CONDITIONS OF RELEASE

0094878 300023

********** DRIC U

********** DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

Royal Signals and Radar Establishment

Memorandum 4437

A trainable texture anomaly detector using the Adaptive Cluster Expansion (ACE) method

Stephen P Luttrell
Pattern Processing Principles Section
Royal Signals and Radar Establishment
St Andrews Rd, Malvern, WORCS, WR14 3PS, UK
Janet: luttrell@uk.mod.hermes
Internet: luttrell%hermes.mod.uk@relay.mod.uk

5th November 1990

Abstract

We derive an adaptive hierarchical maximum entropy estimate of probability density functions, whose mathematical structure suggests the name "adaptive cluster expansion" (ACE). We apply ACE to the problem of locating statistically anomalous regions in otherwise homogeneous textured images, which we demonstrate using several images of Brodatz textures.

THIS PAGE IS LEFT BLANK INTENTIONALLY

Contents

List of Figures

1 Introduction

The purpose of this paper is to construct a probabilistic models of images of homogeneous textures for use in Bayesian decision making. In our past work in this area [1, 2, 3, 4, 5] we successfully used entropic methods to design Markov random field (MRF) models to reproduce the observed statistical properties of textured images. However, the main drawback of this method was the need for lengthy computer runs to simulate the MRF. We now wish to formulate a novel MRF structure that requires very little effort to train and use. There are two essential ingredients in our simplification: we do not use hidden variables, and we restrict our attention to hierarchical sampling functions of the data.

The use of hidden variables is a flexible way of modelling high order correlations in data [6], but it needs lengthy Monte Carlo simulations to estimate averages over the hidden variables. An MRF without hidden variables is specified by a set of sampling functions of the data, which transforms the data into a set of numbers containing sufficient information to compute the probability density function (PDF) of the data [4, 5].

The choice of an appropriate set of sampling functions is not easy, because potentially any function of the data might measure a subtle statistical property that had not been discovered by the sampling functions used hitherto. In fact, for this very reason, we conjecture that this problem is insoluble in principle. We can nevertheless obtain a wealth of statistical information about the data by restricting our attention to a finite number of well-defined sampling functions. For instance, in [7] a number of useful textural features are presented, which may be used to model and discriminate between various textures that occur in images. However, we wish to design our sampling functions adaptively in a data-driven manner, so that the resulting set is optimised to capture the statistical properties of the data. We choose to use adaptive *hierarchical* sampling functions, because these not only capture statistical properties at many length scales, but are also very easy to train.

We briefly discussed hierarchical sampling functions in [8], where we conjectured that topographic mappings [9] might be appropriate for connecting together the layers of the hierarchy. We investigated this use of topographic mappings in [10, 11, 12, 13, 14, 15] and found that it could produce useful multiscale representations of the data. We therefore use multilayer topographic mappings (MTM) to adaptively design hierarchical sampling functions of data for use in MRF models. In this type of model different layers of the hierarchy measure statistical structure on different length scales, and shorter length scale structures are clustered together and correlated to produce longer length scale structures. We therefore frequently refer to this type of scheme as an adaptive cluster expansion (ACE).

We should point out that although ACE is based upon a tree-structured representation of the data, it is not equivalent to using a dependence tree as discussed in [16, 17], in which an n-th order PDF was approximated using a tree-like product of 2-nd order PDFs. ACE makes use of the PDFs of transformed versions of the data to model high order properties of the PDF.

We demonstrate the ability of ACE to adapt to statistical structure in texture by designing a pyramid image processor that adapts to and displays the statistical structure of an image. There are many ways of displaying such structure, but we simply display the local PDF of the image: we call this a probability image. By using a representation in which low probability regions appear as bright peaks we can also display what we call an anomaly image. Anomaly images prove to be a very effective way of locating isolated textural anomalies in images that are otherwise texturally homogeneous.

The layout of this paper is as follows. In §2 we use the maximum entropy method to

estimate the PDF of the data, subject to a set of marginal probability constraints measured using hierarchical sampling functions, to yield an MRF model in closed form (ie no undetermined Lagrange multipliers). In §3 we extend this result to remove some of the limitations of its hierarchical structure, such a translation non-invariance, and describe the ACE system for producing probability/anomaly images. In §4 we present the result of applying ACE to some textured images taken from the Brodatz set [18].

2 Maximum entropy PDF estimation

In this section we present a derivation of a hierachical maximum entropy estimate $Q_{mem}(x)$ of an observed true PDF $P(x)$, where we constrain $Q_{mem}(x)$ so that certain marginal PDFs agree with observation. Although we consider only the case of a binary tree, we also present a simple diagrammatic representation of our main result that allows us easily to extend it to general trees.

2.1 Basic maximum entropy method

For completeness, we first of all outline the basic principles [19, 20] of the maximum entropy method of assigning estimates of PDFs. Introduce the entropy functional H

$$H = -\int dx\, Q(x) \log\left(\frac{Q(x)}{Q_0(x)}\right) \tag{1}$$

in which the PDF $Q_0(x)$ acts as a scale[1] to ensure that the argument of the logarithm is dimensionless. $Q_0(x)$ is used to introduce prior knowledge about $P(x)$ into the maximum entropy estimate $Q_{mem}(x)$. Loosely speaking, H measures the extent to which $Q(x)$ is non-committal about the value that x might take. The maximum entropy method consists of maximising H subject to the following set of constraints

$$C_{1,i} = \int dx\, Q(x)\, y_i(x) - \int dx\, P(x)\, y_i(x) = 0 \tag{2}$$

where the $y_i(x)$ are the components of a vector $y(x)$ of sampling functions. These constraints ensure that certain average values are the same whether they are measured using $Q(x)$ (ie our estimated PDF) or using $P(x)$ (ie the observed true PDF). By carefully selecting the $y(x)$ we can optimise the agreement between $Q(x)$ and $P(x)$ as appropriate.

$Q_{mem}(x)$ may be found by introducing a vector λ of Lagrange multipliers, and functionally differentiating $H - \sum_i \lambda_i C_{1,i}$ with respect to $Q(x)$ to yield eventually

$$Q_{mem}(x) = \frac{Q_0(x)\, \exp(-\lambda.y(x))}{\int dx'\, Q_0(x')\, \exp(-\lambda.y(x'))} \tag{3}$$

The undetermined Lagrange vector λ must be chosen in such a way that the constraints are satisfied—this is usually a non-trivial problem[2].

[1] $Q_0(x)$ is frequently, and incorrectly, omitted from the entropy expression when x is a dimensional continuous variable. This is a dangerous practice, leading to results that are difficult to interpret.

[2] Note that equation (3) is one of a family of PDFs called Gibbs distributions which originally made their appearance in statistical thermodynamics.

Now we shall consider another type of maximum entropy problem in which we constraint a set of marginal probabilities [5]. We may impose such constraints by modifying $y_i(\mathbf{x})$ and λ_i as follows

$$\begin{aligned} y_i(\mathbf{x}) &\rightarrow \delta(y - y(\mathbf{x})) \\ \lambda_i &\rightarrow \lambda(y) \end{aligned} \tag{4}$$

where $\delta(y - y(\mathbf{x}))$ is a Dirac delta function. In the $\{y_i(\mathbf{x}), \lambda_i\}$ version of the maximum entropy problem, by varying the value of an index i we could scan through the set of constraint functions $y_i(\mathbf{x})$ and Lagrange multipliers λ_i. However, in the $\{\delta(y - y(\mathbf{x})), \lambda(y)\}$ version of the maximum entropy problem, by varying the value of a variable y we can scan through the set of constraint functions $\delta(y - y(\mathbf{x}))$ and Lagrange multipliers $\lambda(y)$.

The modification in equation (4) causes the constraints in equation (2) to become

$$\begin{aligned} C_2(y) &= \int d\mathbf{x}\, Q(\mathbf{x})\, \delta(y - y(\mathbf{x})) - \int d\mathbf{x}\, P(\mathbf{x})\, \delta(y - y(\mathbf{x})) \\ &= Q(y) - P(y) \\ &= 0 \end{aligned} \tag{5}$$

where we have defined the PDFs over y as

$$\begin{aligned} Q(y) &= \int d\mathbf{x}\, Q(\mathbf{x})\, \delta(y - y(\mathbf{x})) \\ P(y) &= \int d\mathbf{x}\, P(\mathbf{x})\, \delta(y - y(\mathbf{x})) \end{aligned} \tag{6}$$

Thus the delta function constraints force $Q(y) = P(y)$. Note that we have used a rather loose notation for our PDFs—$P(\mathbf{x})$ and $P(y)$ are in fact different functions of their respective arguments. We have made this choice of notation for simplicity, because it will always be clear from the context which PDFs are the same and which are different.

By analogy with the previous maximum entropy derivation, $Q_{mem}(\mathbf{x})$ may be found by functionally differentiating $H - \int dy\, \lambda(y) C_2(y)$ with respect to $Q(\mathbf{x})$ to yield

$$Q_{mem}(\mathbf{x}) = \frac{Q_0(\mathbf{x}) \exp(-\lambda(y(\mathbf{x})))}{\int d\mathbf{x}'\, Q_0(\mathbf{x}') \exp(-\lambda(y(\mathbf{x}')))} \tag{7}$$

$$\rightarrow Q_0(\mathbf{x})\, f(y(\mathbf{x})) \tag{8}$$

where $\lambda(y(\mathbf{x}))$ is an undetermined Lagrange function of $y(\mathbf{x})$. In equation (8) we present a simpler notation by introducing an undetermined function[3] $f(y(\mathbf{x}))$ to absorb the exponential function and the denominator term that appeared in equation (7). We may impose the constraints in equation (5), and use the definitions of $Q(y)$ and $P(y)$ in equation (6) to obtain $f(y)$ in the form

$$f(y) = \frac{P(y)}{\int d\mathbf{x}'\, Q_0(\mathbf{x}')\, \delta(y - y(\mathbf{x}'))} \tag{9}$$

and $Q_{mem}(\mathbf{x})$ in form

$$Q_{mem}(\mathbf{x}) = \frac{Q_0(\mathbf{x})\, P(y(\mathbf{x}))}{\int d\mathbf{x}'\, Q_0(\mathbf{x}')\, \delta(y(\mathbf{x}) - y(\mathbf{x}'))} \tag{10}$$

[3] We shall refer to these $f(y(\mathbf{x}))$ functions as Lagrange functions in our discussion, although this is not an accurate use of nomenclature.

Note that this result is a closed form solution because it contains no undetermined Lagrange functions, unlike equation (3) which contains an undetermined Lagrange vector $\boldsymbol{\lambda}$. The normalisation of this solution can be verified as follows

$$\begin{aligned}\int d\boldsymbol{x}\, Q_{mem}(\boldsymbol{x}) &= \int d\boldsymbol{x}\, dy\, \delta(y - y(\boldsymbol{x}))\, \frac{Q_0(\boldsymbol{x})\, P(y)}{\int d\boldsymbol{x}'\, Q_0(\boldsymbol{x}')\, \delta(y - y(\boldsymbol{x}'))} \\ &= \int dy\, P(y) \\ &= 1 \end{aligned} \tag{11}$$

where we use the identity $\int dy\, \delta(y - y(\boldsymbol{x})) = 1$ to create a dummy integral over y[4].

2.2 Hierarchical maximum entropy method

The purpose of this subsection is to present a generalisation of equation (10) that not only can be solved in closed form, but also allows hierarchical sampling functions to be used.

In practice the result in equation (10) has a limited usefulness. Firstly, we would like to impose many simultaneous constraints, each using its own constraint function $\delta(y_i - y_i(\boldsymbol{x}))$ in equation (5), but this cannot in general be done without sacrificing our closed form solution in equation (10). Secondly, we would like to impose higher order constraints, using a constraint function $\delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))$. This may easily be done by making the replacement $y \rightarrow \boldsymbol{y}$ in equation (10). However, there is a hidden problem, because the greater the dimensionality of $\boldsymbol{y}$, the less easy is it to make the necessary measurements to establish the form of $P(\boldsymbol{y})$. Fortunately, there is a solution to both of these problems, which we shall describe below.

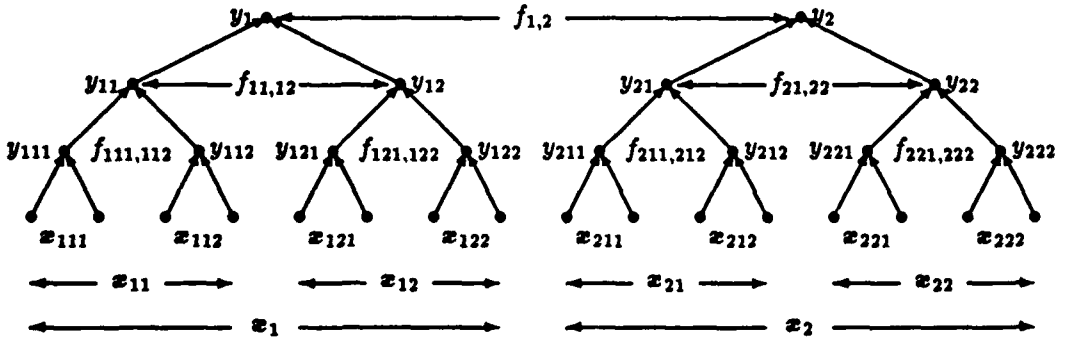

Figure 1: Notation used in the hierarchical maximum entropy derivation.

We shall apply the maximum entropy method with constraints of the form shown in equation (5) to a hierarchy of transformed versions of the input vector $\boldsymbol{x}$. In order to make our calculation tractable we introduce the notation shown in figure (1). The $\boldsymbol{x}_{ijk\ldots}$ are various partitions of the input vector $\boldsymbol{x}$, the $y_{ijk\ldots}$ are various transformed versions $y_{ijk\ldots}(\boldsymbol{x}_{ijk\ldots})$ of the input $\boldsymbol{x}_{ijk\ldots}$, and the $f_{ijk\ldots,i'j'k'\ldots}$ are the Lagrange functions $f_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots})$

[4]**This artifice is a very useful trick for manipulating integrals overs PDFs, by introducing an extra dummy integration, swapping the order of the integrals, and integrating out one or more of the variables that you couldn't integrate in the first place.**

that appear in the generalised version of the maximum entropy solution $Q_{mem}(\mathbf{x})$ in equation (7).

We choose to write the dependence of $y_{ijk\ldots}$ directly on the input $\mathbf{x}_{ijk\ldots}$, even though the value of $y_{ijk\ldots}$ is obtained via a number of intermediate transformations leading from the leaf nodes of the tree up to node $ijk\ldots$, because this leads to a transparent hierarchical maximum entropy derivation. It is convenient to define $\Pi_{ijk\ldots}(\mathbf{x}_{ijk\ldots})$ as the product of the Lagrange functions that appear beneath node $ijk\ldots$ of the tree. $\Pi_{ijk\ldots}(\mathbf{x}_{ijk\ldots})$ has the following recursion property

$$\Pi_{ijk\ldots}(\mathbf{x}_{ijk\ldots}) = f_{ijk\ldots1,ijk\ldots2}(y_{ijk\ldots1}(\mathbf{x}_{ijk\ldots1}), y_{ijk\ldots2}(\mathbf{x}_{ijk\ldots2}))\,\Pi_{ijk\ldots1}(\mathbf{x}_{ijk\ldots1})\,\Pi_{ijk\ldots2}(\mathbf{x}_{ijk\ldots2}) \tag{12}$$

Also introduce a normalisation factor defined as

$$Z_{ijk\ldots}(y_{ijk\ldots}) = \int d\mathbf{x}_{ijk\ldots}\,\delta(y_{ijk\ldots} - y_{ijk\ldots}(\mathbf{x}_{ijk\ldots}))\,\Pi_{ijk\ldots}(\mathbf{x}_{ijk\ldots}) \tag{13}$$

which is a sum of $\Pi_{ijk\ldots}(\mathbf{x}_{ijk\ldots})$ over all states $\mathbf{x}_{ijk\ldots}$ of the leaf nodes beneath node $ijk\ldots$ that are consistent with $y_{ijk\ldots}$ emerging at node $ijk\ldots$.

The proof of the general hierarchical maximum entropy result proceeds inductively. Firstly, we generalise equation (4) to become

$$\begin{aligned} y_i(\mathbf{x}) &\rightarrow \delta(y_{ijk\ldots1} - y_{ijk\ldots1}(\mathbf{x}_{ijk\ldots1}))\,\delta(y_{ijk\ldots1} - y_{ijk\ldots1}(\mathbf{x}_{ijk\ldots1})) \\ \lambda_i &\rightarrow \lambda_{ijk\ldots}(y_{ijk\ldots1}, y_{ijk\ldots2}) \end{aligned} \tag{14}$$

Secondly, we generalise equation (8) to become

$$Q_{mem}(\mathbf{x}) = Q_0(\mathbf{x})\,f_{1,2}(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))\,\Pi_1(\mathbf{x}_1)\,\Pi_2(\mathbf{x}_2) \tag{15}$$

where we use the $\Pi_{ijk\ldots}$ notation to display the Lagrange function $f_{1,2}(y_1, y_2)$ that connects the topmost node-pair (ie node-pair $(1,2)$) in the tree, but conceal the other terms in $Q_{mem}(\mathbf{x})$.

We may determine the exact form of $f_{1,2}(y_1, y_2)$ independently of the rest of the Lagrange functions (which are hidden inside the $\Pi_1(\mathbf{x}_1)$ and $\Pi_2(\mathbf{x}_2)$ functions) by imposing the constraint shown in equation (5) and equation (6) (as applied to node-pair $(1,2)$) to obtain

$$\begin{aligned} P_{1,2}(y_1, y_2) &= \int d\mathbf{x}_1\,d\mathbf{x}_2\,\delta(y_1 - y_1(\mathbf{x}_1))\,\delta(y_2 - y_2(\mathbf{x}_2))\,Q_{mem}(\mathbf{x}) \\ &= f_{1,2}(y_1, y_2)\,Z_1(y_1)\,Z_2(y_2) \end{aligned} \tag{16}$$

which yields

$$f_{1,2}(y_1, y_2) = \frac{P_{1,2}(y_1, y_2)}{Z_1(y_1)\,Z_2(y_2)} \tag{17}$$

Substituting this result back into equation (15) yields

$$Q_{mem}(\mathbf{x}) = P_{1,2}(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))\,\frac{\Pi_1(\mathbf{x}_1)}{Z_1(y_1(\mathbf{x}_1))}\,\frac{\Pi_2(\mathbf{x}_2)}{Z_2(y_2(\mathbf{x}_2))} \tag{18}$$

which correctly obeys the constraint on the joint PDF $P_{1,2}(y_1, y_2)$ of the topmost pair of nodes in the tree.

We now marginalise $Q_{mem}(\mathbf{x})$ in order to concentrate our attention on the left-hand main branch of the tree. Thus

$$\begin{aligned} Q_{1,mem}(\mathbf{x}_1) &= \int d\mathbf{x}_2\, Q_{mem}(\mathbf{x}) \\ &= \int d\mathbf{x}_2\, dy_2\, \delta(y_2 - y_2(\mathbf{x}_2))\, Q_{mem}(\mathbf{x}) \\ &= P_1(y_1(\mathbf{x}_1))\, \frac{\Pi_1(\mathbf{x}_1)}{Z_1(y_1(\mathbf{x}_1))} \end{aligned} \tag{19}$$

We now use the recursion property given in equation (12) to extract the Lagrange function associated with node-pair (11,12). Thus $Q_{1,mem}(\mathbf{x}_1)$ becomes

$$Q_{1,mem}(\mathbf{x}_1) = P_1(y_1(\mathbf{x}_1))\, f_{11,12}(y_{11}(\mathbf{x}_{11}), y_{12}(\mathbf{x}_{12}))\, \frac{\Pi_{11}(\mathbf{x}_{11})\, \Pi_{12}(\mathbf{x}_{12})}{Z_1(y_1(\mathbf{x}_1))} \tag{20}$$

As before, we may determine the exact form of $f_{11,12}(y_{11}, y_{12})$ independently of the rest of the Lagrange functions by applying the constraints to node-pair (11, 12) to obtain

$$f_{11,12}(y_{11}, y_{12}) = \frac{P_{11,12}(y_{11}, y_{12})}{P_1(y_1)}\, \frac{Z_1(y_1)}{Z_{11}(y_{11})\, Z_{12}(y_{12})} \tag{21}$$

where the value of y_1 is to be understood to be obtained directly from the values of y_{11} and y_{12} via the mapping which connects node-pair (11, 12) to node 1. Substituting this result into equation (20) yields

$$Q_{1,mem}(\mathbf{x}_1) = P_{11,12}(y_{11}(\mathbf{x}_{11}), y_{12}(\mathbf{x}_{12}))\, \frac{\Pi_{11}(\mathbf{x}_{11})}{Z_{11}(y_{11}(\mathbf{x}_{11}))}\, \frac{\Pi_{12}(\mathbf{x}_{12})}{Z_{12}(y_{12}(\mathbf{x}_{12}))} \tag{22}$$

By inspection, we see that equation (18) and equation (22) are identical in form once we have accounted for their different positions in the tree, so we may use induction to obtain all of the rest of the Lagrange functions in the form

$$f_{ijk\ldots1,ijk\ldots2}(y_{ijk\ldots1}, y_{ijk\ldots2}) = \frac{P_{ijk\ldots1,ijk\ldots2}(y_{ijk\ldots1}, y_{ijk\ldots2})}{P_{ijk\ldots}(y_{ijk\ldots})}\, \frac{Z_{ijk\ldots}(y_{ijk\ldots})}{Z_{ijk\ldots1}(y_{ijk\ldots1})\, Z_{ijk\ldots2}(y_{ijk\ldots2})} \tag{23}$$

which is analogous to equation (21), and where $y_{ijk\ldots}$ is obtained directly from the values of $y_{ijk\ldots1}$ and $y_{ijk\ldots2}$. The Π/Z factors may be discarded once we reach the leaf nodes of the tree, because the integral in equation (13) then reduces to $Z = \Pi$.

Finally, by starting with equation (15) and recursively simplifying the $\Pi_{ijk\ldots}$ using equation (12) and substituting for the Lagrange functions $f_{ijk\ldots,i'j'k'\ldots}$ using equation (23) we obtain eventually for an n-layer tree

$$\begin{aligned} Q_{mem}(\mathbf{x}) = &\left[\prod_{k=0}^{n-2} \prod_{i_1,i_2,\ldots,i_k=1}^{2} \frac{P_{i_1 i_2 \ldots i_k 1, i_1 i_2 \ldots i_k 2}(y_{i_1 i_2 \ldots i_k 1}(\mathbf{x}_{i_1 i_2 \ldots i_k 1}), y_{i_1 i_2 \ldots i_k 2}(\mathbf{x}_{i_1 i_2 \ldots i_k 2}))}{P_{i_1 i_2 \ldots i_k 1}(y_{i_1 i_2 \ldots i_k 1}(\mathbf{x}_{i_1 i_2 \ldots i_k 1}))\, P_{i_1 i_2 \ldots i_k 2}(y_{i_1 i_2 \ldots i_k 2}(\mathbf{x}_{i_1 i_2 \ldots i_k 2}))}\right] \\ &\left[\prod_{i_1,i_2,\ldots,i_n=1}^{2} P_{i_1 i_2 \ldots i_n}(\mathbf{x}_{i_1 i_2 \ldots i_n})\right] \end{aligned} \tag{24}$$

where we have rearranged the terms to collect together the factors that each node-pair $(i_1 i_2 \ldots i_k 1, i_1 i_2 \ldots i_k 2)$ contributes.

Although we have concentrated on deriving $Q_{mem}(\mathbf{x})$ for a binary tree, the principle of the derivation carries over unchanged to arbitrary tree structures, and equation (24) may easily be generalised.

2.3 Diagrammatic notation

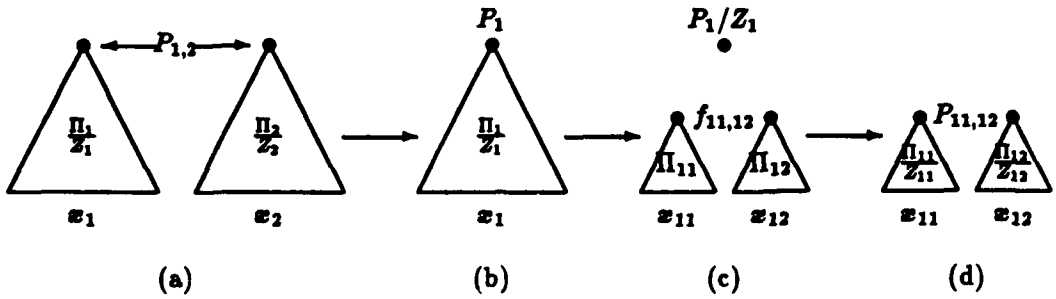

Figure 2: The individual steps of the inductive hierarchical maximum entropy derivation.

We now present the steps in the inductive derivation leading from equation (18) to equation (22) as a diagram in figure (2). We use a triangle to represent a subtree, and we indicate its apex node, its associated Π or Π/Z factor, and its dependence on x. Figure (2a) represents equation (18), which is a pair of trees connected by the joint PDF of their apex nodes. By integrating over x_2 we remove the right hand tree to obtain figure (2b), which corresponds to equation (19). We then explicitly display the two daughter nodes to obtain figure (2c), which corresponds to equation (20), although we have grouped the terms together slightly differently, for simplicity. This exposes one of the Lagrange functions which we determine explicitly to obtain figure (2d), which corresponds to equation (22). One cycle of the inductive proof is completed by noting the correspondence between figure (2a) and figure (2d).

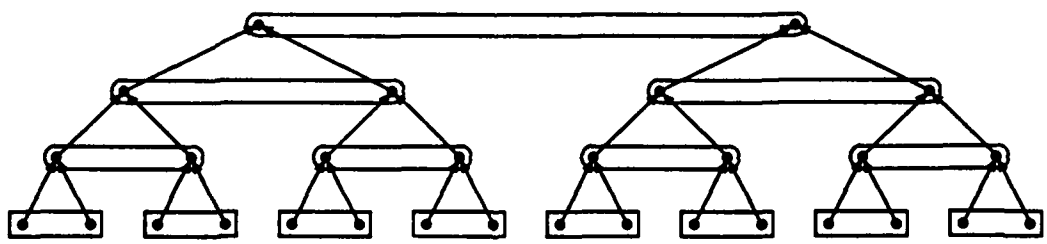

Figure 3: A diagrammatic representation of the hierarchical maximum entropy result.

We represent equation (24) in diagrammatic form in figure (3). The tree structure represents the flow of the transformations of the original input data x. Each square cornered rectangle represents the marginal PDF of the enclosed node-pair (ie one $P_{i_1 i_2 \ldots i_n}$ term from the second factor in equation (24)). Each round cornered rectangle represents the normalised marginal PDF of the encosed node-pair (ie one $P_{i_1 i_2 \ldots i_k 1, i_1 i_2 \ldots i_k 2}/(P_{i_1 i_2 \ldots i_k 1} P_{i_1 i_2 \ldots i_k 2})$ term from the first factor in equation (24)). Overall, we obtain equation (24) as the product of the rectangles in figure (3).

This notation makes it easy to generalise the result in equation (24) in a purely diagrammatic fashion, by firstly constructing an arbitrary (ie not necessarily binary) tree-like transformation of the input data, and secondly using as maximum entropy constraints the

marginal[5] PDF of each set of sister nodes in the tree. This prescription permits many possible ACE structures, including those in which different constraints effectively operate between different layers of the hierarchy (by mapping one or more node values directly from layer to layer).

Each rectangle representing a marginal PDF in figure (3) contributes to the maximum entropy estimate of the PDF of a cluster of nodes in the input data. Because of the tree structure, clusters at each length scale are built out of clusters at smaller length scales. Equation (24) tells us exactly how to incorporate into $Q_{mem}(\mathbf{x})$ any additional statistical properties that might be observed when forming larger clusters out of smaller clusters in this way.

3 Implementation of an anomaly detector

Henceforth we shall refer to our hierarchical maximum entropy method as an adaptive cluster expansion (ACE), because this phrase describes its essential properties[6].

In this section we describe how to implement equation (24) in software. We assume that the ACE transformation functions have already been optimised using the method[7] that we describe in §A and in [14], so the purpose of this section is to explain how to manipulate equation (24) into a form that produces a useful output. For concreteness, we produce an output in the form of an image that represents the degree to which each local patch of an input data is statistically anomalous, when compared to the global statistical properties of the input data.

3.1 Two-dimensional array of inputs

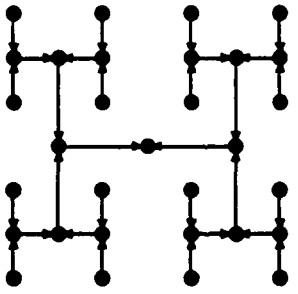

Figure 4: ACE connectivity for processing a 2-dimensional array of inputs.

In §2 we represented ACE as if it were operating on a 1-dimensional array of inputs (eg a time series). In practice this might indeed be the case, but in this paper we choose to study 2-dimensional arrays of inputs (ie images). There is no difficulty in applying ACE to an image, provided that we appropriately assign the leaf nodes to pixels of the image. In figure (4) we show the simplest possibility in which the image is alternately compressed

[5] It is important to include the whole of each family of sister nodes in each marginal PDF, because otherwise the derivation leading to equation (24) fails.

[6] Apparently, the term ACE cannot be used as a registered trademark because it is "laudatory". This will *not* stop me from using the acronym, for obvious reasons!

[7] We use topographic mappings to connect the layers of ACE. However, our maximum entropy results any tree-structured connection of mappings, so one is free to choose other mappings if one so wishes.

in the north-south and east-west directions. A priori, the choice of whether to start with north-south or east-west compression is arbitrary, but if we knew, for instance, that the image had stronger short range correlations in the east-west direction than the north-south direction, then it would be better to compress east-west first of all. Note that in figure (4) the topology of the tree is the same as in figure (3), but the way in which the leaf nodes are identified with the data samples is different.

More generally, we could identify the leaf nodes of the tree with the image pixels in any way that we please, provided that no pixel is used more than once (to guarantee that the tree-like topology is preserved). The problem of optimising the identification of leaf nodes with pixels is extremely complicated, and we shall not pursue it in this paper.

3.2 Histograms

The maximum entropy PDF in equation (24) is a product of (normalised) marginal PDFs. In a practical implementation of ACE the $y_{ijk\ldots}$ are discrete-valued quantities (for instance, integers in the interval [0,255]), and the $P_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots})$ are probabilities (not PDFs). We estimate the $P_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots})$ by constructing 2-dimensional histograms

$$P_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) \simeq \frac{1}{N_{ijk\ldots,i'j'k'\ldots}} h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) \tag{25}$$

where $h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots})$ is the number of counts in the histogram bin $(y_{ijk\ldots}, y_{i'j'k'\ldots})$, and N is the total number of histogram counts given by

$$N_{ijk\ldots,i'j'k'\ldots} = \sum_{y_{ijk\ldots}} \sum_{y_{i'j'k'\ldots}} h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) \tag{26}$$

Note that the estimate in equation (25) suffers from Poisson noise due to the finite number of counts in each histogram bin.

Before training begins the histogram bins are initialised to zero. They are then filled with counts by exposing ACE to many examples of input (or training) vectors. Thus each a vector is propagated up through the ACE-tree, and we then inspect each node-pair $(ijk\ldots, i'j'k'\ldots)$ for which a marginal probability needs to be estimated, and increment its corresponding histogram bin thus

$$h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) \rightarrow h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) + 1 \tag{27}$$

When the training set has been exhausted, histogram bin $ijk\ldots, i'j'k'\ldots$ records the number of times that state $(y_{ijk\ldots}, y_{i'j'k'\ldots})$ occurred.

3.3 Translation invariant processing

We wish to detect statistical anomalies in images which have otherwise spatially homogeneous statistics, such as textures. An invariance of the statistical properties of the true PDF $P(\mathbf{x})$ can be expressed as

$$P(\mathcal{G}\mathbf{x}) = P(\mathbf{x}) \tag{28}$$

where $\mathcal{G}$ is any element of the invariance group, which we shall assume to be the group of translations of the image pixels. In equation (24) $Q_{mem}(\mathbf{x})$ does not respect translation invariance for two reasons. Firstly, we use transformations $y_{ijk\ldots}(\mathbf{x}_{ijk\ldots})$ that are explicitly

translation variant, because the functional form depends on the $ijk\ldots$ indices. Secondly, we connect together these transformations in translation variant way, because the tree structure in figure (1) and figure (4) does not treat all of its leaf nodes equivalently. We shall therefore modify the cluster expansion procedure that we derived in §§2.2 to guarantee translation invariance[8]. This will lead to a much improved maximum entropy estimate $Q_{mem}(\boldsymbol{x})$ of the true $P(\boldsymbol{x})$.

Firstly, use the same transformation function at each position within a single layer of ACE. Thus in equation (24) we make the replacement

$$\begin{aligned} y_{i_1 i_2 \ldots i_k 1}(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}) &\rightarrow y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}) \\ y_{i_1 i_2 \ldots i_k 2}(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}) &\rightarrow y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 2}) \end{aligned} \tag{29}$$

where we indicate that the transformation is associated with the k-th layer of ACE by attaching a superscript k to each function[9]. This yields

$$\begin{aligned} Q_{mem}(\boldsymbol{x}) = & \left[\prod_{k=0}^{n-2} \prod_{i_1, i_2, \ldots, i_k = 1}^{2} \frac{P_{i_1 i_2 \ldots i_k 1, i_1 i_2 \ldots i_k 2}(y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}), y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 2}))}{P_{i_1 i_2 \ldots i_k 1}(y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}))\, P_{i_1 i_2 \ldots i_k 2}(y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 2}))} \right] \\ & \left[\prod_{i_1, i_2, \ldots, i_n = 1}^{2} P_{i_1 i_2 \ldots i_n}(\boldsymbol{x}_{i_1 i_2 \ldots i_n}) \right] \end{aligned} \tag{30}$$

Equation (30) guarantees translation invariance (in the sense of a "single-instruction-multiple-data" computer) of the processing that occurs when the input data is propagated upwards through the overlapping trees.

Secondly, assume that that equation (28) holds for all image translations, so that the marginal PDFs are independent of position. We may make this explicit in our notation by making the following replacement in equation (30)

$$\begin{aligned} \frac{P_{i_1 i_2 \ldots i_k 1, i_1 i_2 \ldots i_k 2}(\cdot)}{P_{i_1 i_2 \ldots i_k 1}(\cdot)\, P_{i_1 i_2 \ldots i_k 2}(\cdot)} &\rightarrow \frac{P^k_{1,2}(\cdot)}{P^k_1(\cdot)\, P^k_2(\cdot)} \\ P_{i_1 i_2 \ldots i_n}(\cdot) &\rightarrow P^{n-1}(\cdot) \end{aligned} \tag{31}$$

where we use the same superscript notation as in equation (29). This yields

$$\begin{aligned} Q_{mem}(\boldsymbol{x}) = & \left[\prod_{k=0}^{n-2} \prod_{i_1, i_2, \ldots, i_k = 1}^{2} \frac{P^k_{1,2}(y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}), y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 2}))}{P^k_1(y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 1}))\, P^k_2(y^k(\boldsymbol{x}_{i_1 i_2 \ldots i_k 2}))} \right] \\ & \left[\prod_{i_1, i_2, \ldots, i_n = 1}^{2} P^{n-1}(\boldsymbol{x}_{i_1 i_2 \ldots i_n}) \right] \end{aligned} \tag{32}$$

Equation (32) guarantees not only translation invariance of the transformations that propagate the data through the tree, but also translation invariance of the marginal PDFs of $P(\boldsymbol{x})$ that are used to construct $Q_{mem}(\boldsymbol{x})$.

[8] The inclusion of this section might seem to be gratuitously pedantic, because usually this translation invariance is assumed tacitly at the outset. Our detailed approach is forced on us by our desire to derive an image processing scheme from first principles, progressing from the general to the specific

[9] Unfortunately, the notation that we introduced in order to clarify our derivations labels the layer of the tree *downwards* as $0, 1, 2, \ldots$, starting with 0 for the top layer.

Both of the simplifications in equation (29) and equation (31) reduce the total number of unknowns that have to be determined. For a given amount of training data we can thus construct a better maximum entropy estimate $Q_{mem}(\mathbf{x})$ of the true $P(\mathbf{x})$. The transformation functions may be optimised better, and the histogram bins have a reduced Poisson noise.

We usually apply ACE to such large input arrays that it is not appropriate to build a single binary tree whose leaf nodes encompass the entire input array. Instead, we divide the input array (which we shall assume is a $2^M \times 2^M$ array of image pixels) into a set of contiguous $2^{m_1} \times 2^{m_2}$ arrays, each of which we analyse using equation (32). There are no constraint functions to measure the mutual dependencies between these subarrays, so the maximum entropy joint PDF of the set of subarrays is a product of terms of the form shown in equation (32).

$$\begin{aligned}\log\left(Q_{mem}(\mathbf{x})\right) &= \sum_{k=0}^{n-2} \sum_{a_1=1}^{2^{M-m_1}} \sum_{a_2=1}^{2^{M-m_2}} \sum_{i_1,i_2,\ldots,i_k=1}^{2} \log\left(\frac{P^k_{1,2}(y^k(\mathbf{x}^{a_1,a_2}_{i_1 i_2 \ldots i_k 1}), y^k(\mathbf{x}^{a_1,a_2}_{i_1 i_2 \ldots i_k 2}))}{P^k_1(y^k(\mathbf{x}^{a_1,a_2}_{i_1 i_2 \ldots i_k 1}))\, P^k_2(y^k(\mathbf{x}^{a_1,a_2}_{i_1 i_2 \ldots i_k 2}))}\right) \\ &+ \sum_{a_1=1}^{2^{M-m_1}} \sum_{a_2=1}^{2^{M-m_2}} \sum_{i_1,i_2,\ldots,i_n=1}^{2} \log\left(P^{n-1}(\mathbf{x}^{a_1,a_2}_{i_1 i_2 \ldots i_n})\right) \qquad (33)\end{aligned}$$

The summation over (a_1, a_2) ranges over the $2^{2M-m_1-m_2}$ contiguous subarrays in the overall $2^M \times 2^M$ array, and the a_1, a_2 superscript on each $\mathbf{x}_{ijk\ldots}$ vector indicates that it belongs to subarray (a_1, a_2). Note that we have transformed $Q_{mem}(\mathbf{x}) \rightarrow \log\left(Q_{mem}(\mathbf{x})\right)$ for convenience.

The final step in constructing a fully translation invariant PDF is to modify the sum over subarrays so that it includes all possible placements of the $2^{m_1} \times 2^{m_2}$ subarray within the overall $2^M \times 2^M$ array. There are $2^{2M-m_1-m_2}$ possible positions when the placement of the subarray is restricted as in equation (33), whereas there are $(2^M - 2^{m_1} + 1)(2^M - 2^{m_2} + 1)$ possible positions when all placements of the subarray are permitted. We therefore make the replacement

$$\begin{aligned}\sum_{a_1=1}^{2^{M-m_1}} \sum_{a_2=1}^{2^{M-m_2}} &\rightarrow \frac{2^{2M-m_1-m_2}}{(2^M - 2^{m_1} + 1)(2^M - 2^{m_2} + 1)} \sum_{p_1=1}^{2^M - 2^{m_1}+1} \sum_{p_2=1}^{2^M - 2^{m_2}+1} \\ &\simeq 2^{-m_1-m_2} \sum_{p_1=1}^{2^M} \sum_{p_2=1}^{2^M} \qquad (34)\end{aligned}$$

in equation (33), where (p_1, p_2) is the coordinate of the pixel in the top left hand corner of the $2^{m_1} \times 2^{m_2}$ subarray. If we ignore edge effects, then we may use the approximation in the final line of equation (34), which is the average of $2^{m_1+m_2}$ separate contributions of the form shown in equation (33). Equation (34) effectively replaces the original maximum entropy PDF $Q_{mem}(\mathbf{x})$ by the geometric mean of a set of maximum entropy PDFs. This averaging effect combines the desirable properties of translation invariance with greatly reduced Poisson noise problems to yield a greatly improved maximum entropy PDF estimate[10].

In practice, we would implement each layer of ACE as a frame store, and the transformation between each pair of adjacent layers as a look-up table. The translation invariant ACE that we derived in equation (33) (with the replacement given in equation (34)) may

[10] Strictly speaking, our averaging prescription is not part of the maximum entropy method, so our averaged result is not a pure maximum entropy estimate.

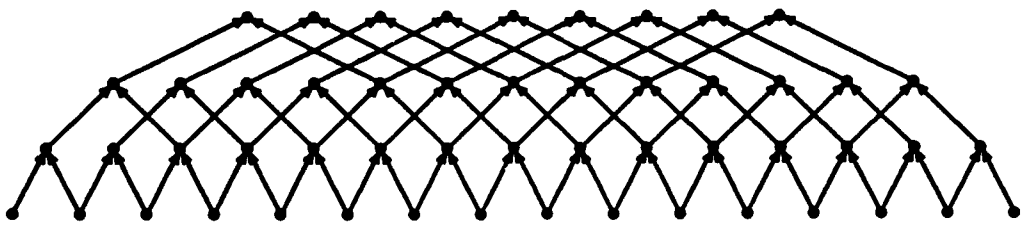

Figure 5: Connectivity for multiple overlapping binary trees.

be implemented using the connectivity shown in figure (5). Ignoring edge effects, we may write equation (33) symbolically as

$$\log\left(Q_{mem}\right) \simeq \sum_{k=0}^{n-2} \frac{1}{2^{n-k}} \sum \log\left(\frac{P_{1,2}^{k}}{P_{1}^{k} P_{2}^{k}}\right) + \frac{1}{2} \sum \log\left(P^{n-1}\right) \tag{35}$$

where the inner summations range over all positions within a single layer of figure (5). We omit all of the functional dependencies, because they are easy to obtain from figure (3). Each $P_{1,2}^{k}/(P_{1}^{k} P_{2}^{k})$ term is represented by a rectangle with rounded corners in figure (3), and each P^{n-1} term is represented by a rectangle with square corners in figure (3). We have not drawn these rectangles in figure (5) because they would overlap, and thus confuse the diagram.

3.4 Forming a probability image

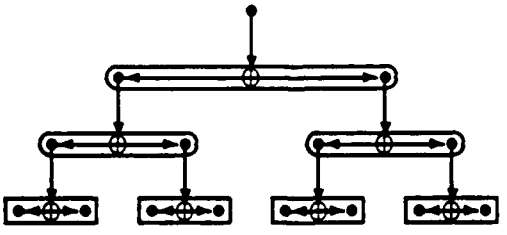

Figure 6: Backpropagation scheme for constructing a probability image.

Equation (35) is the fundamental result that we use to construct useful image processing schemes. However, it would not be very useful simply to calculate the value of $\log\left(Q_{mem}\right)$ as a single global measure of the logarithmic probability associated with an image. We choose instead to break up equation (35) into smaller pieces, and to examine their contribution to the overall $\log\left(Q_{mem}\right)$. In effect, we look at how $\log\left(Q_{mem}\right)$ is built up from the information in each layer of ACE, which in turn we break down into contributions from different areas of the image.

In order to ensure that our decomposition of $\log\left(Q_{mem}\right)$ can be easily computed, we use the backpropagation scheme shown in figure (6) to control the data flow through a translation invariant network of an identical connectivity to the one shown in figure (5). Each node of this backpropagation network records a logarithmic probability, and is cleared

to zero before starting the backpropagation computations. The rectangles in figure (6) represent exactly the same logarithmic probability terms that appeared in figure (3), which we now use as sources of logarithmic probability that we inject into the backpropagating data flow.

The detailed operation of figure (6) is as follows. Each addition symbol takes as input a contribution recorded at a node in the next layer above, adds its own logarithmic probability source $\log(P^k_{1,2}/(P^k_1\,P^k_2))$, scales the result by 1/4, and it finally adds a copy of this result to the value stored at each of its own pair of associated nodes, as shown. The values that accumulate at the leaf nodes represent various contributions to the sum in equation (35). If the translation invariant version of figure (6) is applied to the translation invariant network shown in figure (5), then the sum of the values that accumulate at the leaf nodes reproduces equation (35) precisely.

This method of computing $\log(Q_{mem})$ might seem to be circuitous, but it has the great advantage of both being computationally cheap and forming an image-like representation of $\log(Q_{mem})$, which we call a "probability image". Each $\log(P^k_{1,2}/(P^k_1\,P^k_2))$ term in equation (35) will contribute equally to 2^{n-k} pixels in the probability image. These pixels will be arranged as either a square or a 2-to-1 aspect ratio rectangle according to whether there is an odd number or even number of backpropagation steps from the k-th layer to the leaf nodes. The probability image is therefore a superposition of square and rectangular tiles of logarithmic probability. Each tile corresponds to a node of the network shown in figure (5).

It is useful to display as an image the contributions of a single layer of the network to the probability image, because different layers contribute to the structure of $\log(Q_{mem})$ at different length scales. This image may be displayed in the conventional way, with small probabilities mapped to black, large probabilities mapped to white, and intervening probabilities mapped to shades of grey, in which case we call it a "probabilit image". It is also useful to invert the grey scale so that small probabilities map to black, in which case we call it an "anomaly image", because regions which have statistical properties that occur infrequently show up as bright peaks in the image. We find that the use of probability images and/or anomaly images is an extremely effective way of visually interpreting $\log(Q_{mem})$ in equation (35).

3.5 Modular implementation

For completeness we now present a brief description of a complete system for producing probability and/or anomaly images. This system consists of two tightly coupled subsystems - an ACE subsystem for decomposing the image data, and a probability image subsystem for forming the output image.

Figure (7) combines in one diagram all of the results that we have discussed so far. The upper part of figure (7) is a pure translation invariant ACE subsystem, whereas the lower half is a backpropagating probability image subsystem operating as shown in figure (6). The backpropagating subsystem takes input information from various layers of ACE, as shown. Modules "I" are framestores that record the various transformed images. Modules "M" are look-up tables that record the inter-layer mappings. Modules "T" represent the training algorithm that we explain in §A, which we enclose in a dashed box because the "T" modules are switched out of the circuit once the mappings "M" have been determined[11]. Modules

[11] There is a variety of methods of optimising "T" and "M" as topographic mappings, which are distinguished mainly by their computational cost. The best method that we have found to date fills "M" with the appropriate optimised entries in 2.3 second (using a VAXstation 3100, and assuming 6 bits per pixel).

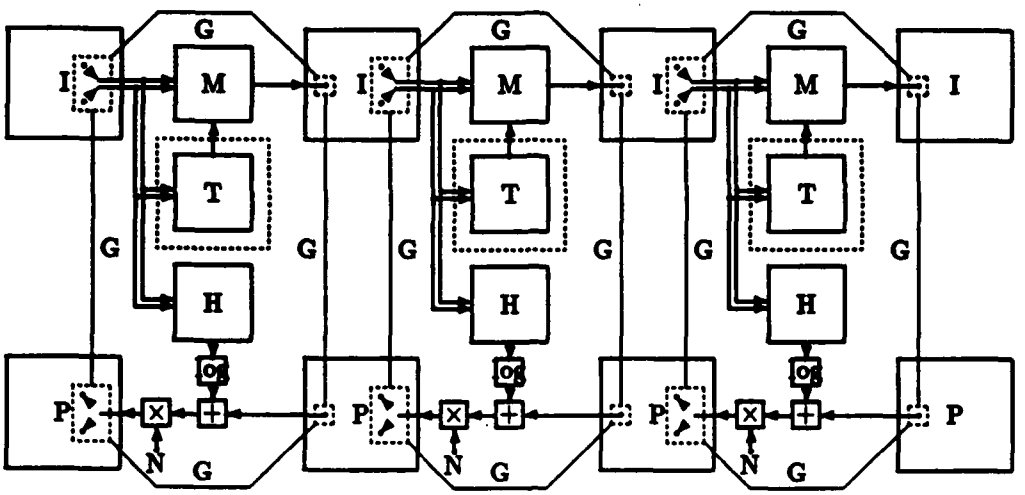

Figure 7: Three layer translation invariant ACE system.

"H" are accumulators that record the 2-dimensional histograms, and then regularise and normalise them appropriately. Modules "P" are framestores that record the various back-propagated probability images. Modules "log" are look-up tables (in fact only one such table is needed) that implement a logarithm function. Modules "⊕" and "⊗" perform the addition and scaling operations that we discussed earlier in connection with figure (6). "N" is scaling factor (which is 1/4 if we wish to reproduce the result in equation (35)). The lines that are annotated "G" represent a ganging together of the (pointers to) pixels in adjacent layers of the ACE subsystem and in the probability image subsystem. These ensure that the entire system works in lockstep, as required.

The simplest mode of operation of this system can be broken down into three stages Firstly, train each layer (from left to right) of the ACE subsystem on a training image. Secondly, propagate a test image (from left to right) through the layers of ACE. Finally, construct a probability image by backpropagating (from right to left) contributions from the various layers of ACE[12]. Furthermore, it is useful to display separately the probability (or anomaly) images that emerge from each layer of ACE, as we shall see in §4.

3.6 Relationship to co-occurrence matrix methods

Both the basic maximum entropy PDF $Q_{mem}(\mathbf{s})$ in equation (24), and the translation invariant version of $\log(Q_{mem}(\mathbf{s}))$ in equation (35) that we implement in practice, depend on various PDFs that are measured in an ACE-tree. The second term of equation (35) may be written as

$$Q_{mem}(\mathbf{s}) = \sqrt{\prod P^{n-1}} \tag{36}$$

Each P^{n-1} factor is the spatial average of the marginal PDF of pairs of adjacent pixel values, assuming that we use the identification of leaf nodes with pixels that we show in

[12]There are more complicated schemes in which different layers are simultaneously trained, whilst communicating information with each other to improve the global performance of ACE. We will explain, and demonstrate the utility of, these refinements in a future communication.

figure (4). The square root in equation (36) compensates for the fact that the product of P^{n-1} factors generates the product of two maximum entropy PDFs shifted by one pixel relative to each other.

By using equation (25) we may approximate equation (36) as a product of histograms. In this case each histogram is the spatial average of the co-occurrence matrix of pairs of adjacent pixel values, as commonly used in image processing [7]. Thus we may use conventional co-occurrence matrix methods to construct a simple form of maximum entropy PDF, which corresponds to using only one layer of ACE.

This co-occurrence matrix result can be generalised, using equation (24) or equation (35), to model higher order statistical behaviour. Although these results depend on co-occurrence matrices measured at various places in the ACE-tree, the contributions which do not depend directly on the input data (ie the first term of equation (35)) actually model higher order statistics of the input data. This is because the value $y_{ijk\ldots}$ that emerges from node $ijk\ldots$ of the ACE-tree depends on $x_{ijk\ldots}$, so the joint PDF $P_{ijk\ldots1,ijk\ldots2}(y_{ijk\ldots1}, y_{ijk\ldots2})$ depends on the statistics of the pair $(x_{ijk\ldots1}, x_{ijk\ldots2})$. Thus ACE is a very convenient way of combining together the various orders of statistical information that are contained in co-occurrence matrices at various places in the ACE-tree, as shown in figure (3).

4 Numerical results

In this section we explain the finer details of how to implement figure (7) in software, and we present the results of applying the system to four 256×256 images of textiles taken from the Brodatz texture set [18].

4.1 Experimental procedure

We compensated for some of the effects of non-uniform illumination by adding to each image a grey scale wedge whose gradient was chosen in such a way as to remove the linear component of the non-uniformity. Not only does this improve the translation invariance of the image statistics, but it also improves the quality of the hierarchical coding of the image, because we reduce the need to develop redundant codes which differ only in their overall grey level.

Throughout our experiments we generate optimal inter-layer mappings using the training methods that we explain in §A. These are known as topographic mappings in the neural network literature, and we showed in [13] why they are appropriate for building multistage vector quantisers. We choose to compress the image in alternate directions using the following sequence: north/south, east/west, north/south, east/west, etc[13]. This compression sequence leads to the following sequence of rectangular image regions that influence the state of each pixel in each stage of ACE: 1×2, 2×2, 2×4, 4×4, etc, using (east/west, north/south) coordinates. In all of our experiments we use an 8 stage ACE.

The number of bits per pixel that we use in each layer of ACE determines the quality of the hierarchical vector quantisation that emerges. Increasing the number of bits improves the quality of the vector quantisation but increases the training time: we need to compromise between these two conflicting requirements. In our work on simple Brodatz texture images we have found that 6-8 bits per pixel is sufficient.

[13] In practice, each image would have its own optimal identification of leaf nodes with image pixels (see figure (4)).

It is important to note that there is a limit to the statistical complexity (ie entropy) of the input data that can be faithfully encoded in a given number of bits. This problem becomes more severe the greater the data compression factor (ie the further we progress through the layers of ACE). For instance, if the input image is very noisy then 6-8 bits will be sufficient only to give good vector quantisation performance in the first few layers of ACE. This problem arises because ACE does not have much prior knowledge of the statistical properties of the input data, so each node of ACE encodes its input without assuming a prior model[14]. A prior model would allow us to reduce the bit rate. This is a fundamental limitation to the capabilities of the current version of ACE, which we intend to address in future.

The choice of the size of the 2-dimensional histogram bins is also important. A property of the topographic mappings that we use to to connect the layers of ACE is that adjacent histogram bins derive from input vectors that are close to each other (in the Euclidean sense), so it is sensible to rebin the histogram by combining together adjacent bins. Thus we control the histogram bin size by truncating the low order bits of each binary vector that represents a pixel value. If we do not truncate any bits, then the 2-dimensional histogram faithfully records the number of times that a pair of pixel values has occured. However, if we truncate b low order bits of each pixel value then effectively we sum together the histogram bins in groups of 2^{2b} ($= 2^b \times 2^b$) adjacent bins, which smooths the histogram. The more smoothing that we impose the less Poisson noise the histogram suffers. However, as we smooth the histogram we run the danger of smoothing away significant structure that might usefully be used to characterise the input image: so we need to make a compromise. In our Brodatz texture work we use only 4-6 bits of each pixel value to generate the histograms in each stage of ACE. Note that we use more bits for vector quantisation than for histogramming because the vector quantisation needs to be good enough to preserve information for encoding by later layers of the hierarchy, whereas the histogramming information is not passed to later layers.

In equation (35) we need to estimate the logarithm of various probabilities from the histograms. We do this in two stages. Firstly, we regularise the histograms by placing a lower bound on the permitted number of counts. One possible prescription is to ensure that each histogram bin has a number of counts at least as large as the average number of counts in all the histogram bins (as determined before regularising the histogram). Thus

$$h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) \rightarrow \begin{cases} h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) & h \geq < h > \\ \langle h_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots}) \rangle & h < < h > \end{cases} \quad (37)$$

where the angle brackets $< \ldots >$ denote an average over histogram bins, rounded up to the next largest integer to avoid setting histogram bins to zero. Secondly, we estimate the probabilities $P_{ijk\ldots,i'j'k'\ldots}(y_{ijk\ldots}, y_{i'j'k'\ldots})$ by inserting the regularised histograms into equation (25). We use a marginalised version of equation (25) to estimate the marginal probabilities $P_{ijk\ldots}(y_{ijk\ldots})$. Finally, we compute the logarithmic probabilities in equation (35) by using a table of logarithms of integers, up to the maximum possible number of counts that could occur in a histogram bin—it suffices to tabulate logarithms up to $\log(N)$.

The prescription in equation (37) is crude but effective. We could improve the performance by introducing prior knowledge of the statistical properties of the input data. Our

[14] In fact, there is a model implied by our use of a Euclidean distortion measure to design the vector quantiser. Loosely speaking, we implicitly assume that a hierarchical Gaussian mixtures model can be used to approximate the PDF of the input data.

histogram smoothing prescription already implicity makes use of prior knowledge of the properties of the Posson noise process that affects the histogram counts, and prior knowledge of the fact that adjacent histogram bins correspond to similar input vectors. Additional prior knowledge would further enhance the performance, especially in cases where there is a limited amount of training data (such as small images, or small segments of larger images).

A pitfall that must be avoided is using histogram bins that are too small when one intends to train ACE on one image and then use a different image to generate a probability image. Effectively, the large number of small bins records the details of the statistical fluctuations of the training image (as particular realisations of a Poisson noise process in each bin), which thus acts as a detailed record of the structure in the training image. The histograms thus look very spiky, and in an extreme case there may be a counts recorded in only a few bins with zeros in all of the remaining bins. If this situation occurs then the training image records a large $\log(Q_{mem})$, whereas a test image having the same statistical properties records a small $\log(Q_{mem})$. Effectively, the spikes in the training and test image histograms are not coincident. This problem can be solved by choosing a large enough histogram bin size.

Finally, we display the logarithmic probability image as follows. We determine the range of pixel values that occurs in the image, and we translate and scale this into the range $[0, 255]$. This ensures that the smallest logarithmic probability appears as black, and the largest logarithmic probability appears as white, and all other values are linearly scaled onto intermediate levels of grey. This prescription has its dangers because each probability image determines its own special scaling, so one should be careful when comparing two different probability images. It can also be adversely affected by pixel value outliers arising from Poisson noise effects, where an extreme value of a single pixel could affect the way in which the whole of an image is displayed. However, we find that the overlapping tree prescription in figure (5) together with the backpropagation prescription in figure (6), causes enough effective averaging together of the histogram bins that we do not encounter problems with pixel value outliers.

In all of the images that we present below, we compensate for the uneven illumination by introducing a grey scale wedge as we explained earlier, we use 8 bits per pixel for vector quantisation, we use 6 bits per pixel for histogramming, and we invert the $[0, 255]$ scale to produce an anomaly image, in which a white pixel indicates a small (rather than a large) logarithmic probability.

4.2 Texture 1

In figure (8) we show the first Brodatz texture image that we use in our experiments. The image is slightly unevenly illuminated and has a fairly low contrast, but nevertheless its statistical properties are almost translation invariant.

In figure (9) we show the anomaly images that derive from figure (8). Note how the anomaly images become smoother as we progress from figure (9a) to figure (9h), due to the increasing amount of averaging that occurs amongst the overlapping backpropagated rectangular tiles that build up each image.

Figure (9e) and especially figure (9f) reveal a highly localised anomaly in the original image. Figure (9f) corresponds to a length scale of 8×8 pixels, which is the approximate size of the fault that is about $1/4$ of the way down and slightly to the left of centre of figure (8). The fault does not show up clearly on the other figures in figure (9) because their characteristic length scales are either too short or too long to be sensitive to the fault.

THIS PAGE IS LEFT BLANK INTENTIONALLY

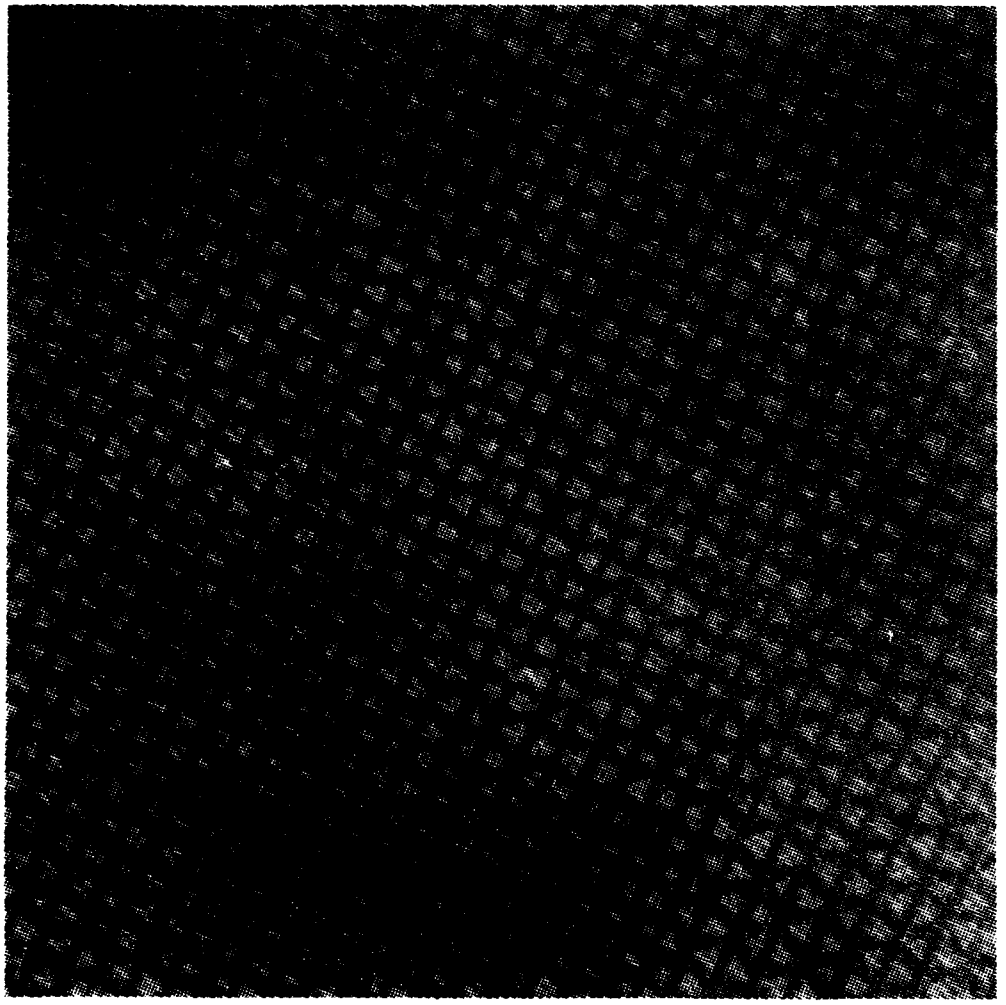

Figure 8: 256x256 image of Brodatz fabric number 1.

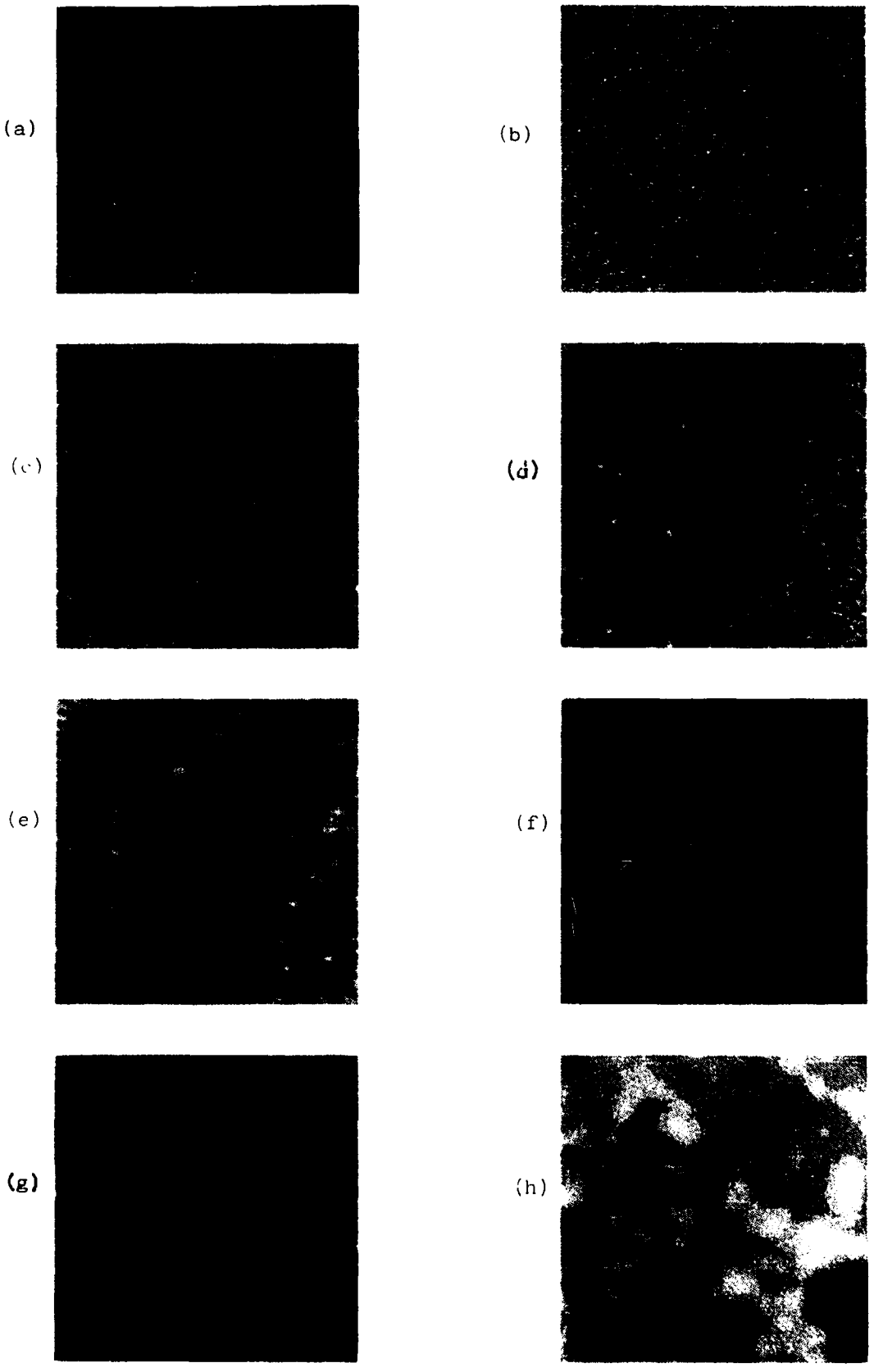

Figure 9: 256x256 anomaly images of Brodatz fabric number 1.

There is a major feature in the bottom right hand corner of figure (9h), where the anomaly image is darker than average, indicating that the corresponding part of the original image has a higher than average probability. This is a different type of anomaly to the sort that we have envisaged so far—it occurs because the corresponding part of original image happens to explore only a high probability part of the space that is explored by the whole image. This part of the anomaly image is surrounded by a brighter than average border, which indicates a conventional anomalous region.

From figure (9) we conclude that ACE can easily pick out localised faults in highly ordered textures.

4.3 Texture 2

In figure (10) we show the second Brodatz texture image that we use in our experiments. The image has a high contrast and translation invariant statistical properties.

In figure (11) we show the anomaly images that derive from figure (10). The most interesting anomaly image is figure (11f) which shows several localised anomalies. About halfway down and to the left of centre of the image is an anomaly that corresponds to a dark spot on the thread in figure (10). The brightest of the anomalies in the cluster just above the centre of the image corresponds to what appears to be a slightly torn thread in figure (10). The other anomalies in this cluster are weaker, and correspond to slight distortions of the threads. There is another anomaly just below and to the right of the centre of figure (11g), which corresponds to what appears to be another slightly torn thread in figure (10). These anomalies all occur at, or around, a length scale of 8×8 pixels. Several of the anomaly images show an anomaly in the bottom left hand corner of the image, which corrsponds to a small uniform patch of fabric in figure (10).

The results in figure (11) corroborate the evidence in figure (9) that ACE can be trained in an unsupervised fashion to pick out localised faults in highly ordered textures.

4.4 Texture 3

In figure (12) we show the third Brodatz texture image that use in our experiments. The image has a very high contrast and statistical properties that are almost translation invariant. However the density of anomalies is much higher than in either figure (8) or figure (10).

In figure (13) we show the anomaly images that derive from figure (12). The most prominant anomaly is in figure (13g), at a length scale of 8×16 pixels, which corresponds to region of figure (12) that is just above and to the left of centre of the image. This region is anomalous because it is both distorted and has slightly thicker threads than elsewhere. The large distorted region in the bottom left hand corner of figure (12) does not show up very clearly to the naked eye in figure (13), but figure (13f) and figure (13h) have significant peaks in this region. There are also many other localised peaks in figure (13) which can be traced back to corresponding faults in figure (12).

Comparing figure (13) with figure (9) and figure (11) we conclude that the ability of ACE to pick out faults is degraded as the density of faults increases. This is because the faults themselves are part of the statistical properties that are extracted by ACE, and if a particular fault occurs often enough in the image then it is no longer deemed to be a fault.

4.5 Texture 4

THIS PAGE IS LEFT BLANK INTENTIONALLY

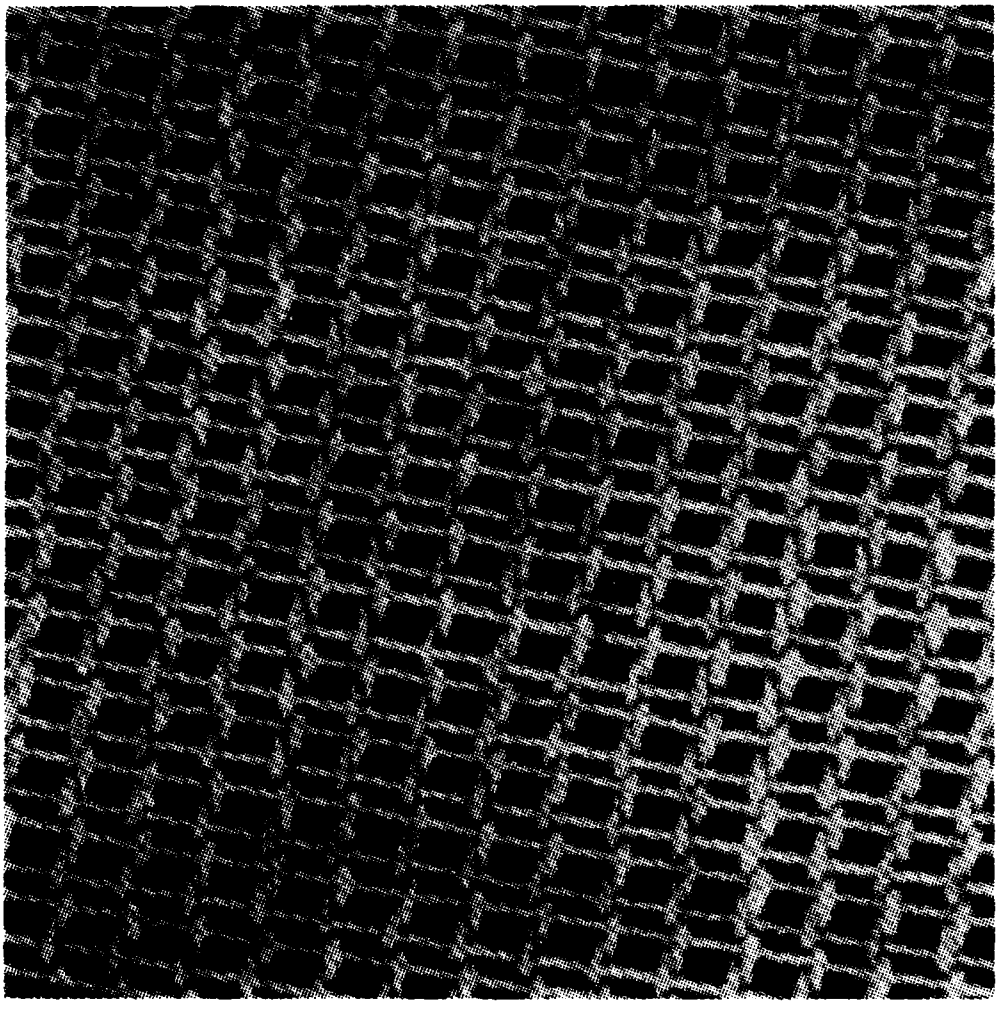

Figure 10: 256x256 image of Brodatz fabric number 2.

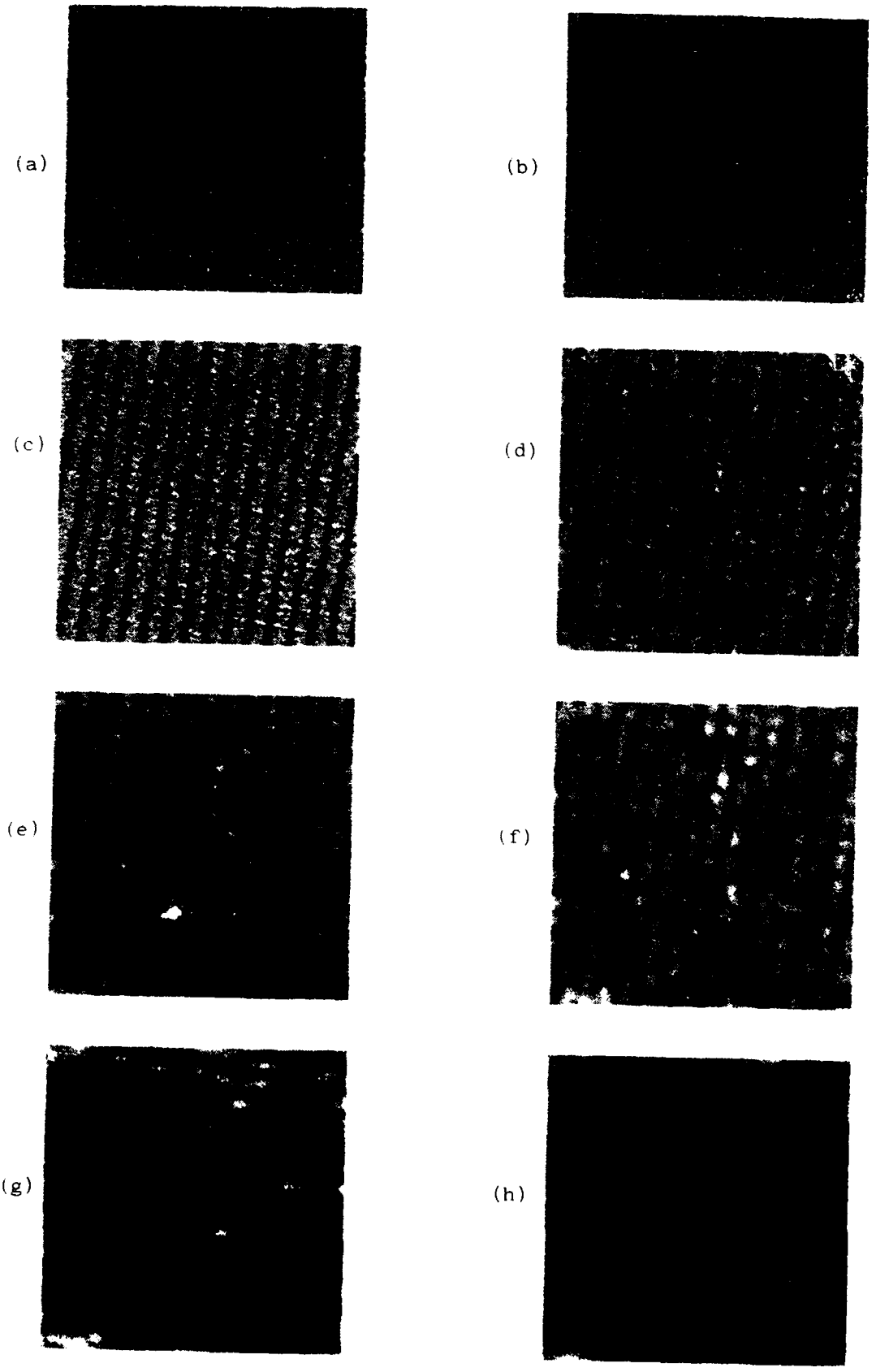

Figure 11: 256x256 anomaly images of Brodatz fabric number 2.

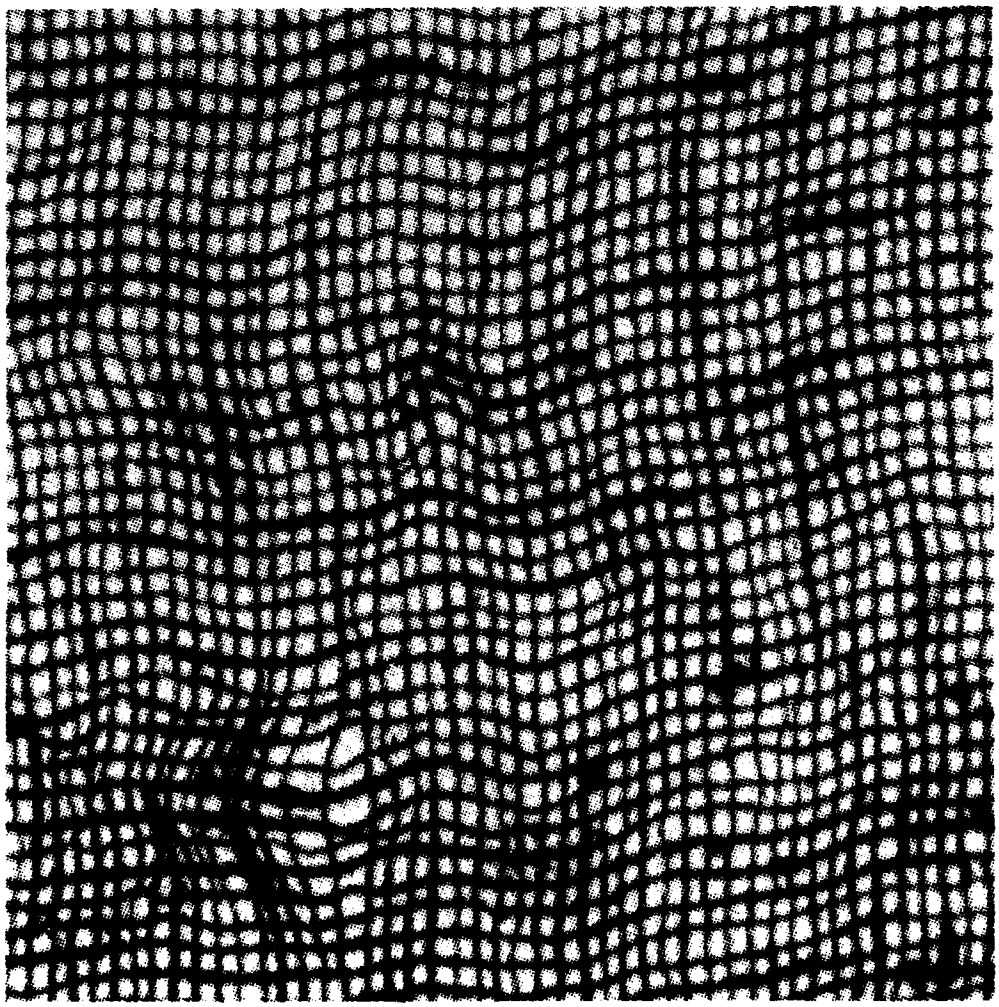

Figure 12: 256x256 image of Brodatz fabric number 3.

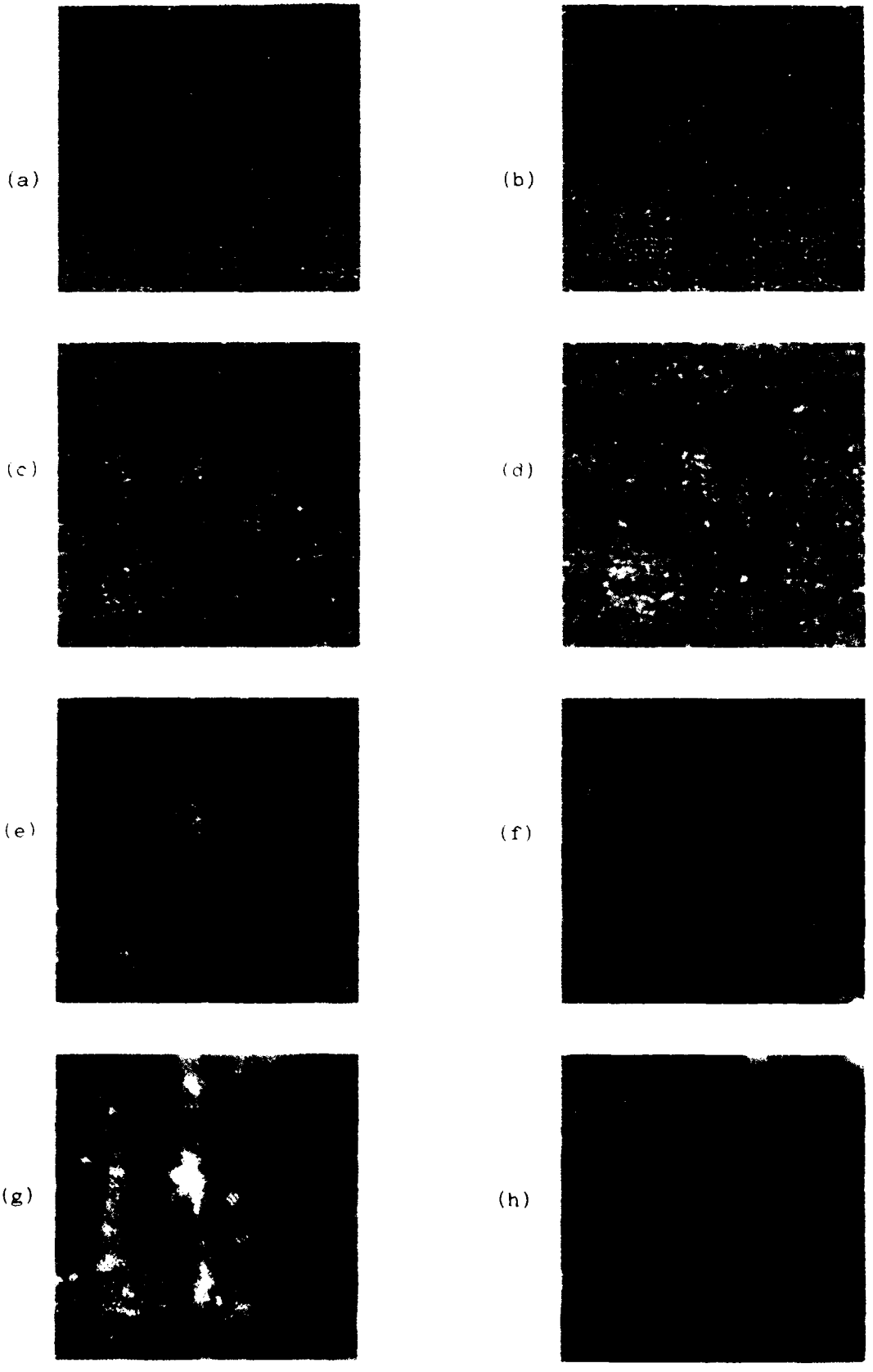

Figure 13: 256x256 anomaly images of Brodatz fabric number 3.

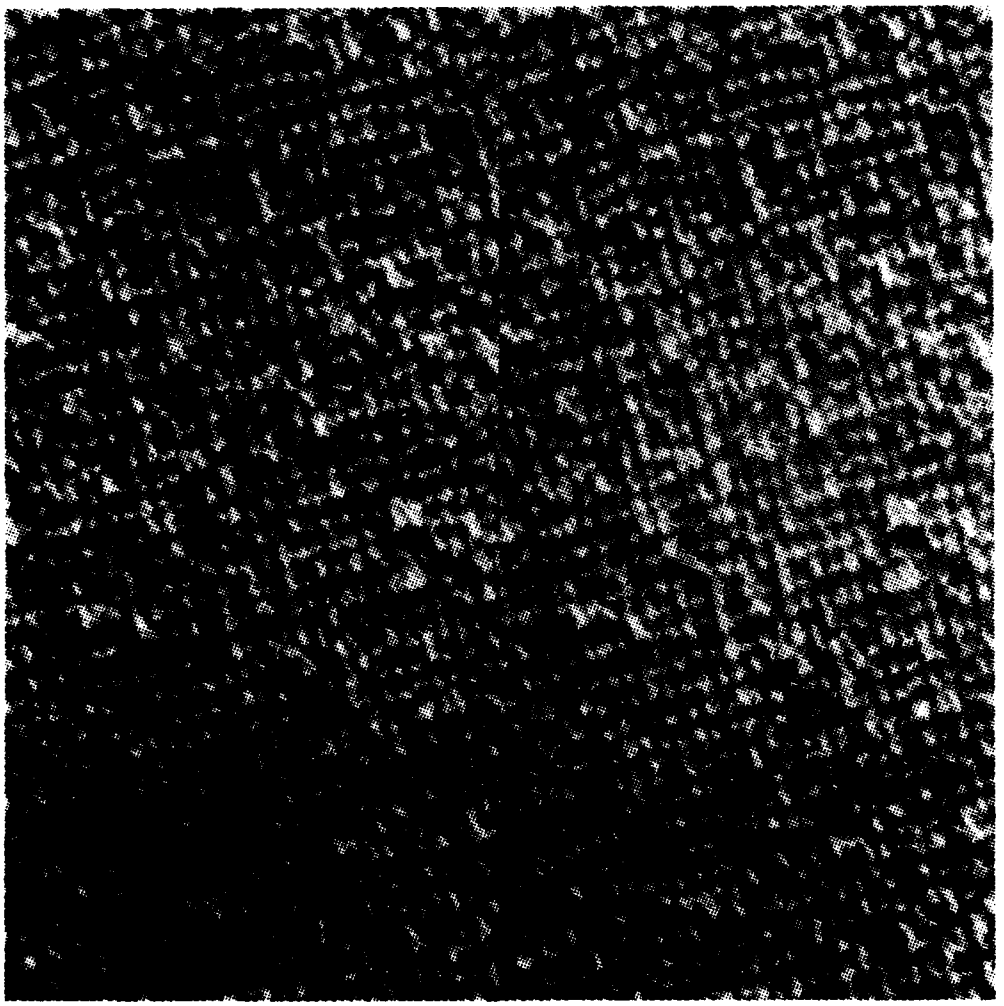

Figure 14: 256x256 image of Brodatz carpet for training.

In this section we present a slightly different type of experiment in which we train ACE on one image and test ACE on another image. To create the two images we start with a single 256 × 256 image of a Brodatz texture, which we divide into a left half and a right half. We then use the left half to build up the training image, and the right half to build up the test image.

In figure (14) we show the training image which is a montage of two copies of the left hand half of a Brodatz texture image. In figure (15) we show the test image which is a montage of two copies of the right hand half of a Brodatz texture image, and superimposed on that is a 64 × 64 patch which we generated by flipping the rows and columns of a copy of the top left hand corner of this image. This patch is a hand crafted anomaly. Note that in constructing these images we have scrupulously avoided the possibility that the training and test images could contain elements deriving from a common source.

In figure (16) we show the anomaly images that derive from figure (15) after having trained on figure (14). Figure (16f) shows the strongest response to the anomalous patch in the centre of the image, corresponding to anomaly detection on a length scale of 8 × 8 pixels.

5 Conclusions

Using maximum entropy methods, we have shown how to construct maximum entropy estimates of PDFs by using adaptive hierarchical sampling functions to record various marginal PDFs of the data. We have also shown how to extend this result so that it can be applied to translation invariant image processing, such as the detection of statistical anomalies in otherwise statistically homogeneous textures. Our methods show great promise, not only because they are amenable to a full theoretical analysis leading to closed-form maximum entropy solutions, but also because they lead directly to a modular system design which can locate anomalies in textures.

We have presented several examples where our "adaptive cluster expansion" (ACE) technique successfully detects anomalous regions in otherwise statistically homogeneous textures. In all cases ACE adaptively extracts the global statistics of an image at various length scales during the unsupervised training. ACE then uses these statistics to form an output image that represents the probability that each local patch of the input image belongs to the ensemble of patches presented during training. We call this a "probability image", and its inverse an "anomaly image".

Some possible applications of our results are as follows. Inspection of textiles: this relies on the assumed statistical homogeneity of an unflawed piece of textile, so that faults show up as anomalies. Detection of targets in noisy background clutter in radar images: this is basically a noisy version of the textile inspection problem. Texture segmentation: this is an ambitious goal which requires further analysis in order to derive a computationally cheap method of handling multiple simultaneous textures.

A Vector quantisation

In this appendix we summarise the hierarchical vector quantisation method that we presented in detail in [14]. In this paper we use this technique to optimise the inter-layer mappings in figure (7). We have applied this technique elsewhere to image compression [11], and multilayer self-organising neural networks [10, 12, 13, 15].

THIS PAGE IS LEFT BLANK INTENTIONALLY

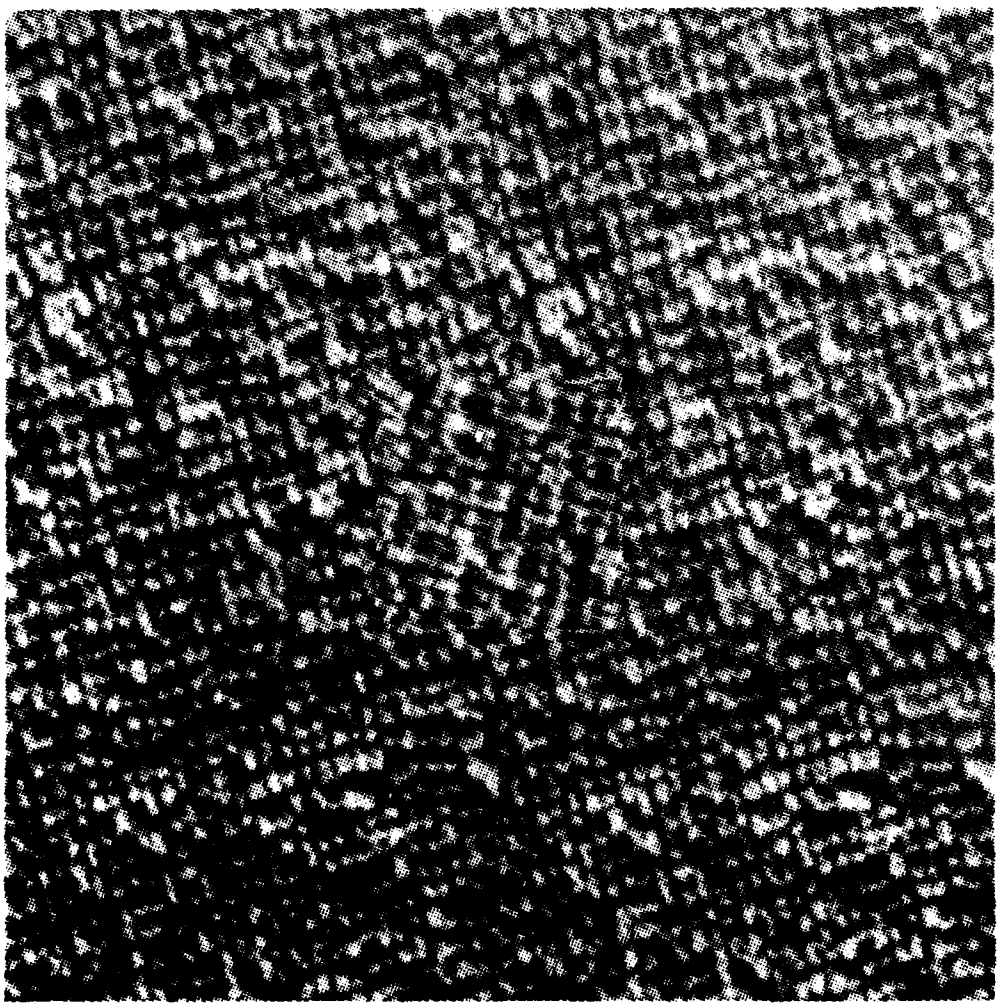

Figure 15: 256x256 image of Brodatz carpet for testing.

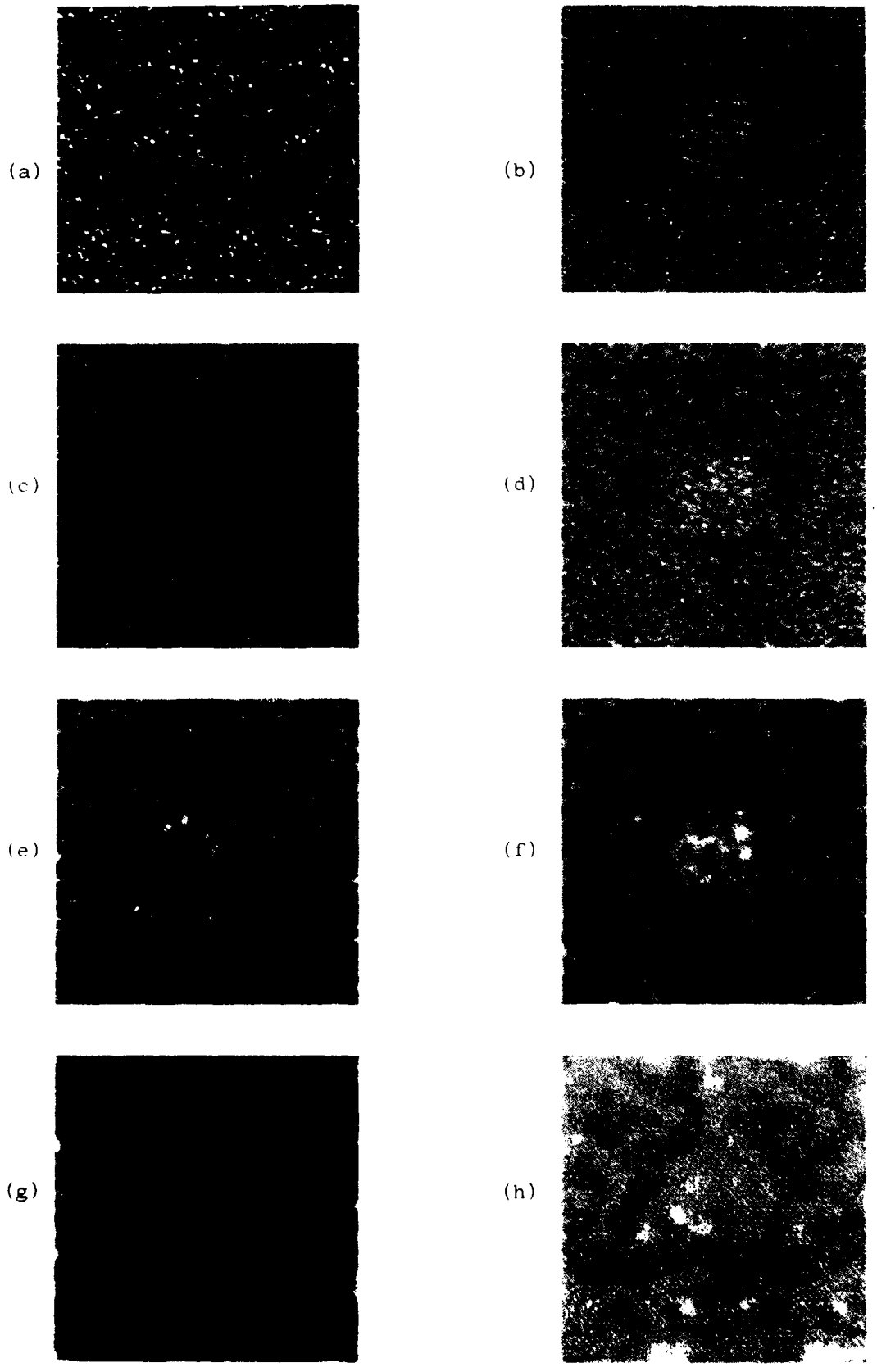

Figure 16: 256x256 anomaly images of Brodatz carpet.

A.1 Standard vector quantisation

This subsection contains those details of the theory of standard vector quantisation that one needs to understand before proceeding to the modified vector quantisation scheme that we present in §§A.2.

The problem is to form a coding y of a vector x in such a way that a good estimate x' of x can be constructed from knowledge of y alone. The sketch derivation in this section is presented in greater detail in [13]. Thus a vector quantiser is constructed by minimising a Euclidean distortion D_1 with respect to the choice of coding function $y(x)$ and decoding function $x'(y)$, where

$$D_1 = \int dx\, P(x)\, \|x - x'(y(x))\|^2 \tag{38}$$

We may represent the encoding and decoding operations diagrammatically as shown in

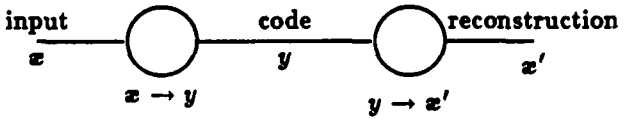

Figure 17: Encoding and decoding in a vector quantiser.

figure (17). By functionally differentiating D_1 with respect to $y(x)$ and $x'(y)$ we obtain

$$\frac{\delta D_1}{\delta y(x)} = P(x) \frac{\partial}{\partial y} \|x'(y) - x\|^2 \Big|_{y=y(x)} \tag{39}$$

$$\frac{\delta D_1}{\delta x'(y)} = 2 \int dx\, P(x)\, \delta(y - y(x))\, (x'(y) - x) \tag{40}$$

Setting $\delta D_1 / \delta y(x) = 0$ in equation (39) yields the optimum encoding function

$$y(x) = \arg \min_y \|x'(y) - x\|^2 \tag{41}$$

which is called "nearest neighbour encoding". Setting $\delta D_1 / \delta x'(y) = 0$ in equation (40) yields the optimum decoding function

$$x'(y) = \frac{\int dx\, P(x) \delta(y - y(x))\, x}{\int dx\, P(x) \delta(y - y(x))} \tag{42}$$

which is the update scheme derived in [21]. Alternatively, we may use an incremental scheme to optimise the decoding function by following the path of steepest descent which we may obtain from equation (40) as

$$\delta x'(y) = \epsilon\, \delta(y - y(x))\, (x - x'(y)) \tag{43}$$

where $0 < \epsilon < 1$.

An iterative optimisation scheme may be formed by alternately applying equation (41) and then either equation (42) or equation (43). This scheme will alternately improve the encoding and decoding functions until a local minimum distortion is located. Alternating equation (41) and equation (42) is commonly called the "LBG" (after the authors of [21]) or "k-means" algorithm.

A.2 Noisy vector quantisation

This subsection contains the theoretical details of the optimisation of inter-layer mappings that we use in our numerical simulations in §4. Thus we generalise the results of §§A.1 to the case where the encoded version of the input vector is distorted by a noise process [22, 23, 14, 15].

Define a modified Euclidean distortion D_2 as

$$D_2 = \int d\boldsymbol{x}\, P(\boldsymbol{x}) \int dn\, \pi(n) \|\boldsymbol{x} - \boldsymbol{x}'(y(\boldsymbol{x}) + n)\|^2 \tag{44}$$

We may represent the encoding and decoding operations together with the noise process

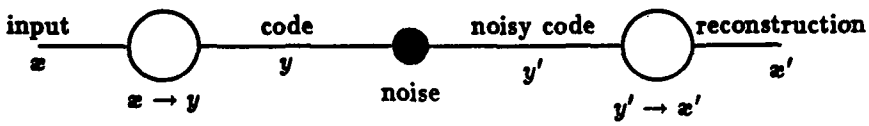

Figure 18: Encoding and decoding in a noisy vector quantiser.

diagrammatically as shown in figure (18), which is a trivially modified version of figure 17. By functionally differentiating D_2 with respect to $y(\boldsymbol{x})$ and $\boldsymbol{x}'(y)$ we obtain

$$\frac{\delta D_2}{\delta y(\boldsymbol{x})} = P(\boldsymbol{x}) \int dn\, \pi(n) \frac{\partial}{\partial y} \|\boldsymbol{x}'(y) - \boldsymbol{x}\|^2 \Big|_{y=y(\boldsymbol{x})+n} \tag{45}$$

$$\frac{\delta D_2}{\delta \boldsymbol{x}'(y)} = 2 \int d\boldsymbol{x}\, P(\boldsymbol{x}) \pi(y - y(\boldsymbol{x}))\, (\boldsymbol{x}'(y) - \boldsymbol{x}) \tag{46}$$

Equation (45) is a "smeared" version of equation (39), so $\delta D_2 / \delta y(\boldsymbol{x}) = 0$ does not lead to nearest neighbour encoding because the distances to other code vectors have to be taken into account in order to minimise the damaging effect of the noise process. However, it is usually a good approximation to use the nearest neighbour encoding scheme shown in equation (41). Setting $\delta D_2 / \delta \boldsymbol{x}'(y) = 0$ in equation (46) yields the optimum decoding function

$$\boldsymbol{x}'(y) = \frac{\int d\boldsymbol{x}\, P(\boldsymbol{x}) \pi(y - y(\boldsymbol{x}))\, \boldsymbol{x}}{\int d\boldsymbol{x}\, P(\boldsymbol{x}) \pi(y - y(\boldsymbol{x}))} \tag{47}$$

which should be compared with equation (42). Alternatively, we may obtain a steepest descent scheme in the form

$$\delta \boldsymbol{x}'(y) = \epsilon\, \pi(y - y(\boldsymbol{x}))\, (\boldsymbol{x} - \boldsymbol{x}'(y)) \tag{48}$$

where $0 < \epsilon < 1$, which should be compared with equation (43).

As in §§A.1, iterative optimisation schemes can be constructed in which we alternate the optimisation of the coding and decoding functions. Alternating equation (41) (which approximately solves $\delta D_2 / \delta \boldsymbol{x}'(y) = 0$) and equation (48) yields the standard topographic mapping training algorithm [9], which is widely used in various forms in neural network simulations.

A.3 Hierarchical vector quantisation

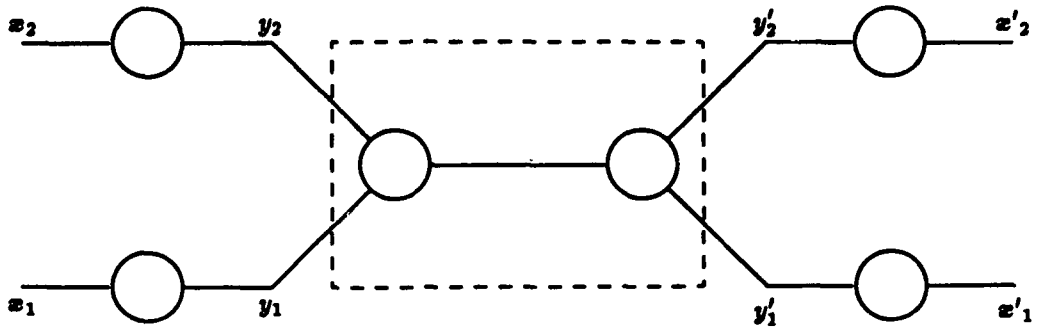

Figure 19: Encoding and decoding in a hierarchical vector quantiser.

In figure (19) we show the simplest type of hierarchical vector quantiser. It consists of an inner quantiser contained in the dashed box, surrounded by a pair of outer quantisers.

If the part of the diagram contained in the dashed box were removed and direct connections made so that $y'_1 = y_1$ and $y'_2 = y_2$, then figure (19) would reduce to a pair of independent vector quantisers of the type shown in figure (17). The dashed box contains a vector quantiser which encodes (y_1, y_2) to produce a code which it subsequently decodes to obtain (y'_1, y'_2).

From the point of view of y_1 the effect of being passed through the inner quantiser is to modify y_1 thus $y_1 \rightarrow y'_1$. A similar argument applies to $y_2 \rightarrow y'_2$. The actual distortions $y'_1 - y_1$ and $y'_2 - y_2$ will be correlated in practice, but we shall model them as if they were independent processes, and thus reduce figure (19) to two independent vector quantisers of the type shown in figure 18.

This procedure can be extended to a hierarchical vector quantiser with any number of levels of nesting. From the point of view of the quantisers at any level, we shall model the effect of the quantisers inwards from that level as independent distortion processes. It turns out not to be critically important what precise distortion model one uses, provided that it approximately represents the overall scale of the distortion due to quantisation.

In [14] we presented in detail a phenomenological distortion model that we used to obtain an efficient training procedure for topographic mappings and their application to hierarchical vector quantisers. Alternatively, the standard topographic mapping training procedure in [9] could be used, but this is a rather inefficient algorithm. The basic training procedure may be obtained from equation (48) as

1. Select a training vector $\mathbf{x}$ at random from the training set.
2. Encode $\mathbf{x}$ to produce $y(= y(\mathbf{x}))$.
3. For all y' do the following:
   (a) Determine the corresponding code vector $\mathbf{x}'(y')$.
   (b) Move the code vector $\mathbf{x}'(y')$ directly towards the input vector $\mathbf{x}$ by a distance $\epsilon \pi(y' - y) |\mathbf{x} - \mathbf{x}'(y')|$
4. Go to step 1.

This cycle is repeated as often as is required to ensure convergence of the codebook of code vectors.

The standard method [9] specifies that $\pi(y'-y)$ should be an even unimodal function whose width should be gradually decreased as training progresses. This allows coarse-grained organisation of the codebook to occur, followed progressively by ever more fine-grained organisation, until finally the algorithm converges towards an optimum codebook.

In our own modification [14] of the standard method we replace a shrinking $\pi(y'-y)$ function acting on a fixed number of code vectors by a fixed $\pi(y'-y)$ function acting on an increasing number of code vectors. There are many minor variations on this theme, but we find that it is sufficient to define

$$\pi(y'-y) = \begin{cases} \epsilon & y'=y \\ \epsilon' & |y'-y|=1 \\ 0 & |y'-y|>1 \end{cases} \tag{49}$$

where we have absorbed ϵ in equation (48) into the definition of $\pi(y'-y)$. We use a binary sequence of codebook sizes $N = 2,4,8,16,32,\ldots$, where each codebook is initialised by interpolation from the next smaller codebook. We find that the following parameter values yield adequate convergence: $\epsilon = 0.1$, $\epsilon' = 0.05$, and we perform $20N$ training updates before doubling the value of N and progressing to the next larger size of codebook. The $N = 2$ codebook can be initialised using a random pair of vectors from the training set.

References

[1] Luttrell S P. Markov random fields: a strategy for clutter modelling. In *Proc. AGARD Conf. on Scattering and Propagation in Random Media*, pages 7.1–7.8, Rome, 1987.

[2] Luttrell S P. The use of Markov random field models in sampling scheme design. In Pike E R, editor, *Proc. SPIE Int. Symp. on Inverse Problems in Optics*, pages 182–188, The Hague, 1987.

[3] Luttrell S P. Designing Markov random field structures for clutter modelling. In *Proc. Radar-87*, pages 222–226, London, 1987.

[4] Luttrell S P. The use of Markov random field models to derive sampling schemes for inverse texture problems. *Inverse Problems*, **3**, 289–300, 1987.

[5] Luttrell S P. A maximum entropy approach to sampling function design. *Inverse Problems*, **4**, 829–841, 1988.

[6] Ackley D H, Hinton G E and Sejnowski T J. A learning algorithm for Boltzmann machines. *Cogn. Sci.*, **9**, 147–169, 1985.

[7] Haralick R M, Shanmugam K and Dinstein I. Textural features for image classification. *IEEE Trans. SMC*, **3**. 610–621, 1973.

[8] Luttrell S P. The use of Bayesian and entropic methods in neural network theory. In Skilling J, editor, *Maximum entropy and Bayesian methods*, pages 363–370, Cambridge, 1989. Kluwer Academic Publishers.

[9] Kohonen T. *Self organisation and associative memory.* Springer-Verlag, 1984.

[10] Luttrell S P. Multilayer topographic mappings. In *Proc. 2nd IEEE Int. Conf. on Neural Networks*, volume 1, pages 93–100, San Diego, 1988.

[11] Luttrell S P. Image compression using a multilayer neural network. *Patt. Recog. Letts.*, **10**, 1–7, 1989.

[12] Luttrell S P. Self-organisation: a derivation from first principles of a class of learning algorithms. In *Proc. 3rd IEEE Int. Joint Conf. on Neural Networks*, volume 2, pages 495–498, Washington DC, 1989.

[13] Luttrell S P. Hierarchical self-organising networks. In *Proc. 1st IEE Conf. on Artificial Neural Networks*, pages 2–6, London, 1989.

[14] Luttrell S P. Hierarchical vector quantisation. *Proc. IEE Part I*, **136**, 405–413, 1989.

[15] Luttrell S P. Derivation of a class of training algorithms. *IEEE Trans. Neural Networks*, **1**, 229–232, 1990.

[16] Chow C K and Liu C N. Approximating discrete probability distributions with dependence trees. *IEEE Trans. IT*, **14**, 462–467, 1968.

[17] Chow C K and Wagner T J. Consistency of an estimate of tree-dependent probability distributions. *IEEE Trans. IT*, **19**, 369–371, 1973.

[18] Brodatz P. *Textures - a photographic album for artists and designers.* Dover, 1966.

[19] Jaynes E T. Prior probabilities. *IEEE Trans. SSC*, **4**, 227–241, 1968.

[20] Jaynes E T. On the rationale of maximum entropy methods. *Proc. IEEE*, bf 70, 939–952, 1982.

[21] Linde Y, Buzo A and Gray R M. An algorithm for vector quantiser design. *IEEE Trans. COM*, **28**, 84–95, 1980.

[22] Kumazawa H, Kasahara M and Namekawa T. A construction of vector quantisers for noisy channels. *Electronics and Engineering in Japan*, **67B**, 39–47, 1984.

[23] Farvardin N. A study of vector quantisation for noisy channels. *IEEE Trans. IT*, **36**, 799–809, 1990.

THIS PAGE IS LEFT BLANK INTENTIONALLY

REPORT DOCUMENTATION PAGE

DRIC Reference Number (If known) .....

Overall security classification of sheet .....Unclassified.....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked with the classification eg (R), (C) or (S).

Originators Reference/Report No.	Month	Year
MEMO 4437	NOVEMBER	1990

Originators Name and Location
RSRE, St Andrews Road
Malvern, Worcs WR14 3PS

Monitoring Agency Name and Location

Title
A TRAINABLE TEXTURE ANOMALY DETECTOR USING THE ADAPTIVE CLUSTER EXPANSION (ACE) METHOD

Report Security Classification	Title Classification (U, R, C or S)
UNCLASSIFIED	U

Foreign Language Title (in the case of translations)

Conference Details

Agency Reference	Contract Number and Period

Project Number	Other References

Authors	Pagination and Ref
LUTTRELL, S P	33

Abstract

We derive an adaptive hierarchical maximum entropy estimate of probability density functions, whose mathematical structure suggests the name "adaptive cluster expansion" (ACE). We apply ACE to the problem of locating statistically anomalous regions in otherwise homogeneous textured images, which we demonstrate using several images of Brodatz textures.

Abstract Classification (U,R,C or S)
U

Descriptors

Distribution Statement (Enter any limitations on the distribution of the document)
UNLIMITED

THIS PAGE IS LEFT BLANK INTENTIONALLY