanal

Abstract

Title of Thesis : On Image Coding and Understanding :
A Bayesian Formulation for the Problem of Template Matching
based on Coded Image Data

Name of degree candidate : Emmanuil N. Frantzeskakis

Degree and Year : Master of Science, 1990

Thesis directed by: John S. Baras,
Professor, Electrical Engineering Department

Some instances of the template matching problem, primarily for binary images corrupted with spatially white binary symmetric noise, are studied. We use the pixel-valued image data as well as data coded by two simple schemes, a modification of the Hadamard basis and the coarsening of resolution. Bayesian matching rules residing on M-ary hypothesis tests are developed. The performance evaluation of these rules is studied.

This approach to the matching problem is intended to show the trade-off between the quantization and external noise with respect to the ability of detecting an object of the image. We consider the case of the black square template in white background or without known background as well as synthetic template without known background. We call external noise the noise generated at the moment we receive the uncoded image, in which case we have a "corrupt-code-detect system", or the noise coming as the effect of the transmission of the coded image over a noisy channel, in which case we have a "code-corrupt-detect system". In both cases the noise is assumed to

be white.

The sum-of-pixels and the histogram statistics are introduced in order to overcome the computational load induced by the correlation statistic with the penalty of an augmented probability of false alarm rate.

What is intended to be shown in the present work is the usefulness and ability of combining an image coding technique with an algorithm for extracting some "base" information used in image understanding. Numerical and simulation results are given.

Acknowledgments

I wish to thank my advisor Dr. John Baras for his decisive contribution to this work and more specifically for pointing to the appropriate research focuses as well as patiently showing the way of writing a scientific document.

I am very thankful to George Panagopoulos for his help in developing the computer code which produced the results of this work. I would also like to thank Ran Xiaonong and Dr. Anderson for the helpful references they provided me with.

Finally I would like to thank the System Research Center for the financial support it provided to me in the form of a research assistantship (grands AFOSR-87–0073E and AFOSR-90–0105).

Contents

List of Figures

List of Tables

Chapter 1
The Template Matching Capability as a Measure of the Image Quality

1.1 The Template Matching as Part of the Image Understanding Problem.

The "Image Understanding Problem" poses the question of understanding what represents a given two-dimensional image. The scope of the problem can further be expanded if we consider a sequence of such images at subsequent time instances, and questioning about time-dependent events, like the relative motion of the objects represented.

Consider now a special case of the problem for fixed time parameter and a multiple gray level digital image. A general structure of a system proposed by Marr [Ros90] for solving this special case of the understanding problem is shown in figure 1.1. The input digital image usually is an array of pixels whose values are the gray levels representing the brightness at a closely spaced grid of points. Segmentation and feature detection can be regarded as assigning labels to the image pixels indicating classes to which the pixels belong, like light/dark, edges/non edges, being a member of some known pattern or not e.t.c. So, after segmentation and feature detection, a symbolic image in which pixel values are labels is obtained.

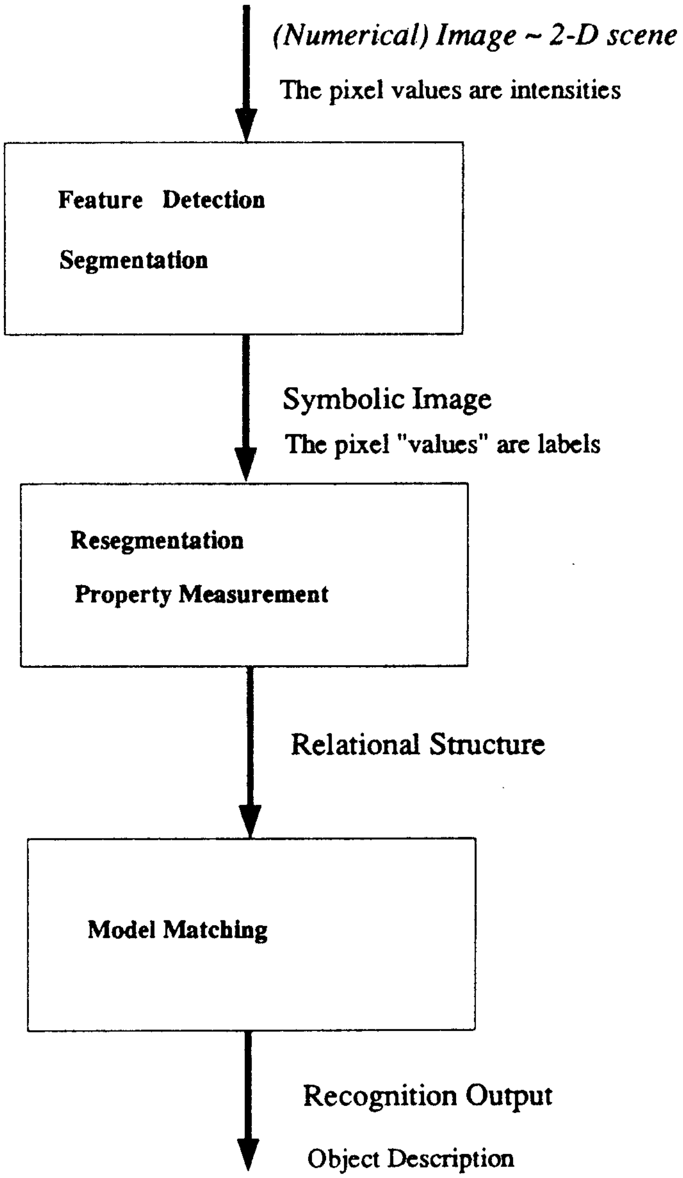


Figure 1.1 Structure of a generic image understanding system as proposed by Marr

Labeled pixels satisfying some specific criteria are grouped together into image parts resulting to a new (re)segmentation output and properties of these parts like area, orientation e.t.c. are measured; in this way a relational graph is formed. Now the object recognition problem can be viewed as finding subcollections of the image parts whose properties and relations satisfy certain constraints. The image understanding problem still requires information about the relations among these objects in the given image, and this wouldn't be difficult to find if the scheme described so far could deal satisfactorily with real world images.

Unfortunately, this is not the case. Some of the reasons for this failure are :

1. The process discussed is a strictly bottom-up one; human vision has a mixed top-down and bottom-up nature. That is we simultaneously process important local characteristics as well as consider global features of an object when we perform object recognition.
2. The algorithms which implement the tasks constituting the recognition process discussed are often ad-hoc and heuristic and consequently do not admit systematic performance evaluation. Typically the latter is based on experiments and simulations with small samples resulting to misleadingly optimistic performance predictions.
3. The noise effects which are present in real world images usually are not adequately or at all taken into account.

1.2 Approaches to the Template Matching Problem

Let **I** be an $n \times n$-pixel multiple gray level image (with Σ gray levels) and **T** be an $m \times m$-pixel template pattern. The exact template matching problem amounts to finding all (i,j) positions in **I**, i,j = 1,2,...,n-m+1, such that **I**[i+k,j+l]=**T**[k,l], k,l = 0,1,...,m-1. It is easy to see that this problem is equivalent to the string matching problem with text string length : $n_1=n^2$ and pattern string length $m_1=m^2$. So, we can take advantage of the existing literature on this problem and solve it in time $O(n_1)$ [KMP77, BoMo77], or in time $O(\log\Sigma n_1 \log m_1)$ [FiPa74], where Σ is the number of gray levels in the image.

Let's consider now a binary image **I**, assigning 0 and 1 as the values of white and black pixels respectively, which is corrupted by Binary Symmetric Noise with pixel inversion probability equal to ϵ. Suppose that given an image **I** and an $m \times m$ window on it , call it $\mathbf{I}_{\mathbf{w}[i,j]}$ (i.e. $\mathbf{I}_{\mathbf{w}[i,j]}[k,l] = \mathbf{I}[i+k,j+l]$, k,l=0,1,...,m-1), we have to decide which one of two possible templates $\mathbf{T}_1$, $\mathbf{T}_2$ should be the original noise-free image window, represented by the observed data $\mathbf{I}_\mathbf{w}$.We will follow a statistical interpretation as in [DuHa73]. We have :

$$Pr\left\{\mathbf{I}_{\mathrm{w}[i,j]} \mid \mathbf{T}_1\right\} = (1-\varepsilon)^{\#of\ matching\ values} \times \varepsilon^{\#of\ non\ matching\ values}$$

$$= \prod_{k,l} (1-\varepsilon)^{1-|\mathbf{I}[i+k,j+l]-\mathbf{T}_1[k,l]|} \times \varepsilon^{|\mathbf{I}[i+k,j+l]-\mathbf{T}_1[k,l]|}$$

and we can get a similar expression for $Pr\left\{\mathbf{I}_\mathrm{w} \mid \mathbf{T}_2\right\}$. In practice we have $\epsilon<0.5$ and so the *maximum likelihood* rule :

$$decide\ \mathbf{T}_1\ if\ Pr\left\{\mathbf{I}_\mathrm{w} \mid \mathbf{T}_1\right\} > Pr\left\{\mathbf{I}_\mathrm{w} \mid \mathbf{T}_2\right\},\ or\ else\ decide\ \mathbf{T}_2$$

reduces to

decide for the template with the least number of mismatces with the window I_w

Pictorially we have the decision regions given on figure 1.2.:

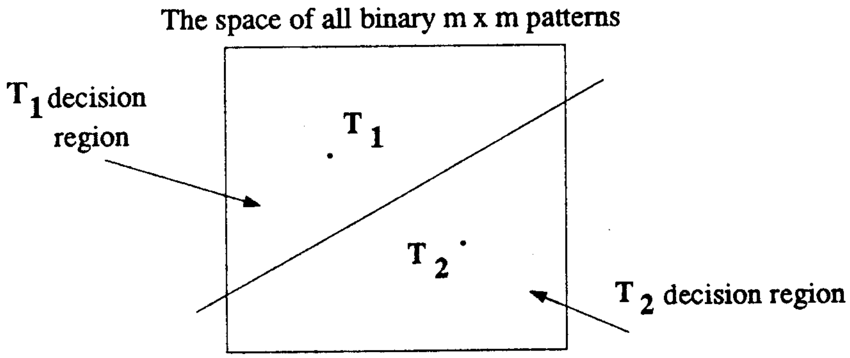


Figure 1.2 Picture of the decision regions for two candidate templates

More practically, let us suppose that we are given only one template T in order to decide whether it does or does not match the image window I_w. Intuitively a “maximum likelihood” way of thinking would lead us to a decision region looking like a sphere in the space of $m^2 - binary$ patterns centered at T and having radious some integer K. In simple words we will decide matching if the number of mismatches is at most equal to K. The distance implied by the sphere is the Hamming distance $d_H(I_w,T)$ between I_w and T. This situation is pictorially shown in the figure 1.3:

For the case of images corrupted by additive white noise our problem is equivalent to the string matching problem with K mismatches and therefore it can be solved in time $O(n_1 \log m_1)$ [AmFa89]. Here we will not consider the computational complexity of the problem; we will rather pose the question of determining some "optimal" value for K, given some information about the source of disruption we want to overcome by introducing a mismatch tolerance. More specifically, in chapter 2 we find K as a function of the noise parameter ϵ; this question has to do with the reliability of the matching rule.

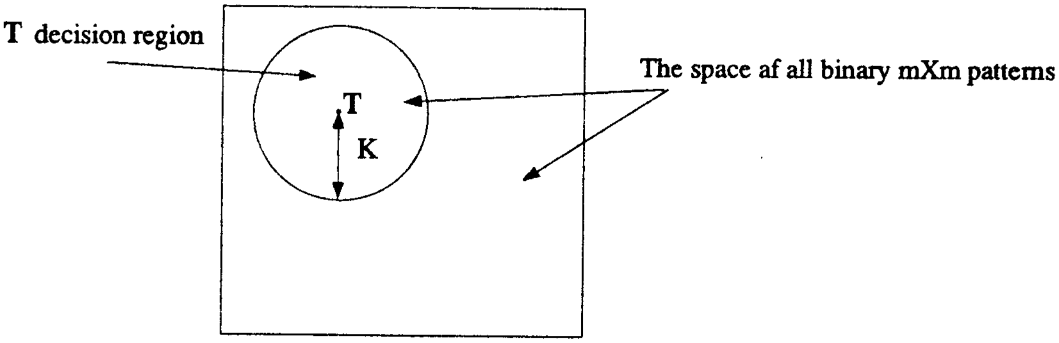


Figure 1.3 Decision regions using Hamming distance

Let us return now to the case of multiple gray level images. We still have fast algorithms for solving the K-mismatches problem as a string matching problem (time $O(n_1\sqrt{m_1}\log m_1)$ [Abr87]). In [AmFa89] the problem of finding the occurrences of a non rectangular pattern of height m and area α in an n×n text with no more than K mismatches is considered and it is shown to be solved in time $O(Kn^2\sqrt{m}\log m + K^2m^2)$.

The number of mismatches is a descriptive statistic when applied to text strings; given that all the letter transitions, e.g. an "a" turned into "e", an "a" turned into "q" e.t.c., are treated the same way, the number of these transitions is good enough to describe the corruption occurred. On the other hand for the case of the template matching in multiple gray level images we wouldn't like to treat transitions among all gray levels the same way, since transitions among neighbor gray levels are more likely to occur.

There are various metrics, or distortion measures used in classical template matching approaches in order to account for the lack of uniformity just mentioned. Some of them [DuHa73] are :

1. The absolute distance : $\sum_{k,l} |\mathbf{I}[i+k,j+l] - \mathbf{T}[k,l]|$
2. The Eucledian distance : $\sum_{k,l} (\mathbf{I}[i+k,j+l] - \mathbf{T}[k,l])^2$
3. The cross correlation : $R(i,j) = \sum_{k,l} \mathbf{I}[i+k,j+l] \times \mathbf{T}[k,l]$

   This is equivalent to the Eucledian distance if we assume constant picture energy in the window, i.e. that $\sum_{k,l} \mathbf{I}^2[i+k,j+l]$ is independent from the position (i,j). It can efficiently be computed by using FFT.
4. The normalized cross correlation : $N(i,j) = R(i,j) / \sum_{k,l} \mathbf{I}^2[i+k,j+l]$

We can now repeat the previous method used with the Hamming distance. That is suppose we are given two candidate templates $\mathbf{T_1}$ and $\mathbf{T_2}$ to associate with $\mathbf{I_w}$. We can use one of the above distortion measures in order to compare them and come up with a decision. Second, if we have a single template to decide about matching, we will compare the (generalized) distance d($\mathbf{I_w}$,T) with some threshold, which can be interpreted as the radius of some sphere centered at T.

If we introduce "maximum likelihood" reasoning for the multiple gray level image, with Σ gray levels, the role of the metric (the Hamming distance for the case of the binary image) is undertaken by the so called "transition probability";

this is based on a vector of Σ^2 different kind of matches /mismatches among the gray levels, with $\Sigma(\Sigma-1)$ degrees of freedom.

In chapters 2 and 4 the Bayesian reasoning is introduced for the template matching problem and is further discussed in chapter 5. In the Bayesian approach the "transition probability" metric is used and the decision regions for matching are determined so that a cost function is minimized.

1.3 Performance Evaluation of "Matching" Algorithms Acting Upon Noisy Image Data

Consider the following simple instance of the Template Matching problem: A binary image represents a black square of size $m \times m$ pixels in white background. We want to find the $m \times m$ windows on the image containing part or all of the black square. Evidently it suffices to detect a single black pixel in the window. We still can compute the correlation R of the test window with the "full black template", i.e. with a square template of size $m \times m$ pixels consisting entirely of black pixels (value = 1); R actually gives the number of black pixels in the window and at the same time it is a measure of overlap of the test window with the black square. We can decide "part or all of the square detected" if R >0 and "square not detected" otherwise.

Suppose now that the image is transmitted over a noisy channel; as an effect of this several pixels are inverted. We model this phenomenon by saying that the image is corrupted by White (in space) Binary Symmetric Noise with some inversion probability ϵ. From now on we will call this kind of noise "BSC noise" since it comes as an effect of the Binary Symmetric Channel :

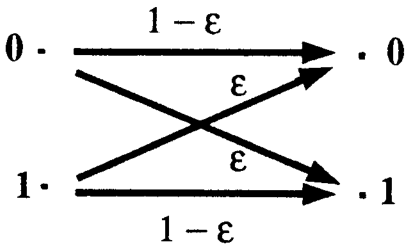


Figure 1.4 The Binary Symmetric Channel

Again, given a window of the noise corrupted image we want to infer about the existence of the black square in it. Now it does not suffice to detect just one black pixel. Though we remind the reader that when we are talking about the noise free image the number of the black pixels in the window gives a measure of the overlap of the window with the square. We will use this number in order to answer the question about the decision for a square or not. Effectively, we have to define some threshold t in the range of $0 to m^2$ which will determine the region $R_1 = t, t+1, \ldots, m^2$ for a positive answer to the question posed. We note that in the noise free image we had $t = 1$ and $R_1 = 1, 2, \ldots, m^2$.

For a given level of noise ϵ we want to determine the threshold value t in such a way that the probability of erroneous decision be minimized; this error probability will be a linear combination [Nar89] of

$$P_{fa} = Pr\{decide\ there\ is\ a\ square\ while\ it\ is\ not\ present\} \quad and$$

$$P_m = Pr\{decide\ there\ is\ not\ a\ square\ while\ it\ is\ present\} \quad .$$

The performance evaluation we want to make for such an optimal rule amounts to:

1. Draw the (P_{fa}, P_d) curve of the optimal Bayesian rules parametrized by the noise parameter ϵ, where P_{fa} stands for probability of "false alarm", and P_d stands for probability of "detection", i.e. probability of deciding there is a square while it is present, Note that $P_d = 1 - P_m$.
2. Draw the Receiver Operating Characteristic (ROC) curves of the test, i.e. the$P_d\,(P_{fa})$ curves, for certain values of ϵ.

In order to obtain the above we use a "hypothesis testing" approach, broadly used in communications detection theory [Poo88, pp7]. A description of the procedure used to build such decision rules is given in chapter 2. At this point we will only state that the square detection problem initially leads to an M-ary hypothesis test; M here equals the number of different Hamming weights[1] in all possible windows of the noise free image. This test is in turn reduced to a binary hypothesis test. The curves 1 and 2 required for the performance evaluation of the test are given in plot 3 and plot 4 respectively in appendix B (the size of the square used for them is m=8); a discussion of these curves will be done in the next chapter. In the table 1.1 a set of (ϵ,t) pairs is given showing the effect of noise upon the threshold selected for the optimal Bayesian rules (recall : $t < m^2 = 64$ and in the ideal no noise case $t = 1$).

[1] Hamming weight of the window here is the number of black pixels in it. In the subsection A.1 it is shown that $M = \frac{m \times (m+1)}{2} + 1$.

epsilon	threshold
0.02	5
0.1	13
0.2	21
0.3	29
0.4	37

Table 1.1 The threshold as a function of the noise

1.4 Image Coding Techniques Appropriate for Input to a Template Matching Algorithm

In the last few years a great variety of image coding techniques has appeared in the literature. The interested reader should refer to [NeLi80], [NaKi88], [KIK85], [KBL87]. The compression obtained reaches the level of 0.25 bpp (bits per pixel), e.g. by using a technique of Entropy Coded Quantization for subband Image Coding [TaFa86].

For our purposes however we need coding techniques producing data acceptable as input to a template matching algorithm. But since such algorithms scan the image and perform some kind of local operations on it (i.e. operations acting on a window of the image and not on the entire extent of it) we should restrict ourselves to coding techniques for which the coded data preserve the local characteristics of the uncoded image; therefore we should invoke some block coding technique.

A block coding scheme splits the image into blocks of some size (e.g. block size of 4×4 pixels) and encodes each block separately of all the others as a vector of random variables. The coded image will be a matrix of size $c_n \times c_n$ *blocks*

, where c_n ="original image dimension" / "block dimension" (both assumed to be square). Each element of the matrix is the coded data for the corresponding image block. For example a 256×256–pixel image coded in blocks of size 4×4 pixels will result in a coded image of size $c_n \times c_n$ =64×64 codewords. There is a number of different approaches to block coding that have been proposed. Here we recall just a few; the ones which appear to be both common in the literature and useful for the present work.

Suppose that the pixel values in the block form an N-component vector X.

1. In [KIK85] the Karhunen-Loeve Transform (KLT) is used to produce a vector Y of uncorrelated coefficients out of the vector X; then the coefficients of Y are quantized by using the Loyd-Max quantizer, while the number of bits assigned to each component is a function of its variance and the total number of bits available for coding the vector Y. We will briefly explain how KLT functions (note also that the Loyd-Max quantizer is presented in [Max60]) : KLT amounts to the transformation : Y = A.X, where A is an N×N matrix computed as follows :
   First find the covariance matrix of X : C_x=E[(X-EX)(X-EX)$^{T]}$. The rows of A are the normalized eigenvectors of the matrix C_x, i.e. they are solutions of the equation $C_xX=\lambda_iX$ where the λ_i's are the eigenvalues of C_x. If we want to reject K out of N coefficients we may do so by leaving out the eigenvectors which correspond to the K smaller eigenvalues of C_x. Then the mean square reconstruction error[2] will be equal to the sum of these eigenvalues. This is the minimum value we can obtain out of a transform of the form Y = A.X

[2] $E[(X-A^{-1}Y)^T(X-A^{-1}Y)]$

[NeLi80]. The KLT has two problems : It defines A in terms of the C_x matrix which in general is not stationary. Also the eigenvectors of C_x are not always distinct [NeLi80].

2. In [MAY82] the coefficients of X are normalized over their variances and the output vector Y is vector-quantized. A review of Vector Quantization (VQ) is presented in [Gra84], while applications in image coding are given in [NaKi88]

3. A popular transform substituting to some extent the KLT is the Discrete Cosine Transform (DCT) which overcomes the problems mentioned for the KLT [ANR74].

   In the DCT we have Y = A.X, where

$$A = \{a_{ij}\},\ a_{ij} = \frac{2K(i)}{\sqrt{N}} \cos\left[(2j+1)\frac{i\pi}{2N}\right].$$

$$K(i) = \begin{cases} 1/\sqrt{2}. & i = 1 \\ 1 & i = 2,\ldots,N \\ 0 & otherwise \end{cases}$$

4. The Symmetric Hadamard Transform [NeLi80] for $N = 2^n$defined by

$$A = \{a_{ij}\},\ a_{ij} = \frac{1}{\sqrt{N}}(-1)^{b(i,j)},\ b(i,j) = \sum_{l=0}^{N-1} i_l j_l$$

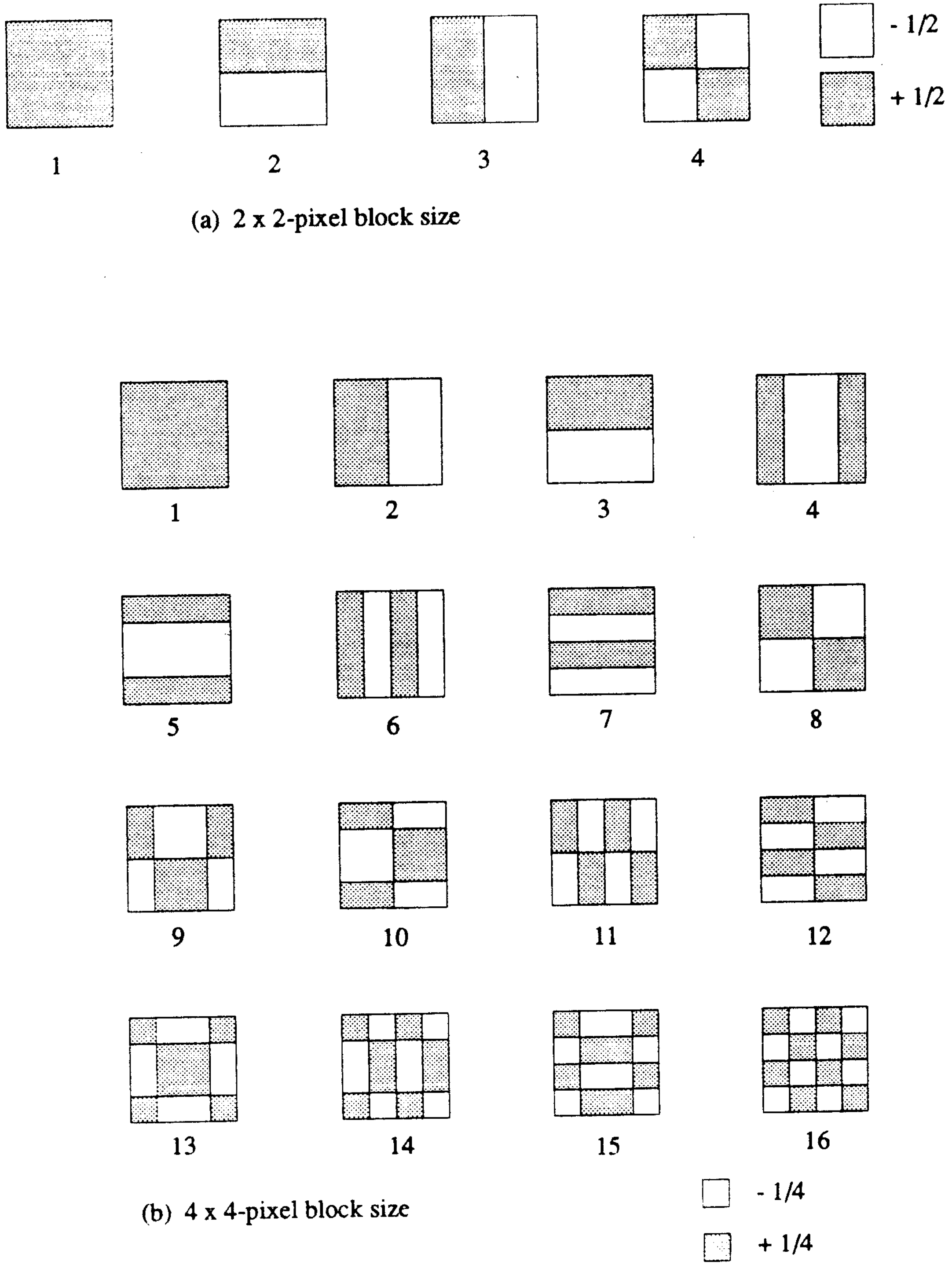
   where i_l, j_l are the bit values in the binary representation of i and j respectively in N bits. For example, for n=2 we have $1_{10} = 0001_2$, $2_{10} = 0010_2$ (where k_b stands for "representation of the number k in base b").

$$a_{11} = \frac{1}{\sqrt{4}}(-1)^{0\cdot 0+0\cdot 0+0\cdot 0+1\cdot 1} = -\frac{1}{2},\quad a_{12} = \frac{1}{\sqrt{4}}(-1)^{0\cdot 0+0\cdot 0+1\cdot 0+0\cdot 1} = \frac{1}{2}.$$

and similarly we can find $A_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$. If we take the 4 row vectors and split each one into two columns we obtain the 2×2 Hadamard basis given in figure 1.5a. The 4×4 Hadamard basis is given in figure 1.5b.

Another constraint posed in the coding procedure is the requirement that all the blocks should be represented by the same number of bits. In other words we should not exploit the Information Theoretic properties of the blocks, as the Huffman codes do [Bla87,pp64]. This is required because at the template matching time we will need to know the precise location of each block in the image file. If on-line search is required we will suffer considerable time delays. On the other hand, if auxiliary pointers are used to avoid this delay, then the space advantage may be lost. Anyway the trade-off between the speed and data compression will not be considered at this point. At this point we will require fixed length code.

A final constraint, which is imposed due to the matching algorithm complexity, is the requirement for the least possible size of the block-codebook. This will be discussed further in chapter 3.



(a) 2 x 2-pixel block size

(b) 4 x 4-pixel block size

Figure 1.5 The Hadamard basis

1.5 The Combined Compression — Feature Detection problem

Consider the following senario : We have a source image I, on which we can perform a template matching algorithm A in order to extract some feature. Alternatively we can compress the image producing the compressed version of it Q(I); after this we can reconstruct it producing R(I) from Q(I) and we perform again the algorithm A on R(I). We expect that a "good" compression procedure will not affect the "quality of the image", or it will not damage the "information content of the image". More explicitly we would like that the algorithm A acting on I give the same output as acting on R(I).

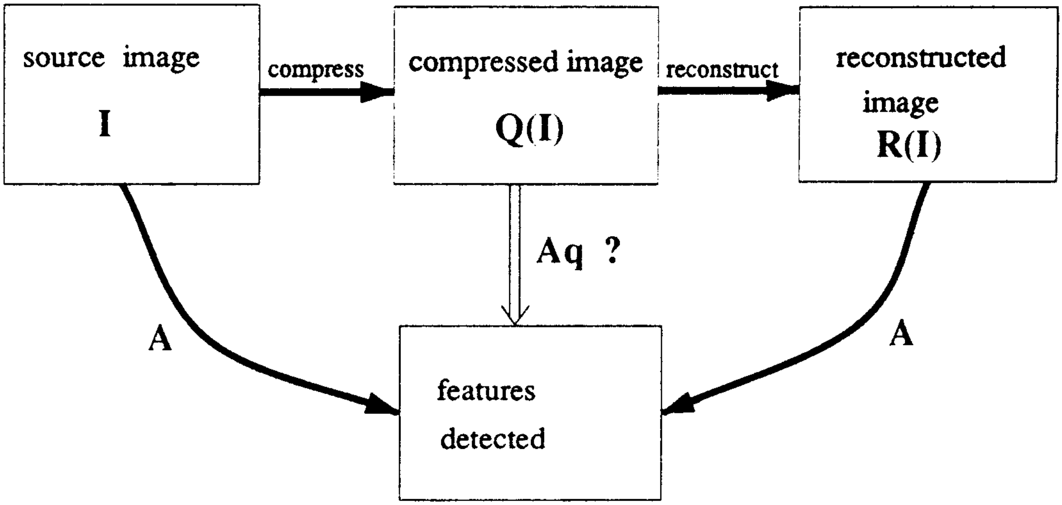


Figure 1.6 The feature detection problem

So the effect of the template matching algorithm can be used to check the "information persistence", i.e. the features identification detection capability after the compression/reconstruction process. (We note that implicitly we admit here that the "information" is also dependent on the detection algorithm A.) In general we expect that R(I) will contain less information (will provoke more decision errors when A is applied) than I contains.

Here we consider quantization as a simple form of compression. We also implicitly assume that the primary purpose of compression is to speed up transmission of the frame, or allow as to store it in an efficient way.

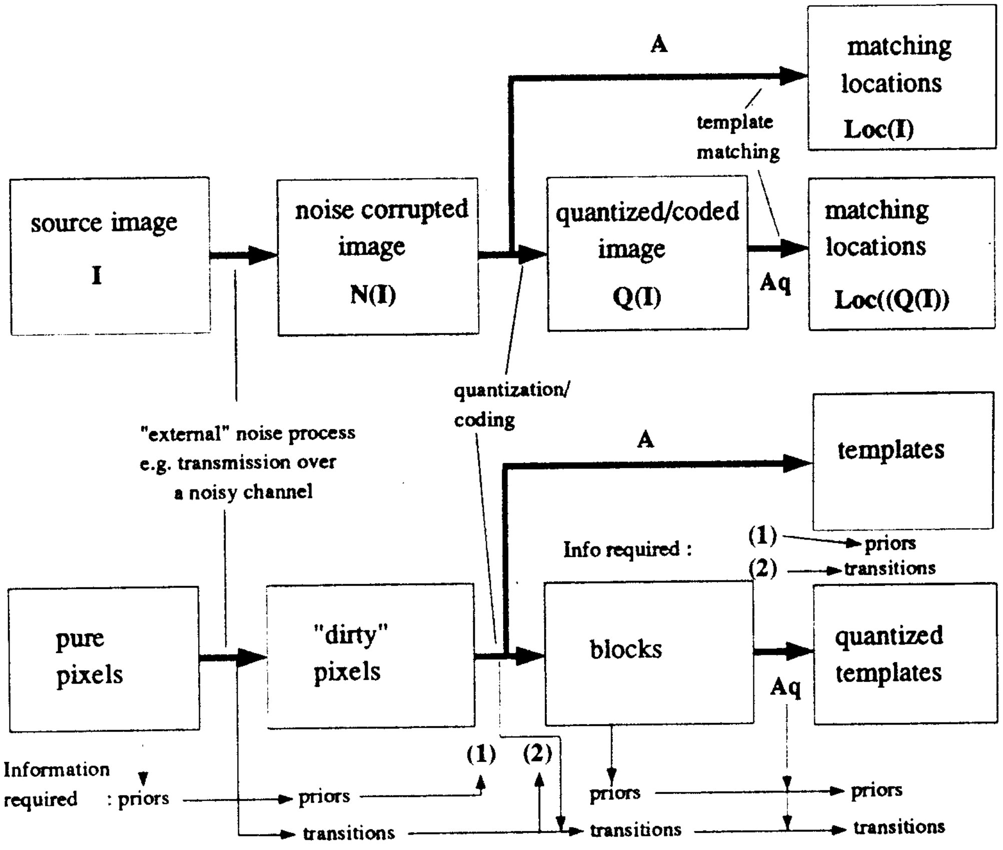


Figure 1.7 The "corrupt-code-detect" model

As it is depicted in the figure 1.6 we are interested in feature detection algorithms A_q acting on the coded data Q(I). More specifically: Can we have any advantage if we apply a feature detection algorithm A_q on the coded data Q(I) instead of applying A on the reconstructed data R(I) ? Or still, can we have any advantage if we apply A_q on the coded data Q(I) instead of applying A on the original data I ? The first question may be answered positively even at an intuitive level.We will argue for a positive answer to the second question as well.

Let us consider a more detailed diagram, see figure 1.7, in which we are not interested in the reconstructed image R(I); the noise free multiple gray level image can be characterized by its first order histogram, i.e. the relative frequencies of the gray levels; these frequencies (normalized to sum to unity) will be called *prior probabilities* or just *priors*. The introduction of noise means that several pixels of certain gray levels will be changed into some other levels. We will call the probabilities of such changes *transition probabilities*; these are determined by the noise parameters. The compression procedure gives rise to some new entities: the *blocks*, which have their own "priors" found rather experimentally, and their "transitions", which are determined by the pixel-level transitions and the compression rule itself. What we want to argue about is that the distortion introduced by the compression scheme may "kill" part of the external noise (transmission channel noise) and therefore the compressed image Q(I) may be more reliable than the noise corrupted N(I) one. Finally, the matching algorithms A and A_q will be applied on data of the size of the template and give the positions in the image where a "match" is decided. A window of an image having the size of the template has also its "priors" and "transitions", which are determined by the ones of the block-level as well as the structure of the template.

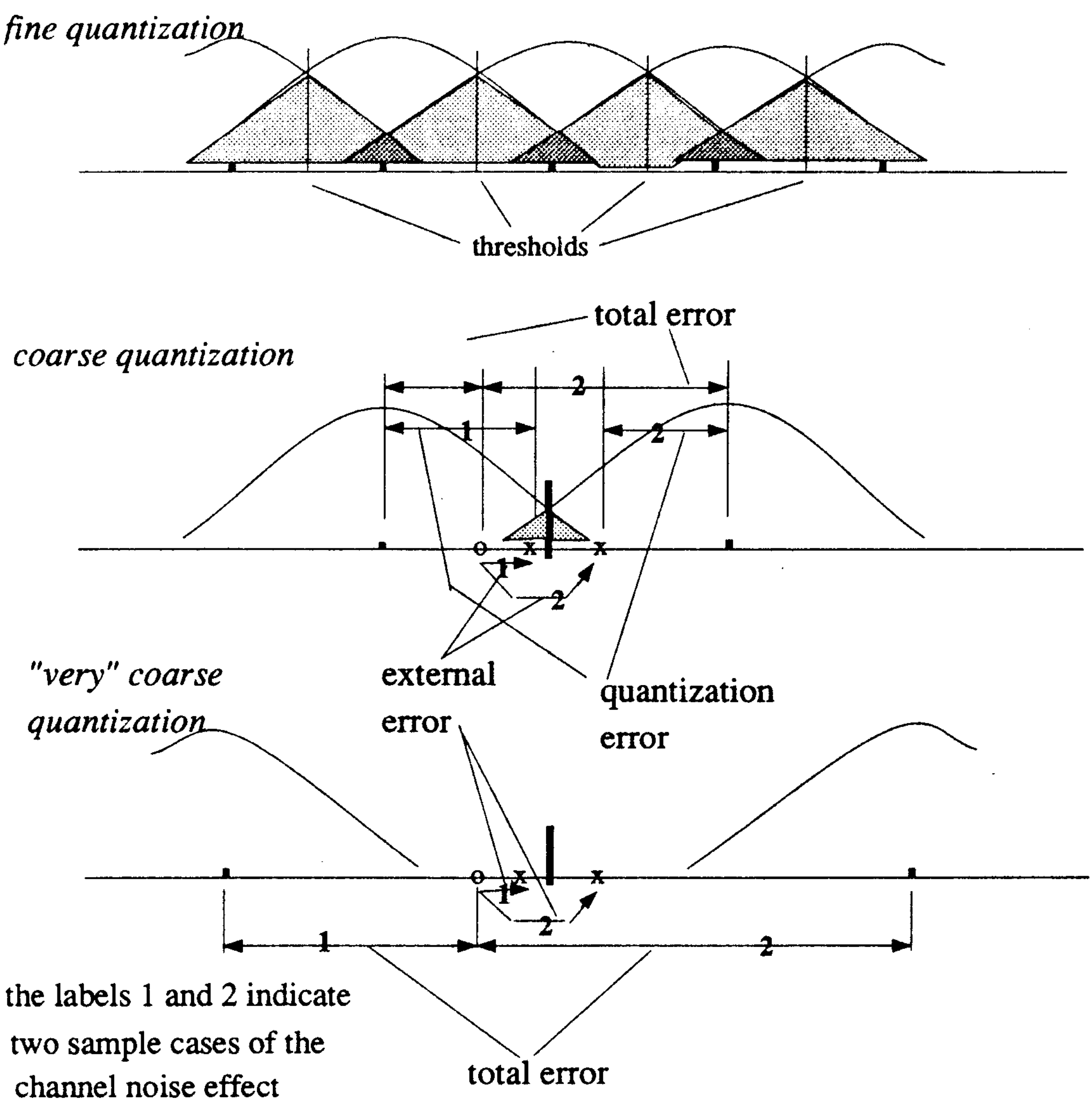


Figure 1.8 The noise effects for various scales of quantization

Suppose we use as coding scheme the Karhunen-Loeve transform described earlier. The vector Y reresenting the data for an image block contains uncorrelated coefficients which are real valued numbers. We scalar-quantize each one of them. As we see in figure 1.8 too fine quantization with respect to the noise parameter (e.g. the variance σ^2 for the case of Gaussian noise) is useless, since precision is lost because of the external noise (large shaded areas correspond to the error probabilities). Somewhat coarser quantization may "kill" part of the error introduced by the channel. Finally, too coarse quantization means that the quantization error dominates the channel error and we have even more loss in the precision. Now, if we suppose that the coefficients in Y are independent random variables, the quantized vector $\hat{Y}$ will also have independent components whose transitions are known. So the block-transition probabilities can be found as the products of the components (pixel level)-transition probabilities. Also the image windows can be thought as vectors with independent block-components and so we can find their "transition probabilities" as well as their "priors".

Hamming weight	quantization value
< 2	white
> 2	black
$= 2$	flip a fair coin and quantize according to the output value

Table 1.2 A simple quantization scheme

Although straightforward the above scheme of quantization/coding is unrealizable because of the very large size of the block codebook, which as described at the end of the previous subsection should be small. A quantization scheme which is simultaneously simple and realizable is the coarsening of the scale of the image. Consider a binary image corrupted by BSC noise with parameter ϵ

and quantize each 2×2 pixel block into a full black or full white block as follows described in table 1.2:

Here we have an effective compression rate equal to 0.25 bpp. The codebook size is 2. The block-priors are both equal to 0.5. The transition probabilities are derived in chapter 2. We can now use the compressed image to test for match with the full black template. The ROC of the test is given in the plot 6. We note that the ROC curve is closer to the point (0,1) than the one in the plot 4 (at the specific noise level ϵ=0.1), which shows that the test on the compressed image is a "more reliable" test as it was anticipated.

1.6 Identification of the Problem Studied

Image compression and image understanding have been traditionally developed as separate fields of research. In image compression one is primarily interested in designing efficient coding schemes which allow the transmission and accurate reconstruction of images at low bit data rates. In image understanding one is primarily interested in extracting high level or "content" information from the image pixels. It is clear that both fields address the problem of efficient information extraction and that in principle they are strongly coupled. We believe that in order to understand the top-down and bottom-up processing performed in human vision, one needs to unify these two fields. It is the purpose of this thesis to perform an initial investigation in this direction. More specifically we are interested in linking image coding and object recognition quantitatively in some simple generic examples. Such progress is necessary for our long range goal of unifying image compression and image understanding.

To be more specific we have considered recognition based on a well structured feature detection algorithm, which we developed, resembling Template Matching algorithms. We first started discussing the template matching problem and we shall present our feature detection algorithm in chapters four and five. In trying to link image processing with image understanding we introduce a novel idea. Namely the "numerical image" for us is not necessarily a pixel image but a block-coded image. In this way the image compression (performed by the code) is linked directly to image understanding. We can now expect to observe the two types of interference or contamination induced on the image data; namely the "external noise" introduced by the physical media (e.g. the transmission channel), and the "quantization noise" introduced by the coding procedure (see fig 1.6). What we are interested to analyze is the process where image recognition can be performed based on the reduced (coded) image data. It is clear that if we reduce the image beyond certain point, recognition capability will be affected. On the other hand it is desirable from a practical point of view to design schemes which can perform object recognition on the basis of block-coded data and not requiring the full image reconstruction. It is clear that what we need to understand and quantify is the explicit relationship between code efficiency and image compression with the required performance. So the reader should not expect to find in this work a review of the image coding techniques or a proposed solution of the 2–D Image Understanding problem.

In the first chapter we introduced the notions of image understanding, template matching, hypothesis testing, image coding and block coding for images. We also identified the problem we will focus on and put it in the broader perspective of the image understanding problem. We further gave examples of the particular aspect we will follow throughout this work.

In the second chapter we show how the template matching problem in a priori known background, based on non coded image data can be formulated as a hypothesis testing problem and we describe the effects of noise on the reliability of the matching test.

In the third chapter we describe a coding procedure upon which a feasible matching test may be based; we give the block-priors and transitions. Related coding schemes which can be used in the future for possible extensions are mentioned.

In the fourth chapter the template matching problem on coded image data is formulated as a hypothesis testing; the noise effects are studied.

In the fifth chapter the Bayesian nature of the developed tests is highlighted; several sample tests are numerically evaluated so that the noise, compression and background knowledge effects be comparatively studied. Related problems are specified and further questions are posed.

Explanations on the algorithms implemented in actual computer code and performance evaluation plots summarizing our analysis are given in the appendix.

Chapter 2 Template Matching on a Binary Image

2.1 Binary Image Corrupted by AWGN: Formulation of the Template Matching Problem as a Hypothesis Testing Problem

We consider the following problem. A binary image represents a black m$\times$m-pixel square in white background. The image is transmitted over a noisy channel that causes some alteration onto the pixel levels. We model this phenomenon by saying that the image is corrupted by Additive White Gaussian Noise (AWGN) of some variance σ^2; we will shortly show the interpretation of this assumption. We want to find the m$\times$m-pixel windows of the image containing part or all of the black square. We will call this problem "the full-black template case in background" of the template matching problem.

We first find the possible relative positions between the target square in the noise free image and a test window scanning this image. Consider the case of m=2. Then all the possible patterns of part of a black square that could appear in a 2$\times$2 test window are shown in figure 2.1.a.

Apart from the all-white window, we notice that all possible patterns are constructed by placing the left-up corner of the window in each one of $3 \times 3 = 9$ places of a square part containing the black square (see fig. 2.1.b) and reading all possible 2$\times$2 squares. This observation applies for the general case too, so we get

(2m-1)×(2m-1) non all-white windows of size m×m when searching for the m×m black square. We also notice that for this case we have $(2m-1)^2+1 = M+1$ distinct patterns out of the 2^{m^2} possible ones.

The *M+1* distinct window patterns may constitute a set of states $H_0, H_1, \cdots, H_M$ (assign H_0 to the white pattern) in a likelihood ratio test [Poo88,pp10]. The state H_i will be characterized by a binary vector s$_i$ of length m^2. The observation will be the image window itself represented by a real valued vector y of length m^2. Corruption by AWGN means that for each component y^j of y we have :

$$y^j = signal + noise = s_i^j + n^j,\ n^j \sim N\left(0, \sigma^2\right)$$

and all n^j's are mutually independent.

```
0        1        2        3                  1     2     3
 o o      o o      o o      o o                o     o     o     o
 o o      o x      x x      x o               4     5     6
         4        5        6                   o     x     x     o
          o x      x x      x o               7     8     9
          o x      x x      x o                o     x     x     o
         7        8        9
          o x      x x      x o                o     o     o     o
          o o      o o      o o
                                                     (b)
              (a)           "o" : white pixel,  "x" : black pixel
```

Figure 2.1 Possible patterns in a 2×2–pixel test window

We now compute the prior probabilities of the states $H_i,\ i = 0, 1, \ldots, M$. For an n×n-pixel image we have a total of $(n-m+1)^2$ possible placements of the test image window; we also have $(2m-1)^2$ distinct placements, including non all-white patterns. So the prior probabilities are :

$$p_i = \frac{1}{(n-m+1)^2} = p, \; i = 1,2,\ldots,M$$

$$p_0 = \frac{(n-m+1)^2 - (2m-1)^2}{(n-m+1)^2} = 1 - Mp$$

The distributions relating the observation $\mathbf{y}$ with a specific state, i.e. the probability of taking $\mathbf{y}$ as a noise corrupted version of $\mathbf{s_i}$ are Gaussian:

$$H_i \; : \; \mathbf{y} \sim f_i(\mathbf{y}) \; : \; N\left(\mathbf{s_i}, \sigma^2 \mathbf{I}_{m^2}\right)$$

where I_{m^2} is the identity matrix of size $m^2 \times m^2$, so

$$f_i(\mathbf{y}) = \frac{1}{(2\sigma^2)^{m^2/2}} \exp\left\{ -\frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{s_i})^T (\mathbf{y} - \mathbf{s_i}) \right\}$$

The detection of the black square problem amounts to deciding for one of the M non all-white patterns, i.e. one of the states $H_i, \; i = 1,2,\ldots,M$, or for the white pattern, i.e. the state H_0. This situation can directly be formulated into a composite binary hypothesis testing problem; for, we group together the states $H_i, \; i = 1,2,\ldots,M$ into one that we will denote as $\tilde{H}_1$. For the resulting binary hypothesis test we will have [Poo88,pp43]:

1. The prior probabilities : $\tilde{p}_0 = p_0 = 1 - Mp, \;\; \tilde{p}_1 = Mp$.
2. The distribution of the observed data is:

$$\tilde{H}_0 \; : \; \mathbf{y} \sim \tilde{f}_0(\mathbf{y}) \equiv f_0(\mathbf{y})$$

$$\tilde{H}_1 \; : \; \mathbf{y} \sim \tilde{f}_1(\mathbf{y}) = E\left[f_i(\mathbf{y})\right] = \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{y})$$

3. The cost function (arbitrarily chosen): $c(\cdot,\cdot) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

4. The decision rule :

$$decide\ 0 \Leftrightarrow \frac{\tilde{f}_1(\mathbf{y})}{\tilde{f}_0(\mathbf{y})} < \frac{1-Mp}{Mp} = t\ .$$

which can be reduced to :

$$decide\ 0 \ \Leftrightarrow \sum_{i=1}^{M} \alpha_i \exp\left(\mathbf{y}^T \beta_i\right) < t\ ,$$

for constant α_i, β_i, t.

The generalized likelihood ratio list approach [Nar89] utilizes the *max* operator instead of the expectation operator E[.] which was used in the previously presented test. The resulting test in our application is :

$$\tilde{H}_0 \ : \ \mathbf{y} \sim \tilde{f}_0(\mathbf{y}) \equiv f_0(\mathbf{y})$$

$$\tilde{H}_1 \ : \ \mathbf{y} \sim \tilde{f}_1(\mathbf{y}) = \max_i f_i(\mathbf{y}), \quad i \in \{1, 2, \cdots, M\}$$

Although these tests guarantee some optimality given $\tilde{f}_1(\cdot)$, they do not guarantee any optimality given the distributions $f_i(\cdot)$, $i = 1, 2, \cdots, M$, which are the actually given ones. This fact has led us to the following formulation of an M-ary hypothesis testing problem.

We will not attempt to construct some pair of distribution functions $\left(\tilde{f}_0, \tilde{f}_1\right)$ as before; instead we will determine a cost function c(.,.) acting directly on the statesH_i, $i = 0, 1, \ldots, M$, which will implicitly do the grouping we need. More explicitly let :

$$\text{l.c.}(\,.\,,\,.\,) = \begin{array}{c|ccccc|} & 0 & 1 & \cdots & & M \\ \hline 0 & 0 & 1 & 1 & \cdots & 1 \\ 1 & 1 & & & & \\ \vdots & 1 & & \bigcirc & & \\ & \vdots & & & & \\ M & 1 & & & & \\ \hline \end{array}$$

We can now apply the theory supporting the M-ary test [Nar89]; this will give the following decision rule :

$$d \;:\; d(\mathbf{y}) = i \;\Leftrightarrow\; g_i(\mathbf{y}) \le g_j(\mathbf{y}),\ 0 \le j \le M,\ where$$

$$g_i(\mathbf{y}) = \sum_{\substack{k=0 \\ k \ne i}}^{M} p_k \left[c(k,i) - c(k,k)\right] f_k(\mathbf{y}),\ or$$

$$g_i = \begin{cases} p \sum\limits_{k=1}^{M} f_k(\mathbf{y}), & i = 0 \\ (1 - Mp) f_0(\mathbf{y}), & i = 1, 2, \cdots, M \end{cases}$$

which minimizes the mean cost value : $J(d) = E\left[c(H, d(\mathbf{Y}))\right]$. We note the fact that all g_i's are identical for i=1,2,...,M, which causes an ambiguity in the rule d. We resolve this ambiguity by defining

$$d \;:\; d(\mathbf{y}) = i = \min_{k} \{k \mid g_k(\mathbf{y}) \le g_j(\mathbf{y}),\ 0 \le j \le M\}$$

which essentially is the intended binary decision rule.

The test can with minor changes to the prior data probabilities be applied to detecting a general geometric pattern[3] and not only the full black one. Nevertheless, it is very expensive computationally, as well as the previously mentioned ones, since for each check we need to evaluate M inner products of size m^2 ; we

[3] We can give equations for squares and triangles parametrized over scale and orientation.

note that such a product evaluation is inherit to the template matching problem. However, since we are dealing with a specific template, the full black one, we will show in chapter 5 that under mild conditions it suffices to use the *sum-of-pixels* statistic instead of the template-vector as the input (the observation) for our test; evidently the correlation in our case reduces to the sum of the values of all the pixels in the image window, since we assign "1" as the value of the noise free black pixel. But note that the equivalence of the correlation and the sum-of-pixels statistic holds only when we have the full black or the full white template.

2.2 Hypothesis Testing Based on the Sum-of-Pixels Statistic

As described earlier the value (intensity) of each pixel in the noise corrupted image is modeled as a random variable $y^j = s^j + n^j$ where s^j takes values in {0,1} and the n^j's are identical, independently distributed (iid) random variables such that : $n^j \sim \mathcal{N}(0, \sigma^2)$. Let us consider now the summation y of all the pixel values in an image window. We will have :

$$y = w + n, \quad where \quad y = \sum_{j \in \mathbf{I}_w} y^j, \quad w = \sum_{j \in \mathbf{I}_w} s^j \ .$$

where $\mathbf{I}_w$ = index set of the pixels of the image test window, w = the *Hamming weight* or simply *weight* of the image window under question, and

$$n = \sum_{j \in \mathbf{I}_w} n^j \sim \mathcal{N}\left(0, \sum_{j \in \mathbf{I}_w} \sigma^2\right) \equiv \mathcal{N}(0, m^2\sigma^2)$$

since n^j's are Gaussian iid random variables. We denote by $w(i)$ the weight of the i'th window pattern, as presented in the previous subsection. This parameter will characterize the states of the M-ary hypothesis test we will construct :

$$H_i \ : \ y \sim N\left(w(i), m^2\sigma^2\right), \ \ i = 0, 1, \cdots, M$$

(M as defined in the previous subsection). In simple words different states correspond to different levels of darkness. This property is sufficient to characterize the all-white pattern, i.e. the one we want to single out and can be obtained as the output of the correlating process with the full black template as described at the end of the last subsection.[4]

The prior probabilities of the states, as well as the cost function, will be identical to the ones found for the M-ary test on the rough data. The decision rule induced will automatically group the states $H_1, H_2, \cdots, H_M$ into one : $\widetilde{H}_1$. So we have to distinguish the two states :

$$\begin{array}{lcl} \widetilde{H}_1 & : & w > 0 \\ \widetilde{H}_0 & : & w = 0 \end{array}$$

The M-ary hypothesis framework [Nar89] provides the rule:

$$d \ : \ d(y) = i \Leftrightarrow g_i(y) \leq g_j(y) \ , \ 0 \leq j \leq M \ ,$$
$$\text{where } g_i(y) = \sum_{\substack{j=0 \\ j \neq i}}^{M} p_j \left[c(j,i) - c(j,j)\right] f_j(y) \ \ .$$

which minimizes the mean cost *E[c(i,j)]*. If we use the cost function

[4] Notice that $w(i)$ will lie in the range $0, 1, \cdots, m^2$; though as i ranges in the set of all possible patterns $w(i)$ will not take all the values in $0, 1, \cdots, m^2$ and also certain duplicates will rise up. So the H_i's will not all be distinct. This is not a problem since we just want to distinguish H_0 from all other states, which is feasible because $w(i)$ exists and has no duplicates. In the subsection A.1 we show how we can take advantage of the nature of the $w(\cdot)$ function in order to reduce the complexity of our computations.

$$\text{l.c.}(\,.\,,.\,) = \begin{array}{c|cccccc|} & 0 & 1 & & \cdots & & M \\ \hline 0 & 0 & 1 & 1 & & \cdots & 1 \\ 1 & 1 & & & & & \\ \vdots & 1 & & & \bigcirc & & \\ & \vdots & & & & & \\ M & 1 & & & & & \end{array}$$

and make a trivial ambiguity waiving assumption we obtain the rule :

$$d \;:\; d(y) = i \;\Leftrightarrow\; g_i(y) \leq g_{1-i}(y),$$

$$g_i(y) = \begin{cases} p \sum_{h=1}^{M} f_h(y), & i = 0 \\ (1 - Mp)\, f_0(y), & i = 1 \end{cases} \quad , f_h(y) \sim \mathcal{N}\left(w(h), m^2\sigma^2\right) \;.$$

For all the sample tests we study in the present work (i.e. for various values of the image and the square size parameters n and m) this rule can numerically be reduced to

$$d \;:\; d(y) = 0 \;\Leftrightarrow\; y \leq y_0$$

for some threshold value y_0 . This threshold determines the tolerance we can have when deciding the matching as a function of the noise level (σ^2). In the figure 2.2 we attempt to give pictorially the intuition of how y_0 is found.

For performance evaluation we need the following :

Probability of false alarm :

$$P_{fa} = Pr\left\{d \neq 0 \mid h = 0\right\} = Pr\left\{y > y_0 \mid H_0\right\} \;\Rightarrow$$

$$P_{fa} = 1 - \Phi\left(\frac{y_0}{m\sigma}\right), \quad where \;\; \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt$$

Probability of detection :

$$P_d = Pr\{d \neq 0 \mid H \neq 0\} = \sum_{i=1}^{M} Pr\{d \neq 0 \mid H = i\}\, Pr\{H = i \mid H \neq 0\} =$$

$$\frac{1}{M}\sum_{i=1}^{M} Pr\{d \neq 0 \mid H = i\} = 1 - \frac{1}{M}\sum_{i=1}^{M} Pr\{d = 0 \mid H = i\} \quad \Rightarrow$$

$$P_d = 1 - \frac{1}{M}\sum_{i=1}^{M} \Phi\left(\frac{y_0 - w(i)}{m\sigma}\right) = 1 - \frac{1}{\sum_{h \in S} p_h} \sum_{h \in S} p_h \Phi\left(\frac{y_0 - w(h)}{m\sigma}\right) .$$

where *h* ranges in the set *S* of all distinct non all-white patterns and p_h are the prior probabilities of finding such a pattern.

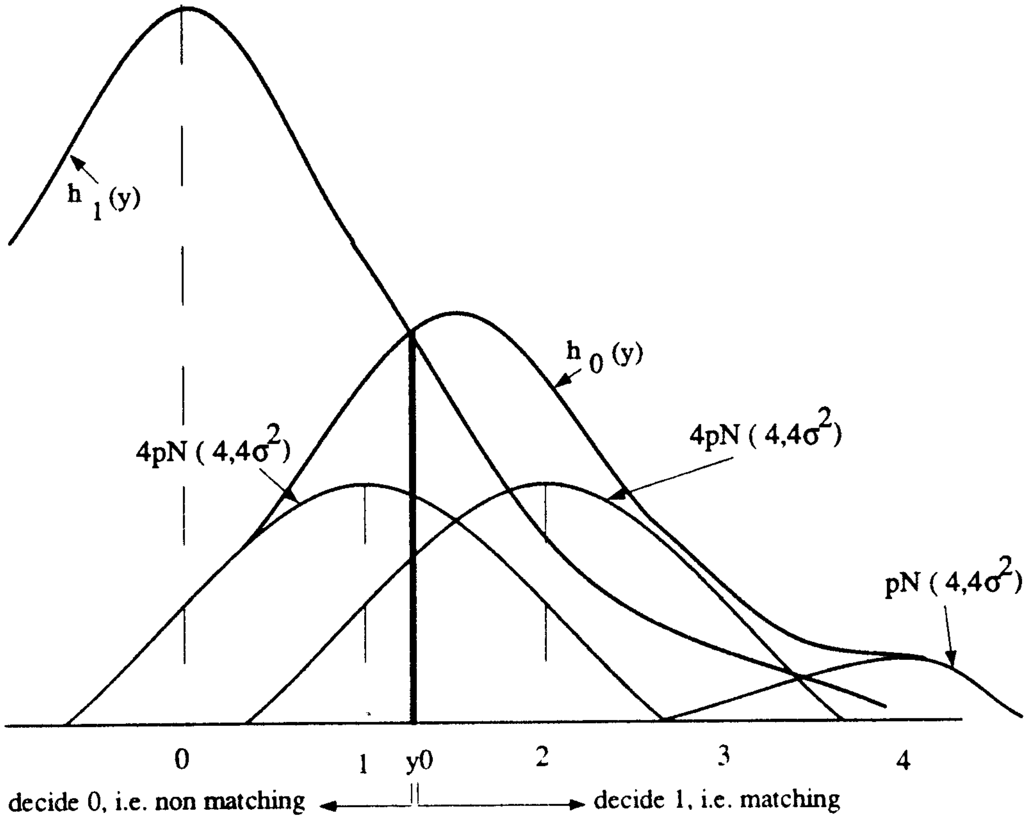


Figure 2.2 The decision regions for a m×m = 2×2 template in the AWGN case

In plot 1 the P_d *vs* P_{fa} curve, parametrized by the noise variance σ^2, for optimal Bayesian tests is shown for both the theoretical analysis and the simulation results. We observe that for little noise (small values of σ^2) P_{fa} is small and P_d is high that happens for small values of the threshold y_0. Adding more noise (letting σ^2 get larger) causes P_{fa} to augment, while P_d gets smaller and y_0 gets larger; y_0 getting larger means that the rule gets conservative i.e. it needs more evidence (darkness) in order to infer "black square detected". If the process of adding noise is continued (σ^2 gets large enough) y_0 gets large; this process gradually leads to the rule "never decide black square detected", i.e. to the point (0,0) of the plot (P_{fa}, P_d). In plot 2 the ROC of the test for m=5, σ^2=0.1 is shown. The simulation results are produced by using one Monte Carlo run and assume that the optimal thresholds are known from the theoretical analysis.

2.3 Testing Under the Effect of BSC Noise

Consider again the problem of finding a black square in a binary image corrupted by noise; but now say that the noise comes as an effect of image transmission over a memoryless binary symmetric channel. The BSC noise, as presented in the previous chapter, will result in a distorted binary image. The use of the sum-of-pixels statistic y is equivalent to counting the black pixels on the image window we scan. A hypothesis testing approach which will give a test with the property of minimizing the probability of an erroneous decision will be described.

Similarly to the AWGN case we construct an M-ary test for which the states

$$H_w,\ w \in S \cup \{0\}$$

are characterized by the Hamming weight of each possible m×m pattern generated by scanning the noise free image ; specifically S is the set of non all-white such patterns. The prior probabilities of these states are denoted as p_h and they are equal to the ones computed in the previous paragraph. The cost function which implicitly transforms the M-ary hypothesis into a binary one again is :

$$\text{l.c.}(\,.\,,.\,) = \begin{array}{c|ccccc|} & 0 & 1 & & \cdots & M \\ \hline 0 & 0 & 1 & 1 & \cdots & 1 \\ 1 & 1 & & & & \\ \vdots & 1 & & \bigcirc & & \\ & \vdots & & & & \\ M & 1 & & & & \end{array}$$

The two hypothesis for the binary test are :

$$\tilde{H}_0 \ : \ h = 0$$

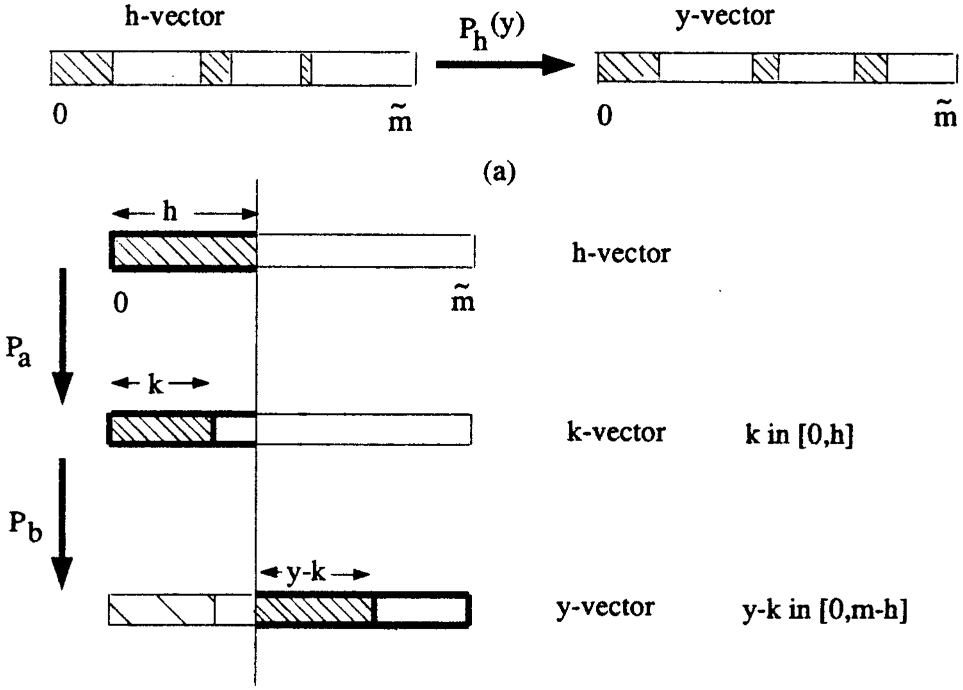
$$\tilde{H}_1 \ : \ h \geq 1$$

The decision rule will look identical to the one derived for the AWGN case :

$$d \ : \ d(y) = i \ \Leftrightarrow \ g_i(y) \leq g_{1-i}(y) \, .$$

$$g_i(y) = \begin{cases} \sum\limits_{h \in S} p_h P_h(y) \, . & i = 0 \\ p_0 P_0(y) \, . & i = 1 \end{cases}$$

where $P_h(y), h \in S \cup \{0\}$, correspond to the $f_h(y)$ for the AWGN case, and are distributions which we still are missing in order to be able to evaluate our test. $P_h(y)$ is the probability of observing an image window with y black pixels in it, while the corresponding noise free window had h ones. As already discussed in chapter 1 we call these discretized probabilities *transition probabilities*. We will show now how to evaluate these probabilities.



(a)

(b)

Figure 2.3 The transition probabilities for the BSC noise case

The image window is represented as a binary vector of length $\tilde{m} = m^2$. The shaded region in the figure 2.3a represents 1's and the blanc region represents 0's. We want to find the probability of getting a configuration with y 1's if our vector with h 1's passes through a BSC with bit inversion probability equal to ϵ. Since we are interested only in the "area" of common and non-common shaded places between the h— and y—vectors and not the specific positions of black and white pixels we will assume that all the shaded area in the h—vector is stacked on the left.

The key idea is as follows : The probability of getting a y—vector from an h—vector is equal to the common probability of getting a k—vector from the h—vector, where $k < h, y$ is the number of common shaded places between the h— and y—vectors and getting a y—vector from the k—vector, where $y - k$ shaded places are not common with the h—vector. The 2–step process is depicted on the figure 2.3b

Since the two events *a* and *b* (see fig. 2.4.b) are related with uncommon areas (different bits) and the binary noise is white they are independent, i.e.

$$P_{ab}(k) = P_a(k) \cdot P_b(k)$$

We have :

$$P_a(k) = \binom{h}{k} (1-\epsilon)^k \epsilon^{h-k}.$$
$$P_b(k) = \binom{\tilde{m} - h}{y - k} \epsilon^{y-k} (1-\epsilon)^{\tilde{m}-h-y+k}$$

with the constraints :

$$\left.\begin{array}{c} 0 \le k \le h \\ 0 \le y - k \le \tilde{m} - h \Rightarrow h - m \le k - y \le 0 \Rightarrow h + y - \tilde{m} \le k \le y \end{array}\right\} \Rightarrow$$
$$k \in [\max(0, h + y - \tilde{m}), \min(h, y)] \equiv [k_{\min}, k_{\max}]$$

So

$$P_h(y) = \sum_{k=k_{\min}}^{k_{\max}} P_a(k) P_b(k) \Rightarrow$$
$$P_h(y) = \sum_{k=\max(0,h+y-\tilde{m})}^{\min(h,y)} \binom{h}{y} \binom{\tilde{m} - h}{y - k} (1-\epsilon)^{\tilde{m}-h-y+2k} \epsilon^{h+y-2k} \quad .$$

If we substitute this expression of $P_h(y)$ in the decision rule then this becomes:

$$d \ : \ d(y) = 0 \Leftrightarrow \frac{1}{p_0} \sum_{h \in S} p_h \sum_{k=k_{\min}}^{k_{\max}} \binom{h}{k} \binom{\tilde{m} - h}{y - k} \left(\frac{\epsilon}{1 - \epsilon} \right)^{h-2k} < \binom{\tilde{m}}{y}$$

and by a further numerical simplification :

$$d \ : \ d(y) = 0 \Leftrightarrow y \leq y_0 \ .$$

where y_0 is some threshold in the range from 0 to m.

For the performance evaluation of the test we need:

Probability of false alarm :

$$P_{fa} = Pr\{d \neq 0 \mid H = 0\} = Pr\{y > y_0 \mid H_0\} \Rightarrow$$

$$P_{fa} = \sum_{y=y_0+1}^{\tilde{m}} \binom{\tilde{m}}{y} (1 - \epsilon)^{\tilde{m}-y} \epsilon^{y}$$

Probability of detection :

$$P_d = Pr\{d \neq 0 \mid H \neq 0\} = \sum_{h \in S} Pr\{d \neq 0 \mid H = h\} \, Pr\{H = h \mid H \neq 0\} =$$

$$1 - \sum_{h \in S} Pr\{y \leq y_0 \mid H = h\} \frac{p_h}{\sum_{h \in S} p_h} \quad \Rightarrow$$

$$P_d = 1 - \frac{1}{\sum_{h \in S} p_h} \sum_{h \in S} p_h \sum_{y=0}^{y_0} P_h(y)$$

The P_d vs P_{fa} curve for optimal Bayesian rules parametrized by the pixel inversion probability ϵ is given in plot 3 for the case of m=8. The shape of this curve is similar to the one of plot 1, in the sense that for little noise we have small P_{fa} and high P_d, while if we add noise we gradually move towards the point (P_{fa}=0, P_d=0). The optimal Bayesian rules range from "decide black square detected if sum-of-pixels > 5" for ϵ=0.02 to "decide black square detected if sum-of-pixels > 37" for ϵ=0.4 and effectively "never decide black square detected" (i.e. reach the point (0,0) of the curve (P_{fa},P_d)) for even larger amounts of noise. The stairs-like shape of the curve is due to the discrete nature of the threshold and makes apparent the trade-off between the high P_d and low P_{fa} that the optimal Bayesian rule attempts to compensate. In plot 4 we see the ROC of the test produced as a result of the above analysis for the case of m=8 and ϵ=0.1. The ROC found by simulating is also given; again one Monte Carlo run is used, and optimal threshold values are assumed to be known from the numerical analysis.

2.4 Testing Based on Image Data of a Coarsened Resolution

Consider a binary image corrupted by BSC noise with parameter ϵ. We can code each 2×2–pixel block of it with one bit indicating all-black or all-white block, depending on the bit value. The noise of the original image will "propagate" to the coded data, resulting in bit inversions. The white noise in the original data will cause the noise to be white in the coded data too.

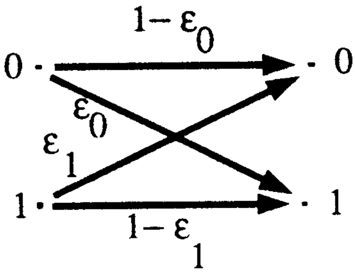


Figure 2.4 The Binary Channel transition diagram

The relation of the noise-free-coded data and the noise-corrupted-and-then-coded data is depicted in the figure 2.4. The inversion probabilities ϵ_0 and ϵ_1 depend on the specific coding rule. If the coding (quantization) rule is the one given in table 1.2 (subsection 1.3), then because of the symmetry of the rule we will have $\epsilon_0=\epsilon_1=\acute{\epsilon}$. So the noise on the coded data can be modeled again as BSC noise with inversion probability $\acute{\epsilon}$, which depends on the noise parameter ϵ of the original data as follows :

$$\acute{\epsilon} = Pr\left\{have\ more\ than\ 2\ bit\ inversions\right\} + \frac{1}{2}Pr\left\{have\ 2\ bit\ inversions\right\}$$
$$= \sum_{i=3}^{4} \binom{4}{i} \epsilon^i (1-\epsilon)^{4-i} + \frac{1}{2}\binom{4}{2}\epsilon^2(1-\epsilon)^2 \quad .$$

An analysis for the detection of the black square based on these coded data is identical to the one we made in the last subsection. The performance evaluation curves for this test are given in plots 5 and 6. We observe that they have the same shape as the plots 3 and 4 respectively which were drawn for the uncoded data. However note that the ROC curve for ϵ=0.1 is closer to the point (P_{fa}=0, P_d=1); this means that tests acting on data of coarsened resolution are more reliable (for this specific object target and amount of noise) than these acting on the original pixel data. Similar observations are further discussed in the subsection 5.1.

2.5 Summary

In this chapter we described a simple instance of the template matching problem using noise corrupted data. Initially we considered Gaussian noise and tried to find some optimal matching test. We briefly examined three tests acting on the rough data, starting from the most straightforward approach of the composite binary hypothesis test, continuing with the generalized likelihood ratio list test and reaching the most structured M-ary hypothesis test.

We noticed that for the case of the full black template we can avoid the computational load of inner products, or equivalently of convolutions, and introduced the sum-of-pixels statistic. We developed and evaluated a binary hypothesis test "residing" on an M-ary test, which was using the sum-of-pixels statistic as its observation. This test designed for the AWGN case was properly modified for the case of the BSC noise and further modification to the latter gave a test based on coarser resolution image data. Sample simulation results were given for both tests.

An attempt was made to show the steps followed which led to the specific, described formulation of the BSC noise case of the problem. This formulation will be expanded in the fourth chapter in order to implement a feature extraction algorithm applicable to either multiple-gray level images or to block coded image data.

Chapter 3
Image Coding

3.1 Block Coding Techniques

A block coding technique transforms the n×n-pixel valued matrix representing an image into a $c_n \times c_n$–block code valued matrix ($c_n = n/$”block size”), representing the encoded version of the original image; each block is encoded separately from all the others; we consider multiple level gray images. The coding process may be analyzed into three steps, namely preprocessing, quantization and the coding itself, as shown in the table 3.1 for three different block coding schemes.

scheme	input	preprocessing	quantization	source coding	output
1	pixel image (nxn-pixel valued matrix)	whitening (KLT, DCT, Hadamard basis)	scalar quant. (Max-Loyd qu.)	Huffman coding or enumeration	block image (C_n x C_n-block code valued matrix)
2		normalization	vector quant.		
3		–	modified Hadamard basis	enumeration	

Table 3.1 Block quantization / coding schemes applied to image data

For the schemes 1 and 2 the pixel values in a block form a vector of random variables; the randomness is due to the image statistics and assumed to be sufficiently described by the first and second order histograms. In [HuSc63] the block-vector, composed of 4 Markovian random variables, is whitened by using the KLT (see paragraph 1.4, also [NeLi80],[San89]) and then each component is separately quantized by the Max-Loyd quantizer [Max60]. It turns out that 8 bits are adequate for representing a quantized vector. In [LaS171] the 4×4–pixel blocks are coded by the Hadamard basis (see par. 1.4). It is shown that 32 bits are adequate for representing a quantized block. Observe that both techniques require a rate of 2 bpp (bits per pixel).

Vector Quantization (VQ) [NaKi88] can give better compression ratios for a given performance than the scalar quantization. This is an information theoretic result broadly used in several applications in the last few years. In [MAY82] VQ is used for the image block-vectors while their components are normalized over their variances. We should note that the VQ procedure has some underlying distortion measure, applied on pairs of quantized vectors; the selection of this distortion measure is application dependent [MRG85] and determines the VQ performance to a great extent.

Both of the above schemes provide us with a block alphabet of quantized vectors to which a code must be assigned. This code may amount to simply enumerating the codewords or using a Huffman code, so that the information theoretic properties of the alphabet may be exploited. However as pointed out in chapter 1 we still require the same number of bits for coding each quantized block-vector and therefore we should use simple enumeration in our application.

The scheme 3 of the table 3.1 is described in subsection 3.3 while the reasoning that led us to admit it is given in the following paragraph.

3.2 Constraints on the Block Alphabet Size Due to the Matching Test

Let l be the number of bits assigned to each codeword (block) of the image block-alphabet. The size of the alphabet will be $M = 2^l$. We usually have

$$2^l << g^s = \# \; of \; possible \; non \; quantized \; blocks,$$

where g is the number of gray levels in the original image and s is the size of the blocks in pixels. So l parametrizes the quantization level of the original image and therefore affects the quality of the image.

Suppose now that the original image has been corrupted by a noise characterized by some parameter, e.g. ϵ or σ^2 as we have seen in the previous chapters. Consequently we erroneously may take a codeword i to be some other, say j. We remind the reader that we call these probabilities ϵ_{ij} *transition probabilities.* Evidently the noise parameter (ϵ) affects the quality of the reconstructed image as well.

As already discussed we intend to apply some feature extraction algorithm (template matching) onto the coded data; we will obtain a performance evaluation measure (P_{fa}, P_d) which will be interpreted as the image quality index; thereafter we will exploit the effects of the two noise sources (having the parameters l and ϵ)

$$\left. \begin{array}{lll} quantization & : & l \rightarrow M \\ external \; noise & : & \epsilon \rightarrow \epsilon_{ij}'s \end{array} \right\} \rightarrow (P_{fa}, P_d)$$

onto the (reconstructed) image quality. We will do so by formulating the template matching as a hypothesis testing problem. In the next chapter it is shown that if the template is of size *n* blocks, then the complexity of the test will be parametrized over the template codebook size

$$S(M,n) = \binom{M+n-1}{n}.$$

We observe that :

1. The coding schemes presented in the previous paragraph will give rise to a very large *S(M,n)* parameter to allow a test to be implemented.[5]
2. We cannot afford *l*>4 (i.e. M>16) and n>9, since $S(16.9) \approx 1.3 \times 10^6$.

Note that in the numerical analysis implemented on a Sun work station the values of the parameters are *l*=4, n=4; so S(M,n)=3876. If we want to have a reasonable size for our template, e.g. 12×12–pixel or equivalently 3×3–block of 4×4–pixel blocks we effectively need to restrict ourselves to the case of the binary image.

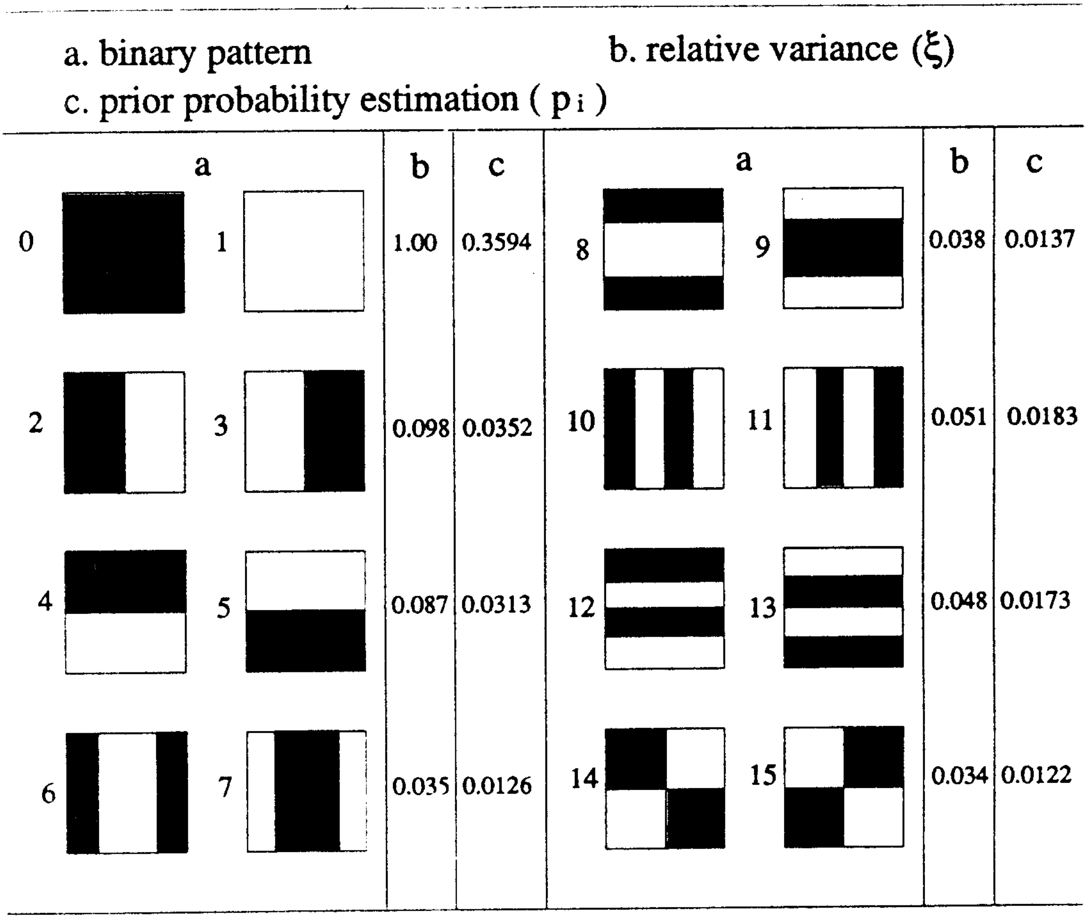
A quite different approach for solving the "image evaluation" problem as it has so far been formulated, will be mentioned, even though it will not be examined in the present work. This comes from the VQ technique using the Itakura-Saito measure [MRG85]; the performance evaluation of the matching test can then be expressed through this distortion measure and thus help us avoid the formulation of the hypothesis testing problem.

[5] Nevertheless it is worthwhile to:

a. Try to adapt the coding scheme of [TaFa86] to our problem since it furnishes rates up to 0.25 bpp and it is robust to the noise.
b. Exploit the fact that most [illegible]'s equal 0 and thus the complexity may be reduced from the order of the codebook size to the size of a "small" hypercube around the point of interest.

3.3 A modified Hadamard Basis

The available 4–bit codeword for a block can represent $2^4 = 16$ different blocks. We assign all bits to one variable which is intended to map the 16 "most possible" 4×4 blocks.



a. binary pattern b. relative variance (ξ)
c. prior probability estimation (p_i)

a	b	c	a	b	c
0 1	1.00	0.3594	8 9	0.038	0.0137
2 3	0.098	0.0352	10 11	0.051	0.0183
4 5	0.087	0.0313	12 13	0.048	0.0173
6 7	0.035	0.0126	14 15	0.034	0.0122

Figure 3.1 The modified 4×4 Hadamard basis

How to determine the 16 patterns : In [LaSl71] the Hadamard basis for 4×4–pixel blocks is given (see also fig. 1.5), along with a measure of variance ξ, of the random variable corresponding to each element i. We will use this measure for making an estimation of the prior probabilities of having these blocks. The Hadamard basis is used for multiple gray level images; negative intensity factors can reverse the pattern, e.g. a black block may be converted into a white one. In our application we do not use such a factor; instead we use the first 8 patterns of the Hadamard basis plus their reversed versions. We will take the prior probabilities of them to be

$$p_{2i} = p_{2i+1} = \frac{\xi_i/2}{\sum\limits_{j=0}^{7} \xi_j} \ , i = 0, 1, \cdots, 7 \ .$$

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1		0	8	8	8	8	8	8	8	8	8	8	8	8	8	8
2			0	16	8	8	8	8	8	8	8	8	8	8	8	8
3				0	8	8	8	8	8	8	8	8	8	8	8	8
4					0	16	8	8	8	8	8	8	8	8	8	8
5						0	8	8	8	8	8	8	8	8	8	8
6							0	16	8	8	8	8	8	8	8	8
7								0	8	8	8	8	8	8	8	8
8									0	16	8	8	8	8	8	8
9										0	8	8	8	8	8	8
10					symmetric						0	16	8	8	8	8
11					elements							0	8	8	8	8
12													0	16	8	8
13														0	8	8
14															0	16
15																0

Table 3.2 The Hamming distances among the elements of the modified Hadamard basis

The 16 patterns we will use and their prior probabilities are shown in the figure 3.1. The Hamming distances between pairs of block patterns are shown on the table 3.2

Transitions are caused by the white binary noise corruption. Nevertheless they are governed by the quantization procedure, i.e. the rule used to map the noise corrupted data back to the limited alphabet of the 16 blocks. These transitions are discussed in the next paragraph.

3.4 The Quantization/Coding Procedure

We will start by making some observations : First the Hamming distance d(i,j) between two different codewords i,j is at least 8 (see table 3.2); therefore we may think of the codewords as the centers of spheres in the $\{0,1\}^{16}$ –space with radius half the Hamming distance ρ=4. Secondly, not all codewords comming up from the Hadamard basis are used; actually exactly half of them are considered in our coding scheme.

The quantization rule :

For each 4×4–pixel block (call it y) of the noise corrupted image do:

1. if d(y,i) < 4 for some codeword i then decode $y \rightarrow i$
2. if d(y,i) > 4 for all i
   a. if "weight of y" (w(y)) > 8 decode $y \rightarrow 0$ (full black block)
   b. if w(y) < 8 decode $y \rightarrow 1$ (full white block)

c. if w(y) = 8,

decode y→0 with probability 0.5,

decode y→1 with probability 0.5

3. if d(y,i) = 4 for one or more i's then

decode y→j_m, where $j_m = \arg\max_j \{prior(j)\}$.

What are the transition probabilities :

We assume that the original scene of the image can accurately be represented by the block alphabet we have, i.e. the image can be decomposed into blocks each of which represents an alphabet element, say i. The image is corrupted by binary noise with parameter ϵ ; so the element i will be translated into a 4×4 binary block (call it y) which may not be an alphabet element. Nevertheless, the quantization rule given above will decode it into some alphabet element, say j, which may be different from i. Once more we repeat that we call *transition probability* Pr{i→j} = P(j / i) = Pr{d=j / H=i} the probability of deciding that the element (codeword) j appears in some place of the image, given that the element (codeword) i was at that place of the original image. We have to calculate the probabilities $P(i \mid j)$, $i, j = 0, 1, \cdots, 15$.

We have :

$$Pr\{d = j \mid H = i\} = \sum_{y \in R_j} Pr\{observe \;\; y \mid H = i\} \quad (1)$$

where R_j is the decision region for the codeword j; note that

$$Pr\{Y = y \mid H = i\} = \epsilon^{d(y,i)}(1-\epsilon)^{16-d(y,i)}$$

The decision regions, according to the quantization rule are :

$$R_0 = \{y : (d(y,0) \leq 4) \vee cond0(y)\} \qquad (2)$$

$$cond0(y) = (\ ((w(y) > 8) \vee (w(y) = 8 w.p.1/2)) \wedge (d(y,i) > 4, i = 0,\cdots,15)\ )$$

$$R_1 = \{y : (d(y,1) \leq 4) \vee cond1(y)\} \qquad (3)$$

$$cond1(y) = (\ ((w(y) < 8) \vee (w(y) = 8 w.p.1/2)) \wedge (d(y,i) > 4, i = 0,\cdots,15)\ )$$

$$R_i = \left\{ y : (d(y,i) \leq 3) \vee (d(y,i) = 4) \wedge (i = \arg\max_{j \in I(y)} prior(j)) \right\} \qquad (4)$$

$where \quad i = 2,3,\cdots,15 \qquad and \qquad I(y) = \{j : d(y,j) = 4\}$

For j = 0 :

Since $d(y,0) \leq 4 \Leftrightarrow w(y) \geq 12$ **we get :**

$$(1),(2) \Rightarrow Pr(0 \mid i) = \sum_{y:w(y)\geq 12} Pr\{y \mid i\} + \sum_{\substack{y:d(y,j)>4, j=0,\cdots,15 \\ w(y)>8 \ or \ w(y)=8w.p.1/2}} Pr\{y \mid i\}$$

$$= Pr\{y : w(y) \geq 12 \mid i\} \ + \ Pr\{y : d(y,j) > 4, \ j = 0,\cdots,15 \mid i\} \times$$

$$\times \left[Pr\{w(y) > 8 \mid i\} + \frac{1}{2} Pr\{w(y) = 8 \mid i\} \right] \qquad (5)$$

For the first term of (5) we have :

$$Pr\{y : w(y) \geq 12 \mid i\} = Pr\{w(i) \rightarrow w, w = 12,\cdots,16\} = \sum_{j=12}^{16} P_{w(i)}(j)$$

and

$$P_{w(i)}(j) = Pr\{get \ \ a \ \ 16 - bit \ length \ word \ \ with \ j \ 1's$$

$$out \ \ of \ a \ same \ \ length \ \ word \ \ with \ \ w(i) \ \ 1's\}$$

$$= \sum_{k=\max\{0,w(i)+j-16\}}^{\min\{w(i),j\}} \binom{w(i)}{k} \binom{16-w(i)}{j-k} (1-\epsilon)^{16-j-w(i)+2k} \epsilon^{j+w(i)-2k},$$

$$j = 12,\cdots,16$$

If i = 2,3,...,15 then :

$$P_{w(i)}(j) = P_8(j) = \sum_{k=j-8}^{8} \binom{8}{k} \binom{8}{j-k} (1-\epsilon)^{8-j+2k} \epsilon^{8+j-2k}$$

$$= [(1-\epsilon)\epsilon]^8 \left(\frac{\epsilon}{1-\epsilon}\right)^j \sum_{k=j-8}^{8} \binom{8}{k} \binom{8}{j-k} \left(\frac{\epsilon}{1-\epsilon}\right)^{-2k}$$

If i = 0 then :

$$P_{w(i)}(j) = P_{16}(j) = \sum_{k=j}^{j} \binom{16}{k} \binom{0}{j-k} (1-\epsilon)^{2k-j} \epsilon^{16-2k+j}$$
$$= \binom{16}{j} (1-\epsilon)^{j} \epsilon^{16-j}$$

If i = 1 then :

$$P_{w(i)}(j) = P_{0}(j) = \sum_{k=0}^{0} \binom{0}{k} \binom{16}{j-k} (1-\epsilon)^{16-j+2k} \epsilon^{j-2k}$$
$$= \binom{16}{j} (1-\epsilon)^{16-j} \epsilon^{j}$$

Now we will evaluate the second term of (5). Let's return to the spheres concept. The 16–element original Hadamard basis along with the "complements" of these elements may be thought as the centers of spheres which constitute a 32–partition of the $\{0, 1\}^{16}$ space in which y lies. If i is the element we originally have and if due to the noise corruption it is translated into a y such that d(i,y) > 4, y will lie with the same probability in any of the rest 32–2 = 30 regions of the partition. Note that we exclude the complementary block which is unlikely to occur. If y lies in one of the 16–2 = 14 left regions with center one of the alphabet elements we use, then we decode y as being this element. If not we decode it into "0" or "1" depending on its weight w(y). Therefore we have :

$$Pr\{y : d(y, j) > 4,\ j = 0, \cdots, 15 \mid i\} =$$

$$Pr\{y : d(y, i) > 4\}\ Pr\{y : d(y, j) > 4, j = 0, \cdots, 15 \mid d(y, i) > 4, i\} =$$
$$= \left(\sum_{d=5}^{16} \binom{16}{d} \epsilon^{d} (1-\epsilon)^{16-d} \right) \frac{14}{30}$$

Also

$$Pr\,\{w(y) > 8 \mid 0\} = \sum_{j=9}^{16} \binom{16}{j} \epsilon^{16-j} (1-\epsilon)^{j}$$

$$Pr\,\{w(y) = 8 \mid 0\} = \binom{16}{8} \epsilon^{8} (1-\epsilon)^{8} = Pr\,\{w(y) = 8 \mid 1\}$$

$$Pr\,\{w(y) > 8 \mid 1\} = \sum_{j=9}^{16} \binom{16}{j} \epsilon^{j} (1-\epsilon)^{16-j}$$

and by symmetry

$$Pr\,\{w(y) > 8 \mid i\} + \frac{1}{2} Pr\,\{w(y) = 8 \mid i\} = \frac{1}{2}. \quad i = 2, \cdots, 15$$

By putting them all together now we get :

$$P(0 \mid 0) = \sum_{j=12}^{16} \binom{16}{j} (1-\epsilon)^{j} \epsilon^{16-j} +$$

$$\frac{14}{30} \left[\frac{1}{2} \binom{16}{8} (1-\epsilon)^{8} \epsilon^{8} + \sum_{j=9}^{16} \binom{16}{j} \epsilon^{16-j} (1-\epsilon)^{j} \right] \times \sum_{d=5}^{16} \binom{16}{d} \epsilon^{d} (1-\epsilon)^{16-d} \quad (6)$$

$$P(0 \mid 1) = \sum_{j=12}^{16} \binom{16}{j} (1-\epsilon)^{16-j} \epsilon^{j} +$$

$$\frac{14}{30} \left[\frac{1}{2} \binom{16}{8} (1-\epsilon)^{8} \epsilon^{8} + \sum_{j=9}^{16} \binom{16}{j} \epsilon^{j} (1-\epsilon)^{16-j} \right] \times \sum_{d=5}^{16} \binom{16}{d} \epsilon^{d} (1-\epsilon)^{16-d} \quad (7)$$

$$P(0 \mid i) = [(1-\epsilon)\epsilon]^{8} \sum_{j=12}^{16} \left(\frac{\epsilon}{1-\epsilon} \right)^{j} \sum_{k=j-8}^{8} \binom{8}{k} \binom{8}{j-k} \left(\frac{\epsilon}{1-\epsilon} \right)^{-2k} +$$

$$\frac{14}{30} \times \frac{1}{2} \sum_{d=5}^{16} \binom{16}{d} \epsilon^{d} (1-\epsilon)^{16-d}. \qquad i = 2, 3, \cdots, 15 \qquad (8)$$

For j = 1 :

Because of symmetry we have :

$$P(1 \mid 1) = P(0 \mid 0)$$

$$P(1 \mid 0) = P(0 \mid 1)$$

$$P(1 \mid i) = P(0 \mid i). \quad i = 2, 3, \cdots, 15$$

For j = 2,3, ..., 15 :

$$
\begin{aligned}
P(j \mid i) = & \sum_{y:d(y,i)\leq 3} P\{y \mid i\} + \sum_{\substack{y:d(y,i)=4, \\ i=\arg\max prior(j), j\in I(y)}} P\{y \mid i\} \\
& \leq \sum_{y:d(y,i)\leq 4} P\{y \mid i\} \quad = \quad Pr\{y : d(y,j) \leq 4 \mid i\} \ . \\
where \quad & I(y) = \{j : d(y,j) = 4\}
\end{aligned}
$$

The bound above is adequate for our needs since the transition probabilities are treated as error probabilities. Therefore from now on we will approximate the transition probability with its corresponding bound. We expect that this approximation will lead to slightly more pessimistic results than the actual ones we would obtain via simulations.

Let's now call $\bar{i}$ the "complementary element" of i. Then :

$$
\begin{aligned}
& Pr\{j, j \neq i \mid i\} = Pr\{y : d(y,j) \leq 4 \mid i, d(y,i) > 4\} \times Pr\{d(y,i) > 4\} = \\
& Pr\{y \in in\ 30\ equaly\ likely\ regions\} \times Pr\{d(y,i) > 4\} \Rightarrow \\
& Pr(j \mid i) = \frac{1}{30} \sum_{k=5}^{16} \binom{16}{k} \epsilon^k (1-\epsilon)^{16-k} \ , \quad i,j = 2,\cdots,15, \quad j \neq i, j \neq \bar{i} \qquad (9)
\end{aligned}
$$

Finally we have :

$$
\begin{aligned}
& Pr(\bar{i} \mid i) = Pr\{y : d(y,i) > 12\} \Rightarrow \\
& Pr(\bar{i} \mid i) = \sum_{k=12}^{16} \binom{16}{k} \epsilon^k (1-\epsilon)^{16-k} \ , \quad i = 2,\cdots,15 \qquad (10) \\
& also \\
& Pr(i \mid i) = \sum_{k=0}^{4} \binom{16}{k} \epsilon^k (1-\epsilon)^{16-k} \ , \quad i = 2,\cdots,15 \qquad (11)
\end{aligned}
$$

A summary of the transition probabilities is given in the table 3.3.

i	j	$P(j\|i)$ given by formula
0	0	(6)
1	1	
0	1	(7)
1	0	
2,...,15	0,1	(8)
2,...,15	2,...,15 , $j \neq i, \tilde{i}$	(9)
2,...,15	i	(11)
2,...,15	$\tilde{i}$	(10)

Table 3.3 Block transition probabilities summary

3.5 Summary

In this chapter we first saw how a matrix representing a multiple gray level pixel image can be converted into one representing a block coded image, so that the image can efficiently be stored and transmitted. We recalled that template matching can be considered as a quality measure for the reconstructed image. Based on a result presented in the next chapter we showed that in order to be able to implement such a matching test acting on coded image data we need to compress the image more than the storage and transmission schemes allow; this effectively leads to some restricted class of images (binary images representing objects which can be described by the modified Hadamard basis). We developed a coding scheme meeting our needs and calculated its statistical properties (block priors and transitions) in terms of the noise characteristics.

Chapter 4
The Histogram Matching Problem

4.1 The Template Histogram

In the previous chapter we introduced a block coding scheme transforming a binary image into a matrix of codewords in the range 0, ..., 15; we also computed the transition probabilities among the blocks in terms of the probability of bit inversion in the original image which also was the cause of these transitions. In this way a 12×12–pixel image window is transformed into a 3×3–code sub-matrix, since 4×4–pixel blocks were considered. The 9 elements of this sub-matrix are independent random variables, because the BSC noise was supposed to be white (in space) and their values define a configuration which will be called *the original state*. If we now group together all the original states having the same (1st order) histogram (i.e. all sub-matrices having the same multitude of each element) then we come up with a new set of states. We will call this new set *the histogram alphabet*. Evidently the size of this set is much smaller than the one of the original set; the size of the histogram alphabet will be discussed in the next chapter.

Let M be the number of all possible distinct components (block codes) which may constitute the template. Let also n be the length of the template in blocks (in the example above we had $n = 9 = 3\times3$ blocks). We will denote by $S(M,n)$ the size of the histogram alphabet. Call $\mathbf{h}$ the M-vector representing the histogram of the noise-free coded template of size n; call also y the M-vector representing the histogram of a noise corrupted and afterwards coded image window of size n. The vector **h** represents some feature we want to detect on the image; our problem amounts to comparing the two histograms **h** and y and infer a decision about their matching.

We will formulate this matching as an M-ary hypothesis testing problem which will efficiently lead to an optimal binary decision rule. To do so we will assume we are given the prior probabilities and the transition probabilities of the histogram elements (i.e. the block priors p_i, $i = 0, 1, \cdots, M-1$ and block transitions ϵ_{ij}, $i, j = 0, 1, \cdots, M-1$) so that we will be able to find the *histogram priors* and *histogram transitions*.

Observe that if we set $M = 2$, i.e. if we have a binary code, then the histogram reduces to the "sum-of-pixels" statistic we discussed in chapter 2. In the subsection 4.2 we develop a rule which is a generalization of the one developed in chapter 2. This rule can be applied to compare arbitrary histograms, provided we have the information about the histogram elements mentioned above.

4.2 The Histogram Matching as an M-ary Hypothesis Test

4.2.1 The Histogram Alphabet

Let $\{a_0, a_1, \cdots, a_{M-1}\}$ be the block alphabet. Evidently we will have M^n possible n-tuples of blocks. We introduce an equivalence relation onto this set of n-tuples. An equivalence class will be composed of all n-tuples having the same histograms, i.e. two n-tuples are in the same class if one is a permutation of the elements of the other. A representative element of an equivalence class will be called *a histogram pattern.*

Consider now the following power expansion [Spi68] :

$$(\alpha_0 + \alpha_1 + \cdots + \alpha_{M-1})^n = \sum_{\{n_i\}:\sum_i n_i = n} \frac{n!}{n_0! n_1! \cdots n_{M-1}!} \alpha_0^{n_0} \alpha_1^{n_1} \cdots \alpha_{M-1}^{n_{M-1}} \cdot$$

Observe that :

1. Each term in the summation above can be considered as a histogram pattern, i.e.

   $$\alpha_0^{n_0} \alpha_1^{n_1} \cdots \alpha_{M-1}^{n_{M-1}}$$

   represents an *n*-tuple in which we have n_0 times the element α_0, n_1 times the element α_1 and so on; note that if $n_i = 0$ for some i, then α_i is not in the n-tuple.

2. The size of each equivalence class will be equal to

   $$\frac{n!}{n_0! n_1! \cdots n_{M-1}!}$$

3. If p_i is the prior probability for $\alpha_i, i = 0, 1, \cdots, M-1$, then the independence assumption implies that the prior probability for a specific element in the equivalence class will be

   $$p_0^{n_0} p_1^{n_1} \cdots p_{M-1}^{n_{M-1}} \cdot$$

   while the prior for the class itself. i.e. the prior of the histogram pattern will be

   $$\frac{n!}{n_0! n_1! \cdots n_{M-1}!} p_0^{n_0} p_1^{n_1} \cdots p_{M-1}^{n_{M-1}} \cdot$$

4. We can systematically construct the histogram patterns with the following procedure :

   a. Find all integer partitions of the number n in at most M places (see subsection A.2)

   $$\{n_0, n_1, \cdots, n_{M-1}\} \ .$$

   b. For each such partition find all the distinct permutations of n_i's in M places. Note that we have

   $$\binom{M}{n_0}\binom{M-n_0}{n_1}\cdots\binom{n_{M-1}}{n_{M-1}}$$

   of them.

   Each permutation will determine a set of exponents in the expansion formula and therefore a histogram pattern.

5. The number of histogram patterns, i.e. the size of the histogram alphabet, *S(M,n)* will be equal to the number of summands and therefore

   $$S(M,n) = \binom{M+n-1}{n} \ .$$

   Note that in the case of the binary block alphabet (e.g. black/white) we get *S(2,n)* = *n+1* = the number of possible variations of black-white mixtures in a pattern of size n.

In the table 4.1 we see the integer partitions of the number $n = 9$, the number of classes generated and the size of each class (number of original states grouped together) for the special case of $M = 16$.

The equivalence classes described (histogram patterns) will determine the states for the M-ary test we will develop; the state space will be denoted by H.

integer partition	# of classes generated	size of class	integer partition	# of classes generated	size of class
54	240	126	33111	43680	20160
531	3360	504	3111111	80080	60480
522	1680	756	222111	160160	45360
5211	21840	1512	2211111	240240	90720
51111	21840	3024	21111111	102960	181440
441	1680	630	111111111	11440	362880
4311	21840	2520	22221	21840	22680
4221	21840	3780	63	240	84
42111	21840	7560	621	3360	252
411111	48048	15120	6111	7280	504
432	3360	1260	72	240	36
3312	21840	50 40	711	1680	72
3222	1680	7560	81	240	9
32211	131040	15120	9	16	1
321111	240240	30240			

Table 4.1 The histogram pattern classes

4.2.2 The Observation and the Cost Function

The observation data in our test correspond to some n-vector of block codewords. This information can equivalently be represented by an M-vector y being the histogram pattern of the image window we scan. For example if n = 4 and M = 16 we may have the observation *[1 5 15 1]* which is equivalent to the histogram pattern $\mathbf{y} = [0\ 2\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$.

We introduce a partition H_0, H_1 of the state space H. Given an observation y we want to decide "matching" (H_1) or "not matching" (H_0). We will do so by imposing the cost function :

$$c(\mathbf{h},\mathbf{y}) = \begin{cases} 0, & if\ both\ \mathbf{h},\mathbf{y}\ belong\ to\ either\ H_0\ or\ H_1 \\ 1, & if\ \mathbf{h}.\mathbf{y}\ do\ not\ belong\ to\ the\ same\ set\ H_q,\ q = 0,1 \end{cases}$$

Note that H_1 is not necessarily a singleton. This means we may consider matching with multiple histogram patterns simultaneously.

4.2.3 The Transitions Among the Histogram Patterns

A block alphabet of size M = 3 will first be considered and afterwards we will generalize our result.

Let's summarize our notation and introduce some new one. We have :

h : Histogram pattern, $\mathbf{h} \in H$; e.g. for n = 4 : the template *[1 0 2 2]* gives histogram **h** = *[1 1 2]*

$h_i, i = 0,1,2$: Number of occurrences of the block i in the pattern, e.g. $h_0 = 1,\ h_1 = 1,\ h_2 = 2$

y : Observation pattern : it has the same format as **h** and is subject to comparison with it.

$y_i,\ i = 0,1,2$: Number of occurrences of the block *i* in the observation vector.

ϵ_{ij} : Block transition probability *Pr{i→j}*.

k_{ij} : Number of (original) blocks *i* in **h** "transformed" (as a result of noise corruption) into blocks *j* in the observation pattern **y**.

We want to evaluate the histogram transition probabilities :

$$\begin{aligned} P_{\mathbf{h}}(\mathbf{y}) = P_{[h_0,h_1,h_2]}([y_0,y_1,y_2]) &= Pr\{\mathbf{y} \textit{ is composed by } y_0 \textit{ elements (blocks) of type } 0, \\ &\qquad\qquad y_1 \textit{ elements} \qquad \textit{of type } 1, \\ &\qquad\qquad y_2 \textit{ elements} \qquad \textit{of type } 2\ / \\ &\quad \mathbf{h} \textit{ is composed by } h_0 \textit{ elements (blocks) of type } 0, \\ &\qquad\qquad h_1 \textit{ elements} \qquad \textit{of type } 1, \\ &\qquad\qquad h_2 \textit{ elements} \qquad \textit{of type } 2\ \} \\ &\quad \times \textit{ the size of the histogram class of } \mathbf{y} \end{aligned}$$

All pairs of (**h**, **y**) can be represented in the form shown in figure 4.1.

Note that :

- $k_{i2} = h_i - k_{i0} - k_{i1},\ \ i = 0,1,2$
- $k_{2i} = y_i - k_{0i} - k_{1i},\ \ i = 0,1,2$

So the free parameters are : $k_{00}, k_{01}, k_{10}, k_{11}$.

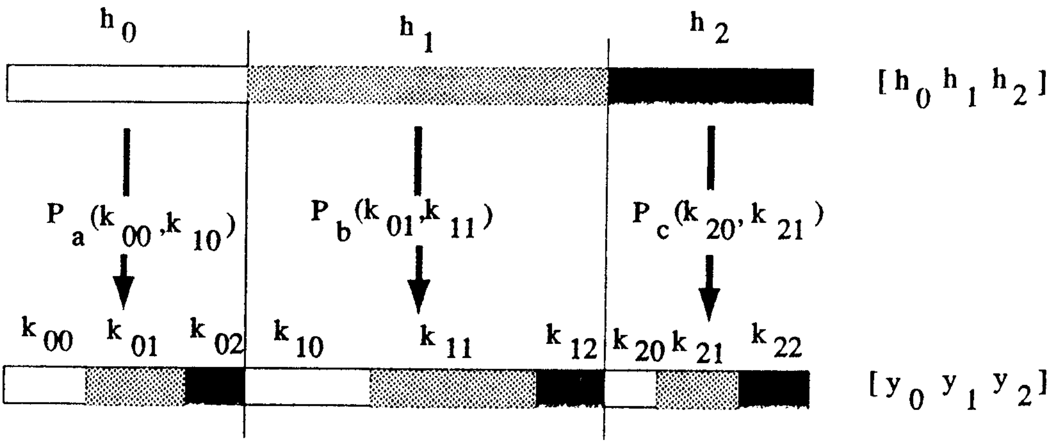


Figure 4.1 Transition probabilities for the case of ternary (M=3) image

Compute the probabilities P_a, P_b, P_c : The number of combinations of k_{00}, k_{01}, k_{02} elements of type 0, 1, 2 respectively in h_0 places is :

$$\binom{h_0}{k_{00}} \binom{h_0 - k_{00}}{k_{01}} \; .$$

For a specific configuration of the h_0 first elements of the y-pattern we have the transition probability

$$\epsilon_{00}^{k_{00}} \epsilon_{01}^{k_{01}} \epsilon_{02}^{h_0 - k_{00} - k_{01}} \; .$$

Therefore we get

$$P_a\,(k_{00}, k_{01}) = \binom{h_0}{k_{00}} \binom{h_0 - k_{00}}{k_{01}} \epsilon_{00}^{k_{00}} \epsilon_{01}^{k_{01}} \epsilon_{02}^{h_0 - k_{00} - k_{01}} \; .$$

Similarly

$$P_b\,(k_{10}, k_{11}) = \binom{h_1}{k_{10}} \binom{h_1 - k_{10}}{k_{11}} \epsilon_{10}^{k_{10}} \epsilon_{11}^{k_{11}} \epsilon_{12}^{h_1 - k_{10} - k_{11}}$$

$$P_c\,(k_{00}, k_{01}, k_{10}, k_{11}) = \binom{h_2}{k_{20}} \binom{h_2 - k_{20}}{k_{21}} \epsilon_{20}^{k_{20}} \epsilon_{21}^{k_{21}} \epsilon_{22}^{h_2 - k_{20} - k_{21}} .$$

$$where \;\; k_{20} = y_0 - k_{00} - k_{10}, \;\; k_{21} = y_1 - k_{01} - k_{11} .$$

We can now compute the probability

$$P_{\mathbf{h}}(\mathbf{y}) = \frac{n!}{y_0!y_1!y_2!} \sum_{\substack{\text{all acceptable combinations} \\ \text{of } k_{00}, k_{01}, k_{10}, k_{11}}} P_a(\cdot)\,P_b(\cdot)\,P_c(\cdot) \quad ,$$

provided we have the ranges for the elements in the 4–tuple of k's.

From the figure 4.1 we get the following constraints :

$$0 \le k_{00} + k_{10} \le y_0 \qquad (1)$$

$$0 \le k_{01} + k_{11} \le y_1 \qquad (2)$$

$$0 \le (h_0 - k_{00} - k_{01}) + (h_1 - k_{10} - k_{11}) \le y_2 \qquad (3)$$

$$0 \le k_{00} + k_{01} \le h_0 \qquad (4)$$

$$0 \le k_{10} + k_{11} \le h_1 \qquad (5)$$

$$0 \le (y_0 - k_{00} - k_{10}) + (y_1 - k_{01} - k_{11}) \le y_2 \qquad (6)$$

The inequalities (1), (2), (4) and (5) are the conditions for $k_{02}, k_{12}, k_{20}, k_{21}$ respectively to be positive numbers. The inequalities (3) and (6) are necessary for k_{22} to be a positive number but they are not sufficient.

We also have to impose the following reasonable constraint :

$$k_{ij} \ge 0, \;\; i, j = 0, 1 \;. \qquad (7)$$

We have :

$$(2),(6) \;\Rightarrow\; 0 \le y_0 - k_{00} - k_{10} \le h_2 \;\Rightarrow\; k_{00} + k_{10} \ge y_0 - h_2 \qquad (8)$$

$$(1),(8) \;\Rightarrow k_{c0}^{\min} = \max\{y_0 - h_2, 0\} \le k_{00} + k_{10} \le y_0 \qquad (9)$$

Similarly we also get :

$$k_{c1}^{\min} = \max\{y_1 - h_2, 0\} \le k_{01} + k_{11} \le y_1 \qquad (10)$$

$$k_{r0}^{\min} = \max\{h_0 - y_2, 0\} \le k_{00} + k_{01} \le h_0 \qquad (11)$$

$$k_{r1}^{\min} = \max\{h_1 - y_2, 0\} \le k_{10} + k_{11} \le h_1 \qquad (12)$$

The inequalities (7) and (9)-(12) constitute a set of necessary conditions for $k_{ij} \geq 0,\ i,j = 0,1,2$; they become sufficient if the induced value for k_{22} is also checked to be a positive integer. In the figure 4.2 the inequalities are pictorially summarized. The summations of the row elements as well as the summations of the column elements have to lie in the ranges provided; the minimum values are given up and left, while the maximum values are at the bottom and right.

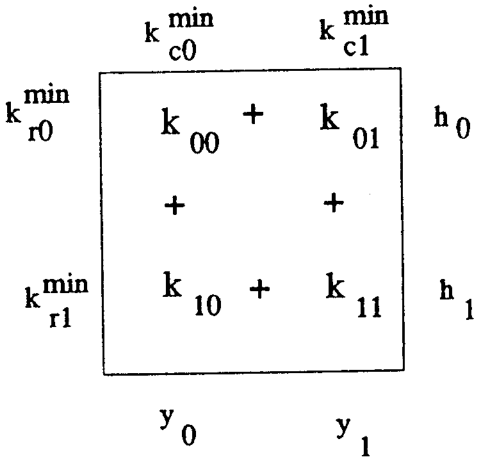


Figure 4.2 The parameter (k's) constraints for the case of the ternary (M=3) image

Now we can generalize the result produced for the special case of alphabet size M = 3 to hold for a block alphabet of arbitrary size. The problem amounts to finding the transitions probabilities

$$P_{\mathbf{h}}(\mathbf{y}) = Pr\left\{\mathbf{h} = [h_1\ h_2\ \cdots\ h_{n_h}] \rightarrow \mathbf{y} = [y_1\ y_2\ \cdots\ y_{n_y}]\right\} \ .$$

where h_i (or y_j) are the numbers of occurrences of a symbol in the vector **h** (or y) and n_h, n_y, $(n_h, n_y \leq M)$ are the numbers of non zero bars in the**h** and y histograms.

First we will find the probabilities $P_i(\cdot)$, $i = 1, 2, \cdots, n_h$, which correspond to the probabilities $P_a(\cdot)$, $P_b(\cdot)$, $P_c(\cdot)$ for the case of $M = 3$. As a direct extension of the special case we get :

$$P_i(k_{i1}, k_{i2}, \cdots, k_{in_y-1}) = \underset{\longleftarrow \qquad\qquad n_y - 1 \qquad\qquad \longrightarrow}{\binom{h_i}{k_{i1}} \binom{h_i - k_{i1}}{k_{i2}} \cdots \binom{h_i - \sum\limits_{j=1}^{n_y-2} k_{ij}}{k_{n_y-1}}} \times$$

$$\times \underset{\longleftarrow \quad n_y \quad \longrightarrow}{\epsilon_{i1}^{k_{i1}} \epsilon_{i2}^{k_{i2}} \cdots \epsilon_{in_y}^{h_i - \sum\limits_{j=1}^{n_y-1} k_{ij}}}$$

For $i = n_h$, i.e. for $P_{n_h}(\cdot)$, the above formula holds but we we also have

$$k_{n_h j} = y_j - \sum_{i=1}^{n_h-1} k_{ij} \ , \quad j = 1, 2, \cdots, n_y$$

and therefore $P_{n_h}(\cdot)$ depends on all k_{ij} , $i = 1, 2, \cdots, n_h - 1$, $j = 1, 2, \cdots, n_y - 1$.

The probability $P_h(y)$ will be of the form :

$$P_h(y) = \frac{n!}{y_1! y_2! \cdots y_{n_y}!} \sum_{\substack{\text{all acceptable} \\ k_{ij} \text{ tuples}}} \prod_i P_i(k_{i1}, k_{i2}, \cdots, k_{in_y-1}) \ .$$

The necessary conditions to be satisfied from the k's are summarized in the figure 4.3. An additional condition providing sufficiency is that the induced $k_{n_h n_y}$ element be a positive integer.[6]

[6] In the subsection A.3 a systematic way is described for producing the acceptable tuples of k's.

$$\begin{array}{c|ccccccc|c}
 & k^{min}_{c\,0} & & k^{min}_{c1} & & \cdot\,\cdot & & k^{min}_{c\;ny-2} & \\ \hline
k^{min}_{r0} & k_{00} & + & k_{01} & + & \cdot\,\cdot & + & k_{0\;ny-2} & h_{0} \\
 & + & & + & & & & + & \\
k^{min}_{r1} & k_{10} & + & k_{11} & + & \cdot\,\cdot & + & k_{1\;ny-2} & h_{1} \\
 & + & & + & & & & + & \\
\cdot & \vdots & & \vdots & & & & \vdots & \cdot \\
\cdot & + & & + & & & & + & \cdot \\
k^{min}_{r\;nh-2} & k_{nh-2\;0} & + & k_{nh-2\;0} & + & \cdot\,\cdot & + & k_{nh-2\;ny-2} & h_{nh-2} \\ \hline
 & y_{0} & & y_{1} & & \cdot\,\cdot & & y_{ny-2} &
\end{array}$$

Figure 4.3 The parameter (k's) constraints for the case of multiple valued image cells (blocks or pixels)

4.2.4 The Decision Rule

Given the histogram alphabet, the prior probabilities of the histogram patterns (states), the transitions among them (conditional probabilities) and the cost function performing the desirable grouping of the states, we may find a decision rule which minimizes the mean cost of the "matching"/"non matching" decision. The rule d(.) as an application of the M-ary hypothesis testing is as follows :

Let y be the observation pattern, p_q , $q \in H$ be the prior probabilities for the histogram patterns and $P_h(y)$, $h, y \in H$ be the transition probabilities.

Then the rule is

$$d(\mathbf{y}) = i \Leftrightarrow g^{i}(\mathbf{y}) \leq g^{1-i}(\mathbf{y}), \quad g^{i}(\mathbf{y}) = \begin{cases} \sum\limits_{\mathrm{q} \in H_0} p_{\mathrm{q}} P_{\mathrm{q}}(\mathbf{y}), & i = 1 \\ \sum\limits_{\mathrm{q} \in H_1} p_{\mathrm{q}} P_{\mathrm{q}}(\mathbf{y}), & i = 0 \end{cases} .$$

We can numerically determine a partition $\{R_0, R_1\}$ of H (see appendix) such that

$$d(\mathbf{y}) = i \Leftrightarrow \mathbf{y} \in R_i \, , \quad i = 0, 1 \, .$$

We recall that

$$d(\mathbf{y}) = \begin{cases} 1 & \textit{if we decide matching} \\ 0 & \textit{if we decide non matching} \end{cases} .$$

Note also that the calculation of the partition $\{R_0, R_1\}$ corresponds to the calculation of a threshold value in the case of the binary block alphabet as we have seen in chapter 2.

4.2.5 Performance Evaluation of the Decision Rule

We will calculate the (P_{fa}, P_d) pair for our rule.

Probability of false alarm :

$$\left.\begin{array}{l} P_{fa} = Pr\{\mathbf{d} \in H_1 \mid \mathbf{h} \in H_0\} = \sum\limits_{\mathbf{i} \in H_1} Pr\{\mathbf{d} = \mathbf{i} \mid \mathbf{h} \in H_0\} \\ Pr\{\mathbf{d} = \mathbf{i} \mid \mathbf{h} \in H_0\} = \sum\limits_{\mathbf{i} \in H_0} Pr\{\mathbf{d} = \mathbf{i} \mid \mathbf{h} = \mathbf{j}\} \, Pr\{\mathbf{h} = \mathbf{j} \mid \mathbf{h} \in H_0\} \end{array}\right\} \Rightarrow$$

$$P_{fa} = \sum_{\mathbf{i} \in H_1} \sum_{\mathbf{j} \in H_0} Pr\{\mathbf{d} = \mathbf{i} \mid \mathbf{h} = \mathbf{j}\} \, Pr\{\mathbf{h} = \mathbf{j} \mid \mathbf{h} \in H_0\} =$$

$$P_{fa} = \sum_{\mathbf{j} \in H_0} Pr\{\mathbf{h} = \mathbf{j} \mid \mathbf{h} \in H_0\} \sum_{\mathbf{i} \in H_1} Pr\{\mathbf{d} = \mathbf{i} \mid \mathbf{h} = \mathbf{j}\} \, .$$

$$\textit{where} \quad Pr\{\mathbf{d} = \mathbf{i} \mid \mathbf{h} = \mathbf{j}\} = \sum_{\mathbf{y} \in R_{\mathbf{i}}} P_{\mathbf{j}}(\mathbf{y})$$

for some decision region $\tilde{R}_i$. So

$$P_{fa} = \frac{1}{\sum\limits_{l\in H_0} p_l} \sum_{j\in H_0} p_j \sum_{i\in H_1} \sum_{y\in \tilde{R}_i} P_j(y) \Rightarrow$$

$$P_{fa} = \frac{1}{\sum\limits_{l\in H_0} p_l} \sum_{j\in H_0} p_j \sum_{y\in R_1} P_j(y) \quad .$$

Similarly the *probability of miss* P_m is found to be

$$P_m = \frac{1}{\sum\limits_{l\in H_1} p_l} \sum_{i\in H_1} p_i \sum_{y\in R_0} P_i(y)$$

and as usually the *probability of detection* P_d is $P_d = 1 - P_m$.

In the plot 9 the Receiver Operating Characteristic (ROC) curve is given for various values of the inversion probability ϵ in the original image data. The curves are drawn for the 2–block template shown in figure 5.1. The subset of the first four elements of the modified Hadamard basis was used as the block codebook. We observe that these ROC curves are much closer to the point (P_{fa}=0, P_d=1) than the ROC curves we already have seen (plots 2 and 4); this is surprising since in those tests we assumed we knew the background of the target object while here we do not. Certainly the coding procedure we used may have contributed to the amelioration of the decision rule but still this does not seem to be a satisfactory explanation. We can find an answer for this question if we think that here the detection region H_1 is a singleton which contains the target object itself, while in the previous tests it was containing the target object (the black square) as well as all the patterns representing part of it in white background. For example the four patterns containing only one black pixel of a corner of the black square belong in H_1 and contribute to the probability of miss in case they are not detected. On the other hand the fact that such patterns belong in H_1 implies a low decision threshold y_0, which forces P_{fa} to augment.

In plot 10 the P_{fa}—P_d curve (for the optimal Bayesian rules) parametrized by the pixel inversion probability in the original image is given. The modified Hadamard basis is used for the coding of the image data; the curve is drawn for the 2×2–block full black template. Further discussion on the comparison of decision rules is made in subsections 5.1 and 5.2.

4.3 Summary

In this chapter we introduced the notion of the histogram and discused the histogram alteration as a result of the alteration of bar elements. We also developed a decision rule for inference about the matching of such histograms. Our rule actually resides on an M-ary hypothesis test and is optimal in the sense of minimizing the probability of taking an erroneous decision. The performance evaluation characteristics of this rule can be computed numerically. Sample results are given.

In the next chapter we will comment on the discrimination power of the histogram matching test and compare it with the template matching test in terms of complexity and reliability.

Chapter 5
Discussion

5.1 Review : Detection in Background

Several instances of the two dimensional matching problem have been studied. Our intention was to show the dependence of the detection capability on the noise level that the image suffers and the effect of quantization on the image data. Throughout all the preceding analysis we primarily concentrated on the 8×8–pixel full black template as the prototype geometrical object to be recognized. In chapter 2 we first considered a binary uncoded image corrupted by AWG noise. In that case we knew we had white background and therefore took advantage of the a priori known possible positions of the image window scanning the image in relation with the target black square. This background knowledge gave rise to an M-ary test, for which the distinct states represented the possible image window positionings (see fig. 2.1) i.e.

1 completely white : view the background

2 completely black : view the target square

3 partially black : view the neighborhood to the target square pattern.

Afterwards we grouped together the 2– and 3–type states, forming thus the binary test which we eventually treated in a Bayesian framework. We gave an insight on how the noise affects the decision making (see figure 2.2 and plot 1). The ROC of the test was also given for noise variance equal to σ^2=0.1 (plot 2).

A similar formulation was presented for the case of the detection in background with BSC noise. The noise effect was studied (plot 3) and the ROC curve for pixel inversion probability ϵ=0.1 was given (plot 4). We will use the notation *(2,64)-KB* to characterize the test for matching a 64–element pattern of 2–valued elements (bits) in known background. In both cases above the sum-of-pixels statistic (equivalent to the Hamming weight for the BSC noise case) was used. This essentially means that we do not take into account the positions of the pixels; we simply count the white and the black ones and use this information for our test. Evidently this affects the reliability of the test when the position information for the template carries important additional information; this will be discussed in the next subsection. Note though that position information is assumed to be of no interest for the case of the full black template.

A simple way of compacting the image data, the coarsening of resolution, is presented at the end of chapter 1. Based on the coded data a rule for detection in background is developed for which the 8×8–pixel template of the previous two cases is replaced by a 4×4–block template of 2×2–pixel blocks. This detection rule turns out to be more robust to the noise than the rule based on uncoded data, provided we do not have much noise (compare the plots 3 and 5 for pixel inversion probability ϵ<0.05). The result is reversed for large values of ϵ, since the $P_{fa}(\epsilon)$—$P_d(\epsilon)$ curve for the uncoded case is higher and closer to the P_d axis. This observation is further validated by the ROC curves for ϵ=0.1 (plot 11 : (2,16)-KB test better) and ϵ=0.3 (plot 12 : (2,64)-KB test better). We will characterize this test as a (2,16)-KB test since it matches a 16–element pattern of 2–valued elements.

In chapter 4 we developed another test acting on coded image data. This time we used a modification of the 4×4 Hadamard basis to code the image data (see fig. 3.1). The detection of the black square in white background eventually is equivalent to the previous scheme, since we have zero prior probabilities for all but the first two elements of the base; though here the original 8×8–pixel template is treated as a 2×2–block template of 4×4–pixel blocks, i.e. we have an even coarser quantization. The ROC for this test, which is treated as a (2,4)-KB test, is rather worse than the one for the (2,16)-KB test, while the optimal Bayesian rules (plot 7) give considerably lower P_d values from the previous tests even for ϵ=0.1 (see plot 8).

In plots 13 and 14 a comparison of these three tests acting on data with different resolutions is attempted. It is apparent that at certain noise level the compression of data favors detection capability, while for higher noise level better detection results are obtained by a rule based on the uncoded data. More specifically in plot 14 the (2,16)-KB test has higher P_d at three distinct intervals of ϵ (around the points ϵ=0.02, ϵ=0.14 and ϵ=0.22). Also the (2,4)-KB test is superior to the other two tests for heavily noise corrupted data (ϵ>0.15). In plot 13 however it becomes evident that these superiorities of the coarsened resolution data always are penalized with peaks of the ϵ—P_{fa} curve. Nevertheless, note that for ϵ<0.05 the (2,16)-KB test performs better than the (2,64)-KB test in terms of both P_{fa} and P_d.

5.2 Review : Detection Without Known Background

When we are looking for an object in an image we usually do not have the luxury of knowing what we expect to find around it and also we do not look only for such a uniform object as the full black template is. These facts led us to the modification of the Hadamard basis (in chapter 3) and to the general type *(M,n)-UB* test acting on n-element patterns of M-valued elements (in chapter 4) in unknown background. This type of test relies on the histogram statistic, which is the generalization of the sum-of-pixels statistic for the case of the M-valued elementary image data.

In this subsection we will see the results of two kinds of experiments. In the first one the image elements in the *(M,n)-UB* test are the codewords of the modified Hadamard basis or some subset of this. So we will continue the discussion on the noise effects as opposed to the quantization effects on the detection capability in the more general and practical case where we have more than two distinct codewords. The resulting tests will usually have M>>n. In the second kind of experiments we will have multiple valued pixels which may be thought as multiple gray level pixels or multiple color pixels. The experiments are relevant to the classification of patterns according to their color. The resulting tests usually have M<<n. In both types of tests we assume we have no knowledge of the background. This means that :

1. When scanning the image we cannot take advantage of the windows viewing the target partially; note that this was the reason that induced the M-ary hypothesis test for the full black template case in the *(2,n)-KB* type test.

2. We need to know the prior probabilities of the image elements we use.For the case of the image elements which are codewords we derived these probabilities based on statistical results given in [LaSl71]. Therefore the results of this analysis apply for searching in natural (not synthetic) images as well. For the case of multiple valued pixels we arbitrarily assume uniform distribution.

Remember that in the *(M,n)-UB* test we have to match an n-element pattern of M-valued elements. We assume that only the number of the distinct element values and not the specific positions is important. Consequently we have to match a histogram of M bins whose bars sum up to n. Each such distinct histogram constitutes a hypothesis of the M-ary test. Recal that we will have $S(M,n) = \binom{M+n-1}{n}$ such hypotheses. We may define one or more of them as target histograms and group the hypotheses into two parts. The resulting binary test distinguishing these two sets is what we call the *(M,n)-UB* type test.

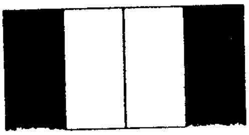


Figure 5.1 A sample template

Consider the (2,4) test we mentioned in the previous subsection with the difference that now we have no knowledge about the background. In plot 10 the noise effect is shown for the corresponding (16,4)-UB test. Note how the performance of the test worsens as the pixel inversion probability ϵ in the original image ranges from 0.05 to 0.35. For ϵ=0.35 the optimal Bayesian rule is "never decide square".[7]

Consider the template of the figure 5.1. A (4,2)-UB test for this template has been studied; the ROC curve for a number of different values of ϵ is given (see plot 9). The first four elements in the modified Hadamard basis with normalized prior probabilities are used. Note that although the blocks constituting the template are selected to be the ones least *a priori* favored, the test is quite reliable (the ROC is close to the P_d axis) even for ϵ=0.4. This was discussed in the subsection 4.2.5.

5.3 The Code-Corrupt-Detect System and the Multiple Gray Level Images

In the model we studied so far the noise process affects the uncoded image and the noise effect propagates in the coded version of the image. In this way we can exploit the effect of the noise generated at the moment we receive the image and thus this model may be called "the corrupt-code-detect system".

A related problem is that of introducing the noise after the image has been coded; this is the case of the degradation of an image during the transmission of the coded version of it over a noisy channel. For the case of the modified Hadamard basis we use the source coding shown in table 5.1.

This specific code is selected because it retains the low transition probabilities between the elements 1 and 2, 3 and 4, e.t.c. in the coded version of the image.

[7] For this test the ROC cannot be computed due to the memory requirements for such a computation. Notice that the decision region which is recursively computed must range from 0 to S(16,4)=3876.

In the plots 15–23 results similar to the ones found for the code-corrupt-detect case are illustrated. It is worthwhile to note that

- The optimal Bayesian rules based on the pixel data always perform better (have higher probability of detection) than the rules based on the coded data.
- However there is a small range for ϵ (close to ϵ=0.06) where the (2,4)-KB test with compression rate 1/16 bpp performs better than both the (2,16)-KB and (16,4)-UB tests with compression rate 1/4 bpp.

base element	code	base element	code
1	0000	9	0100
2	1111	10	1011
3	0001	11	0101
4	1110	12	1010
5	0010	13	0110
6	1101	14	1001
7	0011	15	0111
8	1100	16	1000

Table 5.1 Source coding for the modified Hadamard basis elements

In order to study the histogram matching problem in the case of multiple gray level images now, we need to define a pixel transition matrix, for instance the one in table 5.2 for the case of 3 gray levels.

	0	1	2
0	$1-\varepsilon-\varepsilon^2$	ε	ε^2
1	ε	$1-\varepsilon-\varepsilon^2$	ε
2	ε^2	ε	$1-\varepsilon-\varepsilon^2$

ε = noise parameter

Table 5.2 Ternary pixel transition probability matrix

The prior probability mass function as already discussed is assumed to be uniform. The P_{fa}—P_d curve parametrized by ϵ for the optimal Bayesian rule, concerning the test of 2×3–pixel template of 3–valued pixels, i.e. the (3,6)-UB test is given in plot 24.

5.4 The Bayesian Approach with "Position Independent" Statistics v.s. the Bayesian and Maximum Likelihood Approaches with "Position Dependent" Statistics

The tests we have examined so far rely on the sum-of-pixels or the histogram statistic. As already indicated the common characteristic of these two statistics is that they do not take into consideration the position information, as other statistics like the correlation statistic, do. We will call *position independent* the tests or rules related to the first kind of statistic and *position dependent* the tests or rules related to the second kind of statistic. In this subsection we will see how these statistics compare with the position depended ones in terms of computational complexity and the reliability of the related tests.

Consider the case of a d×d-element template (note : d^2=n) of M-valued elements. The set of possible templates is of size $M^{d^2} = M^n$ (for instance d=8, M=2, $M^n \approx 1.8 \cdot 10^{19}$ and d=2, M=16, M^n=65536) which reflects the size of the state space of a position dependent test. As we have seen in chapters 2 and 4 the template size determines the computational complexity for computing the Bayesian rule, as well as the complexity and memory requirements for evaluating the rule. Consequently for problems where from a practical point of view it is not desirable to develop Bayesian rules, we usually prefer the maximum likelihood (ML) approach (see subsection 1.2) which gives reasonable results. Still both ML and Bayesian position dependent rules imply computationally expensive matching tests, since at each time instant of the image scanning they need O(n^2) operations.

On the other hand the state space size of the position independent tests is $S(M,n) = \binom{M+n-1}{n}$, (for instance d=8, M=2, S(M,n)=65 and d=2, M=16, S(M,n)=3876), which is considerably smaller from the one of the other Bayesian rules[8]. So the histogram related rules as opposed to the position dependent Bayesian rules, turn out to be more computationally feasible for a wider range of templates.

[8] By using Stirling's approximation for the factorials one can easily show that for the hard case of M>>n we have

$$\frac{S(M,n)}{M^n} \underset{M \gg n}{\longrightarrow} \left(\frac{e}{n}\right)^n$$

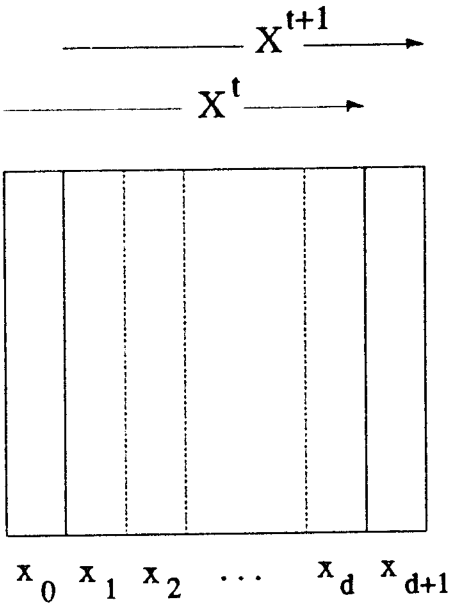


Figure 5.2 One time instant sliding of the scanning image window

Furthermore, the matching test requires *O(min(max(d,M),d²))* operations, as opposed to the *O(d²)* operations for both ML and Bayesian position dependent rules : Consider two consequent positions X^t, X^{t+1} of the d×d window scanning the image (see fig. 5.2). Let $h_0, h_1, \cdots, h_{d-1}$ and $h_1, h_2, \cdots, h_d$ represent the histograms of the elements of the column vectors $x_0, x_1, \cdots, x_{d-1}$ and $x_1, x_2, \cdots, x_d$ of X^t and X^{t+1}.The values of the histogram statistics for the two consequent positions are

$$h^t = h_0 + h_1 + \cdots + h_{d-1}, \quad h^{t+1} = h_1 + h_2 + \cdots + h_d \ ;$$

note that $h^{t+1} = h^t + h_d - h_0$, which requires *O(d)* operations. Afterwards we have to compare the M-vector $\mathbf{h}^{t+1}$ with the template histogram. This is an *O(M)* operation; but since only the non zero elements of the M-vector count and these cannot be more than d^2, the overall test requires $O(min(max(d,M),d^2))$ operations. Note though that this is a conservative estimate; we expect that the complexity will practically be reduced to $O(d)$ because of the smoothness of the image, but this certainly requires simulation verification.

The penalty for the reduced complexity of the position independent statistics is a higher probability of false alarm compared to that of the position dependent ones. Let us suppose that among the $d^2=n$ elements of $\mathbf{X}^t$ we have k distinct ones namely n_0 of type 0, n_1 of type 1, ..., n_k of type k, so that

$$n_0 + n_1 + \cdots + n_k = n \ .$$

This histogram corresponds to

$$c(\mathbf{n}) = \binom{n}{n_0}\binom{n-n_0}{n_1}\binom{n-n_0-n_1}{n_2}\cdots\binom{n_k}{n_k}$$

possible template patterns. Suppose that among these $c = c(\mathbf{n})$ patterns is the one we want to detect. Ideally our rule will respond positively each time it scans one of these patterns. Consequently, each time we scan our target pattern, the rule will recognize it; but it will recognize all the $c-1$ patterns sharing the same histogram with our target as well. So if we denote by $h, P_{fa}, P_d, \tilde{h}, \widetilde{P}_{fa}, \widetilde{P}_d$ the state and performance evaluation parameters for the position independent test and the position depended test respectively, we expect by intuition that

$$\widetilde{P}_d = P_d \ \ and \ \ P_{fa} = P_{fa} + (c-1)P_d \ .$$

We shall see though that things are slightly different.

Let us denote with d the decision outcome 0 or 1 from now on. In figure 5.3 we can see the relations of the regions characterized by the values of $h, \tilde{h}$ and d.

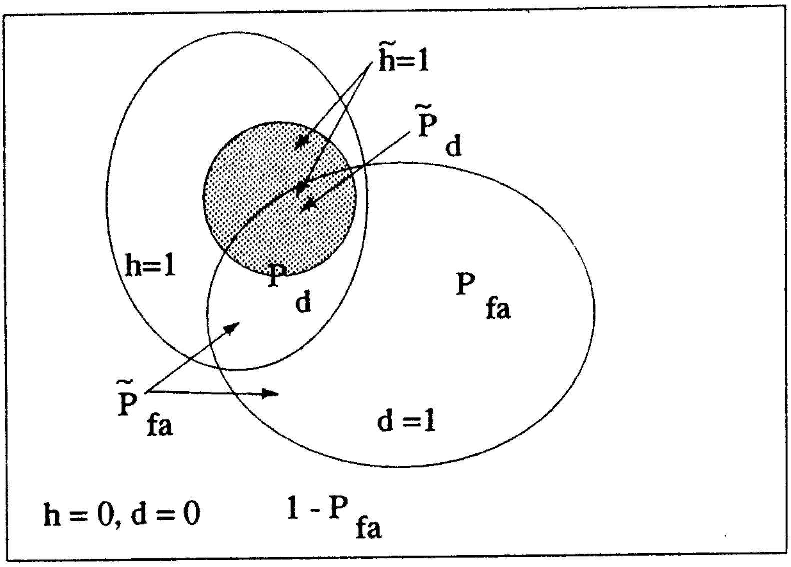


Figure 5.3 Comparison of the performance characteristics for the position dependent and position independent tests.

We know make the following assumption : Given that an image element lies in a template the probability mass function of the random variable indicating its position and ranging over all the allowable positions is a uniform pmf. In other words if c(**n**) distinct patterns result in the same histogram **n** (and no other does), given the histogram **n** any pattern resulting it may occur with probability 1/c(**n**).

From the figure 5.3 we induce that the above assumption implies :

$$Pr\left\{\tilde{h} = 1\right\} = \frac{1}{c} Pr\left\{h = 1\right\} \qquad (1)$$

$$Pr\left\{d = 1, \tilde{h} = 1\right\} = \frac{1}{c} Pr\left\{d = 1, h = 1\right\} . \quad (2)$$

From equation 2 we get :

$$\tilde{P}_d = Pr\left\{d = 1 \mid \tilde{h} = 1\right\} = \frac{1}{c}\frac{Pr\left\{d = 1 \mid h = 1\right\} Pr\left\{h = 1\right\}}{Pr\left\{\tilde{h} = 1\right\}} \quad (3)$$

$$(1),(3) \Rightarrow \tilde{P}_d = \frac{1}{c} P_d \cdot c = P_d$$

as it was expected. We also have :

$$\begin{aligned} Pr\left\{\tilde{h}=0\right\} &= Pr\{h=0\} + Pr\left\{h=1, \tilde{h}=0\right\} \\ &= Pr\{h=0\} + Pr\{h=1\} - Pr\left\{\tilde{h}=1\right\} \\ &= Pr\{h=0\} + Pr\{h=1\} - \frac{1}{c} Pr\{h=1\} \\ &= Pr\{h=0\} + \left(1-\frac{1}{c}\right) Pr\{h=1\} \end{aligned}$$

and similarly

$$Pr\left\{d=1, \tilde{h}=0\right\} = Pr\{d=1, h=0\} + \left(1-\frac{1}{c}\right) Pr\{d=1, h=1\} \quad .$$

So

$$\begin{aligned} \widetilde{P}_{fa} = Pr\left\{d=1 \mid \tilde{h}=0\right\} &= \frac{Pr\left\{d=1, \tilde{h}=0\right\}}{Pr\left\{\tilde{h}=0\right\}} \\ &= \frac{Pr\{d=1, h=0\} + (1-\frac{1}{c}) Pr\{d=1, h=1\}}{Pr\{h=0\} + (1-\frac{1}{c}) Pr\{h=1\}} \\ &= \frac{\alpha P_{fa} + \beta P_d}{\alpha+\beta} \quad , \quad \alpha = \frac{1}{Pr\{h=1\}} \quad , \quad \beta = \frac{1-\frac{1}{c}}{Pr\{h=0\}} \quad . \end{aligned}$$

This is a convex combination of P_{fa} and P_d, which is not exactly what we were expecting to find. We may observe that as *n* gets larger *c* gets larger and consequently β gets larger, so $\widetilde{P}_{fa}$ gets larger. Note also that the prior knowledge $Pr\{h=i\}$, $i = 0, 1$, explicitly affects $\widetilde{P}_{fa}$. For the full black template we have *c*(**n**)=*1*; so for equal priors we have $\widetilde{P}_{fa} = P_{fa}$. This result holds for the all-white template as well. The P_{fa} probability for optimal Bayesian rules as a function of the noise parameter ϵ is given in plot 18.

Thus our position independent tests :

- Are faster but tend to have higher probability of false alarm than both ML and Bayesian position dependent tests.

- Accept analytic performance evaluation and are more powerful than the ML position dependent tests, since they have the same power with the Bayesian position dependent tests.
- Are more attractive from the computational complexity point of view than the position dependent Bayesian tests, but still not adequately attractive for non trivial templates.
- Are equivalent with the Bayesian position dependent tests, in terms of the P_{fa} and P_d characteristics, for the case of the full black and all-white templates, under mild conditions.

5.5 Overview

The template matching problem for noise corrupted binary images was considered. We attempted a Bayesian formulation of the problem that resulted in the use of simplifying statistics. In this way the complexity of determining the optimal Bayesian rule is reduced at the cost of a higher false alarm rate. Matching rules for the original pixel image, as well as data of a coarser resolution and data coded by a modification of the Hadamard basis were developed. These rules gave an intuition of how the data compression and the noise affect the detection capability. Our conclusion is that a certain amount of noise is "killed" by appropriate data compression. Another issue studied was the knowledge about the background surrounding the target object; such knowledge was shown to enhance the reliability of the matching rule.

Practically in "Bayesian template matching", given a target template (or set of target templates) and certain level of noise, we should :

1. Determine the set of patterns for which we decide "matching"; this is in general a very time consuming process.
2. Given this set the search for an object is reduced to searching in the coded image, and checking if the pattern lies in our set.

An application of this process could be characterized as "searching in a bank of images", for a given object. Since we know that we suffer a high false alarm rate but the probability of detection is still high, we can precede the first selection of candidate matching patterns by an ML position dependent test (see subsection 1.2); this second test may be slower but more reliable (lower P_{fa}).

From what we have seen so far our approach has a major weakness: We restrict ourselves to small templates (of size n=4, or 8×8–pixel) of binary images, i.e. we can describe just a small class of possible target objects. This happens because optimal Bayesian rules require computations of mean values over all the possible patterns, which run up to M^n for a matching rule based on position depended statistics and up to $S(M,n) = \binom{M+n-1}{n}$ for the rule based on the histogram statistic. For instance $16^9 \approx 6.9 \cdot 10^{10}$, while $S(16,9) \approx 1.3 \cdot 10^6$ for n=9. Consequently the object we are looking for must belong to some restricted class of such objects. An example of such a class is found by Knudson; in [Knu75] he shows experimentally that only 62 out of 2^{64} 8×8 binary block patterns suffice to give a good quality for newspaper text and graphics. A matching test for 8×8–pixel templates can be implemented with a codebook of size S(62,1)=62. A test for a 2×2–block template of 8×8–pixel blocks with a template codebook of size S(62,4)=677,040 is feasible as well. (Note that the corresponding position dependent test has a codebook of size larger than 11 million codewords).

The "bar code" patterns may constitute some other class of objects to be identified. Nevertheless, the use of a Bayesian matching rule is worthwhile only in the case where we have heavy noise corruption.

An attempt to expand the scope of the Bayesian template matching rules in the class of arbitrary multiple gray level objects can be made by using some efficient coding scheme like one discussed in [TaFa89]. Compaction rates up to 0.25 bpp are achieved by using subband coding [WoNe86]; it is shown that the lowest frequency subband (LFS) contains the most important data needed for the reconstruction of the image; Huffman code is used to encode 4×4–pixel blocks in each subband. Although a fixed length code would give a codebook of size $2^{4\times4\times0.25}=16$ the size of the codebook used for the variable length code must be much larger. The compromise we have to make amounts to :

- Use only the data of the lowest frequency subband.
- Use the M most frequent block codewords, where M is sufficiently large to make the class of the target wide, and at the same time small enough to make the template codebook $S(M,n)$ small.
- The size of the template in blocks is expected to be $n=1$, since Information Theoretic results imply that M (or log_2M) must be large if small compaction distortion is to be obtained[9].

[9] See Shannon's first coding theorem [Bla87, pp74]

An application of the matching test based on the histogram statistic is the detection of patterns based on color information. This rule being faster than the ordinary template matching, reflects the human ability of recognizing the color faster than the shape of an object [Chr75][10].

A Bayesian approach which is not as straightforward as the one already studied could incorporate some probabilistic model for the image (e.g. an Autoregressive model). Under this treatment the matching rule could be formulated as a sequential detection rule, thus avoiding the large state space. However this approach requires the assumption of stationarity for the image statistics, a situation that is not realistic.

[10] Experimental results concerning the time required to locate color targets relative to the time for shape targets localization are given in figures 8–12. It is (experimentally) shown that identification based on color information is faster than the one based on achromatic information, provided the awareness of the color of the targets (i.e. in our formulation, provided we have a known target color histogram).

Appendix A Algorithms

A.1 The Hamming weight of the m×m-pixel patterns of the test in ¶2.1

In the subsection 2.1 we have seen that apart from the all-white pattern $(2m-1)^2$ possible patterns for the test window are considered. For the special case of m=2 the Hamming weights (i.e. the number of black pixels in the pattern) for the 9 possible patterns (see fig. 2.1) are given in figure A.1a .

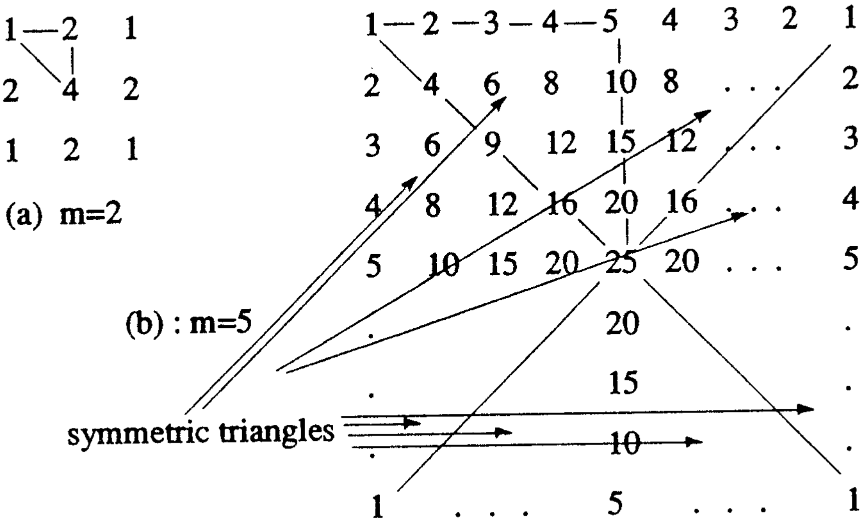


Figure A.1 Hamming weights for some pattern samples

The weights for the case of m=5 are given in figure A.1b. One can observe that all distinct weights appear in the triangle "1–5–25". This property holds for the general case of an arbitrary value of m, because of the symmetric overlaps between the black square on the image and the scanning test window. So we may have at most

$$m + (m-1) + \cdots + 1 = \frac{m(m+1)}{2}$$

distinct weight values. This leads to a reduction of the number of the states of the corresponding M-ary test from $(2m-1)^2 + 1$ to $m(m+1)/2 + 1$ of them, i.e. a reduction by a factor close to 8. The set S of all possible weights we may have is :

$$S = \cup_{i=1}^{m} \cup_{j=1}^{i} \{i \cdot j\} \quad ;$$

duplications in S can be exploited for fixed values of m.

If we are interested in the number of occurrences of each weight value in the set of the $(2m-1)^2$ patterns, we may observe that we have one time the weight m^2, 4 times the weights belonging to the set $\cup_{j=1}^{m-2} \{mj, j^2\}$, and 8 times the weights belonging to the set $\cup_{i=1}^{m-2} \cup_{j=i+1}^{m-1} \{ij\}$

A.2 The integer partitions of a nonnegative integer n

The problem is posed as follows: Given a natural integer n (i.e. $n \in \mathcal{N} = \{1, 2, \cdots\}$) find all the finite sequences of the form $\left\{ n_i / \sum_i n_i = n, n_i \in \mathcal{N} \; for \; all \; i's \right\}$.

The key idea is as follows[11] : We consider the product:

$$\begin{aligned} P(x) &= (1+\alpha_1 x+\alpha_1^2 x^2+\cdots)\left(1+\alpha_2 x^2+\alpha_2^2 x^4+\cdots+\alpha_2^{\lambda} x^{2\lambda}+\cdots\right)\ \cdots \\ &\qquad \left(1+\alpha_k x^k+\alpha_k^2 x^{2k}+\cdots\right)\ \cdots \\ &= 1+\alpha_1 x+\left(\alpha_1^{n_1}\alpha_2^{n_2}\cdots\alpha_k^{n_k}+\cdots\right)x^n\ \cdots \end{aligned}$$

Observe that the term

$$\alpha_1^{n_1}\alpha_2^{n_2}\cdots\alpha_k^{n_k}$$

which appears in the coefficient of x^n is such that $n_1+2n_2+\cdots+kn_k=n$ and thus it determines the following partition of n :

$$n = \underset{\longleftarrow n_k \longrightarrow}{k+k+\ldots+k} + \underset{.}{\ldots} + \underset{\longleftarrow n_2 \longrightarrow}{2+2+\ldots+2} + \underset{\longleftarrow n_1 \longrightarrow}{1+1+\ldots+1}$$

We are interested in the sum-of-products expansion of the n-th coefficient in the polynomial $P(x)=\alpha_1(x)\alpha_2(x)\ \cdots\ \alpha_k(x)\ \cdots$ given above. Obviously this coefficient is not affected by :

1. any multiplicand $\alpha_i(x)$, $i>n$
2. any summands in $\alpha_i(x)$, $i \leq n$ with order greater than *n*.

Therefore we can eliminate all these terms, without the resulting partitions to be affected.

[11] The solution we propose is an expansion of the solution proposed in [ToMe85, pp174] for finding the number of the integer partitions of a nonnegative integer n.

A.3 Solution for a system of inequalities in the domain of nonnegative integers

We have the following system of inequalities :

$$a^i_{\min} < k_{i1} + k_{i2} + \cdots + k_{in} < a^i_{\max}\ ,\ i = 1, \cdots, m$$

$$b^j_{\min} < k_{1j} + k_{2j} + \cdots + k_{mj} < b^j_{\max}\ ,\ j = 1, \cdots, n$$

where all variables belong in the set {0,1,2,...}.

The algorithm we use performs an exhaustive search for the solutions of the system :

1. for each i in $1,...,m$

   for each a^i in $a^i_{min}, ..., a^i_{max}$

   determine the set of integer partitions $\{\mathbf{k}_i\}$, $\mathbf{k}_i = [k_{i1}, k_{i2}, \cdots, k_{in}]$, of a^i in no more than n places

   and all permutations of these partitions in n places.

comment : each m-tuple for which the i-th element belongs to $\{\mathbf{k}_i\}$ satisfies the first set of inequalities, and no other m-tuples do.

2. for each m-tuple constructed as described in the comment above

   check if the second set of inequalities is satisfied;

   if yes obtain a solution $[\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_m]$

A.4 Determining the decision region R_1 of the subsection 4.2.4

We want to determine the set of histograms

$$\begin{aligned} R_1 &= \left\{ \mathbf{y} : f(\mathbf{y}) = \sum_{q \in H_1} p_q P_q(\mathbf{y}) - \sum_{q \in H_0} p_q P_q(\mathbf{y}) \geq 0 \right\} \\ &= \{ \mathbf{y} : f(\mathbf{y}) = S_1(\mathbf{y}) - S_0(\mathbf{y}) \geq 0 \} \end{aligned}$$

which is the region for deciding that the test window matches with the given template.

Consider the case in which the set of target templates H_1 is a singleton $H_1=\{\mathbf{y_0}\}$. The key idea is that the condition $f(y)\geq 0$ can be satisfied only by the points close to $\mathbf{y_0}$; in other words R_1 will be a neighborhood of $\mathbf{y_0}$. Stepping away from $\mathbf{y_0}$ (the given template) will cause $f(y)$ to get gradually smaller and at some point to become negative. Let us define the *set of closest neighbor* of a vector y to be the set of vectors differing at only one coordinate from y. Note that if y has n coordinates and each coordinate may take one of M possible values then y has n(M-1) closest neighbors.

The algorithm, which we call "stepping process", functions as follows :

1. Check if $f(\mathbf{y_0})\geq 0$. If not we have the trivial case of the identity decision rule $d(y)=0$ for every y; exit the algo. Else "put $\mathbf{y_0}$ in R_1 and mark a tag on it as "unchecked".
2. Check which of the closest neighbors of $\mathbf{y_0}$ satisfy the condition $f(\mathbf{y_0})\geq 0$ and put them in R_1 with their tags marked as "unchecked".
3. Mark $\mathbf{y_0}$'s tag as "checked".
4. For the "unchecked" vectors in R_1 repeat steps 2 and 3 until all vectors in R_1 are "checked".

Let us refine the second step of this process : Changing one of the coordinates of $\mathbf{y_0}$, say from $y^i=k$ to $y^i=l$ has the following effects:

1. The $\mathbf{y_0}$ vector is transformed into some other, say $\mathbf{y_1}$ vector differing only at the i-th position from $\mathbf{y_0}$.

2. The $S_1(.)$ term in f(.) gets smaller (usually by several orders of magnitude), since $S_1(\mathbf{y_0})$ will be substituted by

$$S(\mathbf{y}_1) = S(\mathbf{y}_0)\frac{P_k(l)}{P_k(k)} < S_1(\mathbf{y}_0) \quad .$$

where $P_k(l)$ and $P_k(k)$ are the block transition probabilities for which we have $P_k(l)<P_k(k)$ (unless we have a pathological case of noise).

3. the $S_0(.)$ term of f(.) gets larger since $\mathbf{y}_1 \in H_0$ by construction and the summand in $S_0(.)$ corresponding to it will get significantly larger while the other summands will approximately retain their values.

This monotonicity property of f(.) implies that R_l has to be a neighborhood of $\mathbf{y_0}$. The algorithm presented has a recursive nature since each vector entering in the R_l decision region becomes a starting point whose "closest" neighbors are to be checked.

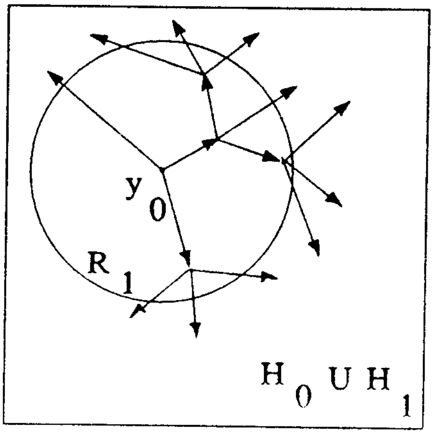
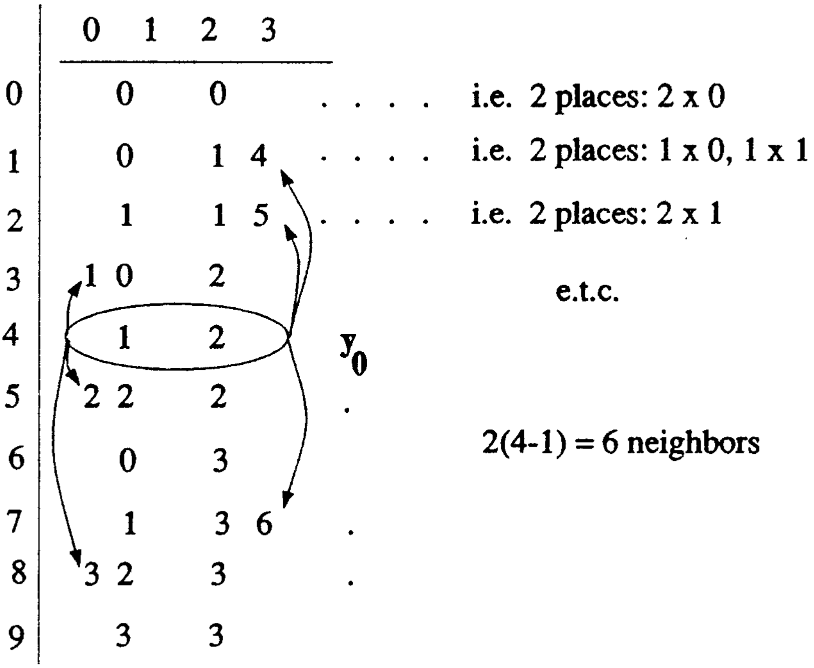


Figure A.2 The flow graph of the "stepping process"

The flow of the stepping process is schematically given in figure A.2. If H_1 is not a singleton we must repeat the proposed algorithm once for each element of H_1 resulting to the decision regions R_1^i, $i = 0, \cdots, k$. The decision region R_l will be the union of them $R_1 = R_1^0 \cup R_1^2 \cup \cdots \cup R_1^k$. An one step instance for the special case of M=4, n=2 is given in the figure A.3.



	0	1	2	3
0		0	0	
1		0	1	4
2		1	1	5
3	1	0	2	
4		1	2	
5	2	2	2	
6		0	3	
7		1	3	6
8	3	2	3	
9		3	3	

. . . . i.e. 2 places: 2 x 0

. . . . i.e. 2 places: 1 x 0, 1 x 1

. . . . i.e. 2 places: 2 x 1

e.t.c.

2(4-1) = 6 neighbors

Figure A.3 An example for the "stepping process"

Let us define N^1 to be the set of closest neighbors of y_0. N^2 the union of the closest neighbor sets of all the elements in N^1 and similarly define the sets N^3, N^4, e.t.c. The ROC plots found in the present work are based on the performance evaluation of a sequence of tests with decision regions N^1, N^2, N^3, e.t.c. Note that the recursive nature of the algorithm presented implies vast memory requirements when the decision region R_I or N^i is relatively large (experimentally found that it should not contain more than 100 elements), that poses limitations to the set of tests for which the ROC curves can be obtained.

Appendix B Performance evaluation plots

The results given concern 64×64–pixel binary images. For the Gaussian noise case we used a 5×5–pixel full black template, while for the BSC noise case we used an 8×8–pixel (or equivalent coded) full black template except otherwise stated.

The compression rates in bits per pixel (bpp) for the tests we studied are given in table B.1

test	compression rate (bpp)
(2,64)	1
(2,16)	1/4
(2,4)	1/16
(4,2)	1/4
(16,4)	1/4
(3,6)	1

Table B.1 Compression rates for the tests studied

A list of the plots is given below. ROC stands for Receiver Operating Characteristic curve, KB stands for "known background", and UN stands for "unknown background". Most of the results presented are based on the analysis described in chapters 2 and 4. The simulation results are produced by one Monte

Carlo run for which the optimal thresholds found by the theoretical analysis are assumed to be known.

□ uncoded image data

Plot 1: Gaussian noise; $P_d(\sigma^2)$-$P_{fa}(\sigma^2)$ curve; KB;

Plot 2: Gaussian noise; ROC, σ^2=0.1; KB;

Plot 3: BSC noise; (2,64)-KB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

Plot 4: BSC noise; (2,64)-KB test; ROC, ϵ=0.1;

□ the corrupt-code-detect system

Plot 5: (2,16)-KB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

Plot 6: (2,16)-KB test; ROC, ϵ=0.1;

Plot 7: (2,4)-KB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve; equivalently: (16,4)-KB test;

Plot 8: (2,4)-KB test; ROC, ϵ=0.1; equivalently: (16,4)-KB test;

Plot 9: (4,2)-UB test; synthetic template (see fig. 5.1); ROC, ϵ=0.05, ϵ=0.1, ϵ=0.2, ϵ=0.3, ϵ=0.4;

Plot 10: (16,4)-UB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

□ comparative plots

Plot 11: (2,64), (2,16)-KB tests; ROC, ϵ=0.1;

Plot 12: (2,64), (2,16)-KB tests; ROC, ϵ=0.3;

Plot 13: (2,64), (2,16), (2,4)-KB (equiv. (16,4)-KB), (16,4)-UB tests; ϵ—P_{fa} curve for optimal Bayesian tests;

Plot 14: (2,64), (2,16), (2,4)-KB (equiv. (16,4)-KB), (16,4)-UB tests; ϵ—P_d curve for optimal Bayesian tests;

□ the code-corrupt-detect system

Plot 15: (2,16)-KB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

Plot 16: (2,16)-KB test; ROC, ϵ=0.1;

Plot 17: (2,4)-KB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

Plot 18: (2,4)-KB test; ROC, ϵ=0.1;

Plot 19: (16,4)-KB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

Plot 20: (16,4)-KB test; ROC, ϵ=0.1;

Plot 21: (16,4)-UB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve;

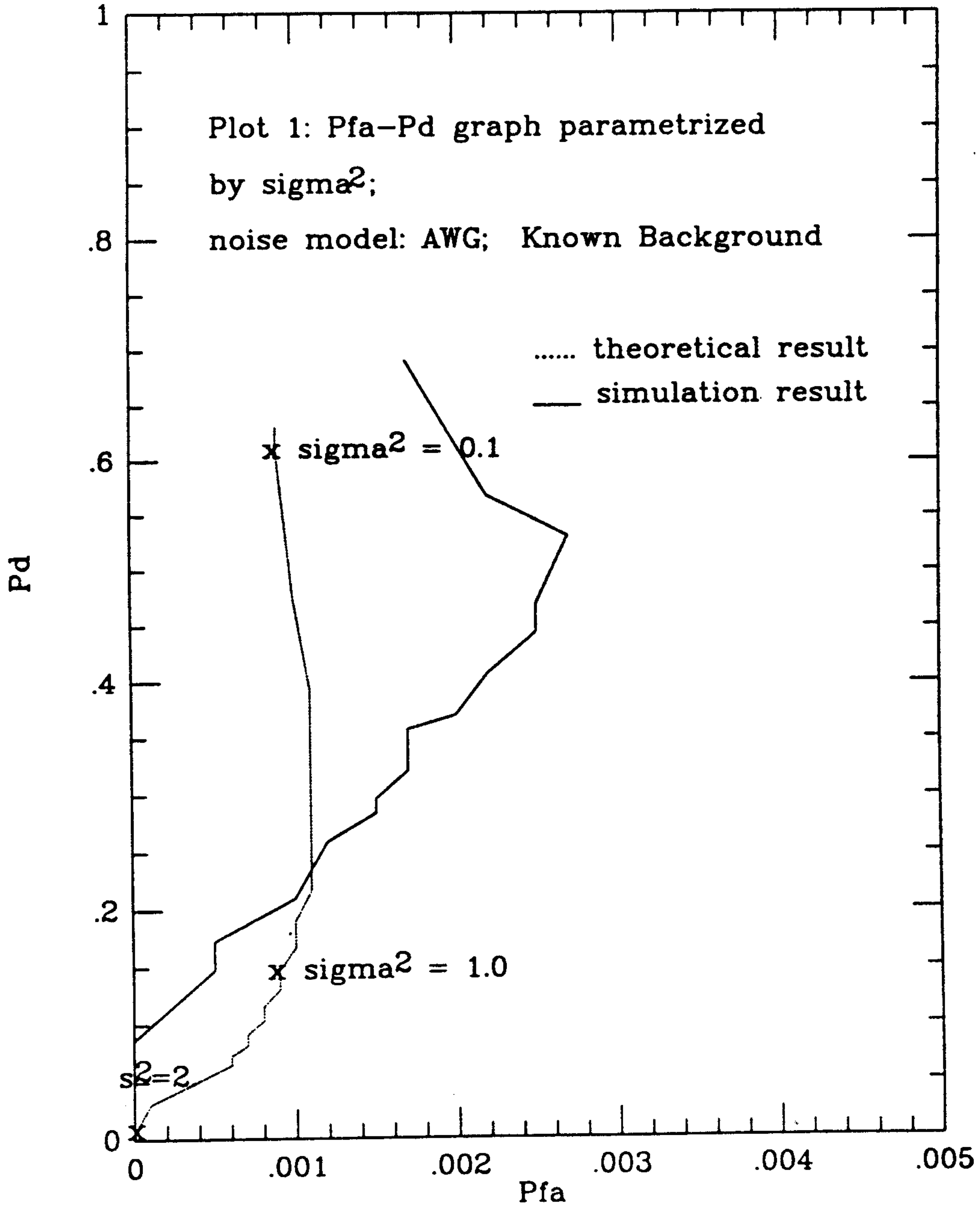
□ comparative plots

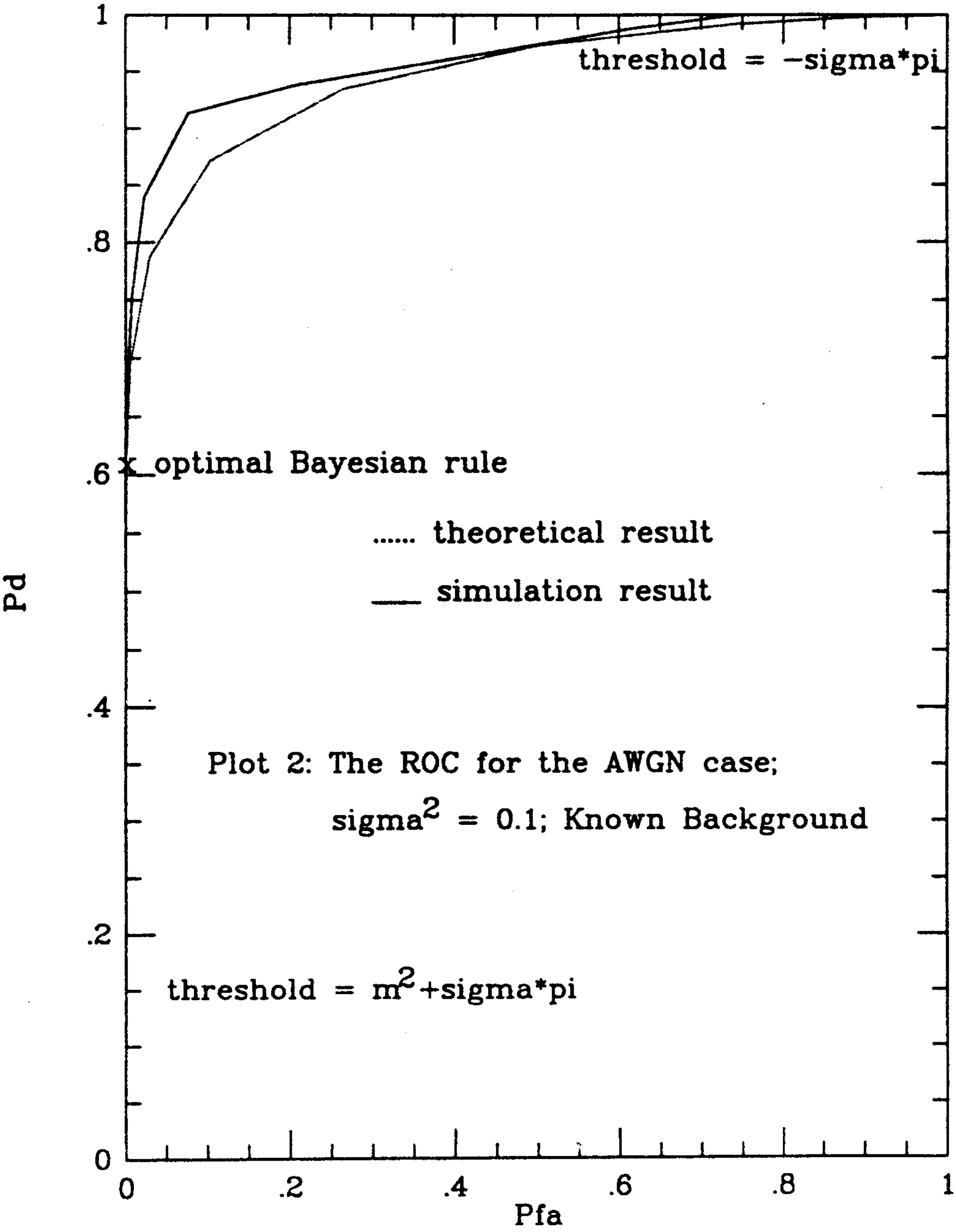
Plot 22: (2,64), (2,16), (2,4)-KB (equiv. (16,4)-KB), (16,4)-UB tests; ϵ—P_{fa} curve for optimal Bayesian tests;

Plot 23: (2,64), (2,16), (2,4)-KB (equiv. (16,4)-KB), (16,4)-UB tests; ϵ—P_d curve for optimal Bayesian tests;

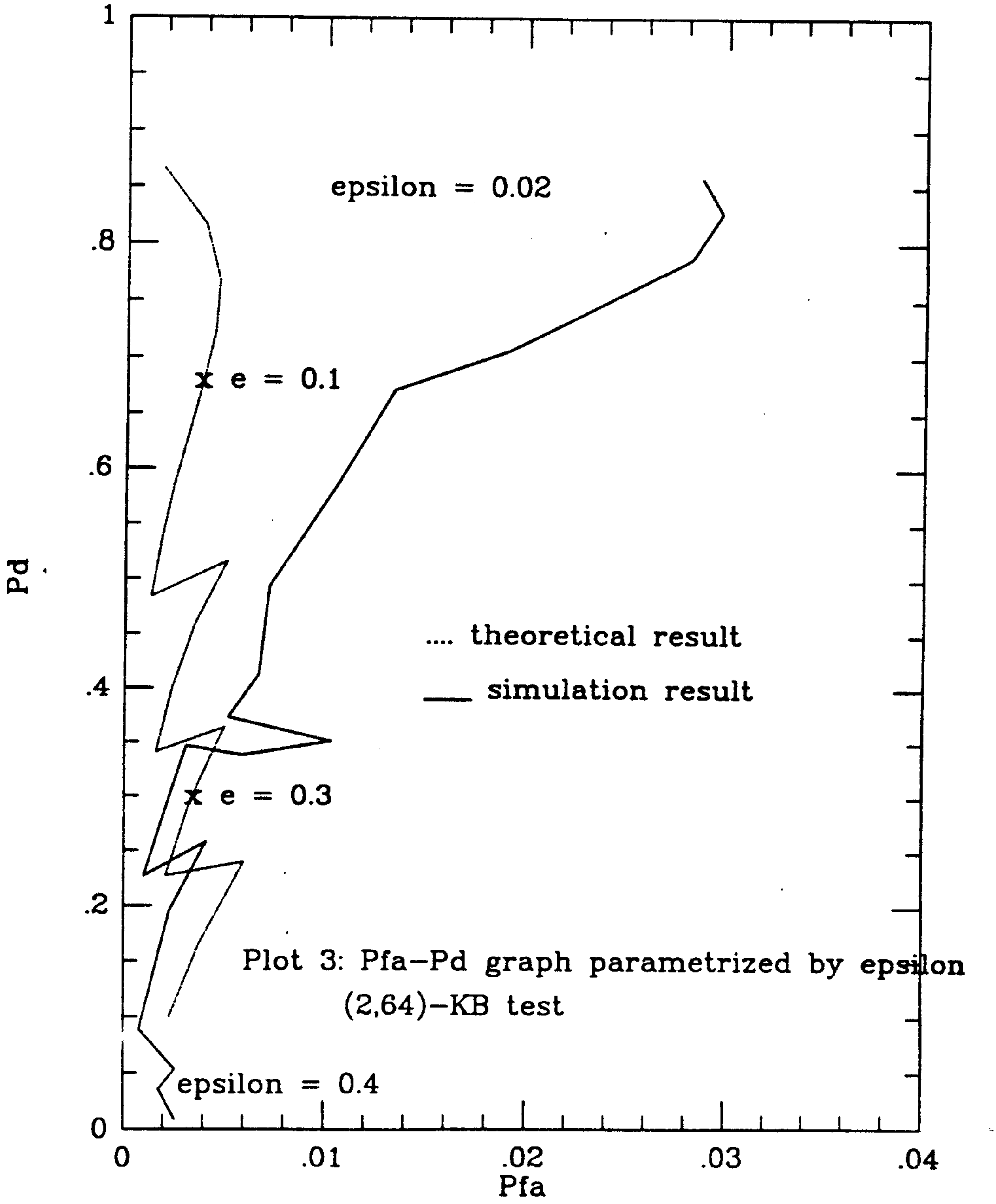
□ multiple gray level pixels

Plot 24: (3,6)-UB test; $P_d(\epsilon)$-$P_{fa}(\epsilon)$ curve.

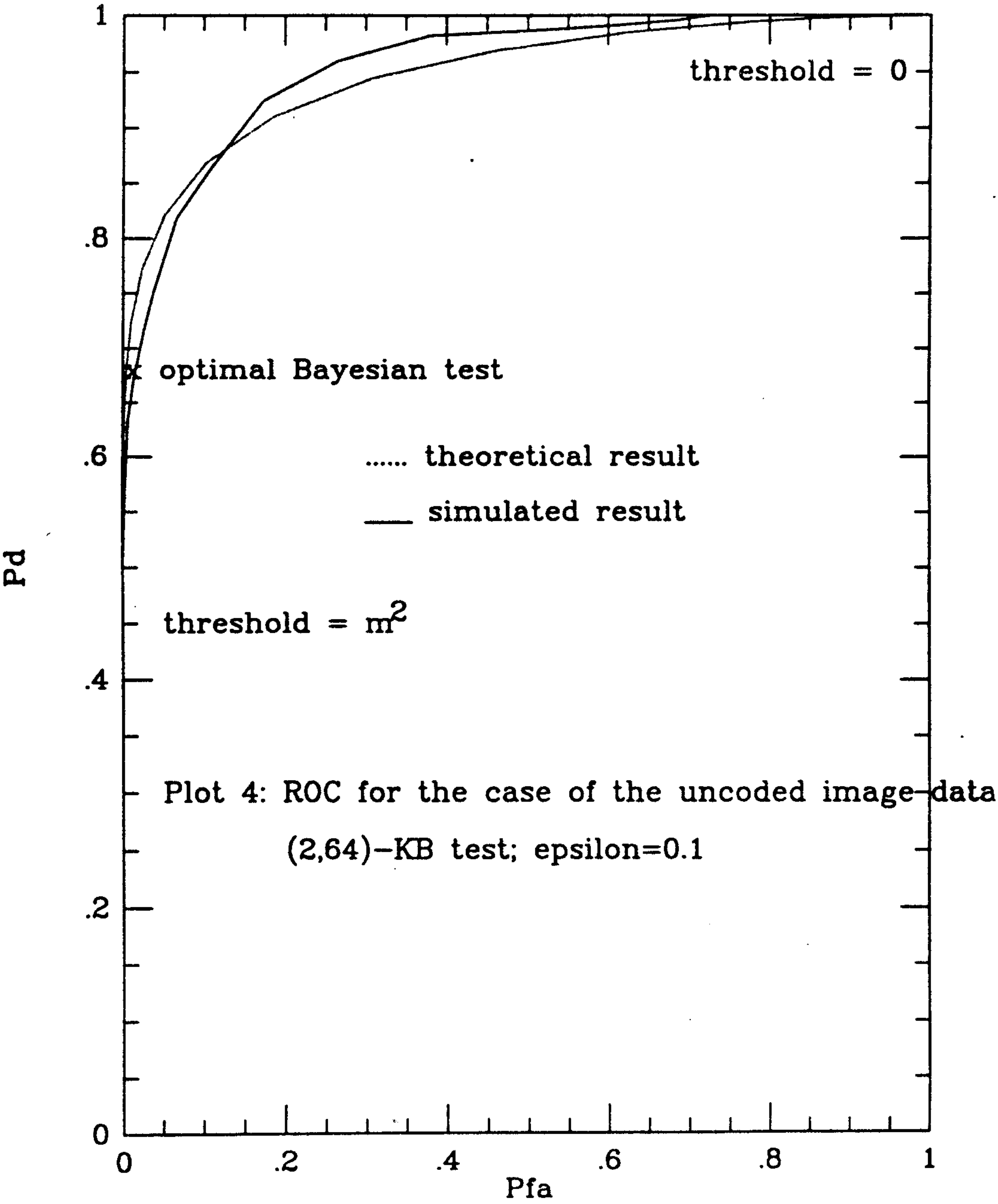

Plot 1: Pfa–Pd graph parametrized
by sigma2;
noise model: AWG; Known Background
...... theoretical result
___ simulation result
x sigma2 = 0.1
x sigma2 = 1.0
s2=2
Pd
Pfa
1
.8
.6
.4
.2
0
0
.001
.002
.003
.004
.005



Plot 2: The ROC for the AWGN case; $\text{sigma}^2 = 0.1$; Known Background

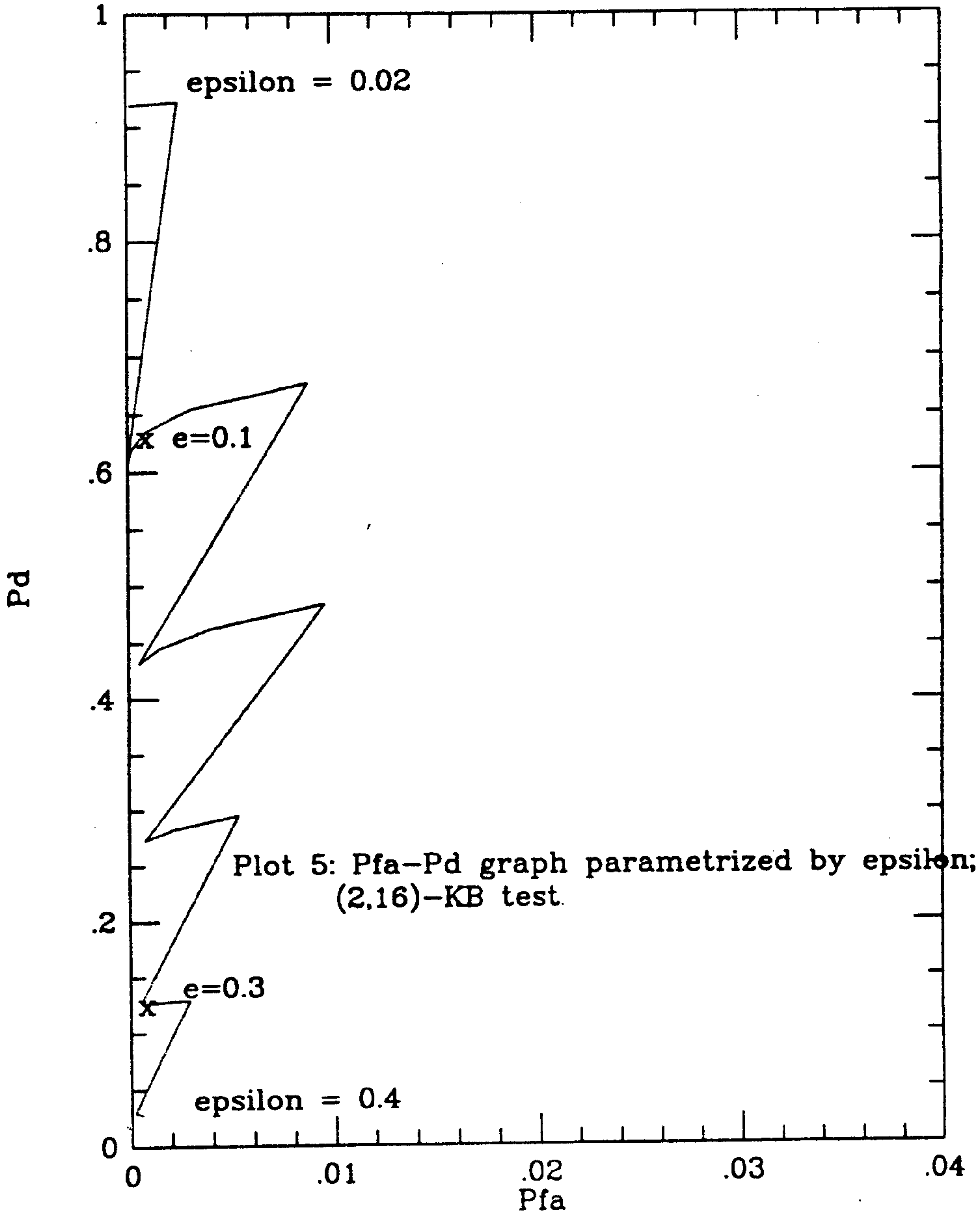


Plot 3: Pfa–Pd graph parametrized by epsilon
(2,64)–KB test

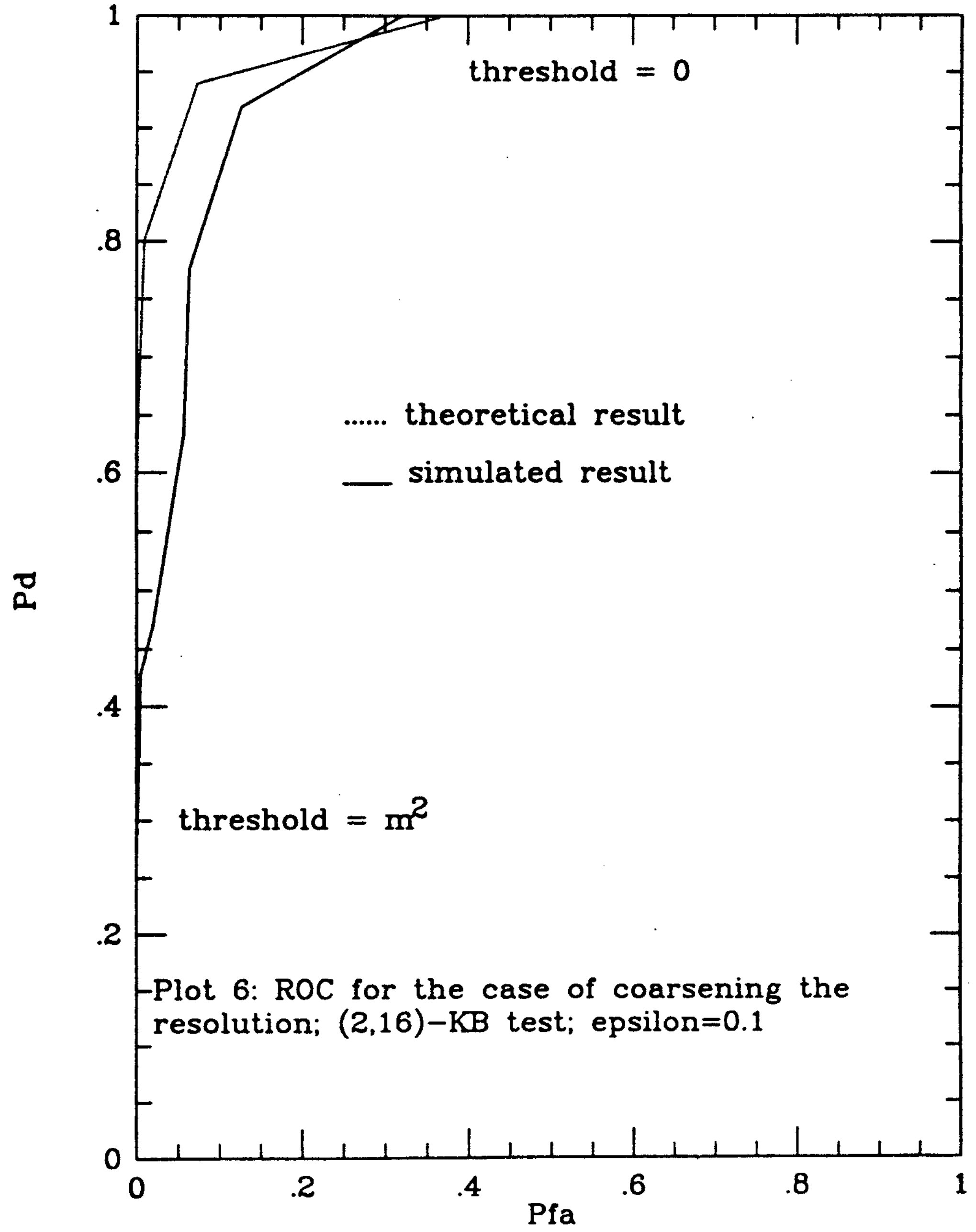


Plot 4: ROC for the case of the uncoded image data

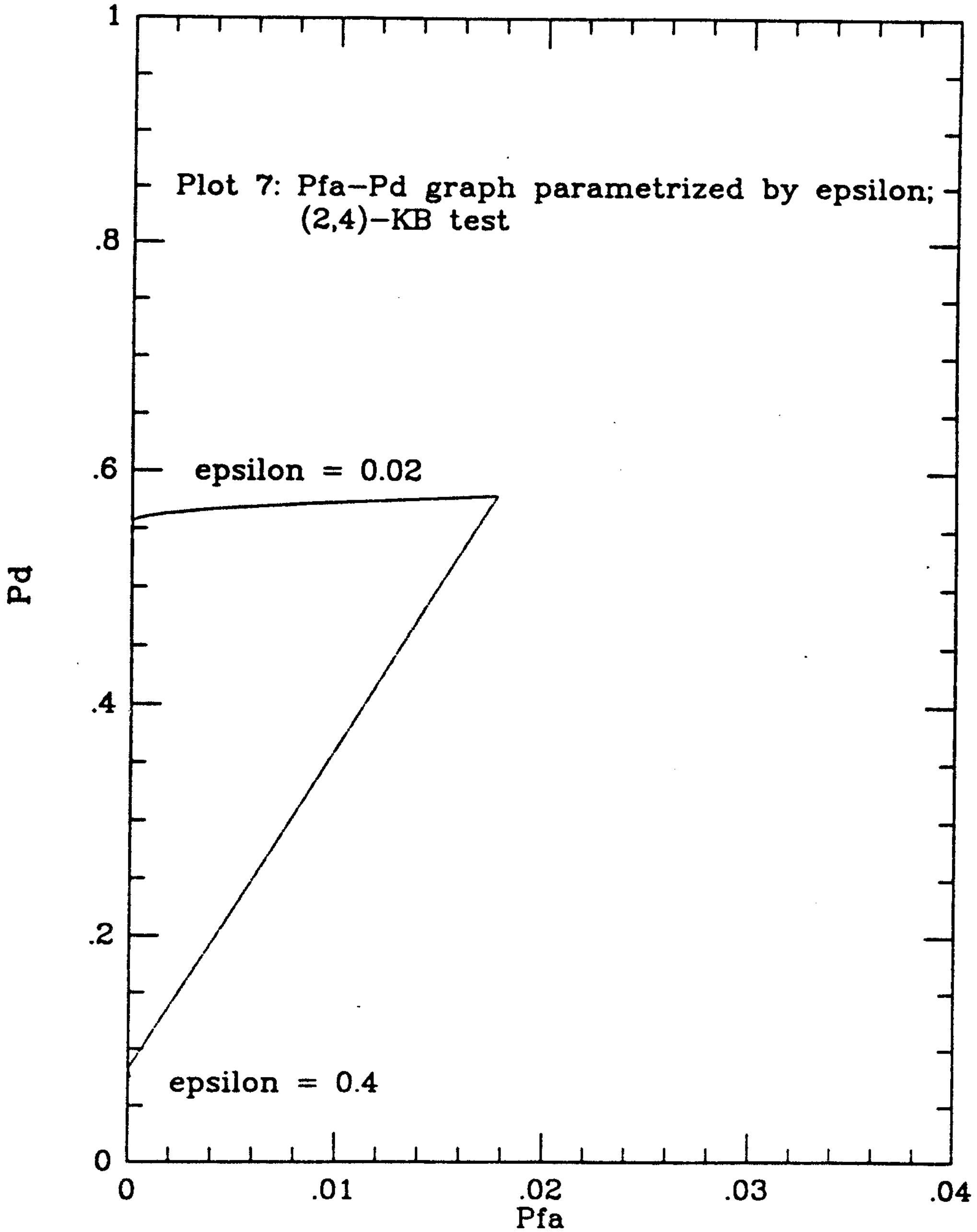
(2,64)–KB test; epsilon=0.1

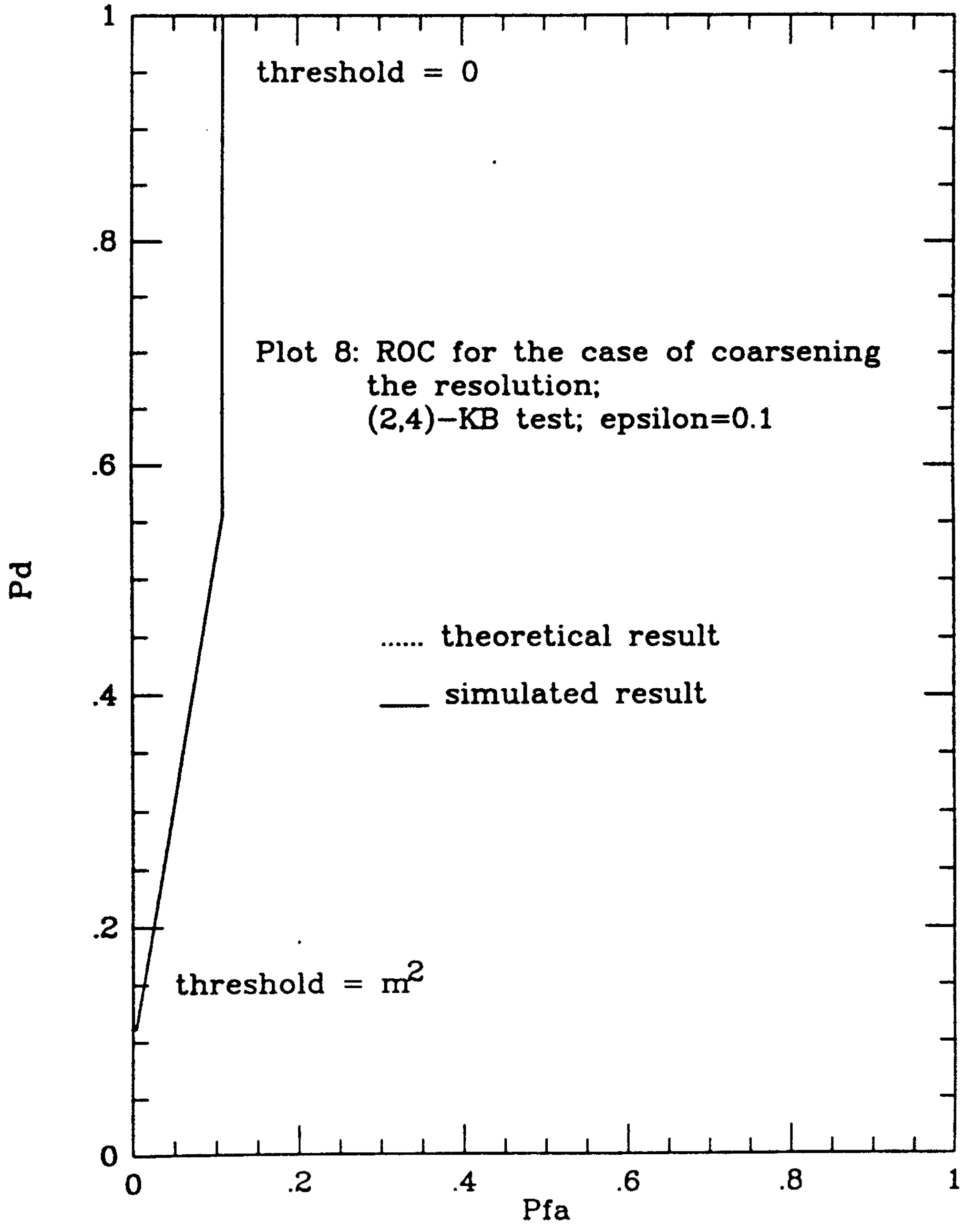


Plot 5: Pfa–Pd graph parametrized by epsilon; (2,16)–KB test.

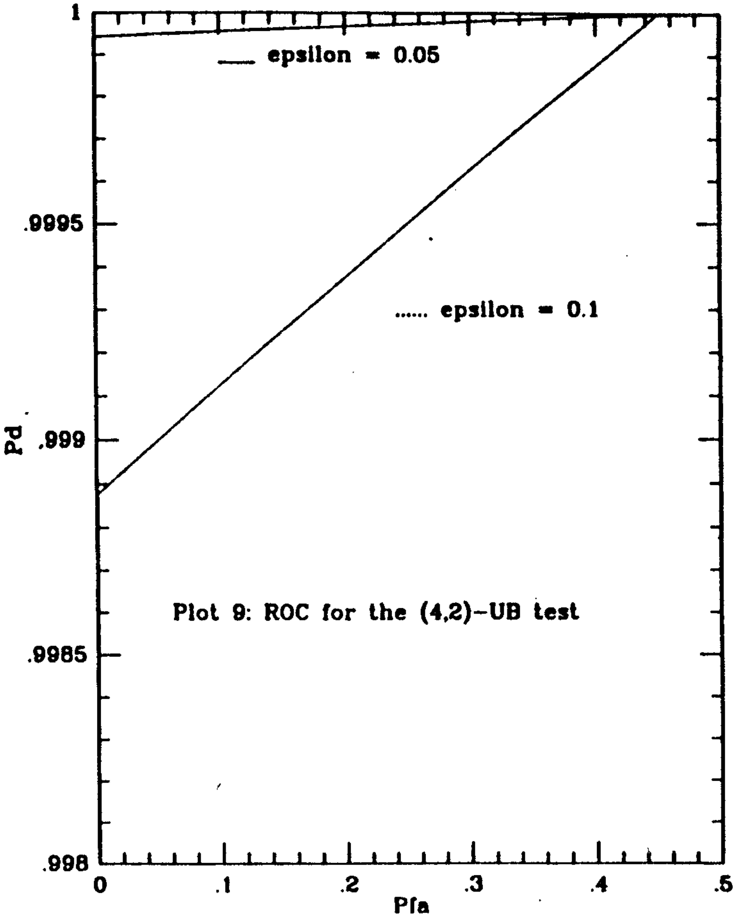


Plot 6: ROC for the case of coarsening the resolution; (2,16)–KB test; epsilon=0.1

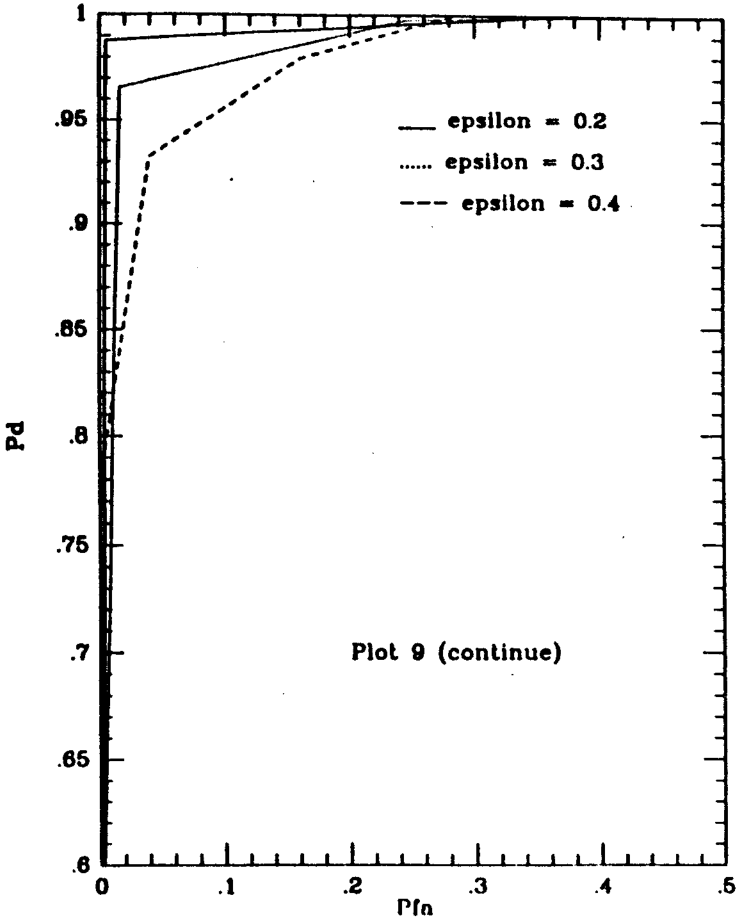

Plot 7: Pfa–Pd graph parametrized by epsilon;
(2,4)–KB test
epsilon = 0.02
epsilon = 0.4
Pd
Pfa
1
.8
.6
.4
.2
0
0
.01
.02
.03
.04



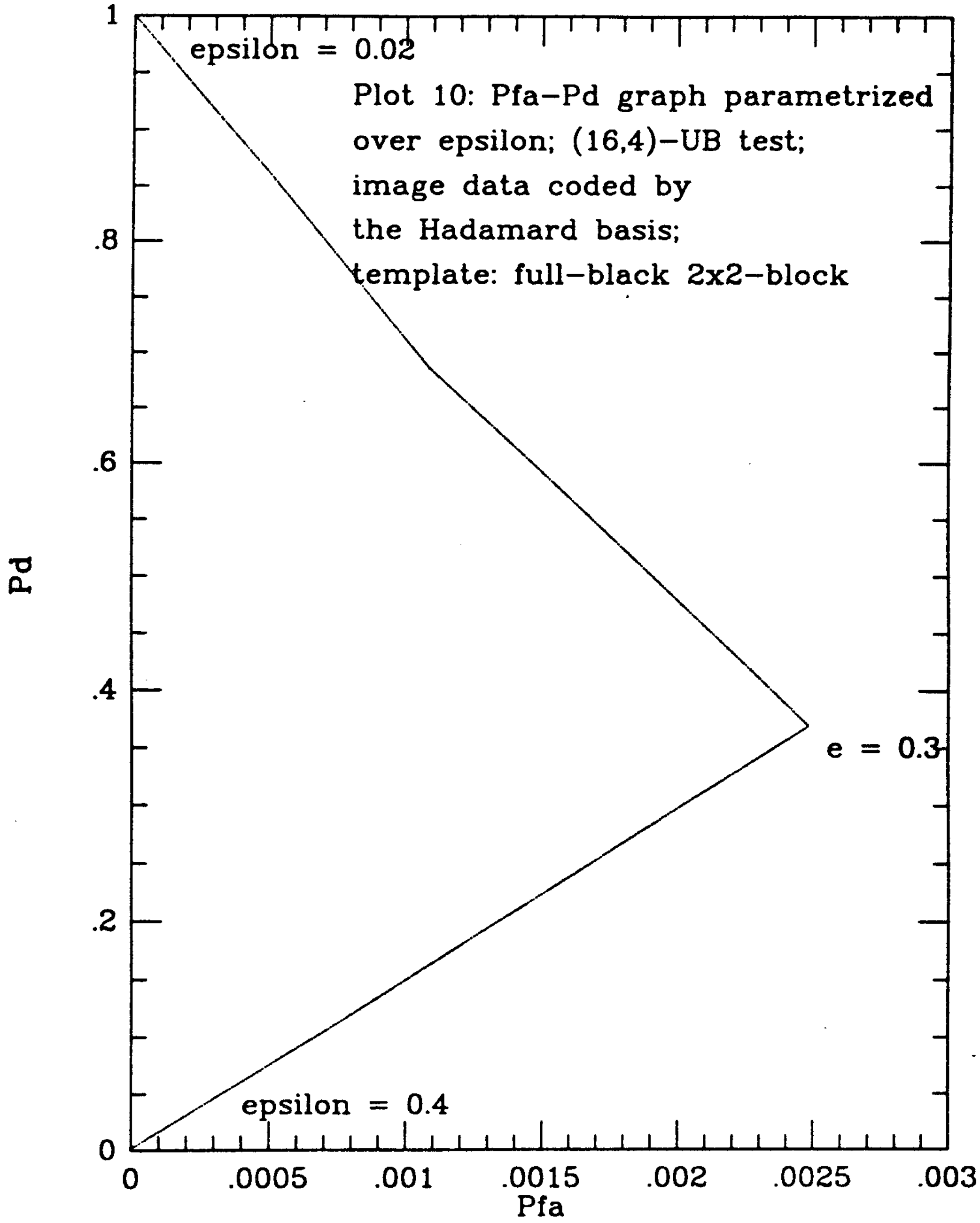
Plot 8: ROC for the case of coarsening the resolution; (2,4)–KB test; epsilon=0.1



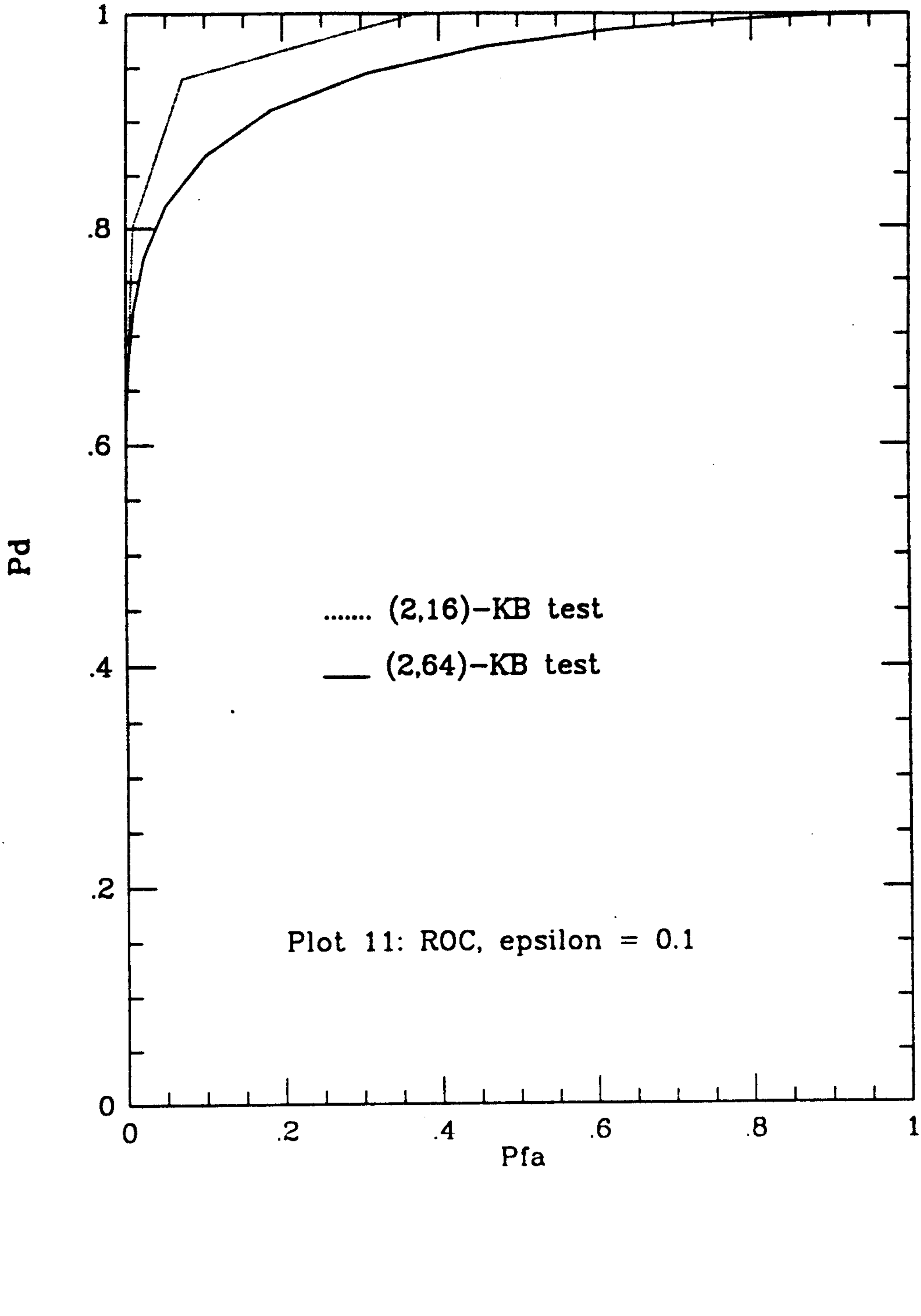
Plot 9: ROC for the (4,2)-UB test



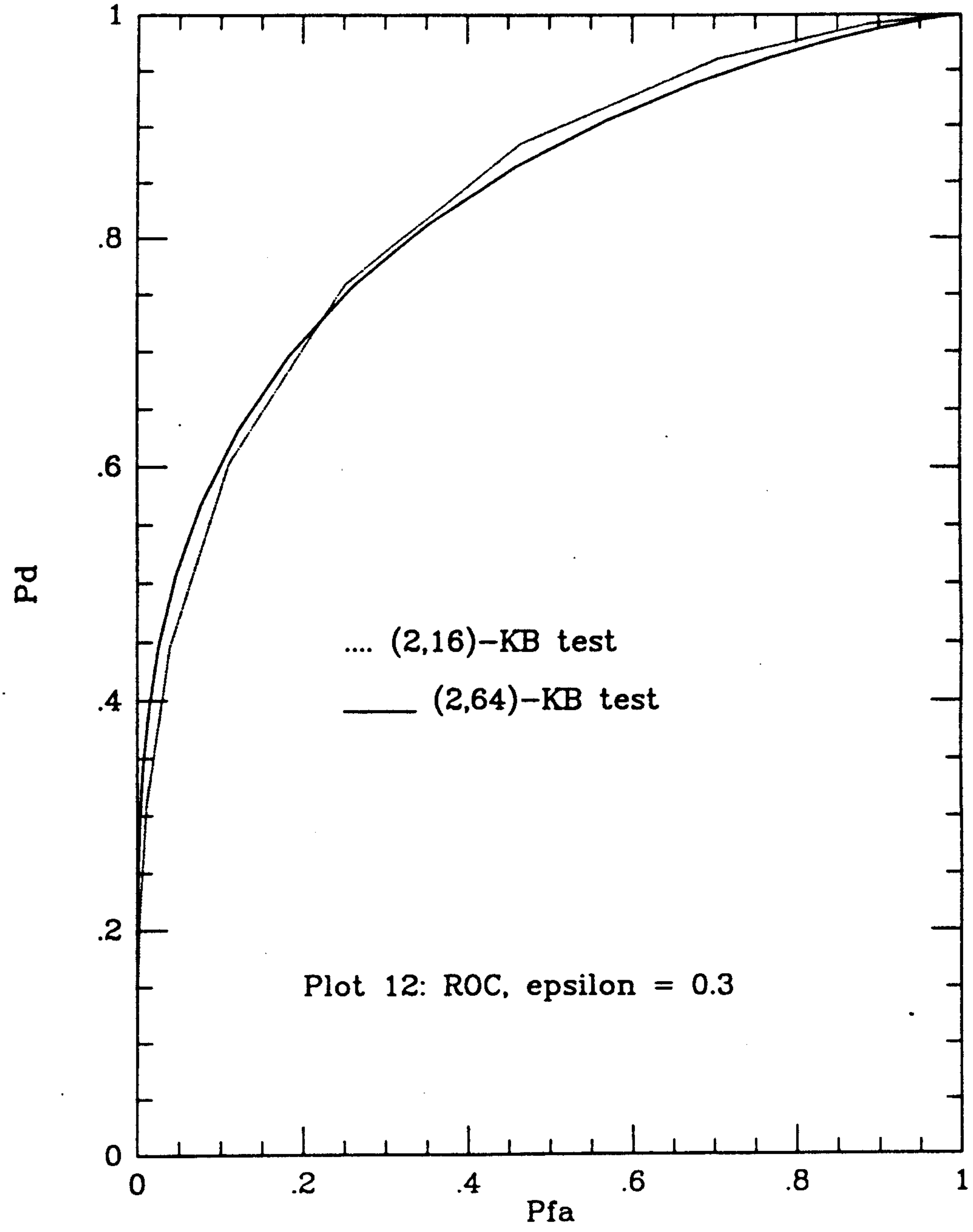
Plot 9 (continue)

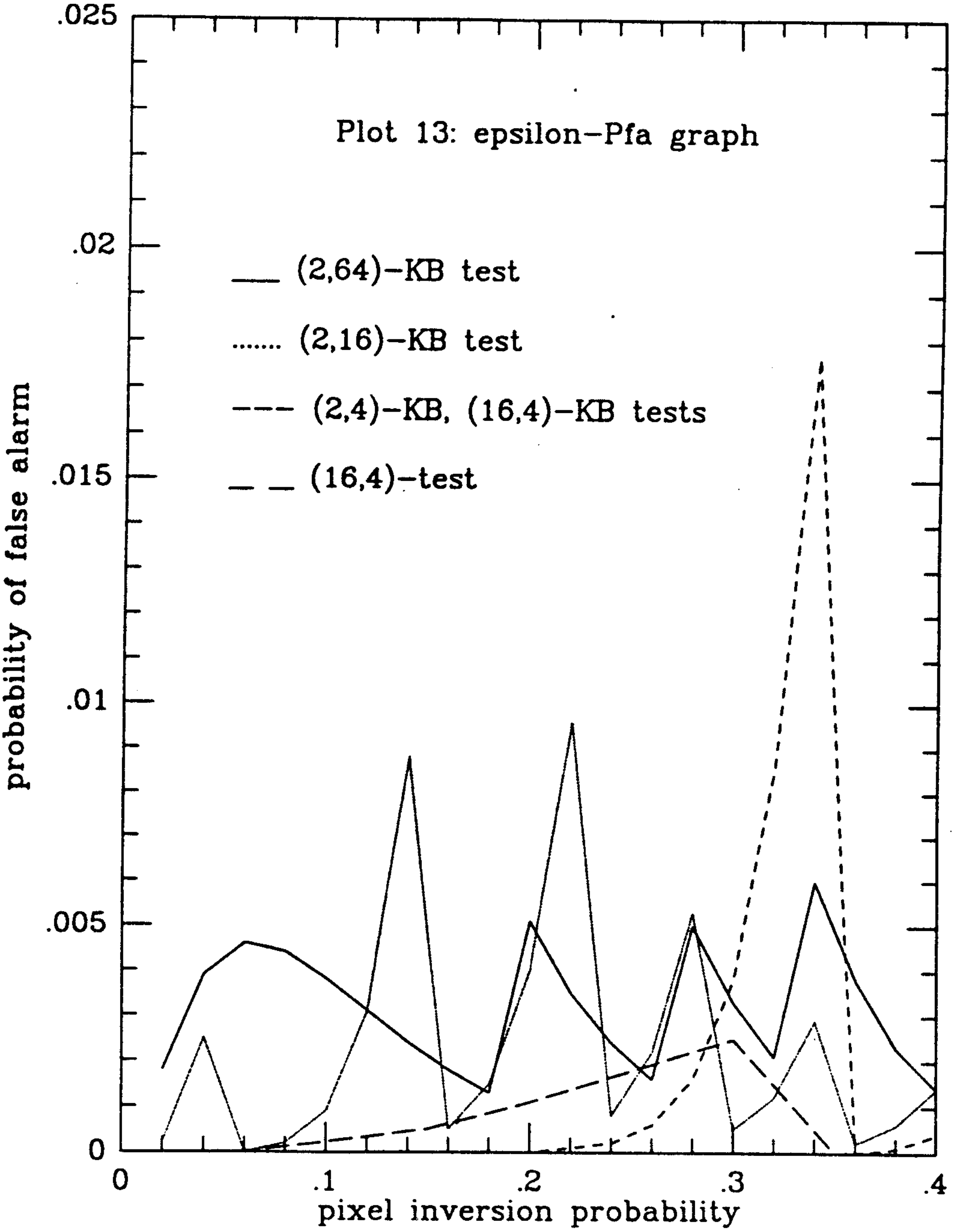


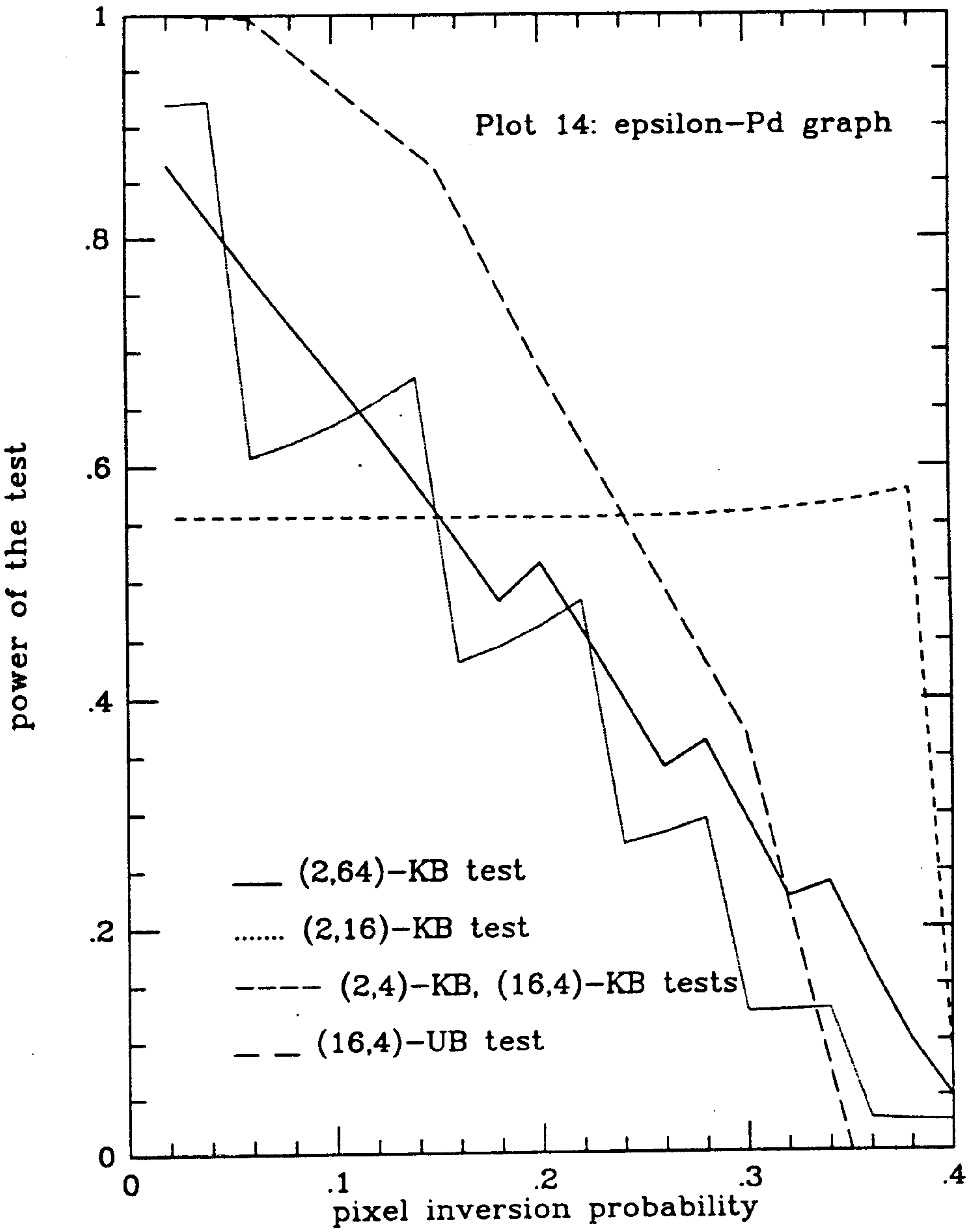
Plot 10: Pfa–Pd graph parametrized over epsilon; (16,4)–UB test; image data coded by the Hadamard basis; template: full–black 2x2–block

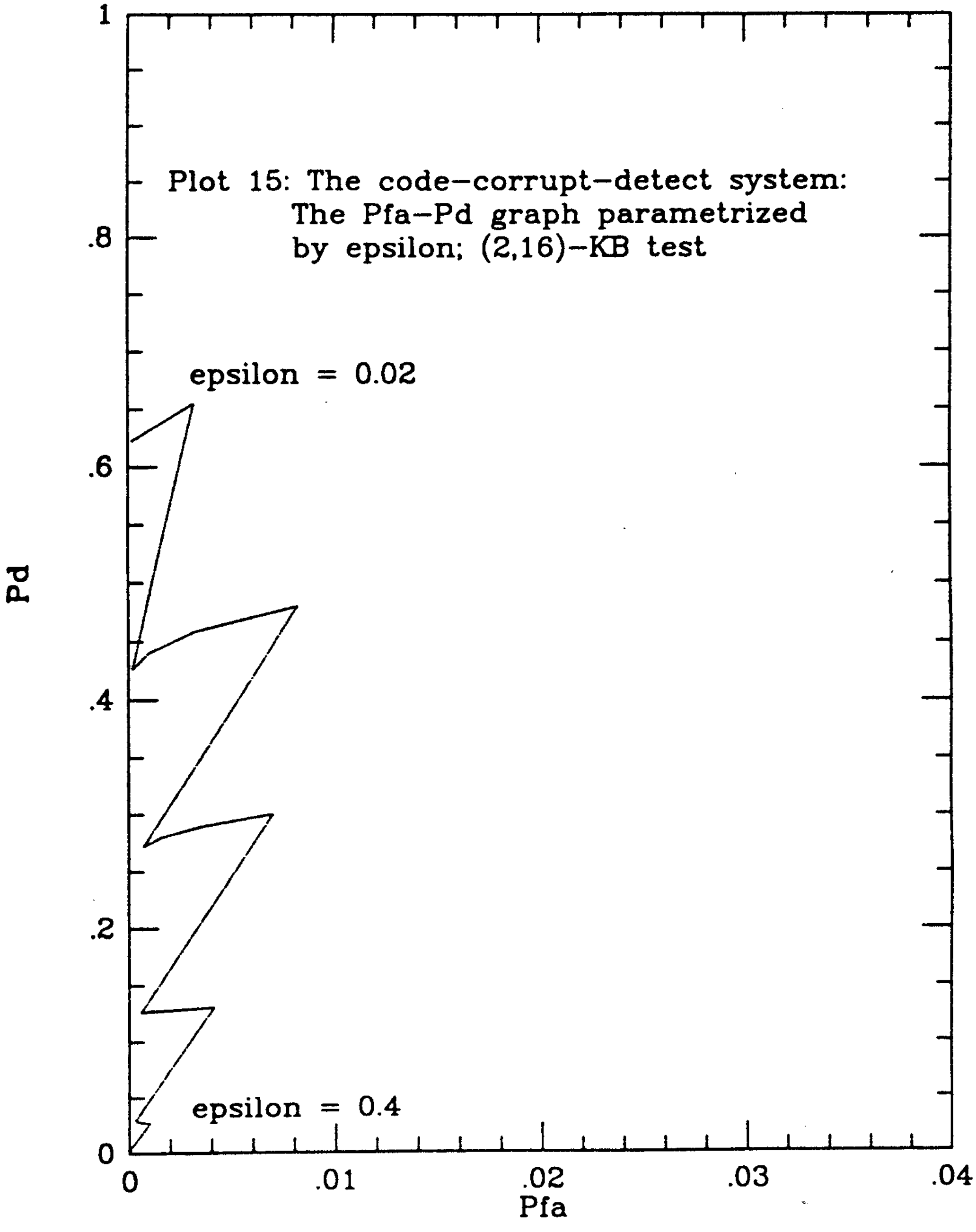


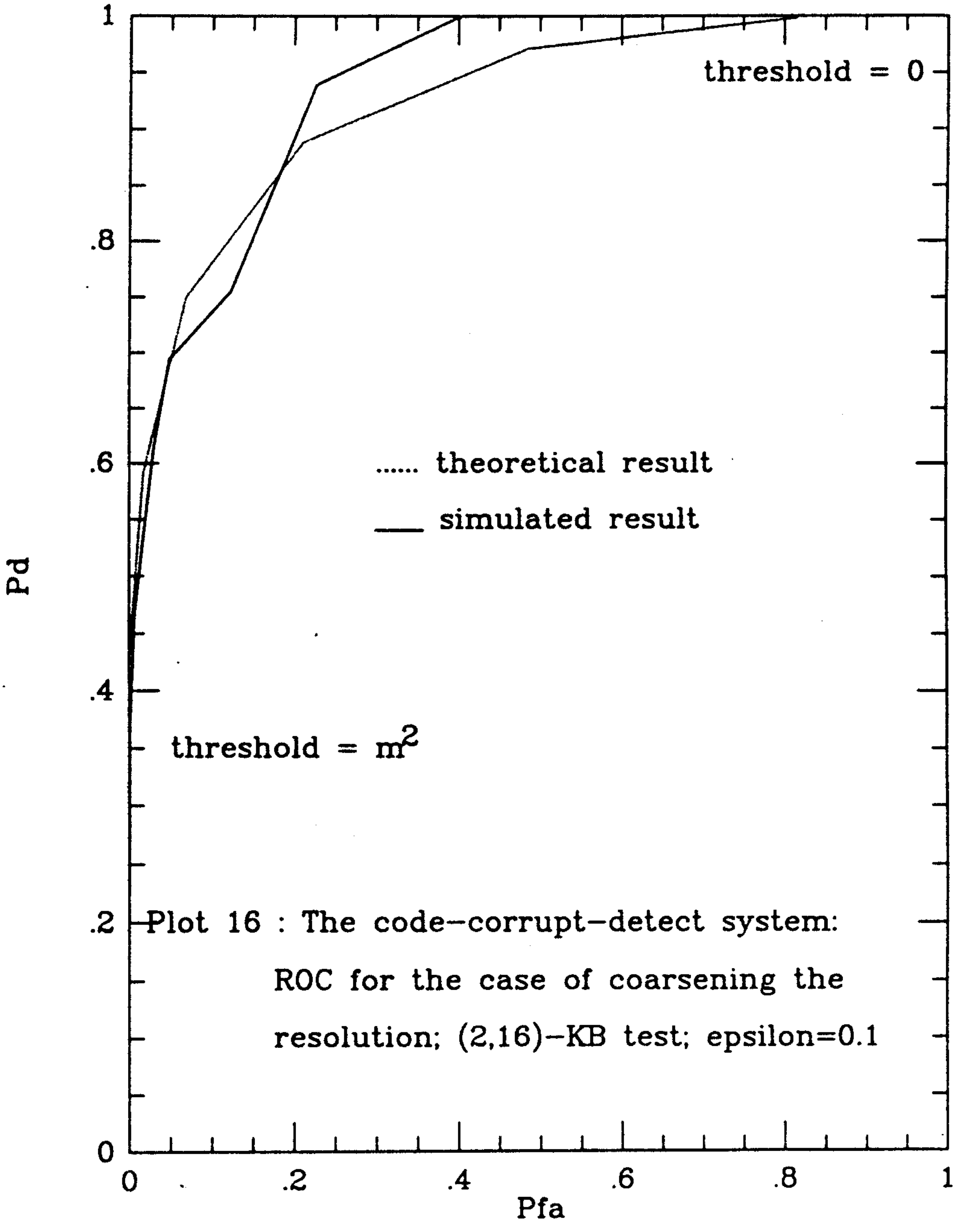
Pd
1
.8
.6
.4
.2
0
0
.2
.4
.6
.8
1
Pfa
.... (2,16)-KB test
___ (2,64)-KB test
Plot 11: ROC, epsilon = 0.1


1
.8
.6
.4
.2
0
Pd
0
.2
.4
.6
.8
1
Pfa
.... (2,16)-KB test
——— (2,64)-KB test
Plot 12: ROC, epsilon = 0.3

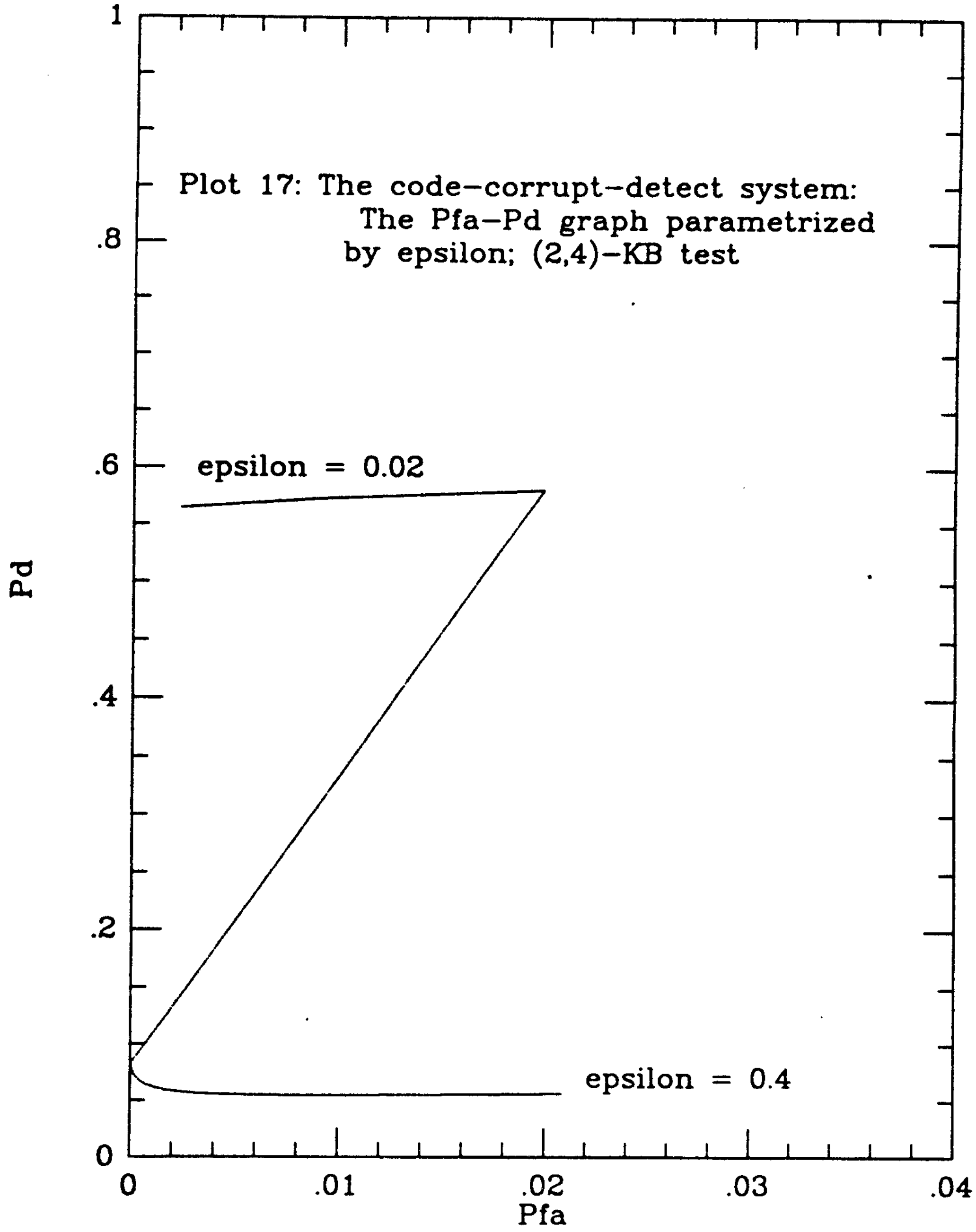

Plot 13: epsilon–Pfa graph
___ (2,64)–KB test
.... (2,16)–KB test
--- (2,4)–KB, (16,4)–KB tests
_ _ (16,4)–test
probability of false alarm
pixel inversion probability
.025
.02
.015
.01
.005
0
0
.1
.2
.3
.4


Plot 14: epsilon–Pd graph
power of the test
pixel inversion probability
1
.8
.6
.4
.2
0
0
.1
.2
.3
.4
(2,64)–KB test
(2,16)–KB test
(2,4)–KB, (16,4)–KB tests
(16,4)–UB test

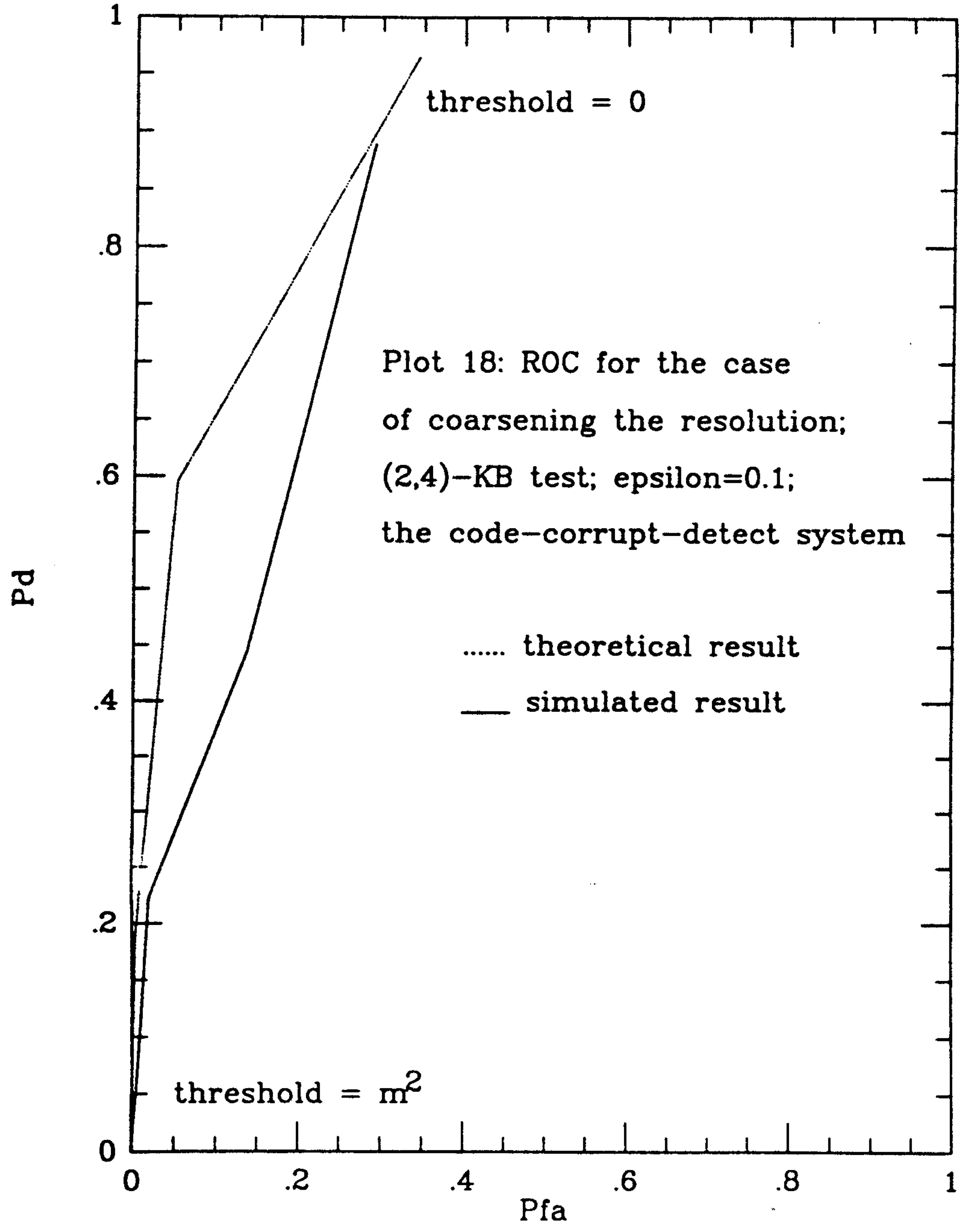
Plot 15: The code−corrupt−detect system:
The Pfa−Pd graph parametrized
by epsilon; (2,16)−KB test
epsilon = 0.02
epsilon = 0.4
Pd
Pfa
1
.8
.6
.4
.2
0
0
.01
.02
.03
.04



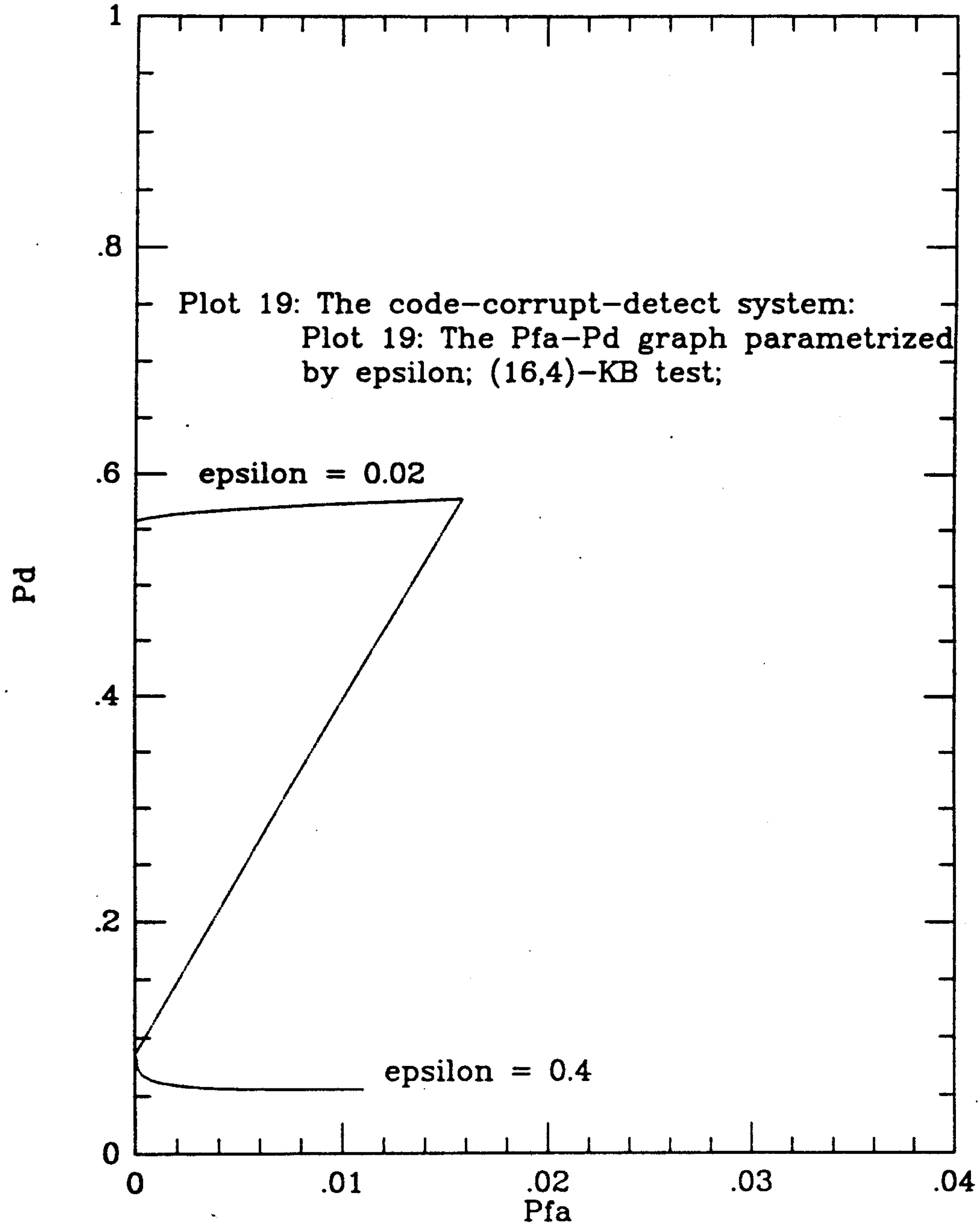
Plot 16 : The code–corrupt–detect system: ROC for the case of coarsening the resolution; (2,16)–KB test; epsilon=0.1

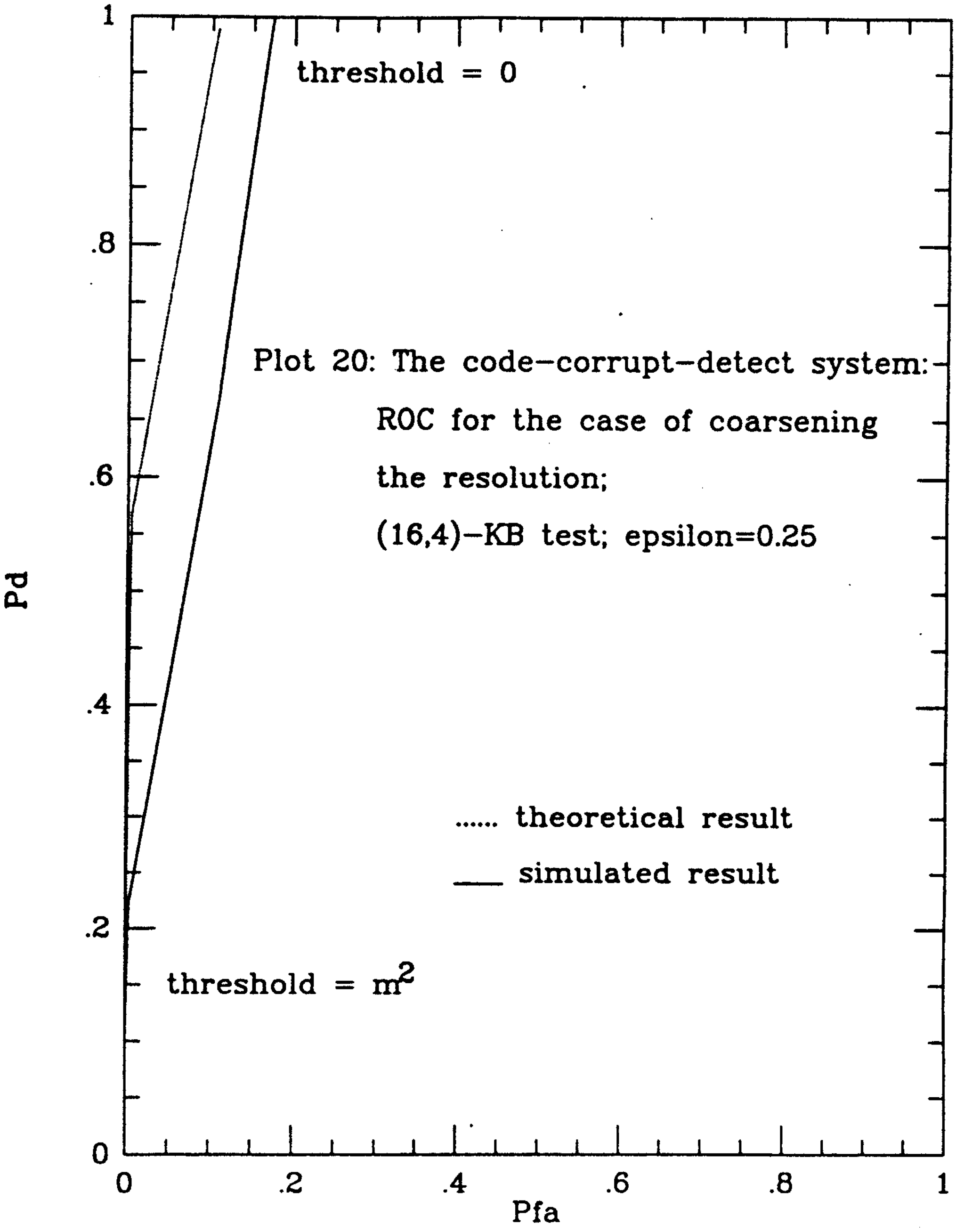


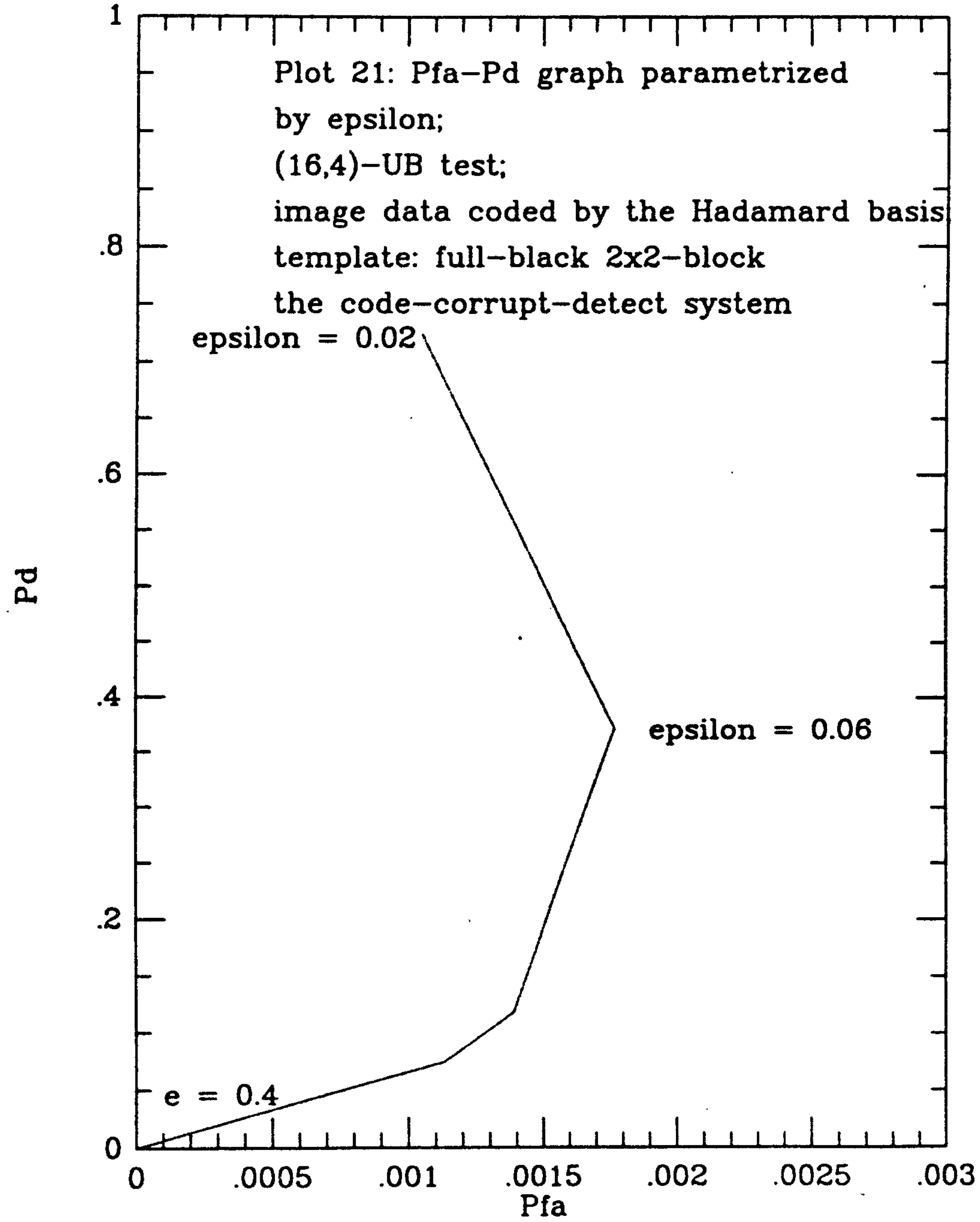
Plot 17: The code-corrupt-detect system:
The Pfa-Pd graph parametrized
by epsilon; (2,4)-KB test

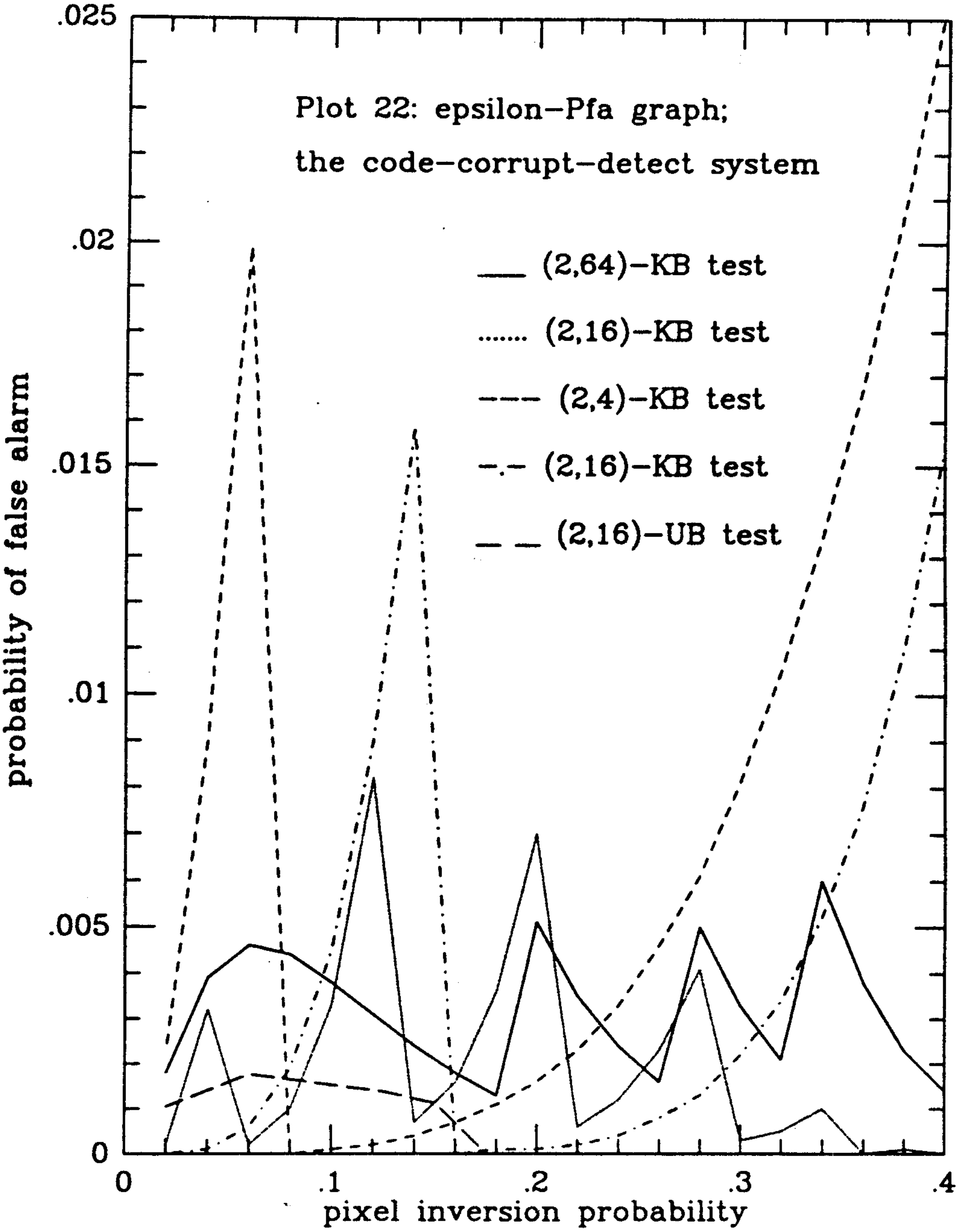


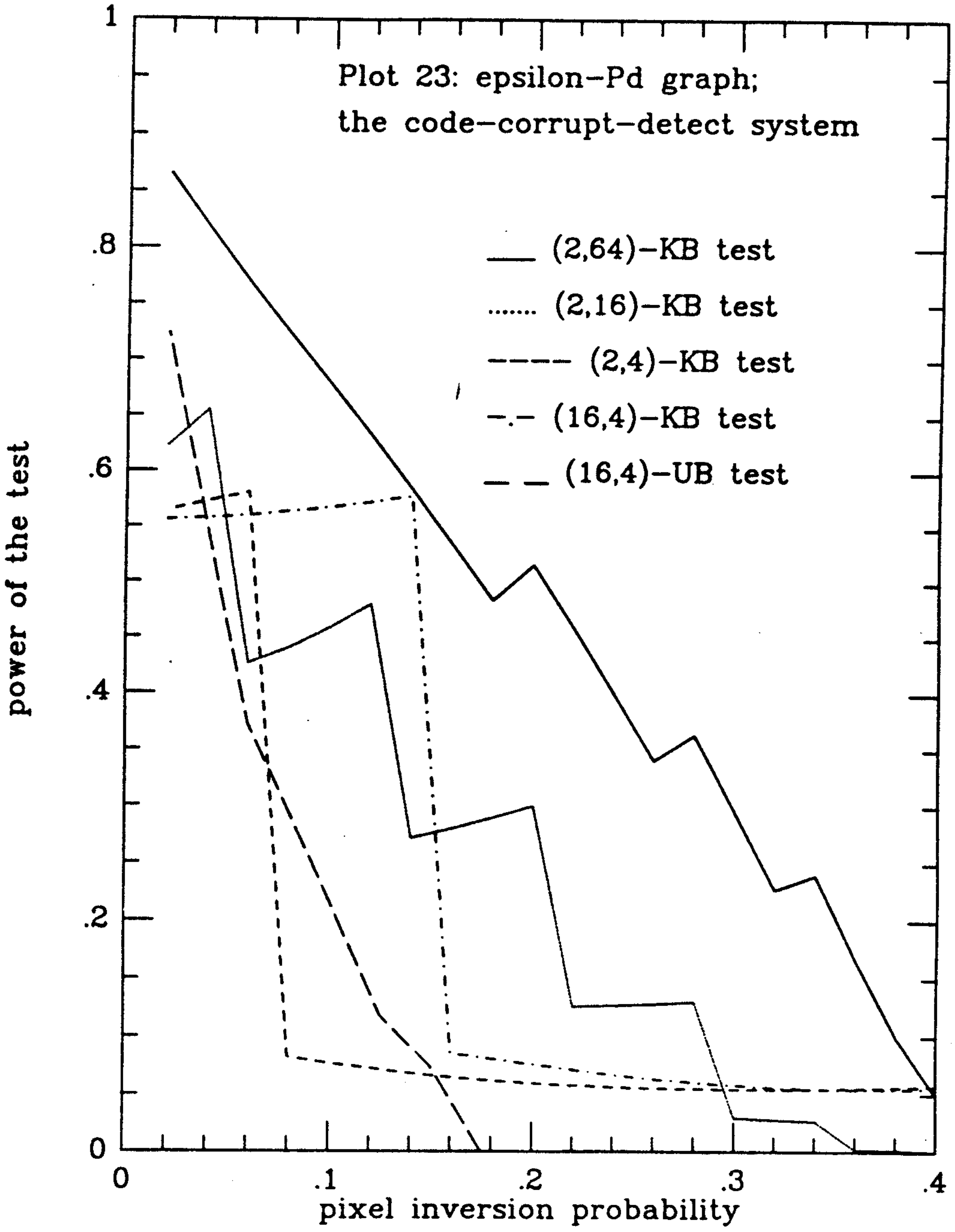
Plot 18: ROC for the case of coarsening the resolution; (2,4)–KB test; epsilon=0.1; the code–corrupt–detect system

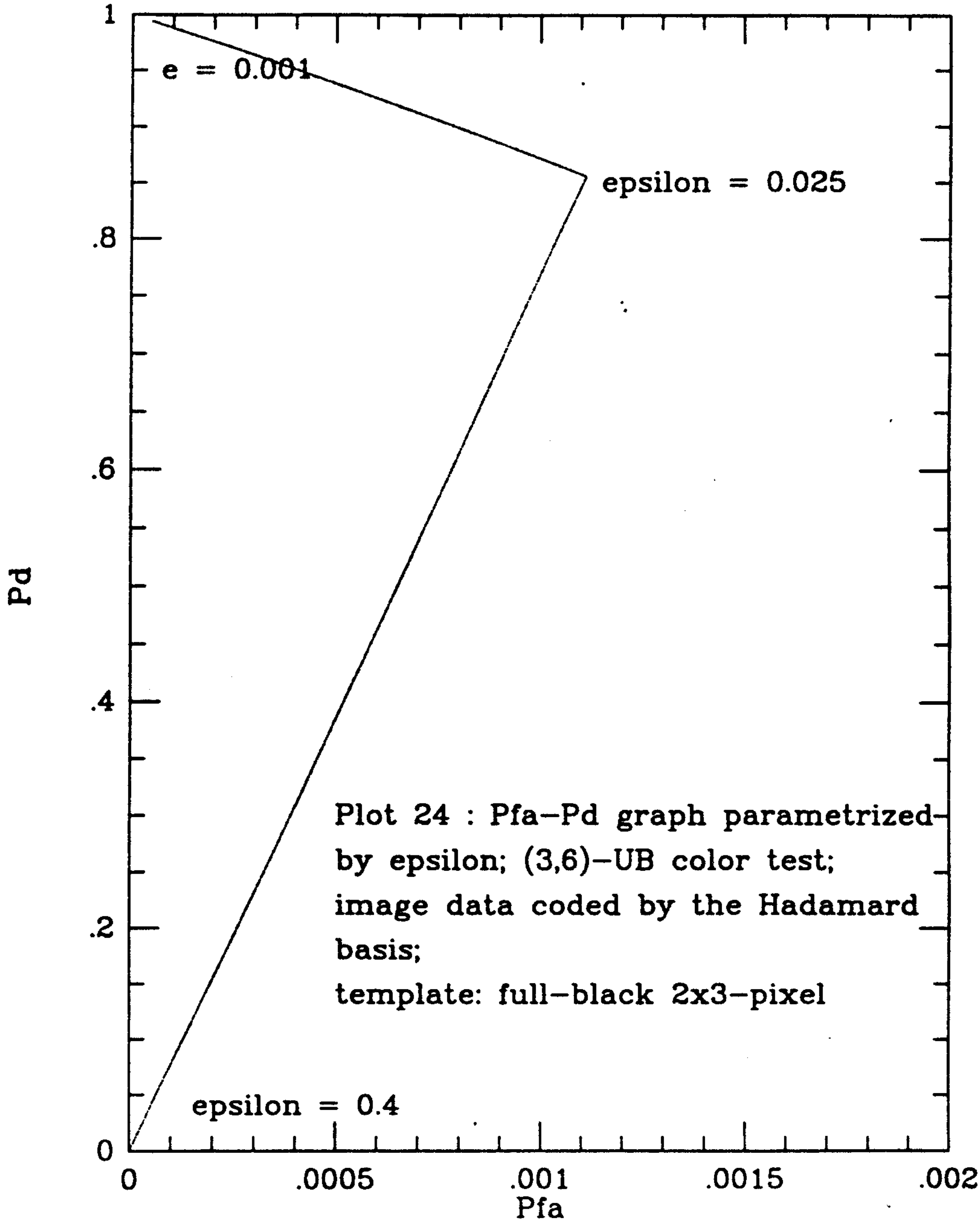

Plot 19: The code−corrupt−detect system:
Plot 19: The Pfa−Pd graph parametrized by epsilon; (16,4)−KB test;
epsilon = 0.02
epsilon = 0.4
Pd
Pfa
1
.8
.6
.4
.2
0
0
.01
.02
.03
.04

threshold = 0
Plot 20: The code−corrupt−detect system:
ROC for the case of coarsening
the resolution;
(16,4)−KB test; epsilon=0.25
...... theoretical result
___ simulated result
threshold = m^2
Pd
Pfa
0
.2
.4
.6
.8
1
0
.2
.4
.6
.8
1


Plot 21: Pfa–Pd graph parametrized
by epsilon;
(16,4)–UB test;
image data coded by the Hadamard basis
template: full–black 2x2–block
the code–corrupt–detect system
epsilon = 0.02
epsilon = 0.06
e = 0.4
Pd
Pfa
1
.8
.6
.4
.2
0
0
.0005
.001
.0015
.002
.0025
.003


Plot 22: epsilon–Pfa graph;
the code–corrupt–detect system
___ (2,64)–KB test
.... (2,16)–KB test
--- (2,4)–KB test
-.- (2,16)–KB test
_ _ (2,16)–UB test
probability of false alarm
.025
.02
.015
.01
.005
0
0
.1
.2
.3
.4
pixel inversion probability


Plot 23: epsilon–Pd graph;
the code–corrupt–detect system
—— (2,64)–KB test
.... (2,16)–KB test
---- (2,4)–KB test
–.– (16,4)–KB test
— — (16,4)–UB test
power of the test
pixel inversion probability
0
.2
.4
.6
.8
1
0
.1
.2
.3
.4



Plot 24 : Pfa–Pd graph parametrized by epsilon; (3,6)–UB color test; image data coded by the Hadamard basis;
template: full–black 2x3–pixel

Bibliography

[Abr87] K. Abrahamson, "Generalized String Matching", *SIAM*, vol.16, pp.1039–1051,1987.

[AmFa89] Amihood Amir, Martin Farach, "Efficient 2–D Approximate Matching of non Rectangular Figures", Manuscript, CS Dept, UMCP, Nov. 1989.

[ANR74] N.Ahmed, T. Natarajan, K. R. Rao, "On Image Processing and a Discrete Cosine Transform", *IEEE Trans. on Comput.*, vol.C-23 pp.90–93, Jan.1974.

[Bla87] Richard E. Blahut, "Principles and Practice of Information Theory", *Addison-Wesley*, 1987.

[BoMo77] R. S. Boyer, J. S. Moore, "A Fast String Searching Algorithm", *Comm. ACM*, vol.20, pp.762–772, 1977.

[Chr75] Richard E. Christ, "Review and Analysis of Color Coding Research for Visual Displays", *Human Factors*, vol.17, pp.542–570 , 1975

[DuHa73] Richard O. Duda, Peter E. Hart, "Pattern Classification and Scene Analysis", *J. Willey and sons*, 1973.

[FiPa74] M. J. Fisher, M. S. Patterson, "String Matching and other products", *Complexity of Computation, R. M. Karp (editor), SIAM-AMS Proceedings*, vol.7, pp.113–125, 1974.

[Gra84] Robert M. Gray, "Vector Quantization", *IEEE, ASSP Magazine*, Apr. 1984.

[HuSc63] J.J.Y.Huang, P.M.Schultheiss, "Block Quantization of Correlated Gaussian Random Variables", *IEEE Trans. on Communication Systems*, Sep. 1963.

[KBL87] M. Kunt, M. Benard, R. Leonardi, "Recent Results in High-Compression Image Coding" *IEEE Trans. on Circuits and Systems*, Nov. 1987.

[KIK85] M. Kunt, A. Ikonomopoulos, M. Kocher, "Second Generation Image-Coding Techniques", *Proc. of the IEEE*, Apr. 1985.

[KMP77] D.E. Knuth, J.H. Morris, V.R. Pratt, "Fast Pattern Matching in Strings", *SIAM J.Comp.* vol.6, pp.323–350, 1977.

[Knu75] D.R. Knudson, "Digital Encoding of Newspaper Graphics", *Electron. Syst. Lab., MIT, Rep. ESL-616*, Aug. 1975.

[LaSl71] H.J.Landau, D. Slepian, "Some Computer Experiments in Picture Processing for Bandwidth Reduction", *The Bell System Technical Journal*, vol.50, pp.1525–1540, May 1971.

[MAY82] T. Murakami, K. Asai, E.Yamazaki, "Vector Quantizer of Video Signals", *Electronic Letters*, vol.18, pp.1005–1006, Nov. 1982.

[Max60] Joel Max, "Quantizing for Minimum Distortion", *IRE IEEE Trans. on Information Theory*, Mar. 1960.

[MRG85] J. Machoul, S. Roucos, H. Gish, "Vector Quantization in Speech Coding", *Proc. of the IEEE,* vol.73, pp.1551–1588, Nov. 1985.

[NaKi88] N.M. Nasrabadi, R.A. King, "Image Coding Using Vector Quantization : A Review", *IEEE Trans on Communications,* vol.36, pp.957–971, Aug. 1988.

[NeLi80] A.N. Netravali, J.O. Limb, "Picture coding : A Review", *Proc. of the IEEE,* vol.68, pp.366–406, Mar. 1980.

[Nar89] Prakash Narayan, "ENEE621 course notes", EE Dept., UMCP, Spring 1989.

[Poo88] H. Vincent Poor, "An Introduction to Signal Detection and Estimation", *Springer-Verlag,* 1988.

[Ros90] Azriel Rozenfield, "Computer Vision for Technical Managers", *The Univ. of MD Instructional Television System,* Jan. 1990.

[San89] T.D. Sanger, "Optimal Unsupervised Learning in Feedforward Neural Networks", M.S. Thesis, Dept. of EE and CS, MIT, Jan. 1989.

[Spi68] M.R.Spiegel, "Mathematical Handbook of Formulas and Tables", *Shaum's Outline Series. McGraw-Hill. NY.* 1968.

[TaFa86] T. Tanabe, N. Farvardin. "Subband Image Coding Using Entropy-Coded Quantization over Noisy Channels". *UMIACS-TR-89–86, UMCP*, Aug . 1989.

[ToMe85] I. Tomescu, R.A.Melter, “Problems in Combinatorics and Graph Theory”, *J. Willey and sons*, 1985

[WoNe86] J.W.Woods and S.D.O’Neil, “Subband Coding of Images”, *IEEE Trans. on ASSP,* vol34, pp1278–1288, Oct. 1986