Pattern Anal Applic (2008) 11:141–157
DOI 10.1007/s10044-007-0085-7

THEORETICAL ADVANCES

A multiscale approach to texture-based image retrieval

Mohammad Faizal Ahmad Fauzi · Paul H. Lewis

Received: 3 November 2005 / Accepted: 19 July 2007 / Published online: 26 September 2007

Abstract This paper presents research on a robust technique for texture-based image retrieval in multimedia museum collections. The aim is to be able to use a query image patch containing a single texture to retrieve images containing an area with similar texture to that in the query. The feature extractor used to build the feature vectors is based on an improved version of the discrete wavelet frames (DWF), proposed elsewhere. In order to utilise the feature extractor on real scene image datasets, a block-oriented decomposition technique, termed the multiscale sub-image matching method, is presented. The multiscale method, together with the DWF, provide an efficient content-based retrieval technique without the need for segmentation. The algorithms are tested on a range of databases of texture images as well as on real museum image collections. Promising results are reported.

Keywords Content-based image retrieval · Texture analysis · Discrete wavelet frames · Multiscale image matching

1 Originality and contribution

The paper investigates the multiscale sub-image matching technique for texture-based image retrieval. The multiscale method, previously used in color-based image retrieval, is modified to suit the texture retrieval applications. Two types of overlapping is considered, namely *case 1* and *case 2 overlapping*, and it was found that even though *case 2 overlapping* is more time consuming during feature extraction stage, the improvement in performance over *case 1 overlapping* is huge. Important parameters of the algorithm, such as the number of sub-images generated, the number of scales involved, sub-image coverage and scale invariance property are described in detail. Comprehensive experiments were conducted to test the reliability of the algorithm, which includes experiment on different locations of the target sub-image, different scales and sizes of the query image, decomposition order issue, as well as experiments on arbitrary size Brodatz database and museum collections. Promising results were reported.

M. F. A. Fauzi (✉)
Faculty of Engineering, Multimedia University,
Jalan Multimedia, 63100 Cyberjaya, Selangor, Malaysia
e-mail: faizal1@mmu.edu.my

P. H. Lewis
School of Electronics and Computer Science,
University of Southampton, Southampton SO17 1BJ, UK

2 Introduction

Image retrieval has been a very active research area since the 1970s, with the thrust from two major research communities; database management and computer vision [1]. Image retrieval can be divided into text-based image retrieval (TBIR) and content-based image retrieval (CBIR). The text-based approaches first annotate the images with text, and subsequently use text-based techniques to perform image retrieval. The use of TBIR was popular in the early days of computer vision, but its usage has been less dominant in recent years. There are two major factors that account for this decline in popularity; the vast amount of labour required in manual image annotation and the subjectivity of human perception. This results in the need for further research effort on CBIR. Using this approach, images are indexed by signatures of their own visual content, such as colour, texture or shape. CBIR has been

increasingly pertinent, particularly with the advancement in computer technology, and it could potentially provide resolutions to the two drawbacks of the TBIR approach mentioned above.

Among the numerous retrieval features associated with CBIR, texture retrieval is one of the most difficult. This is mainly because no widely accepted satisfactory quantitative definition of texture currently exists. Texture analysis has a long history and texture analysis algorithms range from the use of random field models to multiresolution filtering techniques such as the wavelet transform. Unlike colour, texture cannot be represented by a single pixel, thus making texture analysis a more complex challenge. One way to perform CBIR, using texture as the cue, is to first segment an image into a number of different texture regions and then apply a texture analysis algorithm to each texture segment [2–5]. However, segmentation can sometimes be problematic for image retrieval [6]. In this paper, we propose a simple but effective algorithm for texture-based image retrieval without the need for segmentation. Our algorithm uses a multiscale sub-image matching method together with the discrete wavelet frames (DWF) technique. Note that the focus of the paper is on the multiscale sub-image matching and its effectiveness in replacing the segmentation process in CBIR, and not on the texture analysis algorithm itself.

The idea of multiscale feature extraction techniques has found much success in the area of colour analysis. One example can be found in the work of Chan et al. [7], where a multiscale approach is used to extract colour features using the colour coherence vector (CCV) technique. The resulting algorithm, which is called the multiscale colour coherence vector (MCCV), manages to handle the problem of sub-image colour retrieval effectively, without the need for segmentation. With minor modifications, the multiscale approach can be used in a similar way in order to extract texture information using any texture feature extraction method, although those with low computational load are preferable. In this work, the improved DWF proposed in [8] will be used. The multiscale approach, together with the DWF technique, is not only able to provide good retrieval accuracy, but also reduces the scale dependence of the algorithm, hence incorporating scale invariance to the retrieval process. The algorithm, however, is not rotation invariant, but it is one of our targets to include this property in our CBIR system. Wavelet-based methods, to which the DWF technique belongs, can be further improved to achieve rotational invariance, as reported in several papers [9, 10].

This paper is organized as follows. Section 3 briefly reviews some block decomposition algorithms used in image retrieval, while Sect. 4 explains the DWF approach. Section 5 presents the details of the multiscale algorithm, and the experimental evaluation of several image datasets, including real museum collection databases, is highlighted in Sect. 6. Finally the conclusion and potential future work on the proposed algorithm are presented in Sect. 7.

3 Review of block-oriented decomposition techniques

There are several different approaches to using a local mask in CBIR. These include a simple sliding mask [11, 12], the quad-, quin- and nona-tree decompositions [13, 14], and a multiscale decomposition [7], among others.

3.1 Sliding windows

Sliding windows is the simplest block-oriented approach in texture localisation and is usually used in texture retrieval. Given an image, a collection of small to medium sized sub-images is produced by a simple image cropping procedure. The sub-images could be overlapping or non-overlapping, with overlapping windows providing better localisation, but with many more sub-images, which affects the speed of the feature extraction process. The size of windows generally depends on the application, with large windows providing better localisation for coarse texture, but with a risk of failing to capture small texture regions. The feature extraction process is then performed on the sub-images. During matching, the feature vector of the query image is compared with all the feature vectors from the sub-images, and the sub-image with the closest matching feature vector is taken as the region most similar to the query. Manjunath and Ma [11] used non-overlapping windows of size 128×128 for browsing large satellite images and air photos (about $5{,}000 \times 5{,}000$ pixels) with Gabor transform as the texture features. Another example of image retrieval that used sliding windows can be found in the WALRUS system [12].

3.2 Quad-tree decomposition

A quad-tree is a hierarchical image decomposition structure which can provide quick access for image retrieval. A quad-tree is based on the principle of recursive decomposition of images. Each decomposition of an image segment produces four equal-sized quadrants. The root node in the quad-tree represents the entire image, and its four child nodes represent the decomposed segments; these, in turn, become roots for further segmental decomposition. The original quad-tree decomposition for binary images labeled the decomposed segments white if they consist of white pixels only, black if they consist of black pixels only, and

grey if they consist of both black and white pixels. Further decompositions are only carried out on grey segments.

Smith and Chang [13] have presented a *query by texture* approach that uses the quad-tree segmentation and the wavelet transform. In their application of the quad-tree structure, the definition of leaf nodes is slightly changed. Before four children are generated by each parent, conditions for merging are tested. A distance threshold is computed for each child on the basis of extracted texture features. The distances in the feature space are measured from the parent node to each child. If the distance to all four children falls within the thresholds of the children, a single texture is declared in the parent node, and no further decomposition is necessary. Otherwise, pairwise grouping of the children is performed, that is, if the distance between two neighbouring children falls below the thresholds of both, the children are merged as a single child. The quad-tree decomposition is then iterated on each child until the size of the smallest child reaches a certain number of pixels. Generally, the maximum number of children generated by a quad-tree decomposition is 4^i, where i is the number of the decomposition levels.

3.3 Quin-tree decomposition

A quin-tree is a hierarchical image decomposition structure, which is based on a slight modification of the recursive decomposition of images that are used in quad-trees. Each decomposition of an image segment produces five sub-segments of equal size. In addition to the four equal-sized quadrants obtained in quad-tree decomposition, a fifth sub-segment, equal in size to each quadrant, is generated from the central area of the image segment.

Each internal node has a maximum of five children. The strategy of quad-tree decomposition proposed by Smith and Chang [13] was used by Guo and Zhang [14] to guide the decomposition of sub-segments 1, 2, 3 and 4 in the quin-tree. Whether or not these sub-segments are generated determines the generation of sub-segment 5. Generally, the maximum number of children generated by a quin-tree decomposition is $(4^{i+1} - 1)/3$ (after eliminating redundant block segments caused by the overlapping of sub-segment 5 with sub-segments 1, 2, 3 and 4), where i is the number of decomposition level.

3.4 Nona-tree decomposition

A nona-tree is a hierarchical image decomposition which is based on a further modification to the recursive decomposition of images in quin-trees. Each decomposition of an image segment produces nine sub-segments of equal sizes. In addition to the five equal segments in quin-tree, four additional segments, again of the same size, are produced from the central areas of the upper, bottom, left and right halves of the image segment.

Each internal node has a maximum of nine children. Similar to the definition of the leaf nodes in the quin-tree, the strategy of quad-tree decomposition proposed by Smith and Chang [13] is used by Guo et al. [14] to guide the decomposition of sub-segments 1, 2, 3 and 4 in the nona-tree. Whether or not sub-segments 1, 2, 3 and 4 are generated determines the generation of sub-segments 5, 6, 7, 8 and 9. Generally, after eliminating the redundant sub-segments caused by the overlapping of additional sub-segments, the maximum number of children generated by a nona-tree decomposition is $(2^{i+1} - 1)^2$, where i is the number of decomposition level.

3.5 Multiscale image decomposition

Multiscale image decomposition can be viewed as a multiscale version of the sliding windows described previously. In [7], Chan et al. used a multiscale image decomposition approach in order to support colour localisation within high resolution images. Although their application is for colour localisation, it may be appropriate for texture localisation applications as well. The idea of the multiscale approach is to divide the database images into pyramids of patches and record the features for each. All images are first resized to a dyadic size, and overlapping patches of size 64×64 are slid across the image. The patches are slid by an amount equal to half the length of the patch size. The feature vectors computed from each sub-image patch are used as the feature vectors for that particular scale.

The image is then halved, resulting in images corresponding to a lower resolution, and overlapping patches of the same 64×64 size are used to compute the feature vectors at that scale. This process is repeated until the reduced image corresponds to a single patch, i.e. of size 64×64. This decomposition approach results in the patches corresponding to the lowest scale to represent the parent images. The patches corresponding to the higher scales, on the other hand, correspond to specific parts of the parent image. Hence, this method is deemed an appropriate tool to capture both the global and local features of an image.

4 The discrete wavelet frames

Discrete wavelet frames is a variation of the wavelet transform family. The fundamental idea of the wavelet

transform is to analyse data in the spatial-frequency domain. It is based on the dilations and translations of a mother wavelet, ψ, which is a function with some special properties. In image processing terms, the dilations and translations of the mother functions ψ are given by the wavelet basis function:

$$\psi_{(s,l)}(t) = 2^{-s/2}\psi(2^{-s}x - l).$$

The variables s and l are integers that dilate and orientate the mother function ψ to generate a family of wavelets, such as the *Daubechies* wavelet family [15]. The scale index s indicates the wavelet's width, and the location index l gives its position in 2-D.

Wavelets can be divided into orthogonal and non-orthogonal. They are orthogonal if their basis functions are mutually orthogonal. The orthogonality feature results in a series of coefficients that represent the wavelet decomposition for the whole family of a particular wavelet. The series of coefficients is known as the *quadrature mirror filter* (QMF). This concept, along with the theory of filter banks, is the basis with which the famous fast algorithm wavelet transform was produced by Mallat [16]. The QMF is a filter that can be either low-pass or high-pass just by changing signs and rearranging its coefficient. In the wavelet decomposition of an image, the QMFs are used as high-pass as well as low-pass filters in both horizontal and vertical directions, followed by a 2 to 1 subsampling of each output image. This generates four wavelet coefficient images, i.e. the *low–low*, *low–high*, *high–low* and *high–high* (LL, LH, HL, HH, respectively) channels. The process is repeated on the LL channels until some pre-determined condition are met. The number of channels generated for is therefore $3 \times l + 1$, where l is the number of decomposition levels.

The DWF is almost identical to the wavelet transform, except that it upsamples the filters, rather than downsampling the image. While the frame representation is over-complete, and computationally more intensive than the wavelet transform, it holds the advantage of being translationally invariant [17]. Given an image, the DWF decomposes its channel by using the same method as the wavelet transform, but without the subsampling process. This results in four filtered images with the same size as the input image. The decomposition is then continued on the LL channels, as in the wavelet transform, but since the image is not sub-sampled, the filter has to be upsampled by inserting zeros in-between its coefficients. Similar to the wavelet transform, the number of channels for the DWF is $3 \times l + 1$. A more detailed description of DWF and the wavelet family in general can be found in [15–18].

Based on our previous work [8, 19], we find that the DWF technique is appropriate for use as a texture feature for both the segmentation-based and block-based texture retrieval applications. We also proposed suitable parameters to be used with DWF in order to optimise its retrieval performance, and these are summarized in Table 1. Throughout this paper, DWF, with the specifications shown in the table, will be used as the texture feature extraction technique.

5 The multiscale technique for texture retrieval

An improved multiscale decomposition approach, suitable for use with texture in CBIR of museum images, is proposed in this paper. Before a description of the proposed algorithm is presented, it is necessary to point out that the DWF technique is not a scale invariant texture feature.

The distribution of energy in the wavelet decomposition is based on frequency or scale. Hence the energy of a particular texture with a coarser scale focuses within a certain frequency range, for instance the LH channel of the first level decomposition, while the energy of a finer scale version of the same texture focuses within some other frequency range, for instance the LH channel of the second level decomposition. To obtain the features of a particular texture, the DWF texture feature method computes two statistical measures from each channel, namely the standard deviation of the wavelet coefficients and the number of zero-crossings.

To compare the similarity between two textures, the normalized Euclidean distance is employed, where a channel by channel comparison of the two statistical measures mentioned above is carried out. If the channel energy distribution of any two textures differ, the dissimilarity distance will be large and the algorithm will perceive the textures as different. In the case of coarse against fine versions of the same texture, we would like the algorithm to consider them as the same texture, but unfortunately the

Table 1 Summary of the best discrete wavelet frames parameters

Parameters	Identified parameter value/type
Level of decomposition	3
Wavelet basis	Not crucial, but Daubechies 8-tap is chosen
	[–0.2304 0.7148 –0.6309 –0.0280 0.1870 0.0308 –0.0329 –0.0106]
Padding type	Periodic
Distance metric	Normalized Euclidean
Statistical features	Standard deviation energy, zero-crossings
Channels selection	All channels

DWF is unable to attain it, hence the term scale dependence. However, by using the multiscale sub-image matching, the scale dependence of DWF can be reduced. Hence, we opt to use the multiscale decomposition approach instead of other block-oriented decompositions such as the quad-tree decomposition.

5.1 Decomposition algorithm

The proposed algorithm is based on the multiscale algorithm of Chan et. al., where in their work, the CCV is used to extract features from each sub-image. Since the colour property does not change when re-scaling the images without maintaining the aspect ratio, Chan et al. resized all the database images to dyadic sizes to facilitate easier image cropping and re-scaling. This suggests that at the lowest level, the database image will always be of size 64×64, and hence the same size as the sub-image patch. If we are to use the multiscale approach for texture retrieval, a modification is necessary since re-scaling the image without maintaining the aspect ratio tends to alter the properties of the underlying texture. We wish to ensure that the texture properties are unaltered at every level, in order for the retrieved images to offer a fairer resemblance of the query image.

The proposed multiscale image decomposition algorithm is summarized in Fig. 1. The basic sub-image patch used is the same as proposed by Chan et al., that is 64×64, since in real applications, we believe that the query image should not be smaller than this size. Consider a texture of size 256×256. As the image size is a multiple of 64, the sub-images can be fitted to cover the whole image at the root level. Now, consider a texture of size 300×300. As the image size is not a multiple of 64, some overlapping between sub-images are necessary, if the whole image is to be covered. In this case, 25 evenly distributed sub-images are required to cover the root level, with a five-pixel overlap between them in both the horizontal and vertical dimensions. This arrangement, where the overlapping between sub-images is between 0 to 63 pixels in either dimension, is one of two approaches we will consider, and is termed *case 1 overlapping*.

One might argue that better localisation can be achieved by increasing the minimum number of the overlapped pixels. If we increase the minimum number to half the sub-image size, i.e. 32 pixels in either dimension, we will have a much better localisation than in the previous case. The overlapping range is now between 32 and 63 pixels in either dimension. We call this type of localisation *case 2 overlapping*. Although the image is better localised using this arrangement, it requires more sub-images, which suggests more computation for each scale. It is interesting

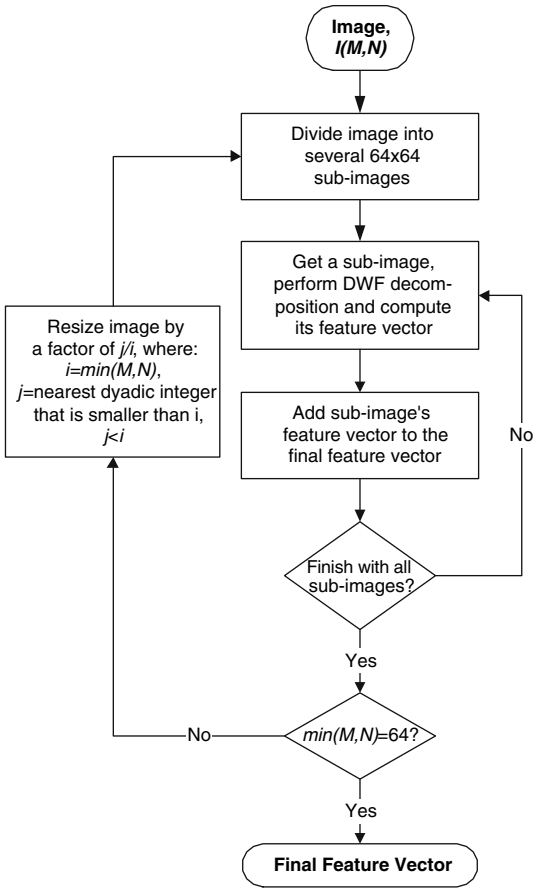

Fig. 1 Flowchart of the proposed multiscale image decomposition technique

to observe the performance of the two approaches, in terms of both accuracy and speed. There are two factors that are important when observing the performance, which are the total number of sub-images and the sub-image coverage. Higher total number of sub-images provides better sub-image coverage, which in theory should lead to better performance, but at the cost of higher computational load. Thus these two factors need to be considered when determining the better approach between *case 1* and *case 2 overlapping*.

For a square-sized sub-image, the number of sub-images generated, K for *case 1* and *case 2 overlapping* for any single level is calculated as

$$K_{\text{case 1}} = \left(\left\lceil \frac{M}{64} \right\rceil \right)^2 \tag{1}$$

$$K_{\text{case 2}} = \left(\left\lceil \frac{M}{64} \right\rceil \times 2 - 1 \right)^2 \tag{2}$$

where M is the length of the sub-image, and $\lceil . \rceil$ is defined to be the smallest integer greater than or equal to a given

number. The $\lceil . \rceil$ operator ensures the sub-images are interconnected and no sections of the image will be left out.

The number of overlapping pixels (which relates to the sub-image coverage) in any particular direction, J can be computed for *case 1* and *case 2 overlapping*, respectively.

$$J_{\text{case 1}} = 64 - \left\lceil \frac{M - 64}{\lceil \frac{M}{64} \rceil - 1} \right\rceil \tag{3}$$

$$J_{\text{case 2}} = 64 - \left\lceil \frac{M - 64}{\lceil \frac{M}{64} \rceil \times 2 - 2} \right\rceil \tag{4}$$

Figure 2a shows the variation of overlapping amount with different sizes of image for the two approaches. In both cases, minimum overlapping is achieved when image sizes are multiples of 64. On the other hand, maximum overlapping is achieved when either the width or the height of the image has a size of 65 (two sub-images, the first sub-image takes the first 64 pixels, and the second sub-image takes the last 64 pixels).

As mentioned earlier, the first level of the decomposition involves the original dimension of the image to be processed. The re-scaling of the image can be described as follows. For an image with $M \times N$ dimensions, the minimum of the two dimensions, min(M,N) is taken as the basis for re-scaling. Hypothetically, say the row, M, is the minimum of the two dimensions. The image is then re-scaled to the nearest dyadic integer that is smaller than M, while maintaining the aspect ratio of the image. The sub-image decomposition described earlier is then performed on the re-scaled image to obtain the sub-images that correspond to the second level. Starting from the second level, to obtain the parent image at the following level, the image is re-scaled by a factor of 2. This process continues until min(M,N) reaches 64.

For example, consider an image of size 783×556. The resolution of the image will be 783×556 at the first level, 721×512 at the second level (512 is the nearest dyadic integer smaller than min(783,556)), 361×256 at the third level, 181×128 at the fourth level, and finally 91×64 at the fifth level. The final level (91×64) will consists of two sub-images for *case 1 overlapping* and three sub-images for *case 2 overlapping*. In general, for an $M \times N$ image, the number of scales, S can be computed as

$$S = \lceil \log_2(\min(M, N) \rceil - 5 \tag{5}$$

5.2 Total number of sub-images

Equations 1 and 2 give the number of sub-images generated at a particular scale for the *case 1* and *case 2 overlapping*, respectively. The total number of sub-images generated by the multiscale algorithm for all scales can be computed by adding the number of sub-images at each scale. Figure 2b shows the total number of sub-images generated for *case 1* and *case 2 overlapping* for a square $M \times M$ image, with M ranging from 64 to 1,024. From the figure, the number of sub-images for *case 2 overlapping* increases almost quadratically with increase in M over the number of sub-images for *case 1 overlapping*.

5.3 Sub-image coverage

We will now discuss the overlapping coverage between a query image and the segments of a database image for instances in which the query image is similar to a sub-image of the database image. Let Q be a query image and D be a database image. We assume, for simplicity sake, that the query image is of size 64×64 and the database image is of size 128×128. Assume that D contains a sub-image d which is similar to Q and d may be located anywhere in D. We now examine the degrees of coverage between the query image and the segments generated by

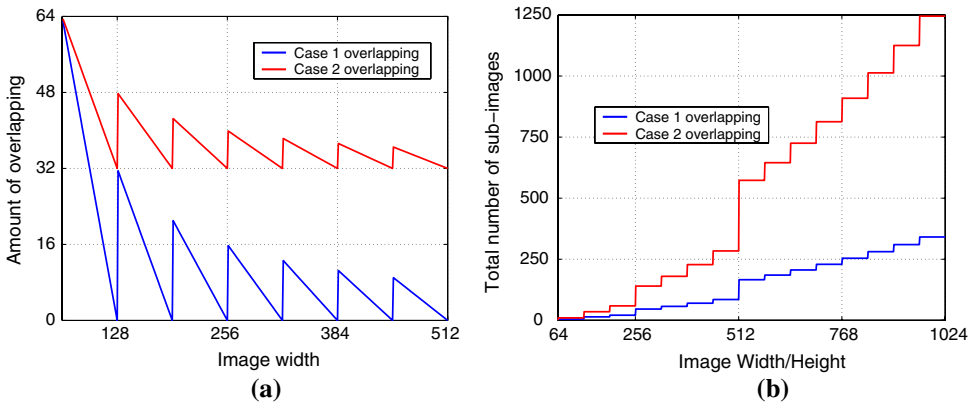

Fig. 2 Comparison between *case 1* and *case 2* overlapping. **a** Amount of pixels overlapping. **b** Total number of sub-images

the two approaches mentioned above, *case 1* and *case 2 overlapping*.

5.3.1 Case 1 overlapping

For a 128 × 128 image, it is decomposed into four quadrants of size 64 × 64. The minimum coverage between these four quadrants for D and Q will be 1/4 of Q, when d is located in the centre of D, as illustrated in Fig. 3. Obviously, when d is located in other places in D, the coverage between Q and any quadrant of D will be larger than 1/4 of Q. When the size of the database image is not a multiple of 64, the coverage will be larger than 1/4 of Q too.

5.3.2 Case 2 overlapping

For a 128 × 128 image, it is decomposed into nine quadrants of size 64 × 64, as is illustrated in Fig. 4. Sixteen non-overlapping sub-images of size 32 × 32 is shown. The nine quadrants can be obtained by grouping any four adjacent 32 × 32 non-overlapping sub-images in squares. Let d be located at an offset of d_1 and d_2 at each side of the image as indicated in Fig. 4.

The overlap between d and the closest quadrant is represented by the shaded area. Let $L = 64$ be the size of the query image. In general, as shown in Fig. 4, the shaded area A of the query image conforms to one of the following cases:

- For $0 \le d_1 \le \frac{1}{4}L$ (Fig. 4a):

  If $0 \le d_2 \le \frac{1}{4}L$, $A = (L - d_1) \times (L - d_2)$,

  If $\frac{1}{4}L \le d_2 \le \frac{1}{2}L$, $A = (L - d_1) \times (\frac{1}{2}L + d_2)$,

  If $\frac{1}{2}L \le d_2 \le \frac{3}{4}L$, $A = (L - d_1) \times (\frac{3}{2}L - d_2)$,

  If $\frac{3}{4}L \le d_2 \le L$, $A = (L - d_1) \times d_2$,

- For $\frac{1}{4} \le d_1 \le \frac{1}{2}L$ (Fig. 4b):

  If $0 \le d_2 \le \frac{1}{4}L$, $A = (\frac{1}{2}L + d_1) \times (L - d_2)$,

  If $\frac{1}{4}L \le d_2 \le \frac{1}{2}L$, $A = (\frac{1}{2}L + d_1) \times (\frac{1}{2}L + d_2)$,

  If $\frac{1}{2}L \le d_2 \le \frac{3}{4}L$, $A = (\frac{1}{2}L + d_1) \times (\frac{3}{2}L - d_2)$,

  If $\frac{3}{4}L \le d_2 \le L$, $A = (\frac{1}{2}L + d_1) \times d_2$,

- For $\frac{1}{2} \le d_1 \le \frac{3}{4}L$ (Fig. 4c):

  If $0 \le d_2 \le \frac{1}{4}L$, $A = (\frac{3}{2}L - d_1) \times (L - d_2)$,

  If $\frac{1}{4}L \le d_2 \le \frac{1}{2}L$, $A = (\frac{3}{2}L - d_1) \times (\frac{1}{2}L + d_2)$,

  If $\frac{1}{2}L \le d_2 \le \frac{3}{4}L$, $A = (\frac{3}{2}L - d_1) \times (\frac{3}{2}L - d_2)$,

  If $\frac{3}{4}L \le d_2 \le L$, $A = (\frac{3}{2}L - d_1) \times d_2$,

- For $\frac{3}{4} \le d_1 \le L$ (Fig. 4d):

  If $0 \le d_2 \le \frac{1}{4}L$, $A = d_1 \times (L - d_2)$,

  If $\frac{1}{4}L \le d_2 \le \frac{1}{2}L$, $A = d_1 \times (\frac{1}{2}L + d_2)$,

  If $\frac{1}{2}L \le d_2 \le \frac{3}{4}L$, $A = d_1 \times (\frac{3}{2}L - d_2)$,

  If $\frac{3}{4}L \le d_2 \le L$, $A = d_1 \times d_2$,

For all cases, A is found to be greater than $\frac{9}{16}L^2$, which suggests the minimum coverage is 9/16 of d, or equivalently 9/16 of Q. The above calculations are also applicable to different image sizes and different scales. Obviously, if the image size is not a multiple of 64, the coverage will be larger than 9/16 of Q, since the overlapping between sub-images increases. Although *case 1 overlapping* decomposition introduces fewer segments than does *case 2 overlapping*, the latter provides a more effective and robust platform for image retrieval. The question of which of these approaches is better can only be determined experimentally, where an evaluation of whether it is worth generating the extra sub-images can be observed in terms of retrieval accuracy.

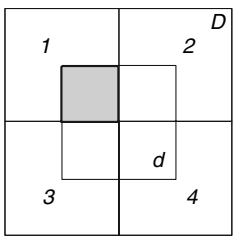

Fig. 3 Minimum coverage for *case 1* overlapping. The *shaded area* represents the overlap between query image Q and database image D

5.4 Scale invariance

As mentioned earlier, the multiscale image decomposition can assist in reducing the scale dependence of the DWF texture features. This section will outline how the multiscale decomposition approach captures different texture scales. Consider the image in Fig. 5, which is of size 256 × 256. The sub-images generated by the multiscale decomposition (using *case 1 overlapping*) are also shown in the figure. There are 16 sub-images corresponding to level 1, four sub-images corresponding to

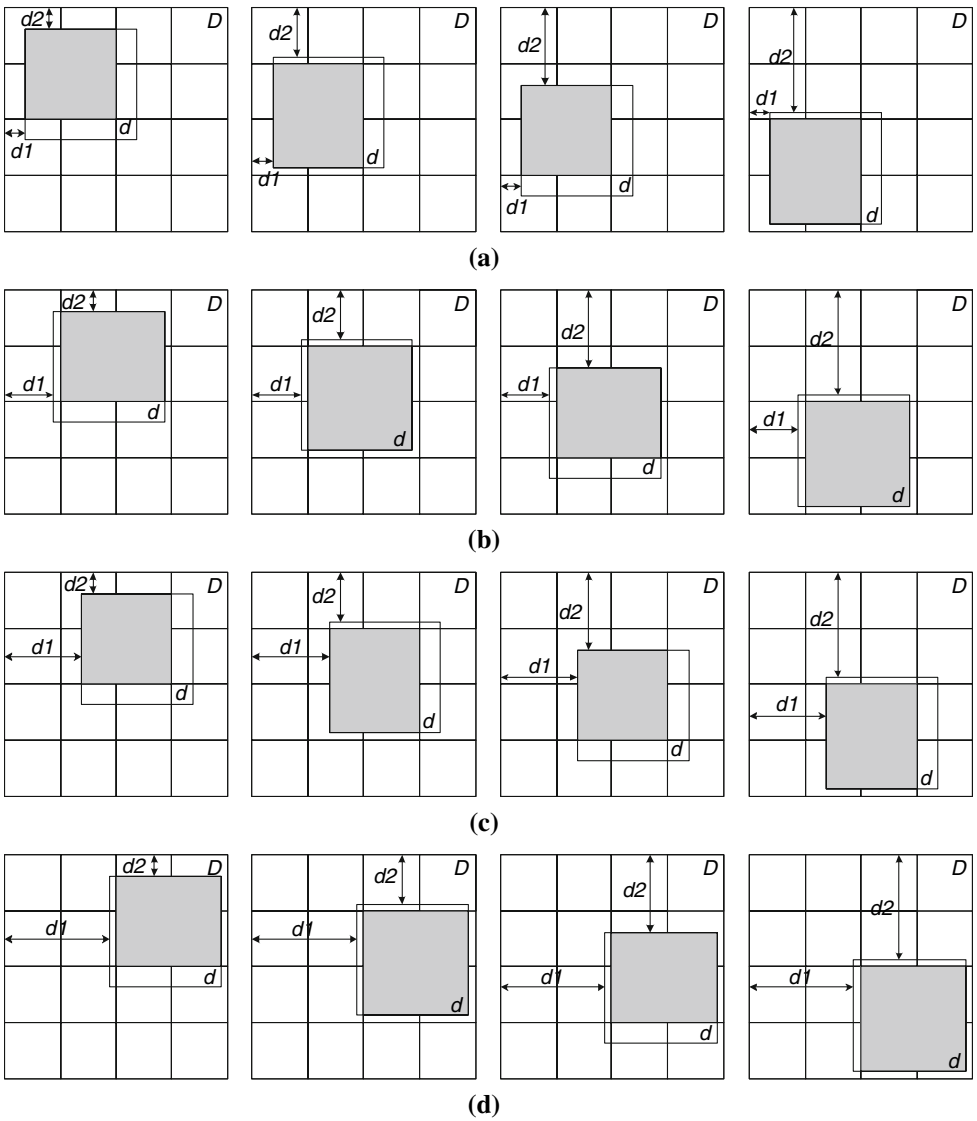

Fig. 4 Minimum coverage for *case 2* overlapping. The *shaded area* represents the overlap between query image Q and database image D

level 2, and one sub-image corresponding to the lowest level. It is clear that the texture scales change from being coarser to finer, with the increase of levels. Now, if a similar texture, but with scale equivalent to the lowest level in Fig. 5 is used as the query, the probability that the parent image will be retrieved should be higher than for the method using only one scale, since the comparison is now performed on three scales, and the sub-image corresponding to the lowest scale should have the least dissimilarity compared to the sub-images corresponding to the other two scales.

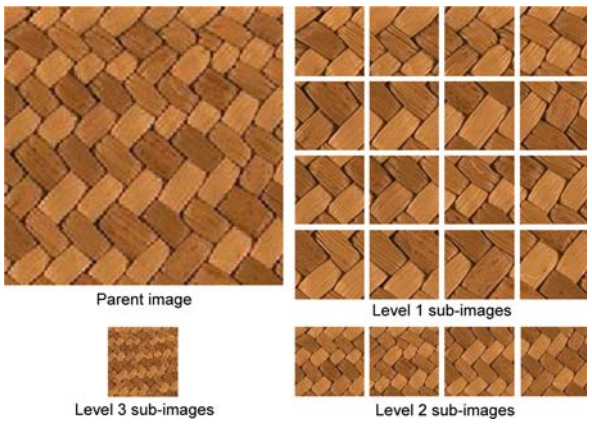

Fig. 5 Example of sub-images generated for image of size 256 × 256

However, the above theory is not true for all cases. Consider the image in Fig. 6. The image consists of a combination of the tile texture and water, where the tile texture is only a fraction of the whole image. From the generated sub-images shown, only one of them manages to capture the whole tile region of the image, i.e. the second sub-image at level 1. Sub-images at levels 2 and 3 fail to capture the unique tile region, thus resulting in no representative of the tile texture at those levels. Hence, only the original scale of the tile texture will be stored in the feature

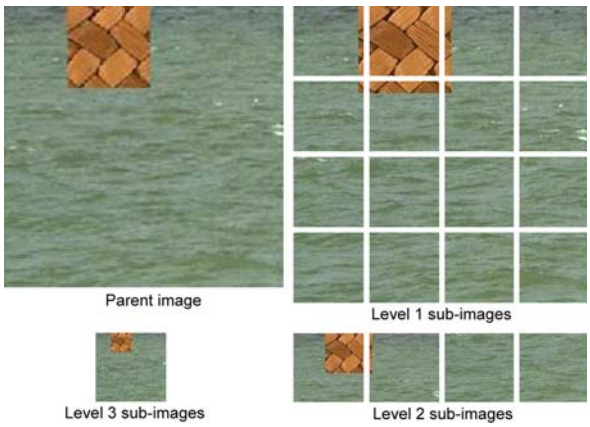

Fig. 6 Another example of sub-images generated for image of size 256 × 256

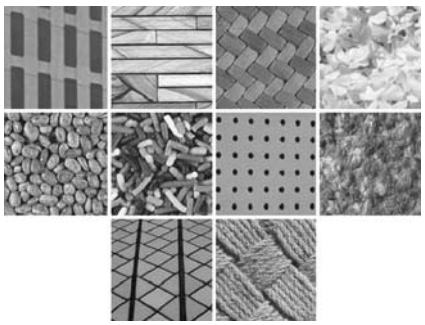

Fig. 7 The ten vision textures selected for use as query in the multiscale experiments. Queries 1–10 is read from *left* to *right*, *top* to *bottom*. The textures are converted to *grey* scale for compatibility with the Brodatz textures

vector. The water texture, however, has representatives at two different scales, i.e. at levels 1 and 2. We can conclude that the multiscale nature of the multiscale decomposition depends on the portion of the texture of interest within the whole image. The larger the portion, the higher the number of scales to be represented for the texture. Nevertheless, since the portion of the texture of interest in the parent image is quite small, it is unreasonable to expect the system to process the texture at many scales, thus, this problem is not considered a serious disadvantage for the multiscale algorithm.

6 Experimental analysis

In this section, we will discuss the effectiveness of the multiscale sub-image matching algorithm for CBIR. During the *offline* feature extraction stage, feature vectors are computed for all the images in the database, using the multiscale image decomposition technique (Fig. 1), and are stored. During the *online* retrieval stage, these feature vectors will be compared to the feature vector of the query texture patch. For a particular database image, each of the sub-image feature vector will be compared to the query feature vector and the one with the lowest dissimilarity score will be considered as the score for that image. After all the database images have been compared, the images are then retrieved with increasing dissimilarity score. As shown in Table 1, the dissimilarity measure used for comparison is the normalized Euclidean distance. Experiments are conducted on three separate databases. The first consists of dyadic size Brodatz textures. This is just to make certain evaluations easier. The second database consists of Brodatz textures of random sizes. Finally, the third database consists of museum image collections. The evaluation takes into account several important factors, including the sensitivity of various sub-image locations within database images to which query images are compared, the size of the query images and the scale of the query images.

6.1 Dyadic size image database

An image data testbed was constructed from Brodatz texture images [20]. For each of the 112 scanned 512 × 512 texture images, 9 overlapping sub-images of size 256 × 256 are produced. Thus, there are a total of 1,008 texture images in the database for the experiments. Ten Vision textures [21] are selected and are cut-and-pasted onto a selected image from the 1,008 Brodatz database to provide target textures for the multiscale algorithm. The colour Vision textures are of course converted to grey-scale before the cut-and-paste process. Each Vision texture is pasted on nine different database images at different locations within the images, giving a total of 90 modified database images. The Vision texture is then used as the query image and the effectiveness of the multiscale algorithm is measured as the ability of the algorithm to retrieve all nine similar textures. Figure 7 shows the ten Vision textures used as query images.

6.1.1 Location of the target sub-image in the database image

In the first experiment, the sensitivity of the sub-image locations is tested in order to compare *case 1 overlapping* with *case 2 overlapping*. For the sake of simplicity, in this experiment, the evaluation is based on a single scale only, that is, the very first scale. In other words, the scale of the query image is the same as the target sub-images, so that the algorithm retrieves sub-images that correspond to only the original scale. The size of the query images is 64 × 64,

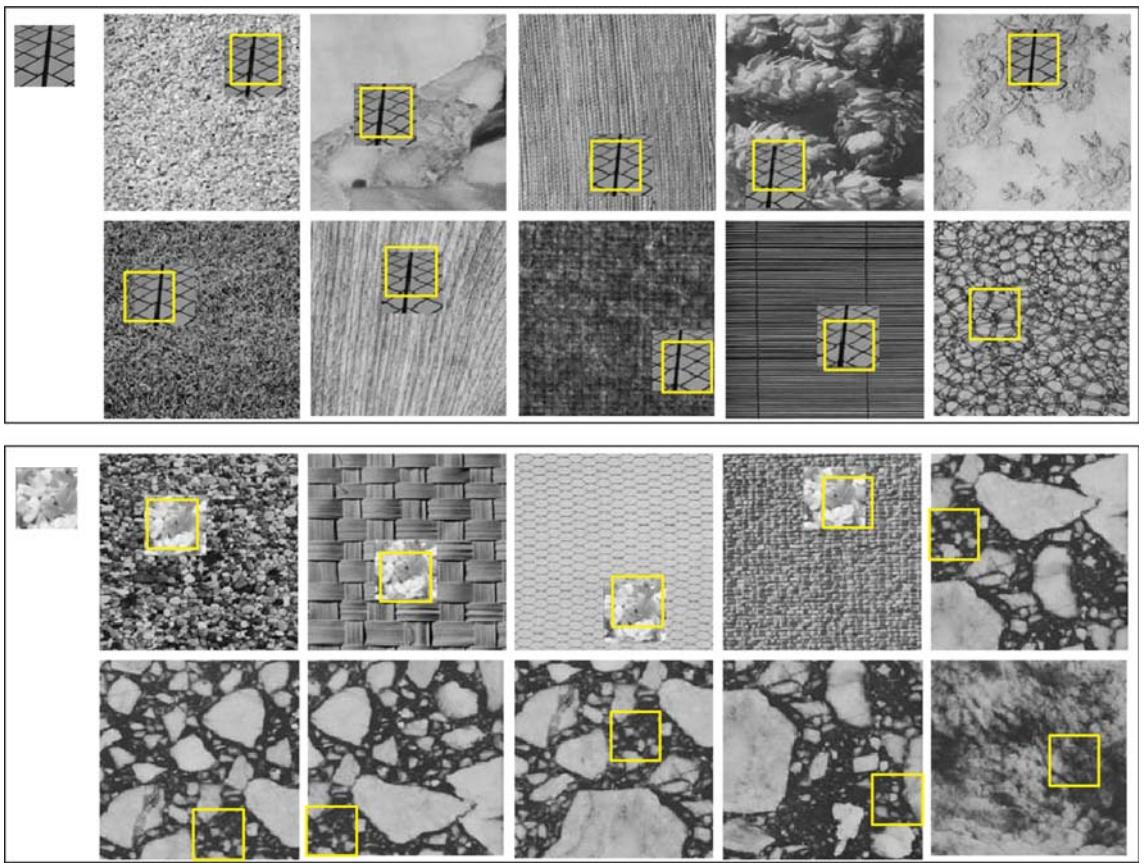

Fig. 8 Example of retrieval results for two different queries for the dyadic database. For each example, the query image is located at the *top left*, and the *top* ten retrieved images are ranked from *left* to *right*, *top* to *bottom*

while the target sub-images pasted on the database images of size 256×256 is set to 80×80 pixels. We consider two sets of experiments. The first was performed by pasting each Vision texture onto nine different locations that are fully covered by *case 2 overlapping* but are only partially covered by *case 1 overlapping*. This is intended to test whether the *case 1 overlapping* approach is severely affected by the location of target sub-images. In the second set of experiments, each Vision texture is pasted at nine random locations within the database images.

The experiment on the first set showed very poor results for the *case 1 overlapping* approach, i.e. less than 10% retrieval rate, compared to *case 2 overlapping*, which showed a perfect 100% retrieval rate. This perfect performance is because, in these specified locations, the sub-image coverage for that particular approach is 100%. This experiment shows that when the sub-image coverage of *case 1 overlapping* is minimum, it results in a very poor retrieval rate, but the *case 2 overlapping* approach showed a very good performance. This is a huge advantage for *case 2 overlapping* as when the coverage of *case 2 overlapping* is minimum (9/16 of the query image), the coverage of *case 1 overlapping* is about the same, hence in overall performance, *case 2 overlapping* should still be much better than *case 1 overlapping*.

This is confirmed in the second set of experiments. Using a randomly pasted target sub-image, the performance of *case 2 overlapping* is almost double the retrieval rate for *case 1 overlapping* (80 against 40%). We can conclude that *case 1 overlapping*, although it has a much lower computational intensity, is far behind *case 2 overlapping* in terms of retrieval accuracy. Figure 8 shows two examples of retrieval results on randomly pasted targets, using *case 2 overlapping*. The first example shows the best recorded result (query image 7) while the second example shows the worst recorded result (query image 4). The box within the image shows part of the image found to be the most similar to the query.

6.1.2 Scale of the query textures

This experiment is aimed at testing the multiscale nature of the proposed algorithm. The same set of database images as in the very first experiment (target located at nine overlapping regions) is used in this experiment, but with

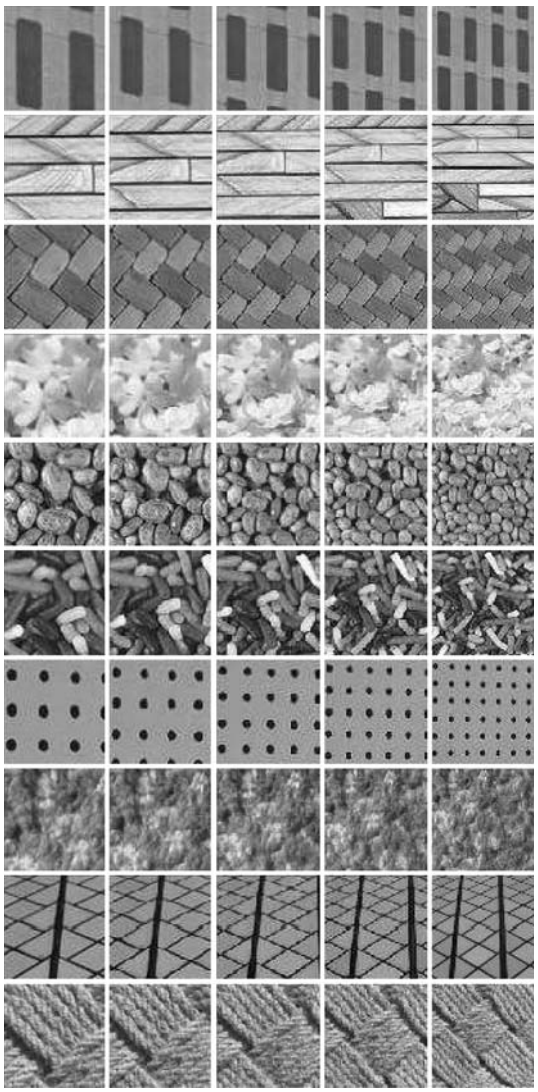

Fig. 9 The five different scales of query images (*left* to *right*) used to test the multiscale nature of the algorithm

different scales of the query images. Five different scales are tested for the query images. The size of the query image remains the same at 64 × 64 pixels. The five different scale query images are produced from the original Vision texture image by appropriate resizing of the original images. Figure 9 shows the five different image scales for each query image. The multiscale algorithm used is *case 2 overlapping*, as from the previous experiment, it is a much better approach.

Table 2 shows the retrieval results for the experiment. From the table, it is obvious that as the scale of the query images moves away from the original scale, the retrieval rate drops dramatically. By using the query images with the same scale as the target sub-images, a 100% retrieval rate is recorded. On the other hand, by using query images of approximately twice the scale of the target sub-images

Table 2 Retrieval rate for five different scales of query images

Query image	Scale 1 (original)	Scale 2	Scale 3	Scale 4	Scale 5
Average retrieval rate (%)	100	71.1	18.9	4.4	5.5

Table 3 Retrieval rate for five different scales of query images, with target region increased to 140 × 140

Query image	Scale 1 (original)	Scale 2	Scale 3	Scale 4	Scale 5
Average retrieval rate (%)	96.7	78.8	37.7	38.9	100

Table 4 Retrieval rate for five different scales of query images, with target region increased to 200 × 200

Query image	Scale 1 (original)	Scale 2	Scale 3	Scale 4	Scale 5
Average retrieval rate (%)	100	86.7	46.7	41.1	94.4

(scale 5), a very poor 5.5% retrieval rate is recorded. However, note that in the database, the target sub-images are only of size 80 × 80 pixels. Therefore, when the image is re-scaled to the next resolution, there are no blocks that manage to capture the homogeneous target sub-images. During the second scale, the 80 × 80 target sub-images are re-scaled to 40 × 40 pixels, hence the 64 × 64 block capturing the target region also consists of another texture. If the size of the target images is increased, the likelihood that the 64 × 64 block will capture the homogeneous target region is much better. To confirm this, another two sets of experiments are conducted. In these experiments, the 80 × 80 pixels target region is replaced by a much larger target. We experimented with 140 × 140 and 200 × 200 pasted sub-images. Note that only the size of the sub-image is different, the scale of the target remains the same.

Tables 3 and 4 show the retrieval results of the respective experiments. The retrieval results, using different scales, have improved tremendously, especially when employing scales 3, 4 and 5. This confirms our previous assumption that the multiscale algorithm works better if the proportion of the target texture is of appropriate size. The larger the proportion, the better the multiscale algorithm works in reducing the scale dependence of the texture feature. The difference will be more obvious if the database images used are larger than 256 × 256, since more scales will be involved. However, without a suitable indexing system at this stage to speed up the matching process, we restrict the database image size to 256 × 256. From

Tables 3 and 4, it can be seen that scale 5 shows a very good retrieval result. This is probably because scale 5 is very close to half of the original scale of the query image, which makes the feature vector of the query image at scale 5 very much closer to the feature vector of the second level sub-images.

Without the multiscale feature, only the original scale is used for comparison, hence the larger the re-scale factor, the higher the dissimilarity to the original texture. Assuming the target texture is of appropriate size, the multiscale algorithm helps in reducing the scale dependence of the DWF texture feature. Without the multiscale feature, textures with different scales will be much harder to match and retrieve. Figure 10 shows an example of retrieval results for different scales using query image 9 (best result). The target region size is 140×140. We can see that when the scale of the query changes, the sub-images corresponding to different scales are retrieved.

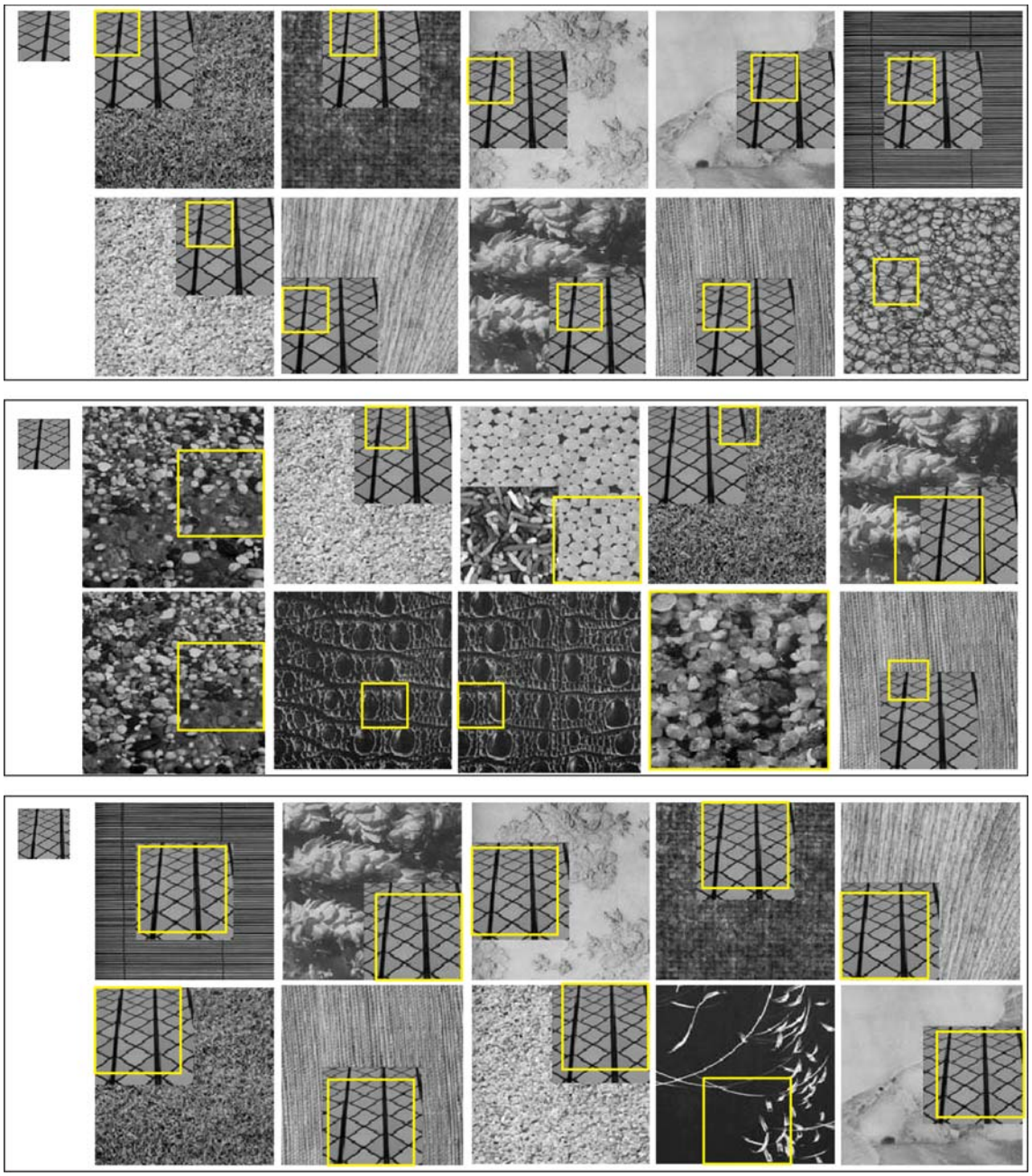

Fig. 10 Example of retrieval result for three different scales of query image

6.1.3 Size of the query images

In the previous experiments, all the query images used were of size 64×64. This section examines whether the size of the query images is crucial to the results of the retrieval. However, it is important not to misinterpret this with the scale of the query which has already been investigated in the last section. What is meant by size is the resolution of the query image itself. When we change the size of the query image from 64×64 to say, 128×128, the scale of the query image remains the same. For instance, consider an image which has a large homogeneous texture region within it. In order to use the textured region in the image as a query, we can simply crop parts of the texture region and compute its feature vector. We can either crop a large rectangle or a smaller one, as long as the texture is well represented. This is what is meant by the size of the query image. We would like to examine if there are any differences if only a small rectangular patch is used, instead of a larger rectangular patch. Theoretically, the feature vectors of the two textures should not be significantly different since the standard deviation and zero-crossing computation is averaged over the total number of pixels, hence the retrieval result should not be affected much. However, to confirm this, a new set of experiments are conducted. The database used is the same as the one used in Sect. 6.1.1, where the target images are pasted on nine different overlapping regions. The query images used are of sizes 48×48, 64×64, 96×96, 128×128 and 150×150.

Table 5 shows the retrieval results for different sizes of query images. The performance of the different query image sizes do not seem to differ much, except for some particular query images, namely queries 6, 8 and 9 (refer Fig. 7). Although the retrieval accuracy for query 6, using query size 48×48, is poor, it was observed that most of the top ten retrieved images consist of visually similar textures from the Brodatz collection. The ten target images are not far down the ranking, hence the retrieval result in general is still relatively good. As for queries 8 and 9, the reason for poor results using sizes 128×128 and 150×150 is probably because, as the size increases, the textures tend to vary a little. For example, certain sections of the textures darkened. Hence, in general we can conclude that, given that the properties of the texture do not change very much, the query image can take any size, which is an advantage if the CBIR system supports image cropping to provide query patches.

Table 5 Retrieval rate for five different sizes of query images

Query image	48×48	64×64	96×96	128×128	150×150
Average retrieval rate (%)	82.2	100	100	82.2	82.2

6.1.4 The decomposition issue

In our previous work [8], we found that periodic padding should be used for the DWF, if the translation invariance property is to be maintained. However, by using the multiscale image decomposition technique, we are dealing with image blocks and not separate entities. Therefore, one might argue that the border information can be extracted from neighbouring image blocks by borrowing border pixels in the filtering operation. Although the DWF decomposition and feature extraction processes are no longer independent for each spatial block, the exchange offers an elegant solution for padding, and the order of operation can be reversed.

First, the entire image is decomposed using wavelet filtering, and then the patches are slid across the stack of DWF coefficients in order to compute the features for each sub-image. After the image is re-scaled to the appropriate size, the DWF decomposition is continuously applied till the lowest scale image. This implies the DWF decomposition only has to be applied s times, where s is the number of scales (once for each scale), instead of applying it to each sub-image generated by the multiscale decomposition. The parent images, however, will need to be padded with a periodic padding. The feature extraction is also simplified by using this technique, where the standard deviation and the number of zero-crossings are computed immediately within each block, like sliding a standard deviation and zero-crossings function over a stack of images.

The database used is the one used in Sect. 6.1.1, where each target image is pasted randomly on nine selected database images. The query images are of the same scale as the target images and of size 64×64. It was observed that the DWF followed by the decomposition approach does not result in satisfactory retrieval rate. The average retrieval rate is recorded as below 50% compared to almost 80% for the alternate approach. This is because of the brightness invariance pre-processing operation of the DWF. Before applying the wavelet frames decomposition to an image, a common practice is to subtract the local mean of the image in order to reduce the bias caused by different illumination conditions. However, applying the DWF before the block decomposition implies that the mean to be subtracted is the global mean, and not the local mean of a particular texture. In other words, only the parent image has a zero-mean. The texture regions of interest will not record a zero-mean. When the feature vectors of these non-zero-mean texture regions are compared to the feature vector of the query texture which is zero-mean, the dissimilarity will be larger

due to the difference in the pixel histogram. We therefore confined ourselves to the original approach, which is first to perform the image decomposition, and second to apply the DWF.

6.2 Arbitrary size images database

Now that important measurements of the multiscale algorithm have been evaluated using a dyadic size image database, the multiscale algorithm will be tested on an arbitrary size image database. The procedure is not significantly different from previous experiments. From each 512×512 scanned Brodatz texture image, nine randomly sized images are produced, which can be of any height, width and location within the parent images. This results in 1,008 random size images in the database. Next for each 10 Vision textures, target images of size 80×80 are randomly pasted onto 9 different database images, resulting in 90 modified database images consisting of target textures. Each of the ten Vision textures (of size 64×64) is used as the query for the retrieval experiment.

An average of 87.8% accuracy is reported from the experiment. If we compare this rate with the rate using the dyadic image sizes, where the retrieval rate is 78.9%, the difference is rather significant. This is mainly due to the fact that, by using randomly sized images, we increase the coverage of sub-images in the multiscale algorithm, hence resulting in much better localisation. Recall that the minimum coverage of the multiscale method is 9/16. This minimum coverage is achieved only when the image size is in multiple of 64. For images that are not of this size, the sub-image coverage will be larger than 9/16. All the results obtained from previous experiments using the dyadic image database present the worst case scenario for each experiment. If the database is not restricted to dyadic image sizes, the retrieval rate will be better. Figure 11 shows some retrieval results using an arbitrary size image database. In order to accommodate the figure, images shown are resized by different factors.

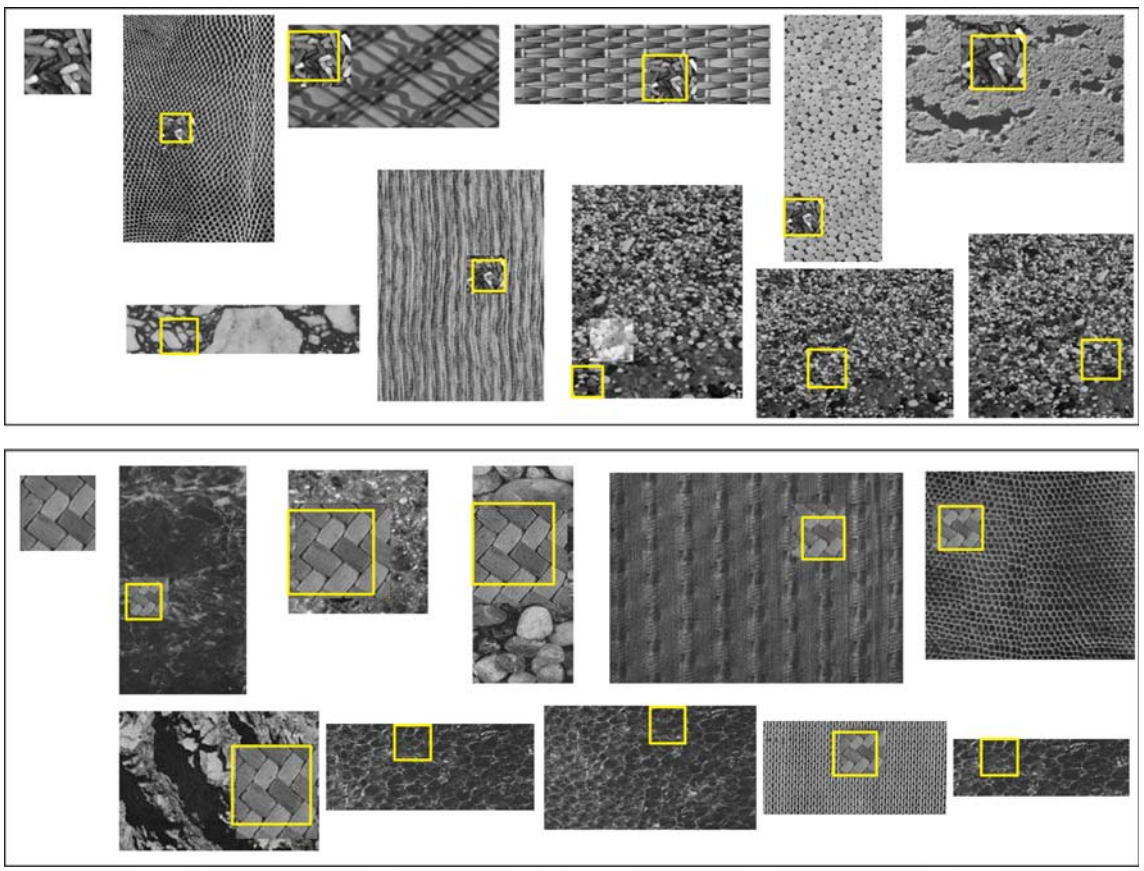

Fig. 11 Example of retrieval results for two different query images, using database of arbitrary size image. The images shown are not according to scale

6.3 Museum image collection

The proposed algorithm is now tested on a real database created from museum image collections. We experimented with three different museum image databases, namely the National Gallery database (more than 1,000 images), the Victoria and Albert Museum database (about 17,000 images) and the Research and Restoration Centre for the Museum of France (C2RMF) database (2,500 images), all of which are provided by the respective galleries or museums as part of the Artiste project [22].

The three databases present quite different challenges. The Victoria and Albert Museum database consists of mainly images of single objects, hence it is not too complex, but the size of the database is the largest. The National Gallery database contains more complex images, as all of its images are of paintings. Finally, the C2RMF database offers the most difficult challenge since some of its images are of paintings in need of restoration and sometimes not many textures are significantly visible.

Figure 12 shows some retrieval results for the museum image database. It is not possible to conduct a quantitative analysis for this experiment as there is no satisfactory ground truth data, i.e. for any particular query patch, we are uncertain as to how many images are in the database containing a texture region similar to the query, unlike in the previous experiments. However, from the figure, it can be observed that the proposed approach works well with all three databases, where almost all images in the top rank contain regions of similar texture to the query. Even for the C2RMF database, the algorithm manages to retrieve visually similar texture, although its performance is not as good as for the other two databases. This is because the nature of the images within the database makes it confusing. The other two databases, on the other hand, show much better retrieval results. Even for a database size of 17,000 images

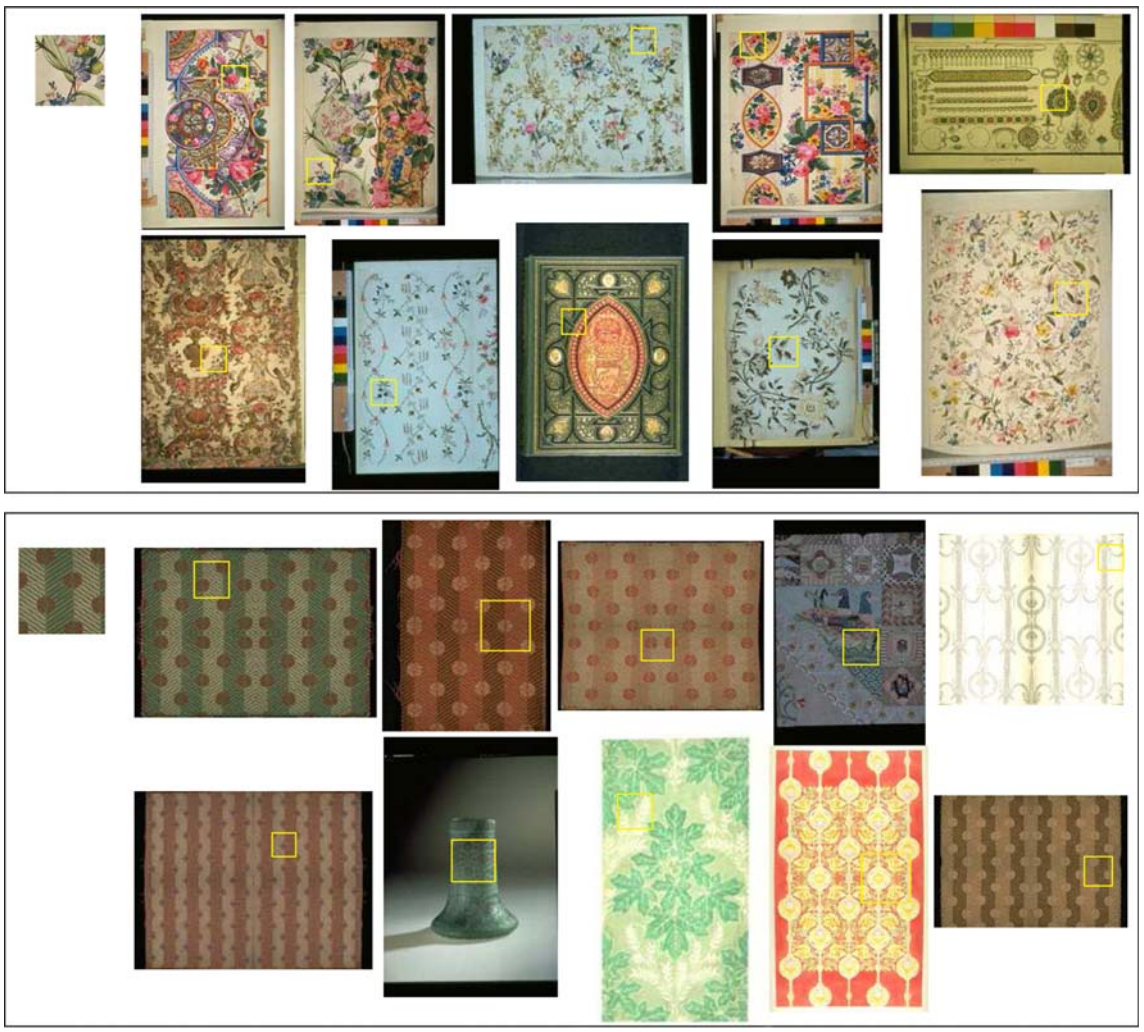

Fig. 12 Example of retrieval results for two different query images, using museum image database. The images shown are not according to scale

for the Victoria and Albert Museum database, the algorithm manages to perform its task well. The proposed algorithm can therefore be applied effectively in CBIR applications and the approach has been incorporated into the Artiste project CBIR system, and useful results have been reported.

7 Conclusion and future works

In this paper, a modified multiscale sub-image matching method is proposed. The multiscale algorithm was chosen because of its simplicity and the fact that it can assist in reducing the scale dependence of the feature extractor. The modifications to the algorithm are necessary in order for it to be used for texture retrieval and to improve performance. These include the scaling factor for different scales, as well as the positioning of the blocks within the entire image of interest. Several experiments were carried out in order to evaluate the performance of the multiscale algorithm. In the first experiment, it was found that *case 2 overlapping* is far better than *case 1 overlapping*, hence making the extra sub-images of the former worthwhile.

The experiments suggest that the multiscale algorithm is in fact a very good method for achieving scale invariant texture retrieval. Although it is not perfectly scale invariant, the algorithm helps to reduce the scale dependence of the texture features. It is also found that the size of the query images has little effect on retrieval results, making it a robust technique. Finally, the combination of the multiscale algorithm with the DWF can be used for an arbitrary size image database, where good retrieval results are observed with the Brodatz database as well as in real databases of museum image collections. The multiscale nature of the approach is especially useful in capturing both coarse and fine textures.

Future improvements to the algorithm will mainly be concerned with reducing the computational load. One of the possibilities might be to use *case 1 overlapping* for sub-image coverage, instead of *case 2 overlapping*, although the accuracy might drop quite drastically as well. Another possibility is to introduce a texture identifier to decide whether a particular sub-image is textured or not. The feature vectors are then created only for the textured patches, reducing the total number of feature vectors to be calculated and compared. We would also like to incorporate rotation invariance into the algorithm and a suitable multidimensional indexing strategy to accelerate the matching process.

Acknowledgments The authors are grateful to the Faculty of Engineering, Multimedia University, Malaysia and the School of Electronics and Computer Science at the University of Southampton, UK for financial support. We are also grateful to the EU for their support under grant number IST_1999_11978 (The Artiste Project), and to our collaborators, the Victoria and Albert Museum (London, UK), the National Gallery (London, UK) and the Research and Restoration Centre for the Museum of France (Paris, France) for use of their images.

References

1. Rui Y, Huang TS (1999) Image retrieval: current techniques, promising directions, and open issues. J Visual Comm Image Rep 10:39–62
2. Hlaoui A, Sun H-J, Wang S-R (2002) Image retrieval using fuzzy segmentation and a graph matching technique. In: Proceedings of the international conference on machine learning and cybernetics, pp 1987–1992
3. Liu Y, Zhou X (2004) Automatic texture segmentation for texture-based image retrieval. In: Proceedings of the international conference on multimedia modelling, pp 285–290
4. Ko B, Peng J, Byun H (2001) Region-based image retrieval using probabilistic feature relevance learning. Pattern Anal Appl 4:174–184
5. Choras RS, Andrysiak T, Choras M (2007) Integrated color, texture and shape information for content-based image retrieval. Pattern Anal Appl. doi: 10.1007/s 10044-007-0071-0(online first)
6. Rubner Y, Tomasi C (1999) Texture-based image retrieval without segmentation. In: Proceedings of the IEEE international conference on computer vision, pp 1018–1024
7. Chan S, Martinez K, Lewis P, Lahanier C, Stevenson J (2001) Handling sub-image queries in content-based retrieval of high resolution art images. In: Proceedings of the international conference in cultural heritage and technologies, pp 157–163
8. Fauzi MFA (2004) Content-based image retrieval of museum images. PhD Thesis, University of Southampton
9. Pun C-M, Lee M-C (2002) Rotation invariant texture feature for content based image retrieval. In: Proceedings of the IEEE international conference on multimedia and expo, pp 173–176
10. Muneeswaran K, Ganesan L, Arumugam S, Soundar KR (2005) Texture classification with combined rotation and scale invariant wavelet features. Pattern Recognit 38:1495–1506
11. Manjunath BS, Ma WY (1996) Texture features for browsing and retrieval of image data. IEEE Trans Pattern Anal Mach Intell 18:837–842
12. Natsev A, Rastogi R, Shim K (1999) WALRUS: a similarity retrieval algorithm for image databases. In: Proceedings of ACM SIGMOD international conference on management of data, pp 395–406
13. Smith JR, Chang S-F (1994) Quad-tree segmentation for texture based image query. In: Proceedings of the ACM multimedia conference, pp 279–286
14. Guo J, Zhang A (1997) Image decomposition and representation in large image database systems. J Visual Comm Image Rep 8:167–181
15. Graps A (1995) An introduction to wavelets. IEEE Comput Sci Eng 2:50–61
16. Mallat SG (1989) A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans Pattern Anal Mach Intell 11:674–693
17. Unser M (1995) Texture classification and segmentation using wavelet frames. IEEE Trans Image Process 4:1549–1560
18. Chang T, Kuo CCJ (1993) Texture analysis and classification with tree-structured wavelet transform. IEEE Trans Image Process 2:429–441

19. Fauzi MFA, Lewis PH (2006) Automatic texture segmentation for content-based image retrieval application. Pattern Anal Appl 9:307–323
20. Brodatz P (1966) Textures: a photographic album for artists & designers. Dover, New York
21. Picard R et al (1995) Vision Texture 1.0. MIT Media Lab, at http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html
22. Artiste Project. At http://www.artisteweb.org

Author Biographies

Mohammad Faizal Ahmad Fauzi received the B.Eng. degree in electrical and electronic engineering from Imperial College, London, UK in 1999, and the Ph.D. degree in electronics and computer science from University of Southampton, Southampton, UK in 2004. He is currently attached to the Multimedia University, Malaysia as a lecturer/researcher. His main research interests are in the area of signal processing, analysis, retrieval and compression of image, audio and video data, as well as biometrics.

Paul H. Lewis received the B.S. degree in physics from Imperial College, London, and a Ph.D. degree in physics from London University in 1972. He is a Professor of Computer Science in the Intelligence, Agents, Multimedia Group in the School of Electronics and Computer Science at the University of Southampton in the UK. His main research interests are in the area of image and video content analysis, semantic analysis and applications to intelligent multimedia information handling. Particular application areas include the medical domain and cultural heritage systems. He is a member of the UK Multimedia Knowledge Management Network Steering group.