Texture Feature Extraction

5

The devil is in the detail.

5.1 Introduction

Texture is a general pattern that can be attributed to almost everything in nature. For a human, texture patterns relate to specific and spatially repetitive structure of surfaces formed by repeating a particular element or several elements in different spatial positions. Generally, the repetition involves local variations of scale, orientation, or other geometric and optical features of the elements.

Texture is an inherent feature of an object. For example, we can easily tell if an object surface is fine or rough, regular or natural, quiet or busy, etc. It is found that human beings tend to recognize texture by its structure or how often it changes. As the result, the texture methods designed in the last few decades are along two directions: spectral methods and spatial methods. Spatial texture methods attempt to capture the primitive patterns of objects and compute the structural features; while spectral texture methods attempt to capture the change patterns of objects and compute the frequency of changes. Spatial methods are generally more intuitive, while spectral methods are generally more efficient and robust. In this chapter, we discuss those important texture methods for image representation.

5.2 Spatial Texture Feature Extraction Methods

In spatial approach, texture features are extracted by computing the pixel statistics or finding the local pixel structures in the original image. These methods include the Tamura textures, co-occurrence matrix method, Markov random field (MRF) method, and fractal dimension (FD) method. Tamura et al. are among the earliest researchers to formally define texture features [1]. The most cited Tamura texture features in literature consist of six perceptual characteristics of images such as the degree of contrast, coarseness, directionality, linearity, roughness, and

D. Zhang, *Fundamentals of Image Data Mining*, Texts in Computer Science, https://doi.org/10.1007/978-3-030-17989-2_5

regularity. In most of the cases, only the first three Tamura features are used as the other three features are defined based on the combinations of the first three features. Tamura features are nice because they are high-level perceptual features and suitable for texture browsing. However, it is difficult to define more such types of high-level features. Therefore, Tamura features are not enough to distinguish all the textures in the world.

5.2.1 Tamura Textures

Tamura et al. [1] introduce six statistical features. These include *coarseness*, *contrast*, *directionality*, *line-likeness*, *regularity*, and *roughness*. The last three features are defined based on the first three features. Therefore, most of the image retrieval systems only use the first three Tamura features.

Coarseness relates to the size of the primitive elements (textons) forming the texture, and it measures the image granularity. It is calculated as the average of the largest window sizes needed to identify texture elements centered at different pixel positions. Formally, it is defined as

$$f_{crs} = \frac{1}{n^2} \sum_{x=1}^{n} \sum_{y=1}^{n} 2^k I(x, y) \tag{5.1}$$

where $n \times n$ denotes the image size of $I(x, y)$, and k is obtained as the value which maximizes the differences of the moving averages of $A_k = \frac{1}{2^{2k}} \sum_{x=1}^{n} \sum_{y=1}^{n} I(x, y)$, taken over a $2^k \times 2^k$ neighborhood along the horizontal and vertical directions. The specific procedure is to compute the differences between the average signals for the nonoverlapping windows of different size:

(1) At each pixel (x, y), compute six averages for the windows of size $2^k \times 2^k$, $k = 0, 1, \ldots, 5$, around the pixel.
(2) At each pixel, compute absolute differences $E_k\ (x, y)$ between the pairs of nonoverlapping averages A_k in the horizontal and vertical directions.
(3) At each pixel, find the value of k that maximizes the difference $E_k\ (x, y)$ in either direction and set the best size $S_{\text{best}}\ (x, y) = 2^k$.
(4) Compute the coarseness feature f_{crs} by averaging $S_{\text{best}}\ (x, y)$ over the entire image. Textures with multiple coarseness can be computed from the histogram of $S_{\text{best}}\ (x, y)$.

Contrast tells how well an object is distinguishable from other objects or background. It measures how gray levels q vary in the image I and to what extent their distribution is biased to black or white. The second-order σ^2 and normalized fourth-order central moments μ_4 of the gray level histogram (empirical probability distribution P) are used to define the contrast feature:

$$f_{con} = \frac{\sigma}{\left(\frac{\mu_4}{\sigma^4}\right)^{\frac{1}{4}}} \tag{5.2}$$

where $\mu_4 = \sum_{q=0}^{q_{max}} (q-m)^4 P(q|I)$ is the *kurtosis*; $\sigma^2 = \sum_{q=0}^{q_{max}} (q-m)^2 P(q|I)$ is the *variance*, and m is the *mean* gray level.

Directionality tells if there exists any directional pattern in an image, like vertical, horizontal, diagonal, etc. The degree of directionality is measured using the frequency distribution of oriented local edges against their directional angles. The edge strength $e(x, y)$ and the directional angle $a(x, y)$ are computed to approximate the pixelwise x and y derivatives of the image:

$$e(x,y) = \left(|\Delta_x(x,y)| + \left|\Delta_y(x,y)\right|\right)/2 \tag{5.3}$$

$$\phi(x,y) = arctan(\Delta_y(x,y)/\Delta_x(x,y)) \tag{5.4}$$

where $\Delta_x(x, y)$ and $\Delta_y(x, y)$ are the horizontal and vertical gray level differences between the neighboring pixels, respectively. They are computed by using Prewitt edge detectors

Δ_x			Δ_y		
−1	0	1	1	1	1
−1	0	1	0	0	0
−1	0	1	−1	−1	−1

A histogram $h_{\text{dir}}(\phi)$ of quantized direction values ϕ is constructed by counting the numbers of the edge pixels with the corresponding directional angles and the edge strength greater than a predefined threshold. The histogram is relatively uniform for images without strong orientation and exhibits peaks for highly directional images. The directionality feature is defined as the *sharpness* of the histogram:

$$f_{dir} = 1 - rn_p \sum_{p=1}^{n_p} \sum_{\phi \in w_p} \left(\phi - \phi_p\right)^2 h_{dir}(\phi) \tag{5.5}$$

where n_p is the number of peaks, ϕ_p is the position of the pth peak, w_p is the range of the angles attributed to the pth peak (that is, the range between valleys around the peak), r denotes a normalizing factor related to quantising levels of the angles ϕ, and ϕ is the quantized directional angle.

Figure 5.1b shows the edge map of an original image in Fig. 5.1a. To compute f_{dir}, h_{dir} is first computed by quantizing ϕ and counting the number of edge pixels with $e(x, y)$ greater than a threshold and the edge histogram of angles is shown in Fig. 5.1c. The angles ϕ in the horizontal axis are in the range of −90° to +90° and

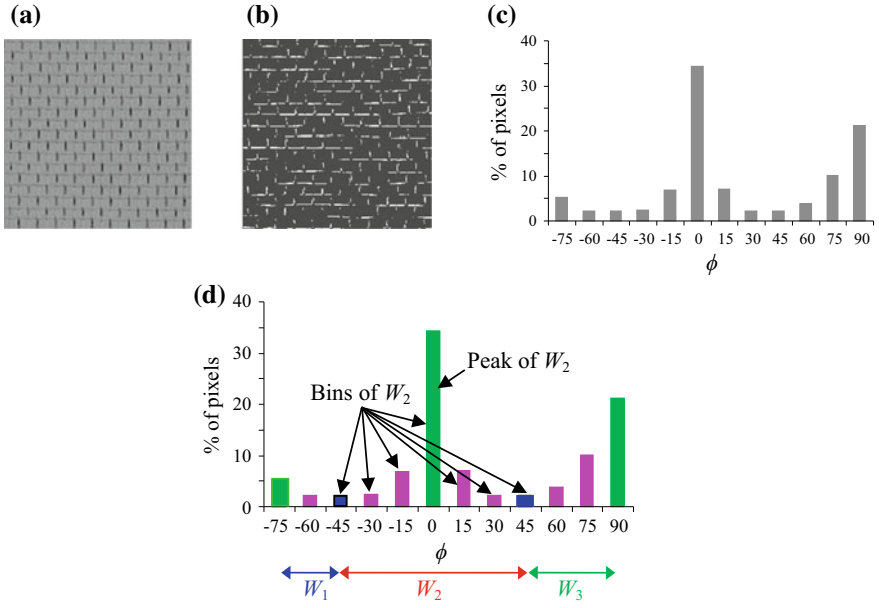


Fig. 5.1 An example of computing directionality

are quantized into 12 intervals and the quantized angles are −75°, −60°, −45°, …, +90°. The vertical axis shows the percentage of edge pixels at different angles [2].

After computing the h_{dir}, all peaks and valleys in h_{dir} are detected. Figure 5.1d shows the peaks and valleys in green and blue, respectively. Suppose, there are n_p peaks in the histogram. For each peak p, let w_p be the window of bins from the previous valley to the next valley (a window contains a peak in it), and ϕ_p be the angular position of the peak in w_p. Based on the definition of f_{dir}, the more directional an image, the higher the directionality the image. However, f_{dir} is not invariant to rotation, rotation of an image causes a circular shift of the h_{dir} histogram, and this can cause false peak detection. For example, the first peak in Fig. 5.1d is actually a part of the hill defined by the last peak of the histogram. Therefore, in practice, several rounds of circular shift are needed to find out the real peaks of the histogram.

Tamura textures are intuitive in terms of definitions, however, the computation processes are complex. This affects the robustness and the overall performance of the computed features.

5.2.2 Gray Level Co-occurrence Matrices

Many statistical texture features are based on gray level co-occurrence matrices (GLCM) or its color counterpart color correlogram. A GLCM represents how

frequent is every particular pair of gray levels in an image, separated by a certain distance d along a certain direction a.

Formally, given an $n \times m$ image $I(x, y)$, a cell in a GLCM is defined as

$$C_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{n}\sum_{y=1}^{m}\begin{cases} 1, & \text{if } I(x,y) = i \text{ and } I(x+\Delta x, y+\Delta y) = j \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

For an image with 256 gray level values, a GLCM is a 256×256 matrix. With the combination of Δx and Δy, a large number of GLCMs can be created. In practice, only four GLCMs are created by capturing the following four structures: horizontal, vertical, left lean diagonal, and right lean diagonal.

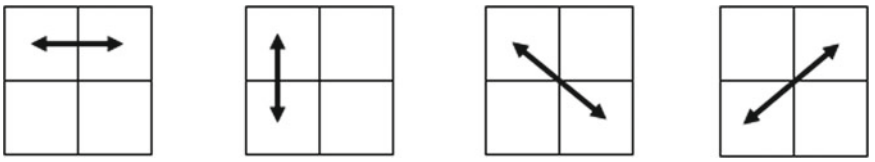


Figure 5.2 shows an example GLCM of an image $I(x, y)$ with 8 gray level values. In this case, the structure to be captured by the GLCM is the *horizontal* structure. The arrows in the figure link the pixel pairs in the image with their corresponding entries in the GLCM.

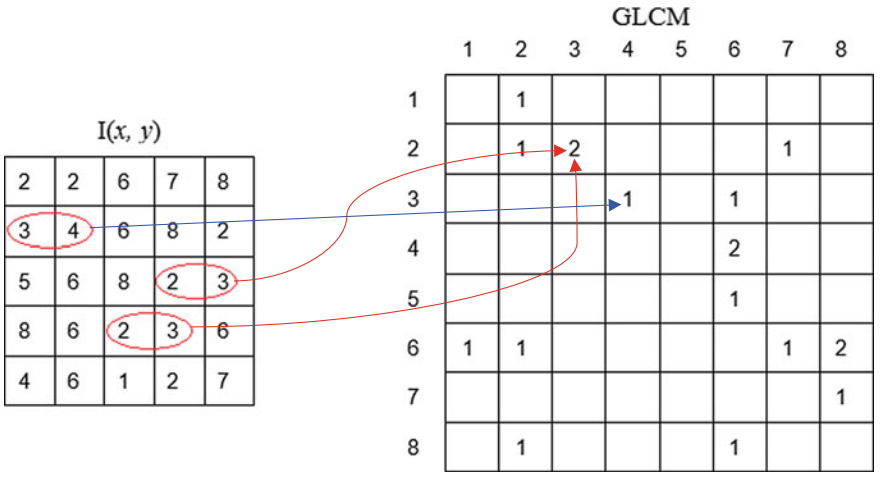


Fig. 5.2 Computation of a GLCM. An 8 gray levels image $I(x, y)$ on the left and its GLCM on the right

A GLCM itself is a gray level image, therefore, a number of statistical features called Haralick features can be computed from each GLCM image, such as the homogeneity, contrast, correlation, energy, entropy, variance, etc. [3].

Suppose $p(i, j)$ is the normalized value of a GLCM entry, it is equivalent to the probability of a particular structural distribution in the image; N_g is the number of gray levels in the quantized image. The three major texture features *angular second moment*, *contrast*, and *correlation* are given in the following:

Angular second moment:

$$f_1 = \sum_i \sum_j \{p(i,j)\}^2 \tag{5.7}$$

Contrast:

$$f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_i \sum_j [p(i,j)||i-j| = n] \right\} \tag{5.8}$$

Correlation:

$$f_3 = \frac{\sum_i \sum_j (ij)p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{5.9}$$

where μ_x, μ_y, σ_x, and σ_y are the means and standard deviations of the marginal probabilities p_x and p_y, respectively. f_1 is a measure of homogeneity of the image and f_3 is a measure of gray tone linear dependencies in the image.

Because a GLCM is usually a large matrix and needs a number of GLCMs to capture different texture structures, GLCM features are expensive to compute.

5.2.3 Markov Random Field

MRF texture methods model image pixel location as a random variable, as a result, an image is a random field. Each type of textures is characterized by a joint probability distribution of signals that accounts for spatial interdependence, or interaction among the signals. The interacting pixel pairs are usually called neighbors, and a random field texture model is characterized by geometric structure and quantitative strength of interactions among the neighbors.

Among the many MRF texture methods, the Simultaneous Auto-Regressive (SAR) model is the most widely used, as it uses fewer parameters. In SAR, the intensity $I(x, y)$ at pixel (x, y) is estimated as a linear combination of the neighboring pixel values $I(s, t)$ and an additive Gaussian noise $\varepsilon(x, y)$.

$$I(x,y) = \mu + \sum_{(s,t)\in N} \theta(s,t)I(s,t) + \varepsilon(x,y) \tag{5.10}$$

where

- μ is the mean of the image,
- N is the neighborhood of (x, y), e.g., a 3 × 3 window,
- $\theta(s, t)$ are the weights or coefficients associated with each of the neighborhood pixels,
- and $\varepsilon(x, y)$ is a Gaussian error with zero mean and standard deviation of σ.

The set of parameters θ and σ are the measurement of the texture, they can be estimated using either the least square error (LSE) technique or the Maximum Likelihood Estimation (MLE). Both LSE and MLE involve complex optimization which is computationally expensive. A higher σ value indicates finer granularity or less coarseness; a higher $\theta(x, y + 1)$ and $\theta(x, y - 1)$ values indicate that the texture is vertically oriented, so on so forth.

Rotation-Invariant SAR model (RISAR) can be created by replacing N with a circular neighborhood. In order to make SAR more robust, Multiresolution MRF (MRMRF) can also be created, where an image is represented by a multiresolution Gaussian pyramid before applying the MRF model.

The number of parameters or the feature dimensions of a SAR depends on the size of the neighborhood, e.g., a 3 × 3 neighborhood results in 9 parameters while a 4 × 4 neighborhood results in 16 parameters. To be scale invariant, SARs with multiple window size are needed, this makes the computation of MRF features prohibitively expensive.

5.2.4 Fractal Dimension

The fractal dimension (FD) method [4] is based on the theory of fractal geometry which characterizes the shapes or patterns of self-similarity. The idea of fractal is to find the smallest structure which replicates the whole pattern. According to fractal theory, a bounded set S in Euclidean space R^n is self-similar if S is the union of $N(r)$ distinct (nonoverlapping) copies of itself scaled up or down by a ratio r, and the relationship between $N(r)$ and r is given by

$$N(r) \approx C\left(\frac{1}{r}\right)^d \tag{5.11}$$

where d is the fractal dimension or FD.

In image applications, FD method models a gray level image as a 3D terrain surface, and a differential box counting is done under the surface to measure how rough the surface is. In logarithm term, the above relationship means the number of boxes under the surface is inversely proportional to the size of the boxes, which is expected. d is given by the following approximation:

$$d = \lim_{r \to 0} \frac{\log N(r)}{\log \frac{1}{r}} \tag{5.12}$$

or

$$d \approx \frac{\log N(r)}{\log \frac{1}{r}} \tag{5.13}$$

From (5.13), the log$N(r)$ and log$(1/r)$ have an approximately *linear* relationship, therefore, FD can be estimated from a least square fitting of the two variables. By fitting a straight line for the log$N(r)$ versus log$(1/r)$ curve, the *slope* of the straight line is taken as the approximation of FD.

Since FD only models the roughness feature, other features like directionality and contrast are missed from FD. Therefore, in [4], six FDs have to be computed from a number of modified images derived from the original image, such as, the original image, low gray-valued image, high gray-valued image, horizontally smoothed image, vertically smoothed image, and the second moment of the original image. Despite of these additional FD features, FD can be very sensitive due to the triple approximation during the box counting, linear fitting, and image modification.

5.2.5 Discussions

Spatial texture methods are based on the ideas of capturing the elemental or microstructures of a textured image. The definitions of the structures are based on how humans describe a textured image, such as rough versus fine, regular versus natural, directional versus random, etc. The advantage of these methods is that they are intuitive and semantically meaningful. However, there are infinite types of textural structures in the nature, and human beings can only define or describe a small number of them. This can limit the application of spatial texture methods.

Another major issue with spatial approach is that spatial features are sensitive to noise. Furthermore, spatial texture methods are usually complex to compute, and they often involve complex optimization which is very expensive to compute. These issues affect the robustness and the overall performance of spatial texture methods.

5.3 Spectral Texture Feature Extraction Methods

Instead of defining and describing specific structures in an image, which is difficult, spectral texture methods attempt to capture how frequent the patterns change in a textured image. Spectral texture methods are based on Fourier Transform (FT), Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), Gabor filters, curvelet transform, etc.

Global power spectra computed from the DFT are not effective in texture classification and retrieval, compared with local features computed from small windows such as DCT. At present, the most promising features for texture retrieval are based on multiresolution features obtained with orthogonal wavelet transforms or Gabor filters. These features describe spatial distributions of oriented edges in the image at multiple scales.

5.3.1 DCT-Based Texture Feature

Compared with the traditional spatial texture methods, DCT is a simple yet robust method to capture local textures of an image. The idea is equivalent to STFT or applying FT on a small window. However, due to the use of 1D cosine transform on both rows and columns, the computation is very efficient.

For a color image I, it is first converted to Y′CbCr or YBR colors. The image is then divided into a set of overlapping 8×8 regions or blocks, which are obtained by a sliding window that moves by two pixels between consecutive samples. At each location of the three YBR color channels, apply the DCT on the local 8×8 window. Each block is then represented by

$$\mathbf{x} = \left[\mathbf{x}^Y, \mathbf{x}^B, \mathbf{x}^R\right] \tag{5.14}$$

where $[\mathbf{x}^Y, \mathbf{x}^B, \mathbf{x}^R]$ is the concatenation of the DCT vectors extracted from each of the YBR color channels by a zigzag scanning. For efficient computation, the 192-dimensional YBR-DCT vector is usually shortened by only retaining the first few coefficients from each of the YBR channels. This is because of the well-known energy compaction properties of the DCT.

To compute the texture features of the image I, a Gaussian mixture model of eight components is computed using the EM algorithm. This produces the following conditional distribution for each image:

$$P(\mathbf{x}|I) = \sum_{k=1}^{8} \pi_I^k G(\mathbf{x}, \mu_I^k, \sigma_I^k) \tag{5.15}$$

where π_I^k is the weight and μ_I^k and σ_I^k are the maximum likelihood parameters of mixture component k. The (μ_I^k, σ_I^k) pair is then organized into a feature vector which

is used for texture representation. For a gray level image, the DCT method just needs to replace the three color channels with a single gray channel.

DCT computation is efficient due to the use of FFT, however, the EM is an optimization method which incurs significant computation cost.

5.3.2 Texture Features Based on Gabor Filters

5.3.2.1 Gabor Filters

Although DCT is efficient to compute locally, the image level texture features are complex to compute due to the use of EM algorithm. An alternative is to use Gabor filters. Gabor filters are based on traditional filter-based image processing approach, which computes one filtered value at each pixel as opposing to computing multiple transformed values at each location as in the DCT. Different from traditional filters, by combining both Gaussian and FT, Gabor filters simulate the powerful properties of perceptual vision of mammals. Furthermore, they can be tuned to different orientations and scales.

Gabor transform creates a filter bank consisting of Gabor filters with various scales and orientations. For a given image $I(x, y)$ with size $P \times Q$, its discrete Gabor transform is given by a convolution:

$$G_{mn}(x,y) = \sum_{s=0}^{K} \sum_{t=0}^{K} I(x-s, y-t) g^*_{mn}(s,t) \tag{5.16}$$

where K is the filter mask size, and g^*_{mn} is the complex conjugate of g_{mn} which is a class of self-similar wavelets generated from dilation and rotation of the following mother wavelet:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \cdot \exp(j2\pi Wx) \tag{5.17}$$

where W is called the modulation frequency. The self-similar Gabor wavelets are obtained through the generating function

$$g_{mn}(x,y) = a^{-m} g(\tilde{x}, \tilde{y}) \tag{5.18}$$

where m and n specify the *scale* and *orientation* of the wavelet respectively, with $m = 0, 1, \ldots M - 1$, $n = 0, 1, \ldots, N - 1$, and

$$\left.\begin{aligned} \tilde{x} &= a^{-m}(x\cos\theta + y\sin\theta) \\ \tilde{y} &= a^{-m}(-x\sin\theta + y\cos\theta) \end{aligned}\right\} \tag{5.19}$$

where $a > 1$ and $\theta = n\pi/N$.

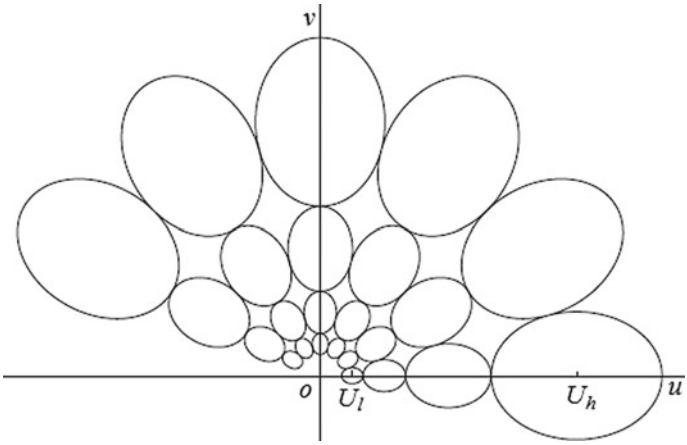


Fig. 5.3 FWHM sampling of spectral responses of Gabor filters in frequency plane

In order to decide the bank of Gabor filters, the parameters in (5.17) to (5.19) have to be determined. The Gabor wavelets generated using (5.17) through (5.19) are complete but not orthogonal wavelets, it implies there is redundancy in the filters. Therefore, we may design a bank of Gabor filters so that redundancy will be significantly reduced while image texture will be well represented by the set of individual filter response. Due to the same functional form of Gabor wavelet function $g(x, y)$ and its frequency response $\Psi(u, v)$ (i.e., its 2-D Fourier transform):

$$\Psi(u,v) = \exp\{-\frac{1}{2}[\frac{(u-W)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}]\} \tag{5.20}$$

where $\sigma_u = (2\pi\sigma_x)^{-1}$ and $\sigma_v = (2\pi\sigma_y)^{-1}$, an optimal representation of an image in spatial domain can be achieved by finding an optimal representation of the image in frequency domain. The idea of achieving so is to use the full width at half maximum (FWHM) of the Gabor spectral functions to form a complete coverage of the frequency plane within the modulation frequency bandwidth (Sect. 2.2). Specifically, the design follows three principles [5]: (i) *Uniform sampling* of orientation angles; (ii) *Exponential sampling* of modulation bandwidth W; (iii) Continuous coverage of the frequency space. This strategy is illustrated in Fig. 5.3.

Let U_l and U_h be the lowest and highest frequencies of interest, such that the coarsest scale filter and the finest scale filter are centered in the frequency domain at distances U_l and U_h from the origin, respectively. By the above strategy of redundancy reduction, the parameters of Gabor filters in spatial domain can be determined as follows:

$$a = (U_h/U_l)^{\frac{1}{M-1}} \tag{5.21}$$

$$\sigma_{x,m,n} = \frac{(a+1)\sqrt{2\ln 2}}{2\pi a^m (a-1) U_l} \tag{5.22}$$

$$\sigma_{y,m,n} = \frac{1}{2\pi \tan\left(\frac{\pi}{2N}\right)\sqrt{\frac{U_h^2}{2\ln 2} - \left(\frac{1}{2\pi\sigma_{x,m,n}}\right)^2}} \tag{5.23}$$

The parameters are independent of orientations (n), in other words, the parameters repeat in every orientation. In practice, the following parameter values are used:

$$U_l = 0.05, U_h = 0.4, K = 60$$

5.3.2.2 Gabor Spectrum

Figure 5.4 shows the Gabor filtered subband images from the lady image of Fig. 3.2. It shows how different image features are captured by Gabor filters from different scales and orientations. It can be observed that low-frequency information are captured at lower scales and as scale increases (*in spectral domain*), more fine details can be seen.

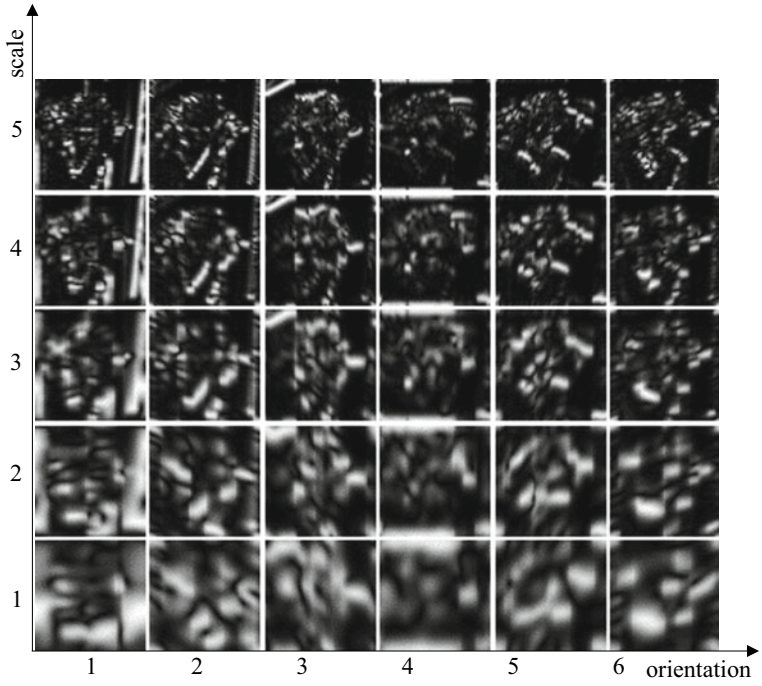


Fig. 5.4 Gabor filtered subbands for the lady image

5.3.2.3 Texture Representation

After applying Gabor filters on the image with different orientation at a different scale, a set of magnitudes is obtained:

$$E(m,n) = \sum_{x=0}^{P} \sum_{y=0}^{Q} |G_{mn}(x,y)|, m = 0, 1, \ldots, M-1;\ n = 0, 1, \ldots, N-1 \quad (5.24)$$

where

- m is the scale
- n is the orientation
- M is the maximal scale
- N is the maximal orientation
- $P \times Q$ is the size of the input image

The magnitudes represent the energy map at a different scale and orientation of the image under transform (Fig. 5.5 right) [6].

The main purpose of texture-based retrieval is to find images or regions with similar texture. It is assumed that we are interested in images or regions that have homogenous texture, therefore the following mean μ_{mn} and standard deviation σ_{mn} of the magnitude of the transformed coefficients are used to represent the homogenous texture feature of the image or region:

$$\begin{aligned} \mu_{mn} &= \frac{E(m,n)}{P \times Q} \\ \sigma_{mn} &= \frac{\sqrt{\sum_x \sum_y \left(|G_{mn}(x,y)| - \mu_{mn}\right)^2}}{P \times Q} \end{aligned} \quad (5.25)$$

A feature vector $\mathbf{f}$ (texture representation) is created using μ_{mn} and σ_{mn} as the feature components. Five scales and six orientations are used in common implementation and the Gabor texture feature vector is thus given by

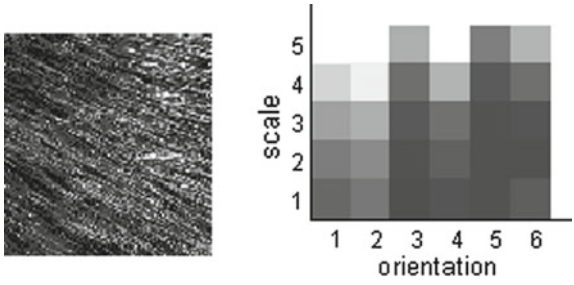


Fig. 5.5 Computation of Gabor texture descriptor. A straw image on the left and its energy map on the right. The higher the energy the brighter the block

$$\mathbf{f} = (\mu_{00}, \sigma_{00}, \mu_{01}, \sigma_{01}, \ldots, \mu_{45}, \sigma_{45}) \tag{5.26}$$

In order to remove the influence of various lighting issues of the camera, the features μ_{mn} and σ_{mn} can be normalized to [0, 1] using the maximum of the respective components. The similarity between two texture patterns is measured by the city block or Euclidean distance between their Gabor feature vectors.

5.3.2.4 Rotation-Invariant Gabor Features

The above-acquired texture feature is not invariant to rotation, similar texture images with different direction may be missed out from the retrieval or get a low rank. A simple circular shift on the feature map can be used to solve the rotation variant problem. Specifically, the orientation with the highest energy is detected as the dominant direction of the image and the feature elements in the dominant direction are moved to the first elements in **f**. The other elements are then circularly shifted accordingly. For example, if the original feature vector is "*abcdef*" and "*c*" is at the dominant direction, then the normalized feature vector will be "*cdefab*" [7].

This circular shift approach is based on the theory that image rotation in spatial domain is equivalent to circular shift in spectral domain. Assume the original image is $I(x, y)$, $I_\phi(x, y)$ is the result of $I(x, y)$ after rotation of angle ϕ, by using (1.40), we have the following:

$$I_\phi(x, y) = I(x, y) \cdot e^{-j\phi} \tag{5.27}$$

For notation convenience, the Gabor transform of $I(x, y)$ and $I_\phi(x, y)$ at scale s and angle θ are denoted as $G(I, s, \theta)$ and $G(I_\phi, s, \theta)$, respectively. Then according to (5.16), we have

$$G(I, s, \theta) = I(x, y) * g_{s\theta}(x, y) \tag{5.28}$$

and by the commutability of convolution, we have

$$\begin{aligned} G\left(I_\phi, s, \theta\right) &= I_\phi(x, y) * g_{s\theta}(x, y) \\ &= \left[I(x, y) \cdot e^{-j\phi}\right] * g_{s\theta}(x, y) \\ &= I(x, y) * \left[g_{s\theta}(x, y) \cdot e^{-j\phi}\right] \end{aligned} \tag{5.29}$$

Equation (5.29) indicates that applying a Gabor filter on a rotated image is equivalent to applying a rotated Gabor filter on the original image. Since $g_{s\theta}(x, y) \cdot e^{-j\phi} = g_{s,\theta+\phi}(x, y)$, we have

$$\begin{aligned} G\left(I_\phi, s, \theta\right) &= I(x, y) * \left[g_{s\theta}(x, y) e^{-j\phi}\right] \\ &= I(x, y) * g_{s,\theta+\phi}(x, y) \\ &= G(I, S, \theta + \phi) \end{aligned} \tag{5.30}$$

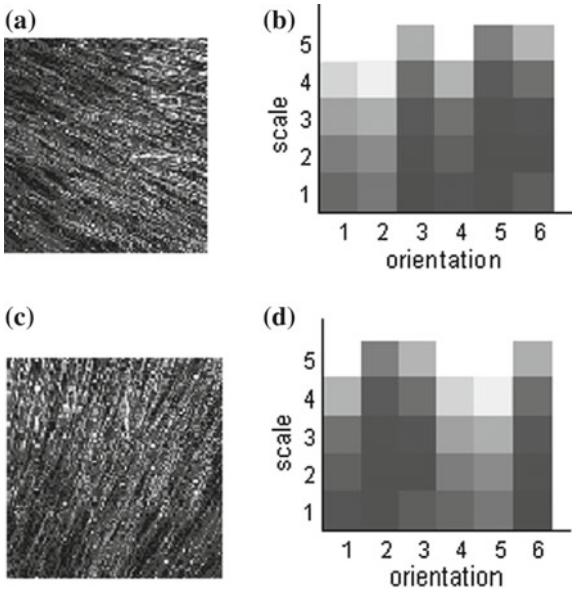


Fig. 5.6 Computation of rotation-invariant Gabor texture descriptor. **a** A straw image; **b** energy map of (**a**); **c** a rotated image of (**a**); **d** energy map of (**c**). The higher the energy, the brighter the block

Equation (5.30) indicates that a rotation of the input image $I(x, y)$ by an angle ϕ is equivalent to a translation of the output energy $G(I, s, \theta)$ by the same amount ϕ along the orientation axis. Figure 5.6 demonstrates this fact. It shows two texture patterns and their feature maps, pattern (c) is a rotation of 90° of pattern (a). Form feature map (b), it can be seen that pattern (a) has a dominant direction feature in orientation 2 (60°), while in feature map (d), this dominant direction feature has moved to orientation 5 (150°) and features in other directions are circularly shifted accordingly. In other words, the spectrum (d) is the circularly shifted version of spectrum (b) [6].

If a texture pattern has directional features, it will show dominant energy at certain direction on the energy map. If the direction of the highest energy is circularly shifted to zero degree, the resulting **f** is a rotation-invariant feature. If a texture pattern does not have dominant direction feature, the matching between rotated patterns can be made at any direction, and the rotation normalization does not affect the matching in this situation.

In image analysis, there is always a compromise between spatial resolution and frequency resolution. Gabor filters achieve optimal joint localization/resolution in both space domain and frequency domain. However, due to the truncation at half peak magnitude, the spectral cover of Gabor filters is not complete, this results in information loss in the spectral domain. For example, in Fig. 5.7, black holes are left at the FWHM in Gabor transformed spectral domain. Consequently, the high-frequency components, which are considered to be the most important in characterizing image textures, are not completely captured. Abundant redundancy also exists between transformed images at different scales because Gabor filters use

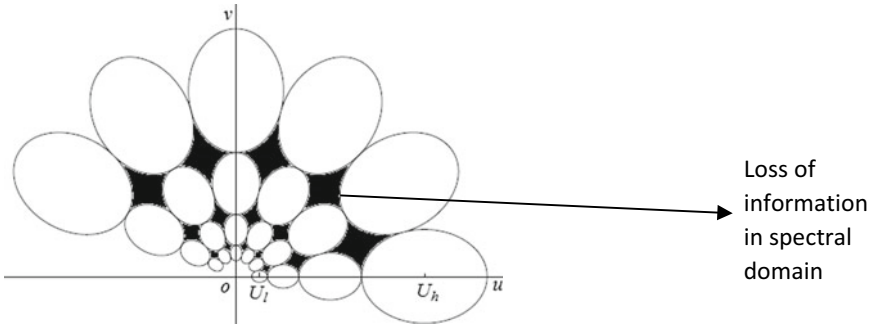


Fig. 5.7 Frequency tiling of half frequency plan by Gabor filters, the ovals are the covered spectrum while the black holes are the lost spectrum

overlapped window and do not involve image down sampling. These limitations can be easily overcome by using wavelet transform.

5.3.3 Texture Features Based on Wavelet Transform

5.3.3.1 Selection and Application of Wavelets

The key idea of wavelet transform is to analyze an image using multiresolution approach by using filters of different size, called wavelets. This is equivalent to look at the image from different distance. The whole decomposition process provides us with an array of DWT coefficients obtained from each subbands at each scale. These coefficients can then be used to represent the texture features of an image.

Given a 2D image $f(m, n)$, $0 \leq m \leq M-1$, $0 \leq n \leq N-1$, its DWT is given by (5.31):

$$W^j(k,l) = \frac{1}{2^j}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f(m,n)\psi^j_{k,l}(m,n) \tag{5.31}$$

where j is the scale and (k, l) is the spatial location of the wavelet and:

$$\psi^j_{k,l}(m,n) = \psi\left(\frac{m-2^j k}{2^j}\right)\psi\left(\frac{n-2^j l}{2^j}\right) \tag{5.32}$$

Different wavelets have been used to capture the texture features of an image. Commonly used wavelets include *Haar*, *Mexican hat*, *Morlet*, *Daubechies*, *biorthogonal*, *symlet*, *Coiflet*, and *Meyer* wavelet. Both Haar and symlet are special members of Daubechies wavelet family. Haar wavelet has been introduced earlier in Sect. 3.3.1. Figure 5.8 shows the 1D profiles six wavelets from some of the wavelet families, while Fig. 5.9 provides a 3D view for 4 of the wavelets.

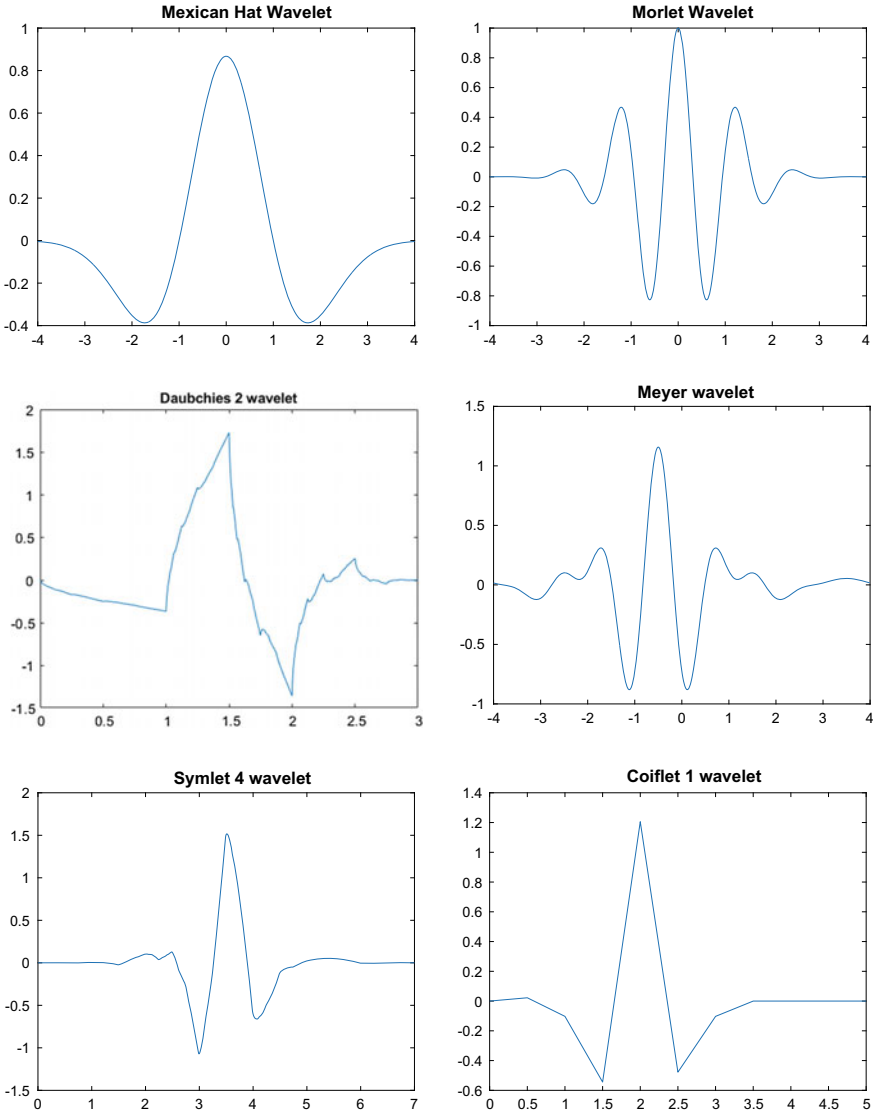


Fig. 5.8 Mexican hat, Morlet, Daubechies, Meyer, Symlet 4, and Coiflet wavelets

Figure 5.10 shows the spectra of the Lena image from some of the common wavelet transforms. It can be seen, that the spectra images are similar but with the subtle difference due to the different shapes of the wavelets. Some are more efficient due to capturing more low-frequency information while discarding more high-frequency information.

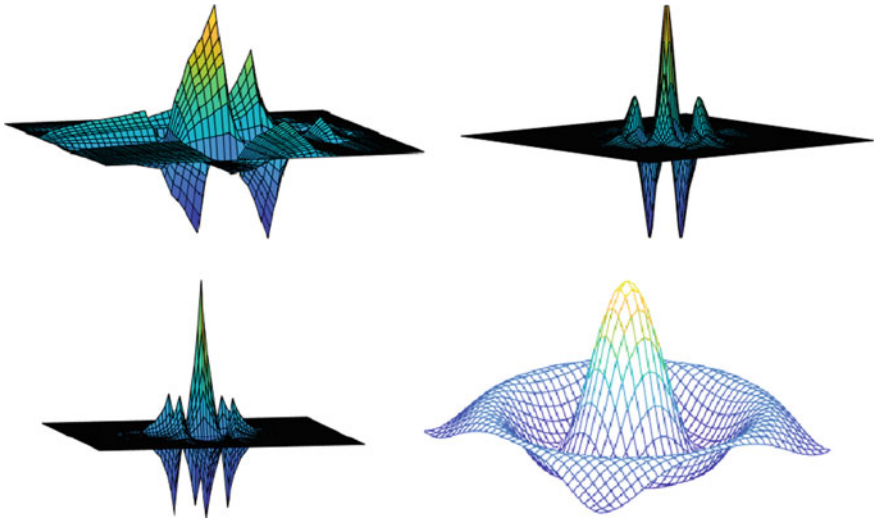


Fig. 5.9 Wavelets in 3D space. Top left: Daubechies 2; Top right: Symlet 4; Bottom left: Coiflet 1; Bottom right: Mexican hat

Discrete wavelets are differentiated by their *shapes*, *orders* (vanishing moments) and *compact support*. The combination of the three factors determines the result of a wavelet transform. The choice of a wavelet usually depends on the nature of the data and applications such as image analysis, image processing, or image compression. Often it requires a number of trials to determine the optimal combination of the three factors. Results of a wavelet transform also depend on whether the wavelet is overlapped or not. The *Maximal Overlap Discrete Wavelet Transform* (MODWT) has found popular application in image analysis.

Shape of a wavelet. The shape of a wavelet is characterized by its *symmetry* and *regularity*. A *symmetric wavelet* shows no preferred direction in time/space, while an asymmetric wavelet gives an unequal weighting to different directions. *Regularity* is related to how many continuous derivatives a function has. Therefore, regularity is a measure of smoothness of a wavelet. Generally, to detect an edge in the data, a wavelet needs to be sufficiently regular. The regularity is also related to the order, the higher the order, the smoother the wavelet.

For image analysis, however, research has shown that the shape of a wavelet does not have a significant influence on classification results [8].

Vanishing moments. An important property of a wavelet function is the number of *vanishing moments*, which characterize how a wavelet interacts with various signals. The names for many wavelets are derived from the number of vanishing moments. For example, db6 is the Daubechies wavelet with six vanishing moments and sym3 is the symlet with three vanishing moments. Generally, a wavelet with N vanishing moments is *orthogonal* to *polynomials* of degree $N - 1$. For example,

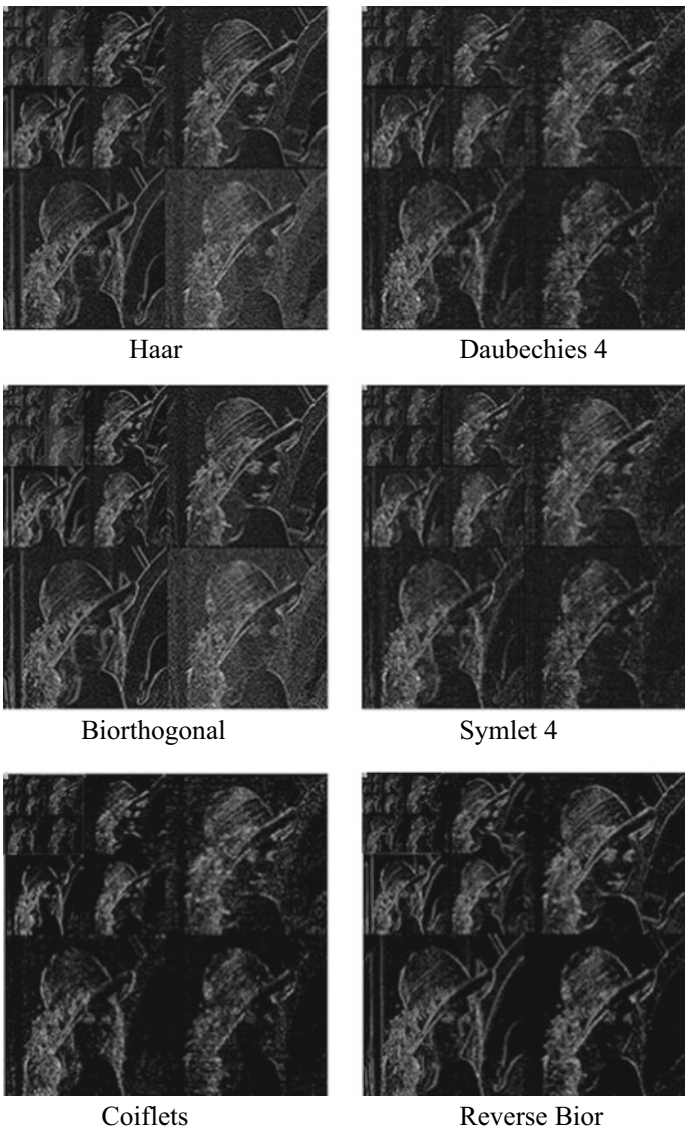


Fig. 5.10 Wavelet spectra for the Lena image

Daubechies 2 wavelet has two vanishing moments. When the Daubechies 2 wavelet is used to transform a data, both the *mean* and any *linear* trend are removed from the data. A higher number of vanishing moments implies that more moments (quadratic, cubic, etc.) will be removed from the data, which results in fewer significant wavelet coefficients. It also means higher order wavelets can capture or represent more frequency bands in the data. Higher order wavelets typically have

more oscillations and require wider support. A wavelet with N vanishing moments must have a support of at least length $2N - 1$.

The choice of wavelet order depends on the nature of the data, while usually, it requires a number of empirical tests on the data. Theoretically, the order of the wavelet should be greater than $2H + 1$, where H is the *Hurst exponent* of a signal or data [8, 9]. H can be determined using a similar technique of finding the *fractal dimension* as described in Sect. 5.2.4.

Compact support. This value measures the effective width of the wavelet function. A narrow wavelet function such as haar, db2, or sym2 can capture closely spaced or finer features and are fast to compute, but a narrow wavelet tends to have low-frequency resolution. Conversely, a wavelet with large compact support such as the Daubechies 24 is smoother and has finer frequency resolution which is usually more efficient for denoising.

Wavelets with large support tend to result in coefficients that do not distinguish individual features. Research has shown that wavelets with wider compact support provide increased sensitivity to group differences, which leads to higher classification accuracy [8]. However, wavelet with very large compact support can decrease the localization of prominent features and more coefficients are affected by boundary conditions. In practice, wavelets with an optimal compact support can be found using an empirical test.

MODWT. MODWT is highly redundant and invariant under circular shift. This causes MODWT preserving the smooth time-varying structure in regional time series that is otherwise lost during the application of DWT [8]. MODWT is adaptive to any signal length and emphasizes on variance analysis, which is desirable for feature extraction. Research shows that MODWT has superior performance than ordinary DWT [8]. This is supported by the fact that in literature, Gabor filters are usually preferred than ordinary DWTs.

5.3.3.2 Contrast of DWT and Other Spectral Transforms

If we give a comparison between wavelets, sinusoids (FT), STFT and Gabor filters, we have the following contrasts:

- **Orthogonality**. Both wavelets and sinusoids are orthogonal (1.2–1.6), while Gabor filters are not.
- **Window**. Wavelets, STFT, and Gabor filters are windowed transforms, while FT is not.
- **Window attenuation**. Both wavelets and Gabor filters attenuate towards the border of a window, while STFT does not.
- **Various window size**. Wavelets vary window size, while STFT and Gabor filters do not.
- **Directionality**. Both wavelets and Gabor filters are directional transforms, while FT and STFT are not.

- **Multiresolution**. Wavelets are multiresolution transforms, while Gabor filters, FT, and STFT are not.

Overall, wavelets have more advantage over FT, STFT, and Gabor filters. It can be observed from Fig. 5.10 that different from both FT and STFT, wavelets successfully capture the edge information of the image which is the most useful texture feature. The next is to focus on extracting texture features from the DWT spectrum.

5.3.3.3 Multiresolution Analysis

Space (time)-frequency methods attempt to find a specific frequency at a specific location, which is the main shortcoming of FT and STFT. However, it is not possible to find a specific frequency at a specific location simultaneously, just like we cannot get both a global view (zooming out) and a local view (zooming in) of a map at the same time (we lose global view when getting too close to the map for details and we lose details when we zoom out for a global view of the map). The solution is to use multiresolution or multi-view to create a tradeoff between spatial resolution and frequency resolution.

Multiresolution methods are designed to obtain a good spatial resolution but less accurate frequency resolution at high frequencies or a good frequency resolution but less accurate spatial resolution at low frequencies (Fig. 3.1 left). By using multiresolution or multi-view approach, both global view and local view of an image are obtained and a complete picture of an image is preserved, although not simultaneously. This approach is useful when a signal or an image contains both fine details in small areas and homogenous patches in larger areas. Usually, 2-D images follow this type of frequency patterns. The multiresolution analysis effectively overcomes the window size problem of STFT. Therefore, multiresolution approaches are more effective in image analysis and they overcome the frequency and location dilemma found in both FT (global view) and STFT (local view).

The coefficients at each subband of a wavelet transform are usually scarce, and they are not suitable for direct image representation. Statistics such as those proposed in the GLCM and Gabor filters can be computed from each subband of a wavelet transform. More robust features can be computed from each subband by using Gaussian mixture model. Since high-frequency components are more important for texture representation, features from lower scale subbands are usually given more weight.

Because digital wavelet transform (DWT) is done by two passes of 1D wavelet transform on rows and columns, respectively, wavelets can only capture edge information on horizontal, vertical, and diagonal directions. This gives Gabor filters an edge over wavelet on texture representation and retrieval because Gabor filters can be tuned to more directions than conventional wavelets. However, neither wavelets nor Gabor filters can effectively capture highly anisotropic elements like the curves from an image, and this is the rational behind the introduction of curvelet in the next section.

5.3.4 Texture Features Based on Curvelet Transform

5.3.4.1 Curvelet Transform

Both Gabor filters and wavelets let us do space–frequency analysis of images. Gabor filter is an improvement to STFT by using a Gaussian window and multi-orientations, however, it still uses a fixed window size for different frequencies. This is equivalent to looking at an image for details from a fixed distance, which is difficult. Wavelets use different window sizes for different frequencies, and this creates a multiresolution view of an image and is equivalent to looking for different levels of details of an image from different distance. Therefore, wavelet is a more accurate simulation to human vision system. However, wavelet can only capture texture features from three directions, which are not sufficient for image analysis.

Curvelet [10] has been introduced in literature to take the advantages of both Gabor filters and wavelet, while overcome the limitations of both. Specifically, a curvelet is orthogonal and multiresolution like a wavelet, while it can be tuned to multi-orientation like Gabor filters as well. In other words, a curvelet is a wavelet tuned to multi-orientations and can capture curved or nonlinear edges in an image instead of just linear edges. Therefore, it is a more powerful tool for image analysis.

Basically, curvelet transform extends the ridgelet transform to multiple scale analysis. Given an image $f(x, y)$, the continuous curvelet transform are defined as [11, 12]

$$\Re_f(a, b, \theta) = \iint \psi_{a,b,\theta}(x, y) f(x, y) dxdy. \tag{5.33}$$

where a ($a > 0$) is the scale, b is the translation, θ is the orientation, and ψ is the curvelet which is defined as follows:

$$\psi_{a,b,\theta}(x, y) = a^{-\frac{1}{2}} \psi\left(\frac{x\cos\theta + y\sin\theta - b}{a}\right) \tag{5.34}$$

where θ is the orientation and a is the scale of the curvelet. A curvelet is constant along the lines: $x\ \cos\theta + y\ \sin\theta = b$ and transverse to these ridges are wavelets. Compared with wavelet definition, the location parameters (b_1, b_2) of wavelet are replaced by the line and orientation parameters (b, θ) in a curvelet. In other words, the two transforms are related by [12]

$$\text{Wavelet: } \psi_{scale,\, point\ position}$$
$$\text{Curvelet: } \psi_{scale,\, line\ position}$$

In contrast, Gabor filters are Gaussian-shaped wavelets tuned to different orientations and scales.

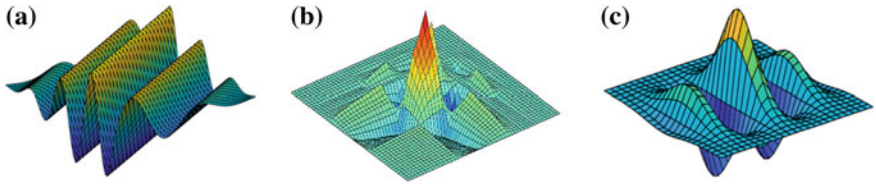


Fig. 5.11 Comparison of curvelet, wavelet and Gabor filter. **a** A curvelet; **b** a Daubechies wavelet and **c** a Gabor filter

$$g_{a,\theta,b_1,b_2}(x,y) = a^{-\frac{1}{2}} g\left(\frac{x\cos\theta + y\sin\theta - b_1}{a}, \frac{-x\sin\theta + y\cos\theta - b_2}{a}\right) \tag{5.35}$$

where the mother wavelet $g(x, y)$ is a Gaussian envelope modulated by a sinusoid wave. The shapes of the three types of wavelets are shown in Fig. 5.11 [11]:

It can be seen from the above figure, compared with both the wavelet and Gabor filter, a curvelet is the most sensitive to lines and edges in an image.

Similar to Gabor filters, a mother curvelet can be tuned to different orientations and different scales to create the curvelets (Fig. 5.12).

Curvelet takes the form of a basis element and obtains a high anisotropy. Therefore, it captures the edge information more effectively because it is sharper than a wavelet and a Gabor filter. Although a curvelet is linear in its edge direction, due to its elongated and orientated design, it aligns with curved edges much better than conventional wavelets do. The contrast between wavelet and curvelet on capturing curved edge information is shown in Fig. 5.13 [12, 13]. It can be observed that curvelets, at higher scales, capture the edge information more accurately and tightly than wavelets.

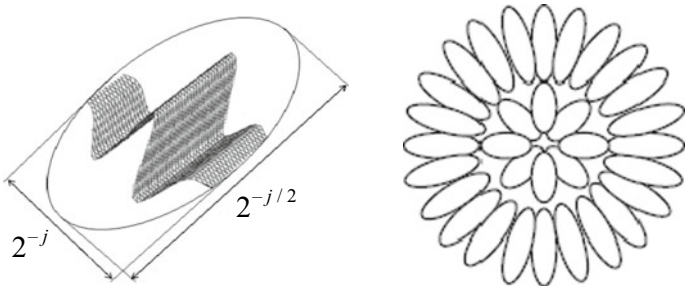


Fig. 5.12 A curvelet and curvelet tiling in spatial domain. Left: a single curvelet with width 2^{-j} and length $2^{-j/2}$; Right: curvelets tuned to 2 scales at different orientations

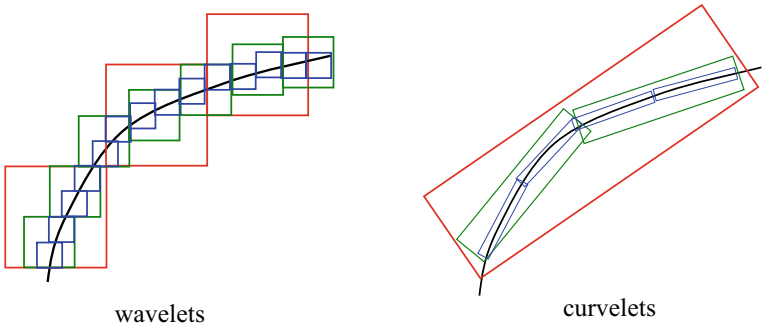


Fig. 5.13 Edge representation using wavelets and curvelets

5.3.4.2 Discrete Curvelet Transform

Given a digital image $f(m, n)$, $0 \leq m \leq M-1$, $0 \leq n \leq N-1$, the discrete curvelet transform is given as follows:

$$C_{j,\theta}(k,l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(m,n) \psi_{j,\theta,k,l}(m,n) \tag{5.36}$$

where $\psi_{j,\theta,k,l}$ (m, n) is a discrete curvelet; j, θ are the scale and orientation respectively; and k, l are the spatial location parameters. The frequency response of a curvelet is a wedge, and the curvelet tiling of frequency plane is shown in Fig. 5.14.

Curvelets exhibit an oscillating behavior in the direction perpendicular to their orientation in frequency domain. A few curvelets at different scales and their frequency responses are shown in Fig. 5.15 [3], and the scales shown on the figure are scales in frequency domain.

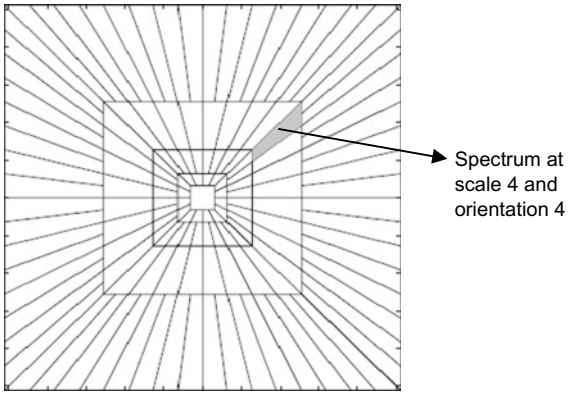


Fig. 5.14 Curvelet tiling of frequency plane with 5 level curvelets

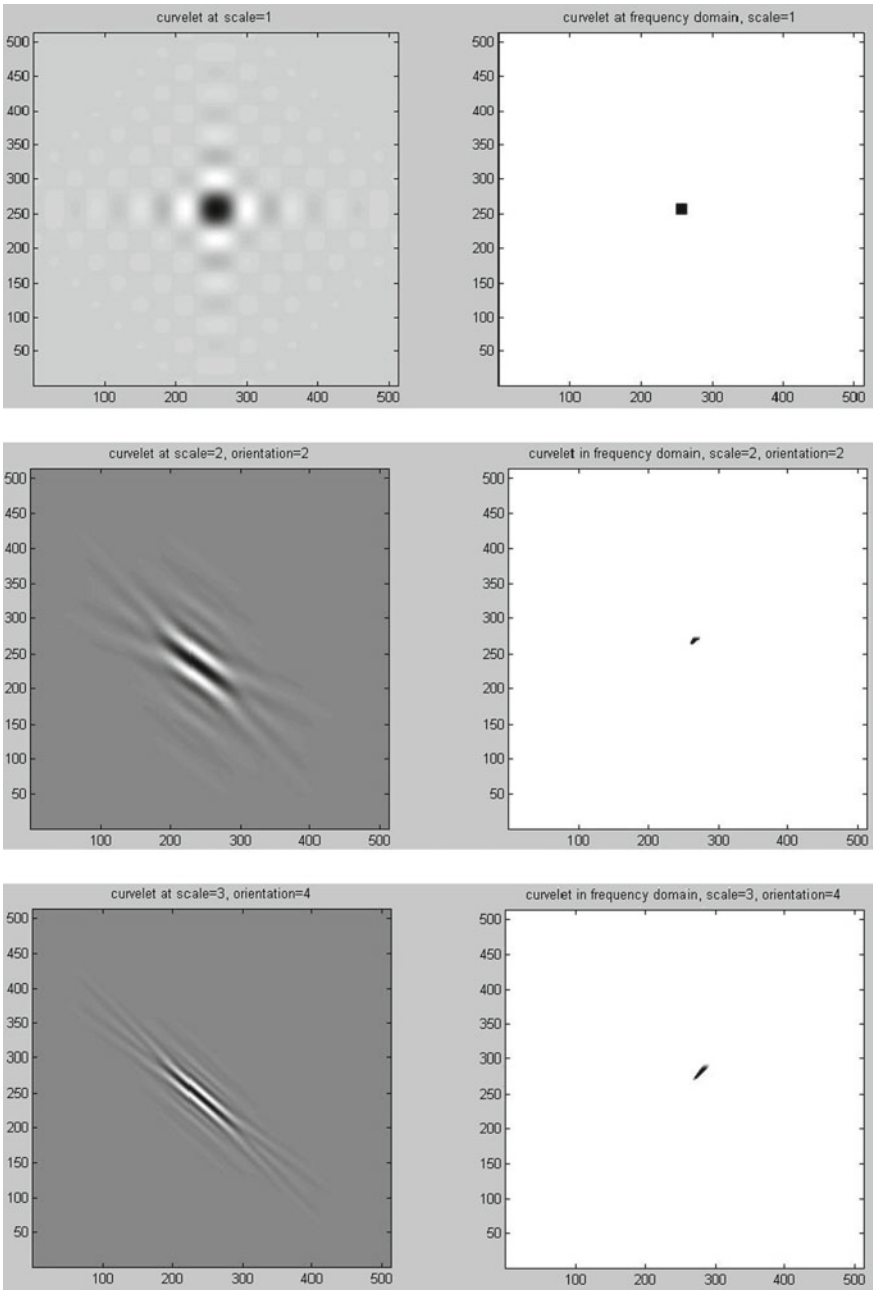


Fig. 5.15 Curvelets at different scales are shown in the spatial domain (left) and in the frequency domain (right) respectively

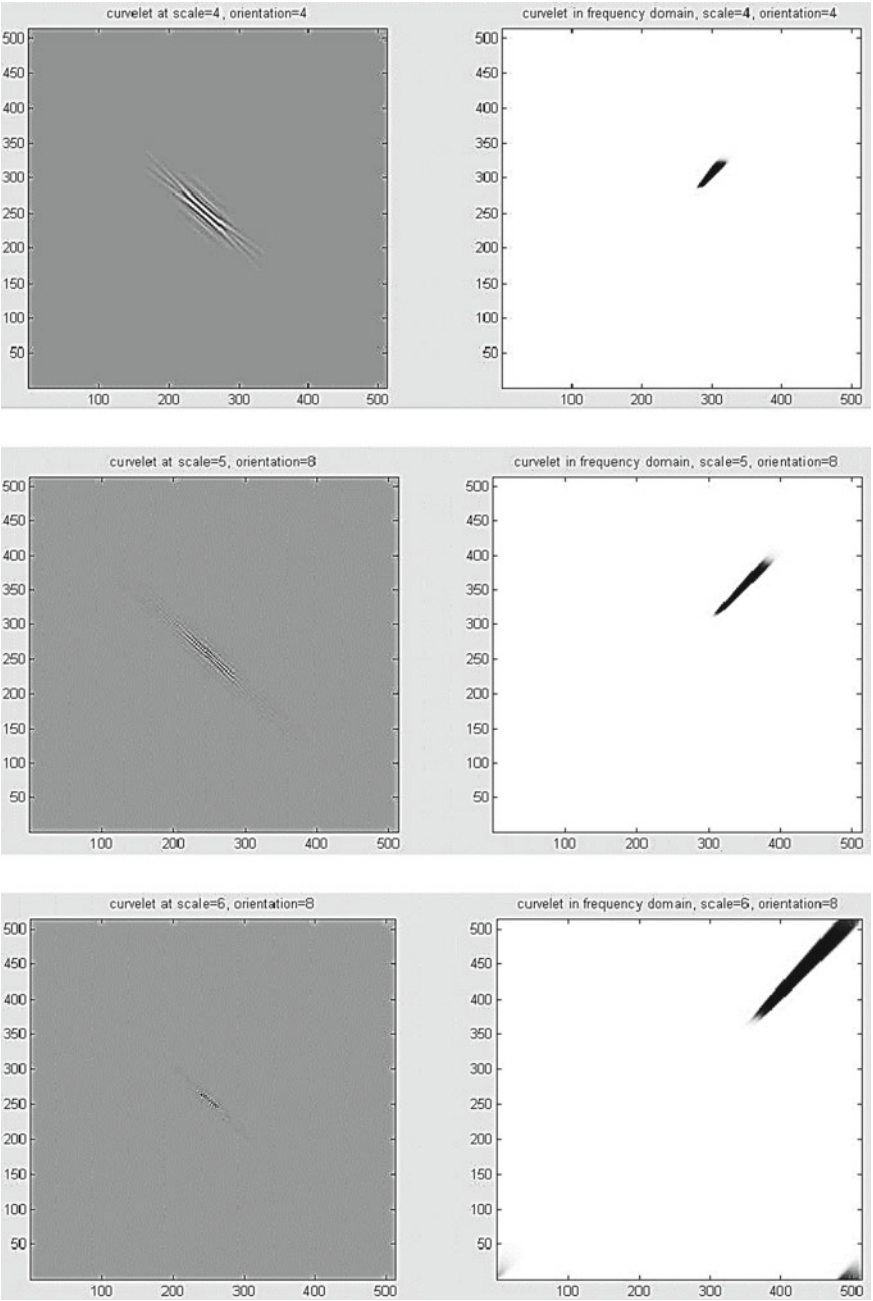


Fig. 5.15 (continued)

It can be observed that the curvelet is nondirectional at the coarsest scale. Whereas, at highest scales, the curvelet waveform becomes so fine that it looks like a needle shaped element. With increase in the resolution level, a curvelet becomes finer and smaller in the spatial domain and shows more sensitivity to curved edges which enables it to effectively capture curves in an image.

Although curvelet has an advantage of capturing nonlinear edges in an image, the computation of curvelet transform is more complex than both the Gabor filters and wavelet. Similar to Gabor filters, to achieve efficiency, curvelet transform is implemented in the frequency domain. That is, both the curvelet and the input image I are transformed into FT domain using FFT and the two FFTs are then multiplied in the FT domain. The product is then transformed back using the inverse fast Fourier transformed (FFT^{-1}) to obtain the curvelet coefficients. The process can be described as

$$\text{Curvelet transform}(I) = \text{FFT}^{-1}[\text{FFT}(\text{curvelet}) \times \text{FFT}(I)] \tag{5.37}$$

However, due to the FFT of the curvelet is a wedge, the product of the two FFTs needs to be wrapped back into a rectangle before it can be used for the FFT^{-1}. This wrapping process increases the computation cost.

5.3.4.3 Curvelet Spectra

Figure 5.16 shows some of the spectra of the curvelet transform on a flower image at different scales (in frequency domain) and orientations [11, 13]. The size of the spectra is adjusted for better viewing.

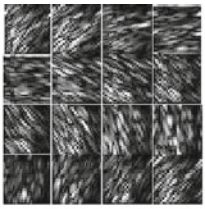
To contrast, the spectra of wavelet and Gabor filters are shown in Fig. 5.17.

It can be observed from the above two figures, the spectra of Gabor filters have more redundancy than both the wavelet spectra and curvelet spectra due to the use of overlapping windows during the transform. The spectra of Gabor filters also look more granular than those of both the wavelets and the curvelets. The spectra of wavelets are the most sparse and the most efficient in terms of reducing redundancy, therefore, wavelets are the choice for compression. Curvelet spectra are in between those of wavelet and Gabor filters. Curvelets are more sensitive to edges than Gabor filters and capture edges from more directions than wavelet. Furthermore, curvelets have little redundancy between different subbands. Curvelets also have a complete covering of the spectrum plane and this has overcome the spectra leakage of Gabor filters. However, due to there is a need to wrapping the wedge shape into a rectangle in order to do the invert FFT in (5.37), the computation is more complex than both Gabor filters and wavelets.

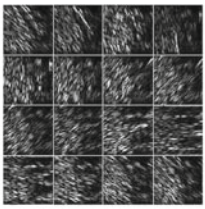
5.3.4.4 Curvelet Features

The feature extraction from curvelet transform is similar to Gabor filters. Once the curvelet transform is applied and the coefficients are obtained at each scale and orientation, the mean μ and standard deviation σ are computed for each of the subbands as follows:

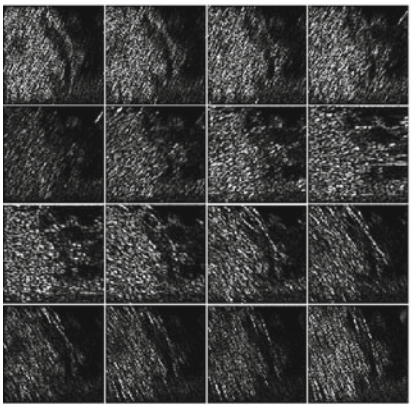
(a) Input image



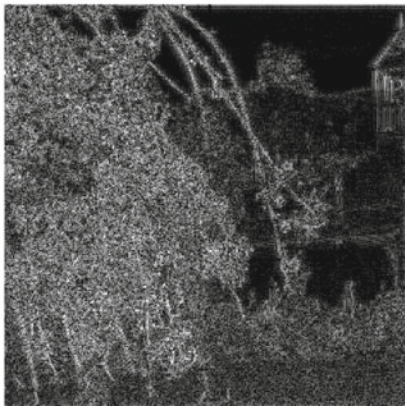
(b) subbands @ scale 3



(c) subbands @ scale 4



(d) subbands @ scale 5



(e) subband @ scale 6

Fig. 5.16 Curvelet subbands at different scales for a flower image (512 × 512). Each subband captures curvelet coefficients of the input image from one orientation

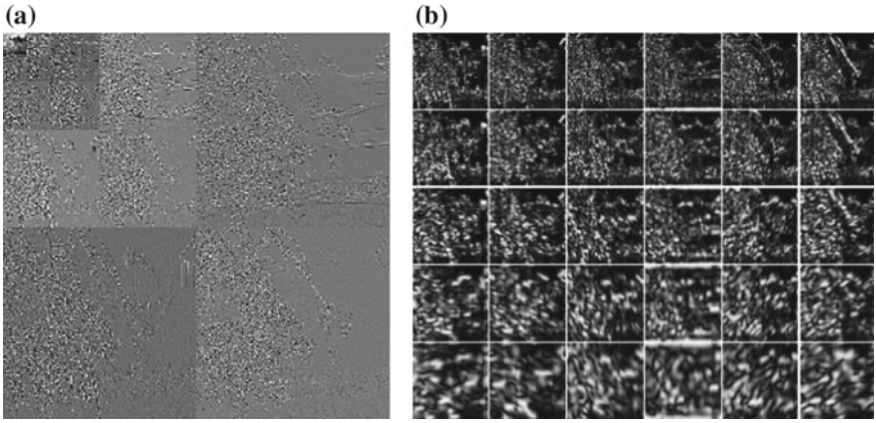


Fig. 5.17 Spectra of wavelets and Gabor filters for the flower image in Fig. 5.16a. **a** Wavelet spectra; **b** Gabor filters spectra

$$\mu_{s\theta} = \frac{E(s,\theta)}{m \times n}, \quad \sigma_{s\theta} = \frac{\sqrt{\sum_x \sum_y \left(|C_{s\theta}(x,y)| - \mu_{s\theta}\right)^2}}{m \times n} \tag{5.38}$$

where s is the scale, θ is the orientation, m and n are the dimensions of the corresponding subband, and $E(s,\theta) = \sum_x \sum_y |C_{s\theta}(x,y)|$ is the total spectral energy of the subband.

Therefore, for each curvelet, two texture features are obtained. If l curvelets are used for the transform, $2l$ texture features are obtained. A $2l$ dimension texture feature vector is used to represent each image in the database for image retrieval. To mitigate the dynamic range of the spectral energy, both the mean and standard deviation features are normalized using the maximum values of the corresponding features in the database.

Based on the curvelet subband division in Fig. 5.14, with 5 levels curvelet decomposition, 82 (= 1 + 16 + 32 + 32 + 1) subbands of curvelet coefficients are computed (only one subband is chosen from the last scale). However, due to the symmetry property, curvelet at angle θ produces the same coefficients as curvelet at angle $\theta + \pi$. Therefore, half of the subbands at scale 2–4 are discarded. As the result, 42 (= 1 + 8 + 16 + 16 + 1) subbands of curvelet coefficients are computed, and a $2 \times 42 = 84$ dimension feature vector is generated for each image. This dimension is higher than the feature vector from Gabor filters with the same scales due to Gabor filters usually use fewer orientations.

Rotation-invariant curvelet features can also be created by using the circular shift method in Gabor filters.

5.3.5 Discussions

Several texture features based on spectral transforms have been introduced in this section, including DCT, Gabor filters, wavelets, and curvelets. Although wavelets are orthogonal and more sensitive to edges, Gabor filters are more directional. This makes Gabor filters a better texture method in many applications. The performance of curvelet based texture features can be affected by the computation complexity due to the irregular frequency response of curvelets.

Generally, spectral texture methods are much more robust than spatial texture methods when they are applied on homogenous texture images, due to their model simplicity and computation efficiency. However, spectral transforms are usually done in a squared window, therefore, it is often difficult to apply spectral texture methods in irregular image regions, especially when the region size is not big enough. In these situations, the DCT-based texture method can be used because the 8×8 window is small enough to fit with most of image regions.