Chapter 4

TEXEMS: Random Texture Representation and Analysis

Xianghua Xie and Majid Mirmehdi*
Department of Computer Science, University of Bristol
Bristol BS8 1UB, England
E-mail: {xie,majid}@cs.bris.ac.uk

Random textures are notoriously more difficult to deal with than regular textures particularly when detecting abnormalities on object surfaces. In this chapter, we present a statistical model to represent and analyse random textures. In a two-layer structure a texture image, as the first layer, is considered to be a superposition of a number of texture exemplars, possibly overlapped, from the second layer. Each texture exemplar, or simply texem, is characterised by mean values and corresponding variances. Each set of these texems may comprise various sizes from different image scales. We explore Gaussian mixture models in learning these texem representations, and show two different applications: novelty detection and image segmentation.

4.1. Introduction

Texture is one of the most important characteristics in identifying objects and understanding surfaces. There are numerous texture features reported in the literature, with some covered elsewhere in this book, used to perform texture representation and analysis: co-occurrence matrices, Laws texture energy measures, run-lengths, autocorrelation, and Fourier-domain features are some of the most common ones used in a variety of applications.

Some textures display complex patterns but appear visually regular on a large scale, e.g. textile and web. Thus, it is relatively easier to extract their dominant texture features or to represent their characteristics by exploiting their regularity and periodicity. However, for textures that exhibit complex, random appearance patterns, such as marble slabs or printed ceramic tiles

*Portions reprinted, with permission, from Ref. 1 by the same authors.

(see Fig. 4.1), where the textural primitives are randomly placed, it becomes more difficult to generalise texture primitives and their spatial relationships.

Fig. 4.1. Example marble tiles from the same family whose patterns are different but visually consistent.

As well as pixel intensity interactions, colour plays an important role in understanding texture, compounding the problem when random textures are involved. There has been a limited but increasing amount of work on colour texture analysis recently. Most of these borrow from methods designed for graylevel images. Direct channel separation followed by linear transformation is the common approach to adapting graylevel texture analysis methods to colour texture analysis, e.g. Caelli and Reye[2] processed colour images in RGB channels using multiscale isotropic filtering. Features from each channel were then extracted and later combined for classification. Several works have transformed the RGB colour space to other colour spaces to perform texture analysis so that chromatic channels are separated from the luminance channel, e.g. Refs. 3–6. For example, Liapis *et al.*[6] transformed colour images into the $L^*a^*b^*$ colour space in which discrete wavelet frame transform was performed in the L channel while local histograms in the a and b channels were used as chromatic features.

The importance of extracting correlation between the channels for colour texture analysis has been widely addressed with one of the earliest attempts reported in 1982.[7] Panjwani and Healey[8] devised an MRF model to encode the spatial interaction within and between colour channels. Thai and Healey[9] applied multiscale opponent features computed from Gabor filter responses to model intra-channel and inter-channel interactions. Mirmehdi and Petrou[10] perceptually smoothed colour image textures in a multiresolution sense before segmentation. Core clusters were then obtained from the coarsest level and initial probabilities were propa-

gated through finer levels until full segmentation was achieved. Simultaneous auto-regressive models and co-occurrence matrices have also been used to extract the spatial relationship within and between RGB channels.[11,12]

There has been relatively limited effort to develop fully 3D models to represent colour textures. The 3D data space is usually factorised, i.e. involving channel separation, then the data is modelled and analysed using lower dimensional methods. However, such methods inevitably suffer from some loss of spectral information, as the colour image data space can only be approximately decorrelated. The epitome[13] provides a compact 3D representation of colour textures. The image is assumed to be a collection of epitomic primitives relying on raw pixel values in image patches. The neighbourhood of a central pixel in a patch is assumed statistically conditionally independent. A hidden mapping guides the relationship between the epitome and the original image. This compact representation method inherently captures the spatial and spectral interactions simultaneously.

In this chapter, we present a compact mixture representation of colour textures. Similar to the epitome model, the images are assumed to be generated from a superposition of image patches with added variations at each pixel position. However, we do not force the texture primitives into a single patch representation with hidden mappings. Instead, we use mixture models to derive several primitive representation, called texems, at various sizes and/or various scales. Unlike popular filter bank based approaches, such as Gabor filters, "raw" pixel values are used instead of filtering responses. This is motivated by several recent studies using non-filtering local neighbourhood approaches. For instance, Varma and Zisserman[14] have argued that textures can be analysed by just looking at small neighbourhoods, such as 7×7 patches, and achieve better performance than filtering based methods. Their results demonstrated that textures with global structures can be discriminated by examining the distribution of local measurements. Ojala *et al.*[15] have also advocated the use of local neighbourhood processing in the shape of local binary patterns as texture descriptors. Other works based on local pixel neighbourhoods are those which apply Markov Random Field models, e.g. Cohen *et al.*.[16]

We shall demonstrate two applications of the texem model to analyse random textures. The first is to perform novelty detection in random colour textures and the second is to segment colour images.

4.2. The Texem Model

In this section, we present a two-layer generative model (see Fig. 4.2), in which an image in the first layer is assumed to be generated by superposition of a small number of image patches of various sizes from the second layer with added Gaussian noise at each pixel position. We define each texem as a mean image patch associated with a corresponding variance which controls its variation. The form of the texem variance can vary according to the learning scheme used. The generation process can be naturally modelled by mixture models with a bottom-up procedure.

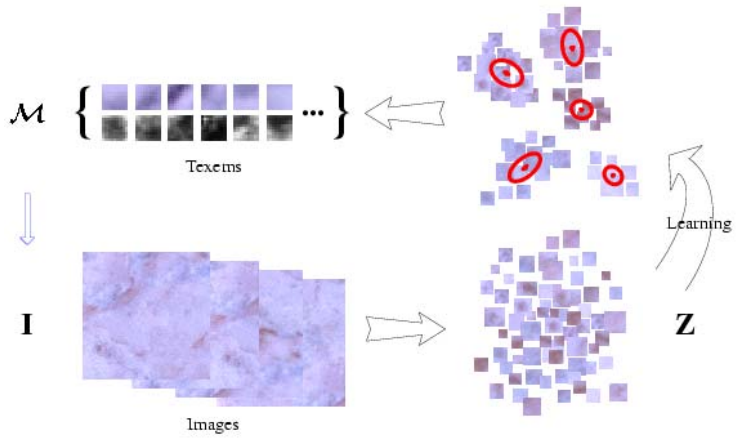

Fig. 4.2. An illustration of the two-layer structure of the texem model and its bottom-up learning procedure.

Next, we detail the process of extracting texems from a single sample image with each texem containing some of the overall textural primitive information. We shall use two different mixture models. The first is for graylevel images in which we vectorise the image patches and apply a Gaussian mixture model to obtain the texems. In the second, colour textures are represented by texems using a mixture model learnt based on joint Gaussian distributions within local neighbourhoods. This extension of texems to colour analysis is examined against other alternatives based on channel separation. We also introduce multiscale texem representations to drastically reduce the overall computational effort.

4.2.1. *Graylevel texems*

For graylevel images, we use a Gaussian mixture model to obtain the texems in a simple and efficient manner.[17] The original image $\mathbf{I}$ is broken down into a set of P patches $\mathbf{Z} = \{\mathbf{Z}_i\}_{i=1}^{P}$, each containing pixels from a subset of image coordinates. The shape of the patches can be arbitrary, but in this study we used square patches of size $d = N \times N$. The patches may overlap and can be of various sizes, e.g. as small as 5×5 to as large as required (however, for large window sizes one should ensure there are enough samples to populate the feature space). We group the patches of sample images into clusters, depending on the patch size, and describe the clusters using the Gaussian mixture model. Here, each texem, denoted as $\mathbf{m}$, is defined by a mean, $\boldsymbol{\mu}$, and a corresponding covariance matrix, $\boldsymbol{\omega}$, i.e. $\mathbf{m} = \{\boldsymbol{\mu}, \boldsymbol{\omega}\}$. We assume that there exist K texems, $\boldsymbol{\mathcal{M}} = \{\mathbf{m}_k\}_{k=1}^{K}$, $K \ll P$, for image $\mathbf{I}$ such that each patch in $\mathbf{Z}$ can be generated from a texem $\mathbf{m}$ with certain added variations.

To learn these texems the P patches are projected into a set of high dimensionality spaces. The number of these spaces is determined by the number of different patch sizes and their dimensions are defined by the corresponding value of d. Each pixel position contributes one coordinate of a space. Each point in a space corresponds to a patch in $\mathbf{Z}$. Then each texem represents a class of patches in the corresponding space. We assume that each class is a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\omega}_k$, which corresponds to $\mathbf{m}_k$ in the patch domain. Thus, given the k^{th} texem the probability of patch $\mathbf{Z}_i$ is computed as:

$$p(\mathbf{Z}_i|\mathbf{m}_k, \psi) = \mathcal{N}(\mathbf{Z}_i; \boldsymbol{\mu}_k, \boldsymbol{\omega}_k), \tag{4.1}$$

where $\psi = \{\boldsymbol{\alpha}_k, \boldsymbol{\mu}_k, \boldsymbol{\omega}_k\}_{k=1}^{K}$ is the parameter set containing $\boldsymbol{\alpha}_k$, which is the *prior* probability of kth texem constrained by $\sum_{k=1}^{K} \boldsymbol{\alpha}_k = 1$, the mean $\boldsymbol{\mu}_k$, and the covariance $\boldsymbol{\omega}_k$. Since all the texems $\mathbf{m}_k$ are unknown, we need to compute the density function of $\mathbf{Z}$ given the parameter set ψ by applying the definition of conditional probability and summing over k for $\mathbf{Z}_i$,

$$p(\mathbf{Z}_i|\psi) = \sum_{k=1}^{K} p(\mathbf{Z}_i|\mathbf{m}_k, \psi)\boldsymbol{\alpha}_k, \tag{4.2}$$

and then optimising the data log-likelihood expression of the entire set $\mathbf{Z}$, given by

$$\log p(\mathbf{Z}|K, \psi) = \sum_{i=1}^{P} \log(\sum_{k=1}^{K} p(\mathbf{Z}_i|\mathbf{m}_k, \psi)\boldsymbol{\alpha}_k). \tag{4.3}$$

Hence, the objective is to estimate the parameter set ψ for a given number of texems. Expectation Maximisation (EM) can be used to find the maximum likelihood estimate of our mixture density parameters from the given data set $\mathbf{Z}$. That is to find $\hat{\psi}$ where

$$\hat{\psi} = \arg\max \log(\mathcal{L}(\psi|\mathbf{Z})) = \arg\max_{\psi} \log p(\mathbf{Z}|K, \psi). \tag{4.4}$$

Then the two steps of the EM stage are as follows. The E-step involves a soft-assignment of each patch $\mathbf{Z}_i$ to texems, $\boldsymbol{\mathcal{M}}$, with an initial guess of the true parameters, ψ. This initialisation can be set randomly (although we use K-means to compute a simple estimate with K set as the number of texems to be learnt). We denote the intermediate parameters as $\psi^{(t)}$ where t is the iteration step. The likelihood of k^{th} texem given the patch $\mathbf{Z}_i$ may then be computed using Bayes' rule:

$$p(\mathbf{m}_k|\mathbf{Z}_i, \psi^{(t)}) = \frac{p(\mathbf{Z}_i|\mathbf{m}_k, \psi^{(t)})\boldsymbol{\alpha}_k}{\sum_{k=1}^{K} p(\mathbf{Z}_i|\mathbf{m}_k, \psi^{(t)})\boldsymbol{\alpha}_k}. \tag{4.5}$$

The M-step then updates the parameters by maximising the log-likelihood, resulting in new estimates:

$$\begin{aligned}
\hat{\boldsymbol{\alpha}}_k &= \frac{1}{P}\sum_{i=1}^{P} p(\mathbf{m}_k|\mathbf{Z}_i, \psi^{(t)}), \\
\hat{\boldsymbol{\mu}}_k &= \frac{\sum_{i=1}^{P} \mathbf{Z}_i p(\mathbf{m}_k|\mathbf{Z}_i, \psi^{(t)})}{\sum_{i=1}^{P} p(\mathbf{m}_k|\mathbf{Z}_i, \psi^{(t)})}, \\
\hat{\boldsymbol{\omega}}_k &= \frac{\sum_{i=1}^{P} (\mathbf{Z}_i - \hat{\boldsymbol{\mu}_k})(\mathbf{Z}_i - \hat{\boldsymbol{\mu}_k})^T p(\mathbf{m}_k|\mathbf{Z}_i, \psi^{(t)})}{\sum_{i=1}^{P} p(\mathbf{m}_k|\mathbf{Z}_i, \psi^{(t)})}.
\end{aligned} \tag{4.6}$$

The E-step and M-step are iterated until the estimations stabilise. Then, the texems can be easily obtained by projecting the learnt means and covariance matrices back to the patch representation space.

4.2.2. *Colour texems*

In this section, we explore two different schemes to extend texems to colour images with differing computational complexity and rate of accuracy.

4.2.2.1. *Texem analysis in separate channels*

More often than not, colour texture analysis is treated as a simple dimensional extension of techniques designed for graylevel images, and so

colour images are decomposed into separate channels to perform the same processes. However, this gives rise to difficulties in capturing both the inter-channel and spatial properties of the texture and special care is usually necessary. Alternatively, we can decorrelate the image channels using Principal Component Analysis (PCA) and then perform texems analysis in each independent channel separately. We prefer this approach and use it to compare against our full colour texem model introduced later.

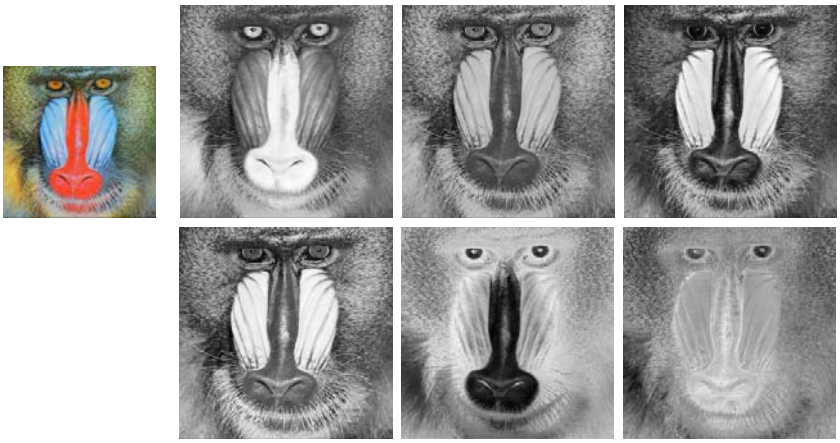

Fig. 4.3. Channel separation - first row: Original collage image; second row: individual RGB channels; third row: eigenchannel images.

Let $\mathbf{c}_i = [r_i, g_i, b_i]^T$ be a colour pixel, $\mathbf{C} = \{\mathbf{c}_i \in \mathcal{R}^3, i = 1, 2, ..., q\}$ be the set of q three dimensional vectors made up of the pixels from the image, and $\bar{\mathbf{c}} = \frac{1}{q}\sum_{\mathbf{c}\in\mathbf{C}} \mathbf{c}$ be the mean vector of $\mathbf{C}$. Then, PCA is performed on the mean-centred colour feature matrix $\mathbf{C}$ to obtain the eigenvectors $E = [e_1, e_2, e_3]$, $e_j \in \mathcal{R}^3$. Singular Value Decomposition can be used to obtain these principal components. The colour feature space determined by these eigenvectors is referred to as the reference eigenspace $\Upsilon_{\bar{\mathbf{c}},E}$, where the colour features are well represented. The image can then be projected onto this reference eigenspace:

$$\mathbf{C}' = \overrightarrow{PCA}(\mathbf{C}, \Upsilon_{\bar{\mathbf{c}},E}) = E^T(\mathbf{C} - \bar{\mathbf{c}}J_{1,q}), \tag{4.7}$$

where $J_{1,q}$ is a $1 \times q$ unit matrix consisting of all 1s. This results in three eigenchannels, in which graylevel texem analysis can be performed separately.

Figure 4.3 shows a comparison of RGB channel separation and PCA

eigenchannel decomposition. The R, G, and B channels shown in the second row are highly correlated to each other. Their spatial relationship (texture) within each channel are very similar to each other, i.e. the channels are not sufficiently complimentary. On the other hand, each eigenchannel in the third row exhibits its own characteristics. For example, the first eigenchannel preserves most of the textural information while the last eigenchannel maintains the ambient emphasis of the image. Later in Sec. 4.3, we demonstrate the benefit of decorrelating image channels in novelty detection.

4.2.2.2. *Full colour model*

By decomposing the colour image and analysing image channels individually, the inter-channel and intra-channel spatial interactions are not taken into account. To facilitate such interactions, we use a different formulation for texem representation and consequently change the inference procedure so that no vectorisation of image patches is required and colour images do not need to be transformed into separate channels. Contrary to the way graylevel texems were developed, where each texem was represented by a single multivariate Gaussian function, for colour texems we assume that pixels are statistically independent in each texem with Gaussian distribution at each pixel position in the texem. This is similar to the way the image epitome is generated by Jojic *et al.*[13] Thus, the probability of patch $\mathbf{Z}_i$ given the k^{th} texem can be formulated as a joint probability assuming neighbouring pixels are statistically conditionally independent, i.e.:

$$p(\mathbf{Z}_i|\mathbf{m}_k) = p(\mathbf{Z}_i|\boldsymbol{\mu}_k, \boldsymbol{\omega}_k) = \prod_{j \in S} \mathcal{N}(\mathbf{Z}_{j,i}; \boldsymbol{\mu}_{j,k}, \boldsymbol{\omega}_{j,k}), \tag{4.8}$$

where S is the pixel patch grid, $\mathcal{N}(\mathbf{Z}_{j,i}; \boldsymbol{\mu}_{j,k}, \boldsymbol{\omega}_{j,k})$ is a Gaussian distribution over $Z_{j,i}$, and $\boldsymbol{\mu}_{j,k}$ and $\boldsymbol{\omega}_{j,k}$ denote mean and covariance at the j^{th} pixel in the k^{th} texem. Similarly to Eq. (4.2) but using the component probability function in Eq. (4.8), we assume the following probabilistic mixture model:

$$p(\mathbf{Z}_i|\Theta) = \sum_{k=1}^{K} p(\mathbf{Z}_i|\mathbf{m}_k, \Theta)\boldsymbol{\alpha}_k, \tag{4.9}$$

where the parameters are $\Theta = \{\boldsymbol{\alpha}_k, \boldsymbol{\mu}_k, \boldsymbol{\omega}_k\}_{k=1}^{K}$ and can be determined by optimising the data log-likelihood given by

$$\log p(\mathbf{Z}|K, \Theta) = \sum_{i=1}^{P} \log(\sum_{k=1}^{K} p(\mathbf{Z}_i|\mathbf{m}_k, \Theta)\boldsymbol{\alpha}_k). \tag{4.10}$$

The EM technique can be used again to find the maximum likelihood estimate:

$$\hat{\Theta} = \arg\max \log(\mathcal{L}(\Theta|\mathbf{Z})) = \arg\max_{\Theta} \log p(\mathbf{Z}|K, \Theta). \tag{4.11}$$

The new estimates, denoted by $\hat{\boldsymbol{\alpha}}_k$, $\hat{\boldsymbol{\mu}}_k$, and $\hat{\boldsymbol{\omega}}_k$, are updated during the EM iterations:

$$\begin{aligned}
\hat{\boldsymbol{\alpha}}_k &= \frac{1}{P}\sum_{i=1}^{P} p(\mathbf{m}_k|\mathbf{Z}_i, \Theta^{(t)}),\\
\hat{\boldsymbol{\mu}}_k &= \{\hat{\boldsymbol{\mu}}_{j,k}\}_{j\in S},\\
\hat{\boldsymbol{\omega}}_k &= \{\hat{\boldsymbol{\omega}}_{j,k}\}_{j\in S},\\
\hat{\boldsymbol{\mu}}_{j,k} &= \frac{\sum_{i=1}^{P} \mathbf{Z}_{j,i} p(\mathbf{m}_k|\mathbf{Z}_i, \Theta^{(t)})}{\sum_{i=1}^{P} p(\mathbf{m}_k|\mathbf{Z}_i, \Theta^{(t)})},\\
\hat{\boldsymbol{\omega}}_{j,k} &= \frac{\sum_{i=1}^{P} (\mathbf{Z}_{j,i} - \hat{\boldsymbol{\mu}}_{j,k})(\mathbf{Z}_{j,i} - \hat{\boldsymbol{\mu}}_{j,k})^T p(\mathbf{m}_k|\mathbf{Z}_i, \Theta^{(t)})}{\sum_{i=1}^{P} p(\mathbf{m}_k|\mathbf{Z}_i, \Theta^{(t)})},
\end{aligned} \tag{4.12}$$

where

$$p(\mathbf{m}_k|\mathbf{Z}_i, \Theta^{(t)}) = \frac{p(\mathbf{Z}_i|\mathbf{m}_k, \Theta^{(t)})\boldsymbol{\alpha}_k}{\sum_{k=1}^{K} p(\mathbf{Z}_i|\mathbf{m}_k, \Theta^{(t)})\boldsymbol{\alpha}_k}. \tag{4.13}$$

The iteration continues till the values stabilise. Various sizes of texems can be used and they can overlap to ensure they capture sufficient textural characteristics. We can see that when the texem reduces to a single pixel size, Eq. (4.12) becomes Gaussian mixture modelling based on pixel colours.

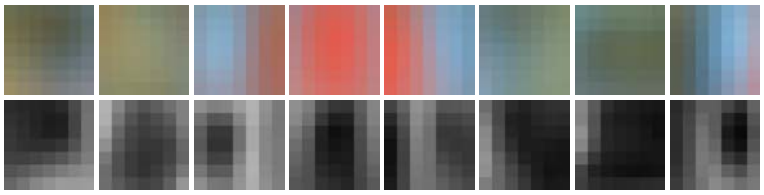

Fig. 4.4. Eight 7×7 texems extracted from the image in Fig. 4.3. Each texem $\mathbf{m}$ is defined by mean values (first row), $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_S]$, and corresponding variance images (second row), $\boldsymbol{\omega} = [\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, ..., \boldsymbol{\omega}_S]$, i.e. $\mathbf{m} = \{\boldsymbol{\mu}, \boldsymbol{\omega}\}$. Note, $\boldsymbol{\mu}_j$ is a 3×1 colour vector, and $\boldsymbol{\omega}_j$ is a 3×3 matrix characterising the covariance in the colour space. Each element $\boldsymbol{\omega}_j$ in $\boldsymbol{\omega}$ is visualised using total variance of $\boldsymbol{\omega}_j$, i.e. $\sum \text{diag}(\boldsymbol{\omega}_j)$.

Figure 4.4 illustrates eight 7×7 texems extracted from the Baboon image in Fig. 4.3. They are arranged according to their descending order of *priors*

$\boldsymbol{\alpha}_k$. We may treat each prior, $\boldsymbol{\alpha}_k$, as a measurement of the contribution from each texem. The image then can be viewed as a superposition of various sizes of image patches taken from the means of the texems, a linear combination, with added variations at each pixel position governed by the corresponding variances.

4.2.3. *Multiscale texems*

To capture sufficient textural properties, texems can be from as small as 3×3 to larger sizes such as 21×21. However, the dimension of the space patches $\mathbf{Z}$ are transformed into will increase dramatically as the dimension of the patch size increases. This means that a very large number of samples and high computational costs are needed in order to accurately estimate the probability density functions in very high dimensional spaces,[18] forcing the procurement of a large number of training samples.

Instead of generating variable-size texems, fixed size texems can be learnt in multiscale. This will result in (multiscale) texems with a small size, e.g. 5×5. Besides computational efficiency, exploiting information at multiscale offers other advantages over single-scale approaches. Characterising a pixel based on local neighbourhood pixels can be more effectively achieved by examining various neighbourhood relationships. The corresponding neighbourhood at coarser scale obviously offers larger spatial interactions. Also, processing at multiscale ensures the capture of the optimal resolution, which is often data dependent. We shall investigate two different approaches for texems analysis in multiscale.

4.2.3.1. *Texems in separate scales*

First, we learn small fixed size texems in separate scales of a Gaussian pyramid. Let us denote $\mathbf{I}^{(n)}$ as the n^{th} level image of the pyramid, $\mathbf{Z}^{(n)}$ as all the image patches extracted from $\mathbf{I}^{(n)}$, l as the total number of levels, and $S^{\downarrow}$ as the down-sampling operator. We then have

$$\mathbf{I}^{(n+1)} = S^{\downarrow} G_{\sigma}(\mathbf{I}^{(n)}), \qquad \forall n, n = 1, 2, ..., l-1, \tag{4.14}$$

where G_{σ} denotes the Gaussian convolution. The finest scale layer is the original image, $\mathbf{I}^{(1)} = \mathbf{I}$. We then extract multiscale texems from the image pyramid using the method presented in the previous section. Similarly, let $\mathbf{m}^{(n)}$ denote the n^{th} level of multiscale texems and $\Theta^{(n)}$ the parameters associated at the same level.

During the EM process, the stabilised estimation of a coarser level is used as the initial estimation for the finer level, i.e.

$$\hat{\Theta}^{(n,t=0)} = \Theta^{(n+1)}, \tag{4.15}$$

which hastens the convergence and achieves a more accurate estimation.

4.2.3.2. *Multiscale texems using branch partitioning*

Starting from the pyramid layout described above, each pixel in the finest level can trace its parent pixel back to the coarsest level forming a unique route or branch. Take the full colour texem for example, the conditional independence assumption amongst pixels within the local neighbourhood shown in Eq. (4.8) makes the parameter estimation tractable. Here, we assume pixels *in the same branch* are conditionally independent, i.e.

$$p(\mathbf{Z}_i|\mathbf{m}_k) = p(\mathbf{Z}_i|\boldsymbol{\mu}_k, \boldsymbol{\omega}_k) = \prod_{n=1}^{l} \mathcal{N}(\mathbf{Z}_i^{(n)}; \boldsymbol{\mu}_k^{(n)}, \boldsymbol{\omega}_k^{(n)}), \tag{4.16}$$

where $\mathbf{Z}_i$ here is a branch of pixels, and $\mathbf{Z}_i^{(n)}$, $\boldsymbol{\mu}_k^{(n)}$, and $\boldsymbol{\omega}_k^{(n)}$ are the colour pixel at level n in ith branch, mean at level n of kth texem, and variance at level n of kth texem, respectively. This is essentially the same form as Eq. (4.8), hence, we can still use the EM procedure described previously to derive the texems. However, the image is not partitioned into patches, but rather laid out in multiscale first and then separated into branches, i.e pixels are collected across scales, instead of from its neighbours.

4.2.4. *Comments*

The texem model is motivated from the observation that in random texture surfaces of the same family, the pattern may appear to be different in textural manifestation from one sample to another, however, the visual impression and homogeneity remains consistent. This suggests that the random pattern can be described with a few textural primitives.

In the texem model, the image is assumed to be a superposition of patches with various sizes and even various shapes. The variation at each pixel position in the construction of the image is embedded in each texem. Thus, it can be viewed as a two-layer *generative* statistical model. The image $\mathbf{I}$, in the first layer, is generated from a collection of texems $\boldsymbol{\mathcal{M}}$ in the second layer, i.e. $\boldsymbol{\mathcal{M}} \rightarrow \mathbf{I}$. In deriving the texem representations from an image or a set of images, a bottom-up learning process can be used as

presented in this chapter. Figure 4.2 illustrates the two-layer structure and the bottom-up learning procedure.

Relationship to Textons - Both the texem and the texton models characterise textural images by using micro-structures. Textons were first formally introduced by Julesz[19] as fundamental image structures, such as elongated blobs, bars, crosses, and terminators, and were considered as atoms of pre-attentive human visual perception. Zhu *et al.*[20] define textons using the superposition of a number of image bases, such as Laplacian of Gaussians and Gabors, selected from an over-complete dictionary. However, the texem model is significantly different from the texton model in that (i) it relies directly on subimage patches instead of using base functions, and (ii) it is an implicit, rather than an explicit, representation of primitives. The design of a bank of base functions to obtain sensible textons is non-trivial and likely to be application dependent. Much effort is needed to explicitly extract visual primitives (textons), such as blobs, but in the proposed model, each texem is an encapsulation of texture primitive(s). Not using base functions also allows texems more flexibility to deal with multi-spectral images.

4.3. Novelty Detection

In this section, we show an application of the texem model to defect detection on ceramic tile surfaces exhibiting rich and random texture patterns.

Visual surface inspection tasks are concerned with identifying regions that deviate from defect-free samples according to certain criteria, e.g. pattern regularity or colour. Machine vision techniques are now regularly used in detecting such defects or imperfections on a variety of surfaces, such as textile, ceramics tiles, wood, steel, silicon wafers, paper, meat, leather, and even curved surfaces, e.g. Refs. 16 and 21–23. Generally, this detection process should be viewed as different to texture segmentation, which is concerned with splitting an image into homogeneous regions. Neither the defect-free regions nor the defective regions have to be texturally uniform. For example, a surface may contain totally different types of defects which are likely to have different textural properties. On the other hand, a defect-free sample should be processed without the need to perform "segmentation", no matter how irregular and unstationary the texture.

In an application such as ceramic tile production, the images under inspection may appear different from one surface to another due to the random texture patterns involved. However, the visual impression of the

same product line remains consistent. In other words, there exist textural primitives that impose consistency within the product line. Figure 4.1 shows three example tile images from the same class (or production run) decorated with a marble texture. Each tile has different features on its surface, but they all still exhibit a consistent visual impression. One may collect enough samples to cover the range of variations and this approach has been widely used in texture classification and defect detection, e.g. for textile defects.[24] It usually requires a large number of non-defective samples and lengthy training stages; not necessarily practical in a factory environment. Additionally, defects are usually unpredictable.

Instead of the traditional classification approach, we learn texems, in an unsupervised fashion, from a very small number of training samples. The texems encapsulate the texture or visual primitives. As the images of the same (tile) product contain the same textural elements, the texems can be used to examine the same source similarity, and detect any deviations from the norm as defects.

4.3.1. *Unsupervised training*

Texems lend themselves well to performing unsupervised training and testing for novelty detection. This is achieved by automatically determining the threshold of statistical texture variation of defect-free samples at each resolution level. For training, a small number of defect free samples (e.g. 4 or 5 only) are arranged within the multiscale framework, and patches with the same texem size are extracted. The probability of a patch $\mathbf{Z}_i^{(n)}$ belonging to texems in the corresponding n^{th} scale is:

$$p(\mathbf{Z}_i^{(n)}|\Theta^{(n)}) = \sum_{k=1}^{K^{(n)}} p(\mathbf{Z}_i^{(n)}|\mathbf{m}_k^{(n)}, \Theta^{(n)})\boldsymbol{\alpha}_k^{(n)}, \tag{4.17}$$

where $\Theta^{(n)}$ represents the parameter set for level n, $\mathbf{m}_k^{(n)}$ is the k^{th} texem at the n^{th} image pyramid level, and $p(\mathbf{Z}_i^{(n)}|\mathbf{m}_k^{(n)}, \Theta^{(n)})$ is a product of Gaussian distributions shown in Eq. (4.9) with parameters associated to texem set $\boldsymbol{\mathcal{M}}$. Based on this probability function, we then define a novelty score function as the negative log likelihood:

$$\mathcal{V}(\mathbf{Z}_i^{(n)}|\Theta^{(n)}) = -\log p(\mathbf{Z}_i^{(n)}|\Theta^{(n)}). \tag{4.18}$$

The lower the novelty score, the more likely the patch belongs to the same family and vice versa. Thus, it can be viewed as a same source simi-

larity measurement. The distribution of the scores for all the patches $\mathbf{Z}^{(n)}$ at level n of the pyramid forms a 1D novelty score space which is not necessarily a simple Gaussian distribution. In order to find the upper bound of the novelty score space of defect-free patches (or the lower bound of data likelihood), K-means clustering is performed in this space to approximately model the space. The cluster with the maximum mean is the component of the novelty score distribution at the boundary between good and defective textures. This component is characterised by mean $u^{(n)}$ and standard deviation $\sigma^{(n)}$. This K-means scheme replaces the single Gaussian distribution assumption in the novelty score space, which is commonly adopted in a parametric classifier in novelty detection, e.g. Ref. 25 and for which the correct parameter selection is critical. Instead, dividing the novelty score space and finding the critical component, here called the boundary component, can effectively lower the parameter sensitivity. The value of K should be generally small (we empirically fixed it at 5). It is also notable that a single Gaussian classifier is a special case of the above scheme, i.e. when $K = 1$. The maximum novelty score (or the minimum data likelihood), $\Lambda^{(n)}$ of a patch $\mathbf{Z}_i^{(n)}$ at level n across the training images is then established as:

$$\Lambda^{(n)} = u^{(n)} + \lambda\sigma^{(n)}, \tag{4.19}$$

where λ is a simple constant. This completes the training stage in which, with only a few defect-free images, we determine the texems and an automatic threshold for marking new image patches as good or defective.

4.3.2. *Novelty detection and defect localisation*

In the testing stage, the image under inspection is again layered into a multiscale framework and patches at each pixel position $\mathbf{x}$ at each level n are examined against the learnt texems. The probability for each patch and its novelty score are then computed using Eqs. (4.17) and (4.18) and compared to the maximum novelty score, determined by $\Lambda^{(n)}$, at the corresponding level. Let $Q^{(n)}(\mathbf{x})$ be the novelty score map at the n^{th} resolution level. Then, the potential defect map, $\mathcal{D}^{(n)}(\mathbf{x})$, at level n is:

$$\mathcal{D}^{(n)}(\mathbf{x}) = \begin{cases} 0 & \text{if } Q^{(n)}(\mathbf{x}) \leq \Lambda^{(n)} \\ Q^{(n)}(\mathbf{x}) - \Lambda^{(n)} & \text{otherwise,} \end{cases} \tag{4.20}$$

$\mathcal{D}^{(n)}(\mathbf{x})$ indicates the probability of there being a defect. Next, the information coming from all the resolution levels must be consolidated to build

the certainty of the defect at position $\mathbf{x}$. We follow a framework[22] which combines information from different levels of a multiscale pyramid and reduces false alarms. It assumes that a defect must appear in at least two adjacent resolution levels for it to be certified as such. Using a logical AND, implemented through the geometric mean of every pair of adjacent levels, we initially obtain a set of combined maps as:

$$\mathcal{D}^{(n,n+1)}(\mathbf{x}) = [\mathcal{D}^{(n)}(\mathbf{x})\mathcal{D}^{(n+1)}(\mathbf{x})]^{1/2}. \tag{4.21}$$

Note each $\mathcal{D}^{(n+1)}(\mathbf{x})$ is scaled up to be the same size as $\mathcal{D}^{(n)}(\mathbf{x})$. This operation reduces false alarms and yet preserves most of the defective areas. Next, the resulting $\mathcal{D}^{(1,2)}(\mathbf{x})$, $\mathcal{D}^{(2,3)}(\mathbf{x})$, ..., $\mathcal{D}^{(l-1,l)}(\mathbf{x})$ maps are combined in a logical OR, as the arithmetic mean, to provide

$$\mathcal{D}(\mathbf{x}) = \frac{1}{l-1}\sum_{n=1}^{l-1}\mathcal{D}^{(n,n+1)}(\mathbf{x}), \tag{4.22}$$

where $\mathcal{D}(\mathbf{x})$ is the final consolidated map of (the joint contribution of) all the defects across all resolution scales of the test image.

The multiscale, unsupervised training, and novelty detection stages are applied in a similar fashion as described above in the cases of graylevel and full colour model texem methods. In the separate channel colour approaches (i.e. before and after decorrelation) the final defective maps from each channel are ultimately combined.

4.3.3. *Experimental results*

The texem model is initially applied to the detection of defects on ceramic tiles. We do not evaluate the quality of the localised defects found (against a groundtruth) since the defects in our data set are difficult to manually localise. However, whole tile classification rates, based on overall "defective" and "defect-free" labelling by factory-floor experts is presented. In order to evaluate texems, the result of experiments on texture collages made from textures in the MIT VisTex texture database[26] is outlined. A comparative study of three different approaches to texem analysis on colour images and a Gabor filter bank based method is given.

4.3.3.1. *Ceramic Tile Application*

We applied the proposed *full colour texem model* to a variety of randomly textured tile data sets with different types of defects including physical damage, pin holes, textural imperfections, and many more. The 256×256

test samples were pre-processed to assure homogeneous luminance, spatially and temporally. In the experiments, only five defect-free samples were used to extract the texems and to determine the upper bound of the novelty scores $\Lambda^{(n)}$. The number of texems at each resolution level were empirically set to 12, and the size of each texem was 5×5 pixels. The number of multiscale levels was $l = 4$. These parameters were fixed throughout our experiments on a variety of random texture tile prints.

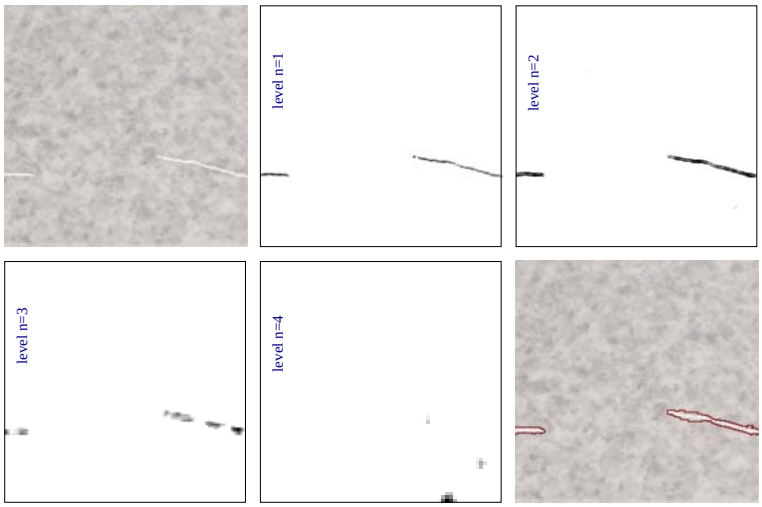

Fig. 4.5. Localising textural defects - from top left to bottom right: original defective tile image, detected defective regions at different levels $n = 1, 2, 3, 4$, and the final defective region superimposed on the original image.

Figure 4.5 shows a random texture example with defective regions introduced by physical damage. The potentially defective regions detected at each resolution level n, $n = 1, ..., 4$, are marked on the corresponding images in Fig. 4.5. It can be seen that the texems show good sensitivity to the defective region at different scales. As the resolution progresses from coarse to fine, additional evidence for the defective region is gathered. The final image shows the defect superimposed on the original image. As mentioned earlier, the defect fusion process can eliminate false alarms, e.g. see the extraneous false alarm regions in level $n = 4$ which disappear after the operations in Eqs. (4.21) and (4.22).

More examples of different random textures are shown in Fig. 4.6. In each family of patterns, the textures are varying but have the same visual

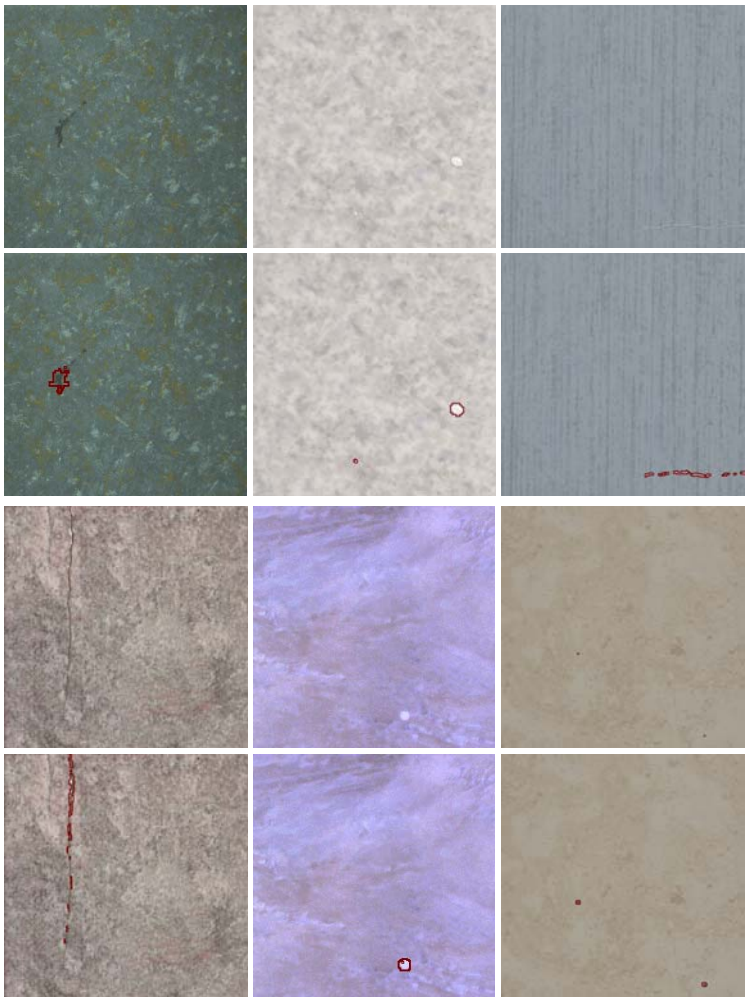

Fig. 4.6. Defect localisation (different textures) - The first row shows example images from three different tile families with different chromato-textural properties. Defects shown in the next row, from left to right, include print error, surface bumps, and thin cracks. The third row shows another three images from three different tile families. Defects shown in the last row, from left to right, include cracks and print errors.

impression. In each case the proposed method could find structural and chromatic defects of various shapes and sizes.

Figure 4.7 shows three examples when using *graylevel* texems. Various defects, such as print errors, bumps, and broken corner, are successfully

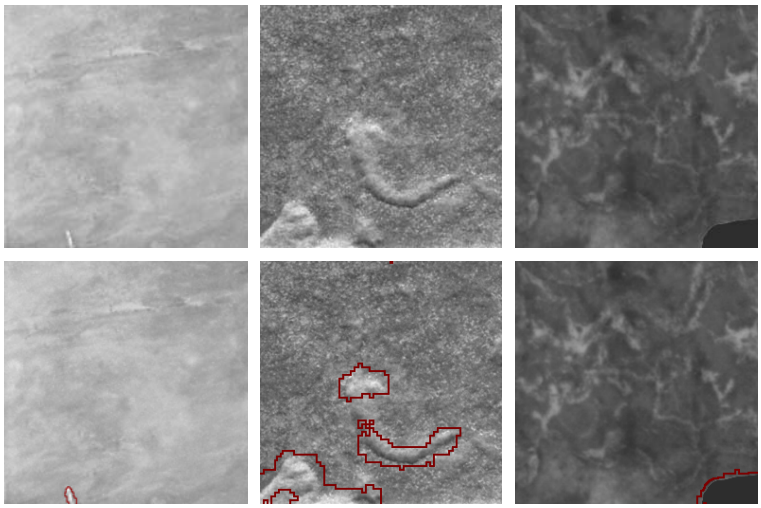

Fig. 4.7. Graylevel texems defect localisation.

detected. Graylevel texems were found adequate for most defect detection tasks where defects were still reasonably visible after converting from colour to gray scale. However, colour texems were found to be more powerful in localising defects and better discriminants in cases involving chromatic defects. Two examples are compared in Fig. 4.8. The first shows a tile image with a defective region, which is not only slightly brighter but also less saturated in blue. The colour texem model achieved better results in localising the defect than the graylevel one. The second row in Fig. 4.8 demonstrates a different type of defect which clearly possesses a different hue from the background texture. The colour texems found more affected regions, more accurately.

The full colour texem model was tested on 1018 tile samples from ten different families of tiles consisting of 561 defect-free samples and 457 defective samples. It obtained a defect detection accuracy rate of 91.1%, with sensitivity at 92.6% and specificity at 89.8%. The graylevel texem method was tested on 1512 graylevel tile images from eight different families of tiles consisting of 453 defect-free samples and 1059 defective samples. It obtained an overall accuracy rate of 92.7%, with sensitivity at 95.9% specificity at 89.5%. We compare the performance of graylevel and colour texem models on the same dataset in later experiments.

As patches are extracted from each pixel position at each resolution

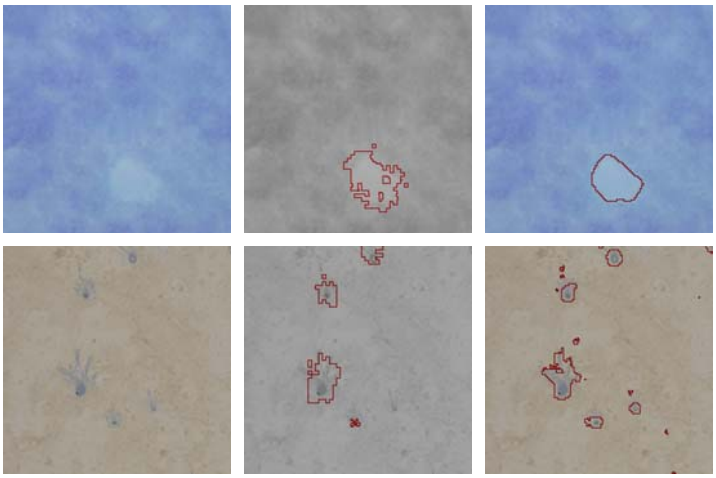

Fig. 4.8. Defect localisation comparison - left column: original texture with print errors, middle column: results using graylevel texems, right column: results using colour texems.

level, a typical *training stage* involves examining a very large number of patches. For the graylevel texem model, this takes around 7 minutes, on a 2.8GHz Pentium 4 Processor running Linux with 1GB RAM, to learn the texems in multiscale and to determine the thresholds for novelty detection. The testing stage then requires around 12 seconds to inspect one tile image. The full colour texem model is computationally more expensive and can be more than 10 times slower. However, this can be reduced to the same order as the graylevel version by performing window-based, rather than pixel-based, examination at the training and testing stages.

4.3.3.2. *Evaluation using VisTex Collages*

For performance evaluation, 28 image collages were generated (see some in Fig. 4.10) from textures in VisTex.[26] In each case the background is the learnt texture for which colour texems are produced and the foreground (disk, square, triangle, and rhombus) is treated as the novelty to be detected. This is not a texture segmentation exercise, but rather defect segmentation. The textures used were selected to be particularly similar in nature in the foreground and the background, e.g. see the collages in the first or third columns of Fig. 4.10. We use *specificity* for how accurately defect-free samples were classified, *sensitivity* for how accurately defective samples were classified, and *accuracy* as the correct classification rate of all

samples:

$$\begin{cases} spec. = \frac{N_t \cap N_g}{N_g} \times 100\% \\ sens. = \frac{P_t \cap P_g}{P_g} \times 100\% \\ accu. = \frac{N_t \cap N_g + P_t \cap P_g}{N_g + P_g} \times 100\% \end{cases} \tag{4.23}$$

where P is the number of defective samples, N is the number of defect-free samples, and the subscripts t and g denote the results by testing and groundtruth respectively. The foreground is set to occupy 50% of the whole image to allow the sensitivity and specificity measures have equal weights.

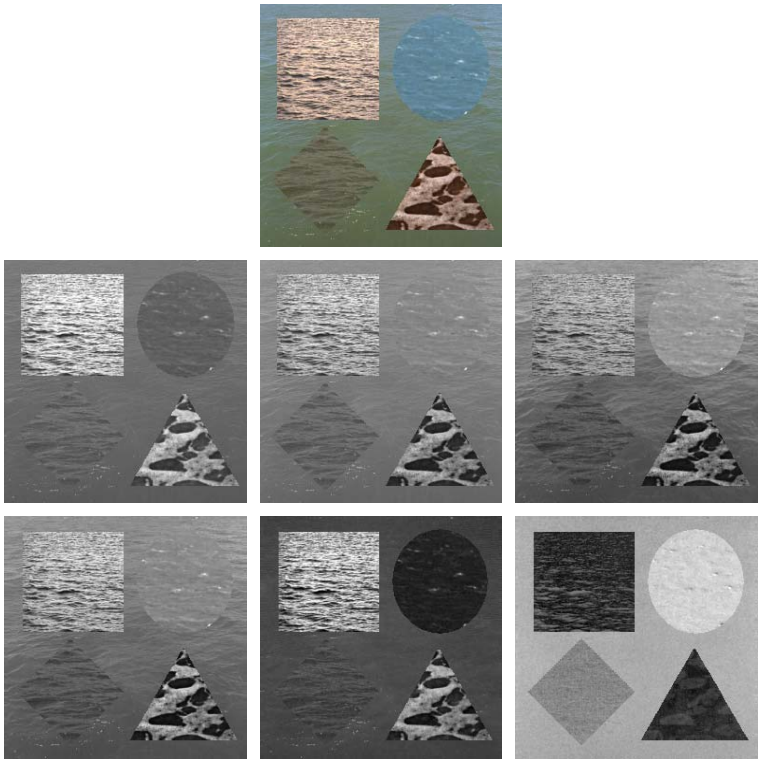

Fig. 4.9. Channel separation - first row: Original collage image; second row: individual RGB channels; third row: eigenchannel images.

We first compare the two different channel separation schemes in each case using graylevel texem analysis in the individual channels. For the RGB

channel separation scheme, defects detected in each channel were then combined to form the final defect map. For the eigenchannel separation scheme, the reference eigenspace from training images was first derived. As the patterns on each image within the same texture family can still be different, hence the individually derived principal components can also differ from one image to another. Furthermore, defective regions can affect the principal components resulting in different eigenspace responses from different training samples. Thus, instead of performing PCA on each training image separately, a single eigenspace was generated from several training images, resulting in a reference eigenspace in which defect-free samples are represented. Then, all new, previously unseen images under inspection were projected onto this eigenspace such that the transformed channels share the same principal components. Once we obtain the reference eigenspace, $\Upsilon_{\bar{\mathbf{c}},E}$, defect detection and localisation are performed in each of the three corresponding channels by examining the local context using the graylevel texem model, the same process as used in RGB channel separation scheme. Figure 4.9 shows a comparison of direct RGB channel separation and PCA based channel separation. The eigenchannels are clearly more differentiating.

Experimental results on the colour collages showed that the PCA based method achieved a significant improvement over the correlated RGB channels with an overall accuracy of 84.7% compared to 79.1% (see Table 4.1). Graylevel texem analysis in image eigenchannels appear to be a plausible approach to perform colour analysis with relatively economic computational complexity. However, the full colour texem model, which models inter-channel and intra-channel interactions simultaneously, improved the performance to an overall detection accuracy of 90.9%, 91.2% sensitivity and 90.6% specificity. Example segmentations (without any post-processing) of all the methods are shown in the last three rows of Fig. 4.10.

We also compared the proposed method against a non-filtering method using LBPs[15] and a Gabor filtering based novelty detection method.[22] The LBP coefficients were extracted from each RGB colour band. The estimation of the range of coefficient distributions for defect-free samples and the novelty detection procedures were the same as that described in Sec. 4.3.2. We found that LBP performs very poorly, but a more sophisticated classifier may improve the performance. Gabor filters have been widely used in defect detection, see Refs. 22 and 23 as typical examples. The work by Escofet et al.,[22] referred to here as Escofet's method, is the most comparable to ours, as it is (a) performed in a novelty detection framework and

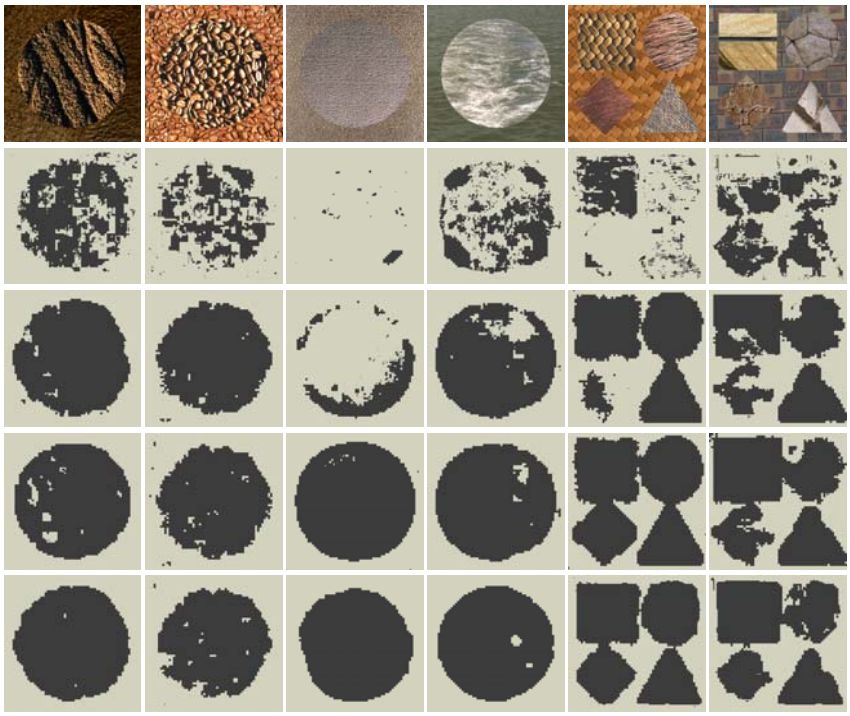

Fig. 4.10. Collage samples made up of materials such as foods, fabric, sand, metal, water, and novelty detection results without any post-processing. Rows from top: original images, Escofet *et al.*'s method, graylevel texems directly in RGB channels, graylevel texems in PCA decorrelated RGB eigenchannels, full colour texem model.

(b) uses the same defect fusion scheme across the scales. Thus, following Escofet's method to perform novelty detection on the synthetic image collages, the images were filtered through a set of 16 Gabor filters, comprising four orientations and four scales. The texture features were extracted from filtering responses. Feature distributions of defect-free samples were then used for novelty detection. The same logical process was used to combine defect candidates across the scales. An overall detection accuracy of 71.5% was obtained by Escofet's method; a result significantly lower than texems (see Table 4.2). Example results are shown in the second row of Fig. 4.10.

There are two important parameters in the texem model for novelty detection, the size of texems and the number of the texems. In theory, the size of the texems is arbitrary. Thus, it can easily cover all the necessary

Table 4.1. Novelty detection comparison: graylevel texems in image RGB channels and image eigenchannels (values are %s).

No.	RGB channels			Eigenchannels		
	spec.	sens.	accu.	spec.	sens.	accu.
1	81.7	100	90.7	82.0	100	90.9
2	80.7	100	90.2	80.8	100	90.3
3	87.6	99.9	93.7	82.4	100	91.1
4	94.3	97.2	95.7	93.9	95.7	94.8
5	87.3	30.7	59.3	77.9	99.6	88.6
6	76.6	100	88.2	77.8	100	88.8
7	96.0	93.4	94.7	90.1	98.6	94.3
8	87.8	97.7	92.7	85.6	95.3	90.4
9	85.5	52.0	68.9	76.1	100	87.9
10	92.2	25.2	59.1	77.8	99.2	88.4
11	89.1	33.6	61.6	80.3	97.2	88.6
12	82.5	88.4	85.4	79.5	97.7	88.5
13	93.5	47.8	70.9	93.0	49.0	71.2
14	80.9	99.9	90.3	81.1	100	90.5
15	98.7	55.3	77.2	98.3	74.8	86.7
16	84.5	78.1	81.3	86.5	92.7	89.6
17	75.1	60.8	67.9	62.3	87.9	73.8
18	64.9	69.5	67.2	60.9	91.9	74.8
19	75.1	60.0	67.5	57.0	87.4	72.2
20	83.9	91.8	87.8	85.4	90.0	87.7
21	78.6	97.3	87.8	88.4	98.4	93.4
22	88.5	49.8	69.4	79.5	76.3	77.9
23	98.2	44.5	71.6	96.6	34.8	66.0
24	60.6	69.8	65.2	64.5	86.8	75.7
25	58.7	100	79.4	64.8	99.9	82.3
26	84.1	91.6	87.9	76.5	94.2	85.3
27	73.2	87.8	80.5	64.7	99.9	82.3
28	74.5	88.3	81.4	65.7	94.6	80.1
Overall	82.7	75.4	**79.1**	78.9	90.8	**84.7**

spatial frequency range. However, for the sake of computational simplicity, a window size of 5×5 or 7×7 across all scales generally suffices. The number of texems can be automatically determined using model order selection methods, such as MDL, though they are usually computationally expensive. We used 12 texems in each scale for over 1000 tile images and collages and found reasonable, consistent performance for novelty detection.

Table 4.2. Novelty detection comparison: Escofet's method and the full colour texem model (values are %s).

No.	Escofet's Method			Colour Texems		
	spec.	sens.	accu.	spec.	sens.	accu.
1	95.6	82.7	89.2	91.9	99.9	95.9
2	96.9	83.7	90.3	84.4	100	92.1
3	96.1	61.5	79.0	91.1	99.8	95.4
4	98.0	53.1	75.8	97.0	92.9	95.0
5	98.8	1.5	50.7	92.1	98.8	95.4
6	96.6	70.0	83.4	96.3	98.6	97.4
7	98.9	26.8	63.2	98.6	79.4	89.0
8	91.4	74.4	83.0	89.6	99.8	94.7
9	90.8	49.0	70.1	86.4	100	93.1
10	94.3	7.2	51.2	92.8	99.6	96.2
11	94.6	8.6	52.1	96.3	90.8	93.6
12	86.9	44.0	65.7	88.4	98.8	93.5
13	96.8	71.0	84.0	91.0	91.9	91.5
14	90.7	95.2	93.0	82.5	100	91.1
15	98.4	27.2	63.2	96.5	76.3	86.5
16	95.5	43.0	69.3	96.3	71.2	83.8
17	80.0	56.5	68.2	83.5	98.7	91.1
18	73.9	60.4	67.2	83.9	96.5	90.2
19	84.9	52.0	68.4	90.4	71.3	80.9
20	94.4	52.0	73.2	95.1	88.8	91.9
21	94.0	48.9	71.6	95.8	75.9	85.9
22	95.8	23.4	60.0	92.2	72.0	82.2
23	97.1	35.1	66.5	93.6	67.8	80.9
24	89.4	46.4	67.9	81.6	98.1	89.8
25	82.6	92.9	87.7	88.3	100	93.9
26	94.5	55.3	74.9	94.3	92.2	93.2
27	93.9	36.5	65.2	85.9	98.9	92.4
28	81.2	55.3	68.3	82.0	95.2	88.6
Overall	92.2	50.5	**71.5**	90.6	91.2	**90.9**

4.4. Colour Image Segmentation

Clearly each patch from an image has a measurable relationship with each texem according to the *posteriori*, $p(\mathbf{m}_k|\mathbf{Z}_i, \Theta)$, which can be conveniently obtained using Bayes' rule in Eq. (4.13). Thus, every texem can be viewed as an individual textural class component, and the *posteriori* can be regarded as the component likelihood with which each pixel in the image can be labelled. Based on this, we present two different multiscale approaches to carry out segmentation. The first, interscale post-fusion, performs segmentation at each level separately and then updates the label probabilities

from coarser to finer levels. The second, branch partitioning, simplifies the procedure by learning the texems across the scales to gain efficiency.

4.4.1. *Segmentation with interscale post-fusion*

For segmentation, each pixel needs to be assigned a class label, $c = \{1, 2, ..., K\}$. At each scale n, there is a random field of class labels, $C^{(n)}$. The probability of a particular image patch, $\mathbf{Z}_i^{(n)}$, belonging to a texem (class), $c = k, \mathbf{m}_k^{(n)}$, is determined by the *posteriori* probability, $p(c = k, \mathbf{m}_k^{(n)} | \mathbf{Z}_i^{(n)}, \Theta^{(n)})$, simplified as $p(c^{(n)} | \mathbf{Z}_i^{(n)})$, given by:

$$p(c^{(n)} | \mathbf{Z}_i^{(n)}) = \frac{p(\mathbf{Z}_i^{(n)} | \mathbf{m}_k^{(n)}) \boldsymbol{\alpha}_k^{(n)}}{\sum_{k=1}^{K} p(\mathbf{Z}_i^{(n)} | \mathbf{m}_k^{(n)}) \boldsymbol{\alpha}_k^{(n)}}, \tag{4.24}$$

which is equivalent to the stabilised solution of Eq. (4.13). The class probability at given pixel location $(x^{(n)}, y^{(n)})$ at scale n then can be estimated as $p(c^{(n)} | (x^{(n)}, y^{(n)})) = p(c^{(n)} | \mathbf{Z}_i^{(n)})$. Thus, this labelling assignment procedure initially partitions the image in each individual scale. As the image is laid hierarchically, there is inherited relationship among parent and children pixels. Their labels should also reflect this relationship. Next, building on this initial labelling, the partitions across all the scales are fused together to produce the final segmentation map.

The class labels $c^{(n)}$ are assumed conditionally independent given the labelling in the coarser scale $c^{(n+1)}$. Thus, each label field $C^{(n)}$ is assumed only dependent on the previous coarser scale label field $C^{(n+1)}$. This offers efficient computational processing, while preserving the complex spatial dependencies in the segmentation. The label field $C^{(n)}$ becomes a Markov chain structure in the scale variable n:

$$p(c^{(n)} | c^{(>n)}) = p(c^{(n)} | c^{(n+1)}), \tag{4.25}$$

where $c^{(>n)} = \{c^{(i)}\}_{i=n+1}^{l}$ are the class labels at all coarser scales greater than the nth, and $p(c^{(l)} | c^{(l+1)}) = p(c^{(l)})$ as l is the coarsest scale. The coarsest scale segmentation is directly based on the initial labelling.

A quadtree structure for the multiscale label fields is used, and $c^{(l)}$ only contains a single pixel, although a more sophisticated context model can be used to achieve better interaction between child and parent nodes, e.g. a pyramid graph model.[27] The transition probability $p(c^{(n)} | c^{(n+1)})$ can be efficiently calculated numerically using a lookup table. The label assignments at each scale are then updated, from coarsest to the finest,

according to the joint probability of the data probability and the transition probability:

$$\begin{cases} \hat{c}^{(l)} = \arg\max_{c^{(l)}} \log p(c^{(l)}|(x^{(l)}, y^{(l)})), \\ \hat{c}^{(n)} = \arg\max_{c^{(n)}} \{\log p(c^{(n)}|(x^{(n)}, y^{(n)})) + \log p(c^{(n)}|c^{(n+1)})\} \quad \forall n < l. \end{cases} \tag{4.26}$$

The segmented regions will be smooth and small isolated holes are filled.

4.4.2. *Segmentation using branch partitioning*

As discussed earlier in Sec. 4.2.3, an alternative multiscale approach can be used by partitioning the multiscale image into branches based on hierarchical dependency. By assuming that pixels within the same branch are conditionally independent to each other, we can directly learn multiscale colour texems using Eq. (4.16). The class labels then can be directly obtained without performing interscale fusion by evaluating the component likelihood using Bayes' rule: $p(c|\mathbf{Z}_i) = p(\mathbf{m}_k|\mathbf{Z}_i, \Theta)$, where $\mathbf{Z}_i$ is a branch of pixels. The label assignment for $\mathbf{Z}_i$ is then according to:

$$\hat{c} = \arg\max_{c} p(c|\mathbf{Z}_i). \tag{4.27}$$

Thus, we simplify the approach presented in Sec. 4.4.1 by avoiding the inter-scale fusion after labelling each scale.

4.4.3. *Texem Grouping for Multimodal Texture*

A textural region may contain multiple visual elements and display complex patterns. A single texem might not be able to fully represent such textural regions, hence, several texems can be grouped together to jointly represent "multimodal" texture regions. Here, we use a simple but effective method proposed by Manduchi[28]to group texems. The basic strategy is to group some of the texems based on their spatial coherence. The grouping process simply takes the form:

$$\hat{p}(\mathbf{Z}_i|c) = \frac{1}{\hat{\beta}_c} \sum_{k \in G_c} p(\mathbf{Z}_i|\mathbf{m}_k)\alpha_k, \qquad \hat{\beta}_c = \sum_{k \in G_c} \alpha_k, \tag{4.28}$$

where G_c is the group of texems that are combined together to form a new cluster c which labels the different texture classes, and $\hat{\beta}_c$ is the *priori* for

new cluster c. The mixture model can thus be reformulated as:

$$p(\mathbf{Z}_i|\Theta) = \sum_{c=1}^{\hat{K}} \hat{p}(\mathbf{Z}_i|\mathbf{m}_k)\hat{\beta}_c, \tag{4.29}$$

where $\hat{K}$ is the desired number of texture regions. Equation (4.29) shows that pixel i in the centre of patch $\mathbf{Z}_i$ will be assigned to the texture cluster c which maximises $\hat{p}(\mathbf{Z}_i|c)\hat{\beta}_c$:

$$c = \arg\max_c \hat{p}(\mathbf{Z}_i|c)\hat{\beta}_c = \arg\max_c \sum_{k\in G_c} p(\mathbf{Z}_i|\mathbf{m}_k)\alpha_k. \tag{4.30}$$

The grouping in Eq. (4.29) is carried out based on the assumption that the *posteriori* probabilities of grouped texems are typically spatially correlated. The process should minimise the decrease of model descriptiveness, D, which is defined as:[28]

$$D = \sum_{j=1}^{K} D_j, \quad D_j = \int p(\mathbf{Z}_i|\mathbf{m}_j)p(\mathbf{m}_j|\mathbf{Z}_i)d\mathbf{Z}_i = \frac{E[p(\mathbf{m}_j|\mathbf{Z}_i)^2]}{\alpha_j}, \tag{4.31}$$

where $E[.]$ is the expectation computed with respect to $p(\mathbf{Z}_i)$. In other words, the compacted model should retain as much descriptiveness as possible. This is known as the Maximum Description Criterion (MDC). The descriptiveness decreases drastically when well separated texem components are grouped together, but decreases very slowly when spatially correlated texem component distributions merge together. Thus, the texem grouping should search for smallest change in descriptiveness, ΔD. It can be carried out by greedily grouping two texem components, $\mathbf{m}_a$ and $\mathbf{m}_b$, at a time with minimum ΔD_{ab}:

$$\Delta D_{ab} = \frac{\alpha_b D_a + \alpha_a D_b}{\alpha_a + \alpha_b} - \frac{2E[p(\mathbf{m}_a|\mathbf{Z}_i)p(\mathbf{m}_b|\mathbf{Z}_i)]}{\alpha_a + \alpha_b}. \tag{4.32}$$

We can see that the first term in Eq. (4.32) is the maximum possible descriptiveness loss when grouping two texems, and the second term in Eq. (4.32) is the normalised cross correlation between the two texem component distributions. Since one texture region may contain different texem components that are significantly different to each other, it is beneficial to smooth the *posteriori* as proposed by Manduchi[28] such that a pixel that originally has high probability to just one texem component will be softly assigned to a number of components that belong to the same "multimodal" texture. After grouping, the final segmentation map is obtained according to Eq. (4.30).

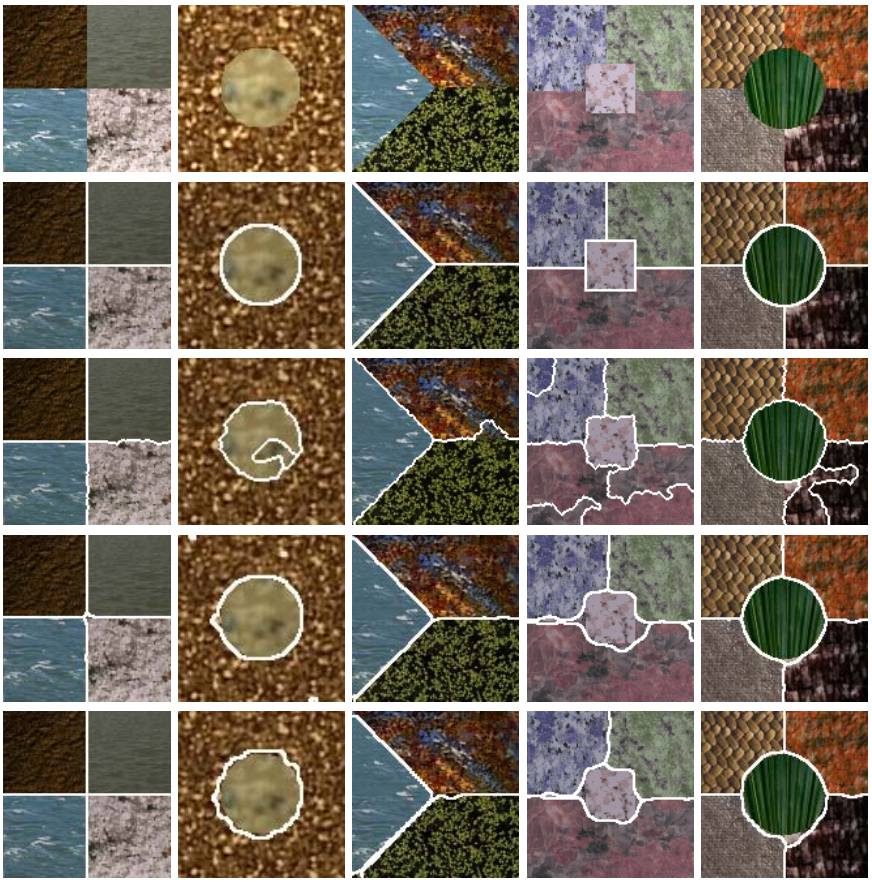

Fig. 4.11. Testing on synthetic images - first row: original image collages, second row: groundtruth segmentations, third row: JSEG results, fourth row: results of the proposed method using interscale post-fusion, last row: results of the proposed method using branch partitioning.

4.4.4. *Experimental Results*

Here, we present experimental results using colour texem based image segmentation with a brief comparison with the well-known JSEG technique.[29]

Figure 4.11 shows example results on five different texture collages with the original image in the first row, groundtruth segmentations in the second row, the JSEG result in the third row, the proposed interscale post-fusion method in the fourth row, and the proposed branch partition method in the final row. The two proposed schemes have similar performance, while

JSEG tends to over-segment which partially arises due to the lack of prior knowledge of number of texture regions.

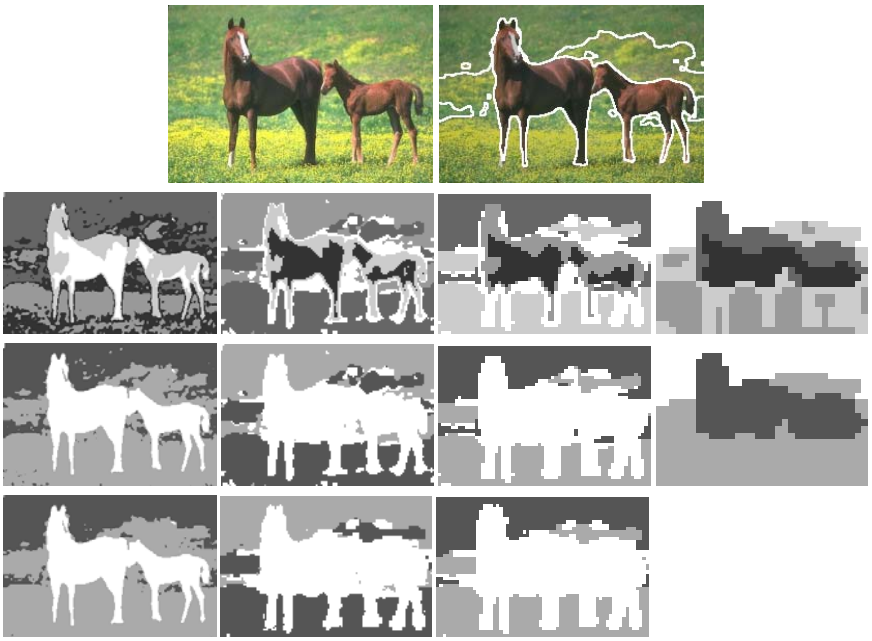

Fig. 4.12. An example of the interscale post-fusion method followed by texem grouping - first row: original image and its segmentation result, second row: initial labelling of 5 texem classes for each scale, third row: updated labelling after grouping 5 texems into 3, fourth row: results of interscale fusion.

Figure 4.12 focuses on the interscale post-fusion technique followed by texem grouping. The original image and the final segmentation are shown at the top. The second row shows the initial labelling of 5 texem classes for each pyramid level. The texems are grouped to 3 classes as seen in the third row. Interscale fusion is then performed and shown in the last row. Note there is no fusion in the fourth (coarsest) scale.

Three real image examples are given in Fig. 4.13. For each image, we show the original images, its JSEG segmentation and the results of the two proposed segmentation methods. The interscale post-fusion method produced finer borders but is a slower technique.

The results shown demonstrate that the two proposed methods are more able in modelling textural variations than JSEG and are less prone to over-segmentation. However, it is noted that JSEG does not require the number

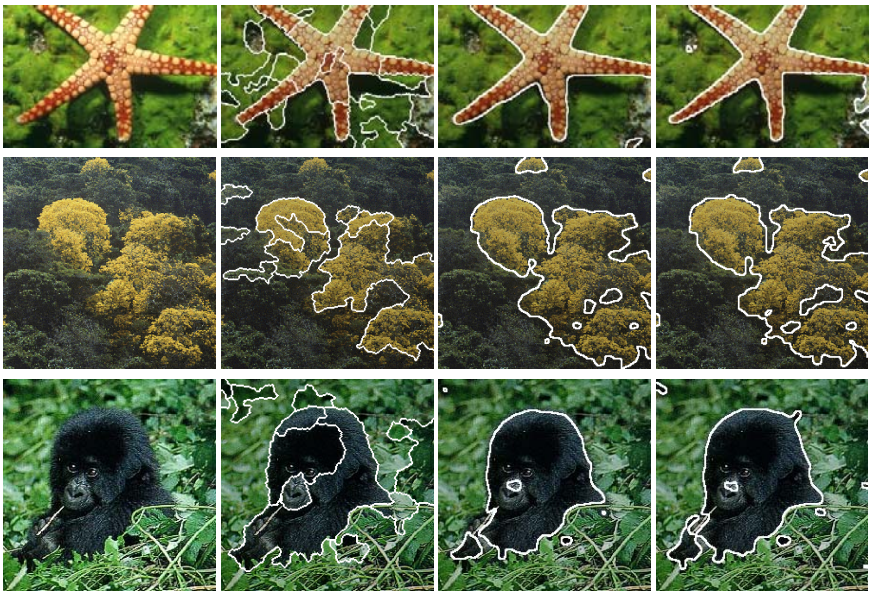

Fig. 4.13. Testing on real images - first column: original images, second column: JSEG results, third column: results of the proposed method using interscale post-fusion, fourth column: results of the proposed method using branch partitioning.

of regions as prior knowledge. On the other hand, texem based segmentation provides a useful description for each region and a measurable relationship between them. The number of texture regions may be automatically determined using model-order selection methods, such as MDL. The post-fusion and branch partition schemes achieved comparable results, while the branch partition method is faster. However, a more thorough comparison is necessary to draw complete conclusions.

4.5. Conclusions

In this chapter, we presented a two-layer generative model, called texems, to represent and analyse textures. The texems are textural primitives that are learnt across scales and can characterise a family of images with similar visual appearance. We demonstrated their derivation for graylevel and colour images using two different mixture models with different computational complexities. PCA based data factorisation was advocated while channel decorrelation was necessary. However, by decomposing the colour image and analysing eigenchannels individually, the inter-channel interac-

tions were not taken into account. The full colour texem model was found most powerful in generalising colour textures.

Two applications of the texem model were presented. The first was to perform defect localisation in a novelty detection framework. The method required only a few defect free samples for unsupervised training to detect defects in random colour textures. Multiscale analysis was also used to reduce the computational costs and to localise the defects more accurately. It was evaluated on both synthetic image collages and a large number of tile images with various types of physical, chromatic, and textural defects. The comparative study showed texem based local contextual analysis significantly outperformed a filter bank method and the LBP based texture features in novelty detection. Also, it revealed that incorporating interspectral information was beneficial, particularly when defects were chromatic in nature. The ceramic tile test data was collected from several different sources and had different chromato-textural characteristics. This showed that the proposed work was robust to variations arising from the sources. However, better accuracy comes at a price. The colour texems can be 10 times slower than the grayscale texems at the learning stage. They were also much slower than the Gabor filtering based method but had fewer parameters to tune. The computational cost, however, can be drastically reduced by performing window-based, instead of pixel based, examination at the training and testing stages. Also, there are methods available, such as Ref. 30, to compute the Gaussian function, which is a major part of the computation, much more efficiently. The results also demonstrate that the graylevel texem is also a plausible approach to perform colour analysis with relatively economic computational complexity.

The second application was to segment colour images using multiscale colour texems. As a mixture model was used to derive the colour texems, it was natural to classify image patches based on posterior probabilities. Thus, an initial segmentation of the image in multiscale was obtained by directly using the posteriors. In order to fuse the segmentation from different scales together, the quadtree context model was used to interpolate the label structure, from which the transition probability was derived. Thus, the final segmentation was obtained by top-down interscale fusion. An alternative multiscale approach using the hierarchical dependency among multiscale pixels was proposed. This resulted in a simplified image segmentation without interscale post fusion. Additionally, a texem grouping method was presented to segment multi-modal textures where a texture region contained multiple textural elements. The proposed methods were

briefly compared against JSEG algorithm with some promising results.

Acknowledgement

This research work was funded by the EC project G1RD-CT-2002-00783 MONOTONE, and X. Xie was partially funded by ORSAS UK.

References

1. X. Xie and M. Mirmehdi, TEXEMS: Texture exemplars for defect detection on random textured surfaces, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (2007). to appear.
2. T. Caelli and D. Reye, On the classification of image regions by colour, texture and shape, *Pattern Recognition*. **26**(4), 461–470, (1993).
3. R. Picard and T. Minka, Vision texture for annotation, *Multimedia System*. **3**, 3–14, (1995).
4. M. Dubuisson-Jolly and G. A., Color and texture fusion: Application to aerial image segmentation and GIS updating, *Image and Vision Computing*. **18**, 823–832, (2000).
5. A. Monadjemi, B. Thomas, and M. Mirmehdi. Speed v. accuracy for high resolution colour texture classification. In *British Machine Vision Conference*, pp. 143–152, (2002).
6. S. Liapis, E. Sifakis, and G. Tziritas, Colour and texture segmentation using wavelet frame analysis, deterministic relaxation, and fast marching algorithms, *Journal of Visual Communication and Image Representation*. **15**(1), 1–26, (2004).
7. A. Rosenfeld, C. Wang, and A. Wu, Multispectral texture, *IEEE Transactions on Systems, Man, and Cybernetics*. **12**(1), 79–84, (1982).
8. D. Panjwani and G. Healey, Markov random field models for unsupervised segmentation of textured color images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **17**(10), 939–954, (1995).
9. B. Thai and G. Healey, Modeling and classifying symmetries using a multiscale opponent color representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **20**(11), 1224–1235, (1998).
10. M. Mirmehdi and M. Petrou, Segmentation of color textures, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **22**(2), 142–159, (2000).
11. J. Bennett and A. Khotanzad, Multispectral random field models for synthesis and analysis of color images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **20**(3), 327–332, (1998).
12. C. Palm, Color texture classification by integrative co-occurrence matrices, *Pattern Recognition*. **37**(5), 965–976, (2004).
13. N. Jojic, B. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *IEEE International Conference on Computer Vision*, pp. 34–42, (2003).
14. M. Varma and A. Zisserman. Texture classification: Are filter banks neces-

sary? In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 691–698, (2003).
15. T. Ojala, M. Pietikäinen, and T. Mäenpää, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **24**(7), 971–987, (2002).
16. F. Cohen, Z. Fan, and S. Attali, Automated inspection of textile fabrics using textural models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **13**(8), 803–809, (1991).
17. X. Xie and M. Mirmehdi. Texture exemplars for defect detection on random textures. In *International Conference on Advances in Pattern Recognition*, pp. 404–413, (2005).
18. B. Silverman, *Density Estimation for Statistics and Data Analysis*. (Chapman and Hall, 1986).
19. B. Julesz, Textons, the element of texture perception and their interactions, *Nature*. **290**, 91–97, (1981).
20. S. Zhu, C. Guo, Y. Wang, and Z. Xu, What are textons?, *International Journal of Computer Vision*. **62**(1-2), 121–143, (2005).
21. C. Boukouvalas, J. Kittler, R. Marik, and M. Petrou, Automatic color grading of ceramic tiles using machine vision, *IEEE Transactions on Industrial Electronics*. **44**(1), 132–135, (1997).
22. J. Escofet, R. Navarro, M. Millán, and J. Pladellorens, Detection of local defects in textile webs using Gabor filters, *Optical Engineering*. **37**(8), 2297–2307, (1998).
23. A. Kumar and G. Pang, Defect detection in textured materials using Gabor filters, *IEEE Transactions on Industry Applications*. **38**(2), 425–440, (2002).
24. A. Kumar, Neural network based detection of local textile defects, *Pattern Recognition*. **36**, 1645–1659, (2003).
25. A. Monadjemi, M. Mirmehdi, and B. Thomas. Restructured eigenfilter matching for novelty detection in random textures. In *British Machine Vision Conference*, pp. 637–646, (2004).
26. MIT Media Lab. VisTex texture database, (1995). URL `http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html`.
27. H. Cheng and C. Bouman, Multiscale bayesian segmentation using a trainable context model, *IEEE Transactions on Image Processing*. **10**(4), 511–525, (2001).
28. R. Manduchi. Mixture models and the segmentation of multimodal textures. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 98–104, (2000).
29. Y. Deng and B. Manjunath, Unsupervised segmentation of color-texture regions in images and video, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **23**(8), 800–810, (2001).
30. L. Greengard and J. Strain, The fast Gauss transform, *SIAM Journal of Scientific Computing*. **2**, 79–94, (1991).

Subject Index