A Combinatorial K-View Based Algorithm for Texture Classification

Yihua Lan, Yong Zhang, Haozheng Ren
School of Computer Engineering, Huaihai Institute of Technology, Lianyungang, China
lanhua_2000@sina.com, zhyhglyg@126.com, renhaozheng666@163.com

***Abstract*—Image texture classification is widely used in many applications and received considerable attention during the past decades. Several efforts have been made for developing image texture classification algorithms, including the Gray Level Co-Occurrence Matrix (GLCM), Local Binary Patterns and several K-View based algorithms. These K-View based algorithms included are K-View-Template algorithm (K-View-T), K-View-Datagram algorithm (K-View-D), Fast Weighted K-View-Voting algorithm (K-View-V), K-View Using Rotation-Invariant Feature algorithm (K-View-R) and K-View Using Gray Level Co-Occurrence Matrix (K-View-G). There are some discussions about a part of these algorithms in the literatures; however, no complete experimental comparisons are made so far. In this paper, by analyzing those K-View based algorithms, an attempt to utilize the advantages of the K-View-R and K-View-V is made. The new approach which we call combinatorial K-View based method was presented. In addition, we review those K-View based algorithms and perform a comparative study based on the experiments using artificial texture images taken from the Brodatz, the evaluation method of performance between the proposed method and five different K-View algorithms are implemented by using classification accuracy, efficiency and stability.**

***Index Terms*—Texture classification, Voting, K-View algorithms**

I. Introduction

Image texture classification plays an essential role in the analysis of many types of images, from the aerial photos which are obtained from aircrafts or satellite platforms to microscopic images of cell cultures or tissue samples, from the medical images such as mammography or magnetic resonance imaging (MRI) to the outdoor natural scenes, to name a few. Therefore, it has been received considerable attention during the past decades. Despite its importance and ubiquity in image, there is not a formal approach or precise definition of texture. Image texture classification is still a hotspot and a challenge in the fields of machine vision and image analysis until now.

How to divide a texture image into regions of homogeneous textural pixels which form a set of texture classes? Extracting texture features is one of the most popular techniques to be used. Many methods have been developed to extract features, including statistics and structure methods, such as Gray Level Co-Occurrence Matrix (GLCM) [1-3] or local neighborhood statistics[4]; frequency or spectrum analysis methods, such as multiple components' frequency estimates[5], texture spectrum[6] and so on.

As stated in [7], to characterize texture, we must characterize the gray level primitive properties as well as the spatial relationship between them. Hung et al [8-9] presented that the texture features called characteristic views are formed with K views, and these views are capable of characterizing the gray level primitive properties as well as the relationship between them, and then they proposed K-View-Template algorithm (K-View-T) and K-View-Datagram algorithm (K-View-D) based on characteristic views [8-10] for image texture classification. In those two algorithms, characteristic views selection is the training process, in which the characteristic views can be selected by using k-means algorithm from the view set of each sample sub-image. In the classification process of K-View-T, the view being classified in the original image is compared with all the views in characteristic views of all sample sub-images. If there is one characteristic view which belongs to texture class M best matches with this view, then class all the pixels in the view being classified to class M (if the view is regarded as a neighborhood of one pixel, classify the pixel to class M). However, K-View-T takes a neighborhood of a classified pixel as a view corresponding to this pixel and decides whether this view is more similar to those characteristic views of the texture classes, K-View-D based on the histogram distribution of the characteristic views. In our experiments, K-View-T always classes pixels which lie in either interior regions or near the boundaries of error class, whereas the K-View-D always has good performance in classifying those pixels in interior regions, but the pixels near boundary areas still cannot be correctly classified. Therefore, Hong Liu et al proposed K-View Using Rotation-Invariant Feature algorithm (K-View-R) [11] and Fast Weighted K-View-Voting algorithm (K-View-V) [12] to improve the original two K-View algorithms by using different approaches and made some progress. GLCM is one of the popular techniques used in describing texture features. In this paper, we proposed a new combinatorial algorithm which incorporates those advantages of K-View-R and K-View-V into utilizing the K-View-V and K-View-R respective advantages in better measure.

This paper is organized as follows: in section 2, the related concepts of view are introduced, then the different K-View based algorithms are briefly reviewed in section 3; Experiments results and discussion and analysis are carried out in section 4; the conclusion gets in section 5 in the end.

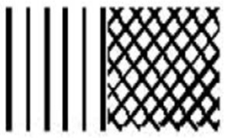
II. The Related Concepts of View

The K-View based algorithms include K-View-T, K-View-D, K-View-R, K-View-V and K-View-G. Those K-View based algorithms need to extract a set of characteristic views from sample sub-images of the texture class. Therefore, we will introduce the related concepts of view at first.

A. The Concept of View and View Set

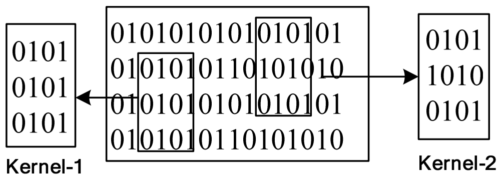
Since Texture is usually looked as a measurement of relationship between the pixels in a local region, Hung et al. presented a texture feature called view to represent a texture class, and used it as a basic unit for image texture classification[10].

Simply put, a view is an element image block in an image. A view has a size of m by n and is a neighborhood of a pixel. A binary image example is shown in Figure 1 (a) that contains two different pattern classes, and shown in Figure1 (b) which are gray values of its pixels by corresponding location. Select randomly in the area of the texture class as a sample of sub-image for each texture class from the original binary image. An example of selected sample sub-images, kernel-1 and kernel-2, are shown in Figure 1(c) (we called these sample image as Kernel). And then we use a sliding window of size 1×2 and scan pixel by pixel on the kernel-1 and kernel-2 respectively. From kernel-1, we obtain nine image blocks (i.e. views) of size 1×2 of (1 0) and (0 1), there are 6(0 1) and 3(1 0) (see Figure 1 (d)), these nine views constitute together a set of primitive views for this texture class, we call this set as primitive view set. In the same way, there are nine image blocks of size 1×2 of (1 0) and (0 1) in kernel-2, and in this primitive view set we also may find two types of views (i.e., (1 0) and (0 1)) in it. The size of the sliding window here is called view size, of course, we may use sliding window by other size such as 2×2 or 2×3 on the two sample images, and then different primitive view sets will be produced in accordance with different view sizes.

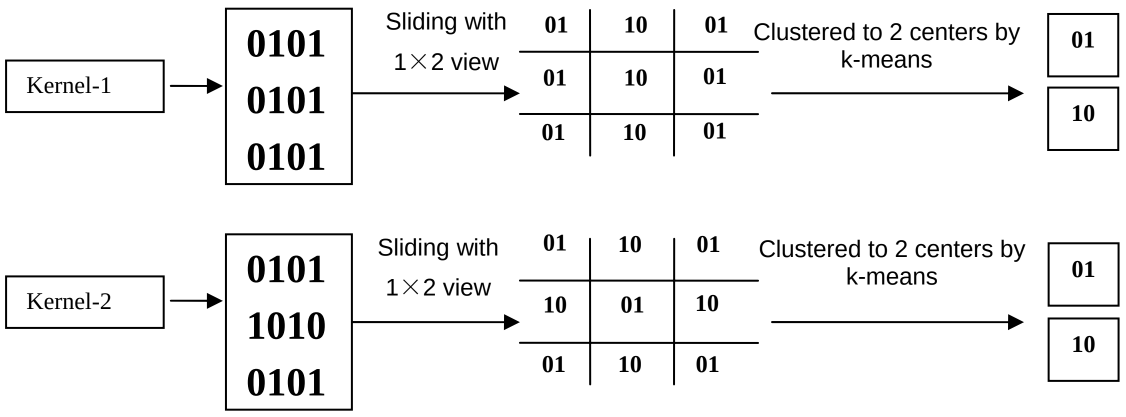


```
0101010101010101
0101010110101010
0101010101010101
0101010110101010
```

(a) An image texture.   (b) Corresponding pixel values.



(c) Kernels (samples) of two different classes taken from the image texture.



(d) Characteristic views of two different classes extracted by 1×2 view

Figure 1. Characteristic views extracted from the kernels of two different classes.

B. The Concept and Extract Method of Characteristic Views

Different views contain texture information from different spatial locations in the kernel. Texture appearances may be recurrent in different locations, and the views will frequently be revealed the repetitions in the same texture class of the image. Therefore, We can certainly imagine that there are a lot of same or similar views in the primitive view set of each texture class of the kernel(here we suppose the size of kernel is big enough).For an example, this circumstance appears in the kernel which is shown in Figure 1(d), in kernel-1, and there are nine views of size 1×2 of (1 0) and (0 1) to form the primitive view set, and these views can be classified into two types(i.e. (1 0) and (0 1)), the kernel-2 may be deduced by analogy. Therefore, we can use views just like (1 0) and (0 1) which are included in these same or similar views to represent the all primitive views of the kernel-1, of course we also can use (1 0) and (0 1) to represent the all primitive views of the kernel-2, in other words, a set of characteristic views in a kernel will have a few representative views which will be a small subset of all primitive views of the kernel, then we used it for image texture classification, it may save a lot of calculation work and computation time but still achieve a reasonable classification accuracy. These views which we selected to describe the characteristic of a texture image and used for image texture classification are called characteristic views. All algorithms based on K-View which will be introduced later used the characteristic views as texture feature.

We can derive characteristic views set which includes K views for each kernel using the K-means algorithm, K may vary for each kernel, a larger K usually means it can be extracted a more representative set of characteristic views. Just like Figure 1 (d), we may obtain 2 characteristic views (i.e. (1 0) and (0 1), K=2) by k-means from Kernel-1. There is a same set of characteristic views (i.e. (1 0) and (0 1)) in kernel-1 and kernel-2, therefore, characteristic views of each texture class can not represent their class good enough to discriminate these two types of texture class, and we don't think this view size an optimum size in this example.

III. The K-view Algorithms

In this section, five different K-View based algorithms will be briefly reviewed.

A. The K-View-Template Algorithm (K-View-T)

The K-View-T algorithm is briefly described in the following parts.

Step 1: Select a sample sub-image for each texture class randomly in the area of the corresponding texture class from the original image. In other words, N sample sub-images will be selected for N texture classes.

Step 2: Extract a primitive view set from each sample sub-image and form a primitive view set S.

Step 3: Determine the value of K for each view set, and derive a K-View of characteristic view set denoted by C_{vs}, from each kernel use the K-means algorithm. The parameter K may vary from each texture class.

Step 4: In the classification process, each view, says V, of an image being classified will be compared to each characteristic view in all the characteristic view sets of all texture classes.

Step 5: If the best matched characteristic view belongs to characteristic view set M, classify all pixels in the view, V, from the original image to class M. (If the view is regarded as a neighborhood of one pixel, classify that pixel only into class M), M is from 1 to N.

Step 6: Repeat steps 4 and 5 for each pixel in the original image being classified.

Steps 1-3 are used for training which establish a prototype for each texture class. Steps 4-6 are used for classification which will class each pixel in original image into a certain type.

B. The K-View-Datagram Algorithm (K-View-D)

The K-View-D algorithm is described in the following steps, similar to the K-View-T algorithm, steps 1-4 are for training and steps 5-7 are for classification:

Step 1-3: Same as Step 1-3 in the K-View-T algorithm.

Step 4: Based on the characteristic view set C_{vs}, we calculated a datagram (D) for each of the N sample sub-images. According to Equation 1, normalize each datagram D (D_N). We call these N normalized datagrams coming from sample sub-images the sample datagrams (D_S).

$$D = (d_1, d_2, d_3, \cdots d_k) \tag{1}$$

$$T = \sum_{i=1}^{k} d_i \tag{2}$$

$$D_N = (d_1/T, d_2/T, d_3/T \cdots d_k/T) \tag{3}$$

Step 5: Scan the entire image using a window of M×M pixels, and obtain the normalized datagram for each window. Calculate the differences between the normalized datagram and each of the N sample datagrams (D_s), and classify the central pixel of the windows to the class, such that the difference between the sample datagram of the class and the normalized datagram is the minimum. The difference (Dif) between a normalized datagram (D_N) and a sample datagram (D_s) can be obtained by the following equations.

$$D_N = (d_{N1}, d_{N2}, d_{N3}, \cdots d_{Nk}) \tag{4}$$

$$D_S = (d_{S1}, d_{S2}, d_{S3}, \cdots d_{Sk}) \tag{5}$$

$$Dif = \sum_{i=1}^{k} |d_{Si} - d_{Ni}| \tag{6}$$

Step 6: Same as Step 5 in the K-View-T algorithm.

Step 7: Repeat steps 5 and 6 for each pixel in the original image being classified.

From the steps mentioned above in K-View-D, we may find that unlike K-View-T algorithm, the decision is made by a single view whose center is located at the current pixel being classified, in K-View-D algorithm, the decision is made by the distribution of all the views contained in a large block in which the current pixel is the center of the block.

C. The K-View Using Rotation-Invariant Feature Algorithm (K-View-R)

The K-View-R algorithm consists of two processes which are the training process and the classification process similar to K-View-T and K-View-D algorithms. The scheme of two processes is shown in Figure 2[11].

There are two enhancements to improve the original K-View-T algorithm and K-View-D algorithms in K-View-R algorithm.

1) The extract method of Characteristic Views Set.

In the original K-View-T and the K-View-D algorithms, the characteristic views can be learned by unsupervised clustering algorithms such as the k-means, however, being different from the method above, the characteristic views are randomly and directly selected from the view set of kernel in each texture class in the K-View-R algorithm.

2) The correlation-matching method between views.

The K-View-T compares the similarity between a view and all characteristic views directly, while the K-View-D computes the histogram distribution of the characteristic views for each texture class and the comparison is transformed into between the histogram of the view and the histograms of each texture class. In the K-View-R algorithm, in order to achieve high classification accuracy, six significant rotation-invariant features named Mean, Standard Deviation, Entropy, Skew, Kurtosis and Histogram are extract, with all the characteristic views used being transformed into rotation-invariant features in this algorithm. Those features provide a good discrimination of textures.

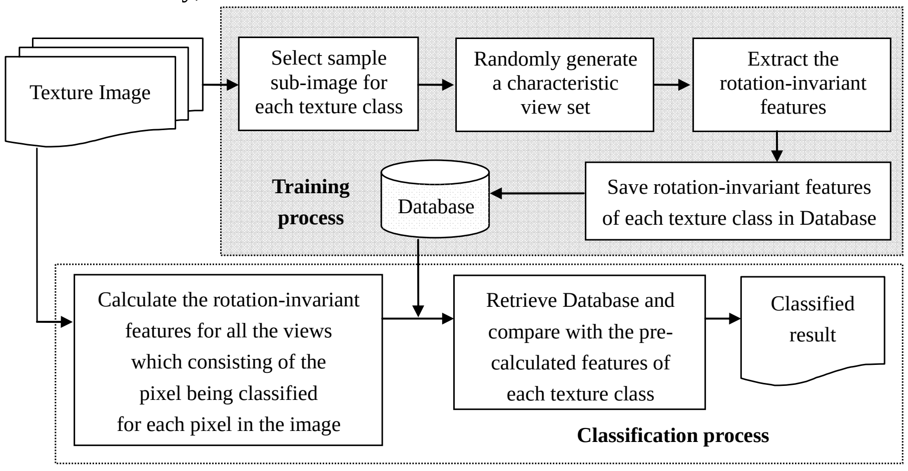

Texture Image
Select sample sub-image for each texture class
Randomly generate a characteristic view set
Extract the rotation-invariant features
Training process
Database
Save rotation-invariant features of each texture class in Database
Calculate the rotation-invariant features for all the views which consisting of the pixel being classified for each pixel in the image
Retrieve Database and compare with the pre-calculated features of each texture class
Classified result
Classification process

Figure 2. Training process and classification process of K-View-R.

D. The Fast Weighted K-View-Voting Algorithm (K-View-V)

The K-View-V algorithm is an innovative efficient approach to improve K-View-T algorithm. Use a novel voting method for texture classification and an accelerating method based on the efficient Summed Square Image (SSI) [13] scheme and Fast Fourier Transform (FFT) for fast processing. The basic ideas of these two different algorithms (i.e. K-View-V and K-View-T) are alike. Therefore, we will be only discussing the crucial differences between them detailed.

Before the introduction of the group decision method made by weighted voting, we explain the concept of neighborhood supporters firstly.

1) Neighborhood supporters

Take an example shown in Figure 3, one pixel of the image may have many views which contain the current pixel are all correlated to it. Therefore, these views which are called neighborhood supporters (i.e. correlative views) should be given an opportunity to decide which texture class that this pixel belongs to. If the view size is m by n, for each pixel there will be m×n different correlative views consisting of the corresponding pixel.

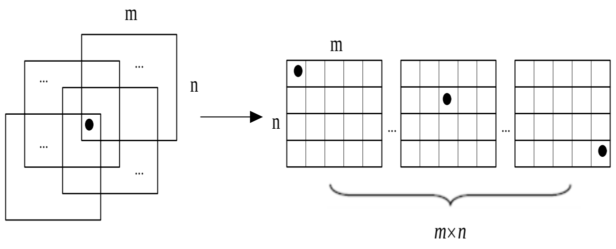


Figure 3. The correlative views of a pixel.

2) Group decision strategy

In K-View-V, let d be the distance between a view and a texture class which is equal to the minimum Euclidean distance between the view and all views in characteristic views corresponding to this texture class. Therefore, the similarity s between a view and a texture class can be defined as the formula:

$$s = \frac{1}{d^2} \tag{7}$$

To some extent, the view with more similarity to a certain texture class should be given the more powerful vote on this class. However, when a view spans the boundary of different texture class images, it contains mixed pixels from different classes, so the appearance of the view looks like several texture classes. This brings that the loyalty of the view to the multi-classes has dispersed. K-View-V quantifies the loyalty for a view to the jth class by uj defined as the formula:

$$u_j = \frac{1/d_j^2}{\sum_{k=1}^{K} 1/d_k^2} \tag{8}$$

Here K is the number of the texture classes. In summary, if the number of views in the neighborhood supporters and texture class are respectively V and K, a voting weight matrix W with the size V×K can be defined as

$$W_{i,j} = u_{i,j} \tag{9}$$

Where $u_{i,j}$ is the loyalty for the ith view to the jth class, and $d_{i,j}$ is the distance between the ith view and the jth class, both of them can be explained above. Then let each view in the current pixel's neighborhood supporters take a vote on each texture class (i.e. the vote is the corresponding weight factor) in matrix W. Therefore, the best-matched texture class is supported by the most weighted votes by taking the maximum among the weights, that is the Pth class with the P which is calculated as in Equation (16):

$$P = \arg(\max(\sum_{i=1}^{V} W_{i,1}, \sum_{i=1}^{V} W_{i,2}, \ldots, \sum_{i=1}^{V} W_{i,K})) \tag{10}$$

3) Summed Square Image (SSI) and FFT method

In K-View-T, it needs to compare an incoming/being classified view with each characteristic view of each texture class directly. Take two views (i.e. V_1 and V_2) with the same view size m×m for example, the Euclidian distance between two views is d which can be obtained by the following equation:

$$d = \sqrt{\| V_1 - V_2 \|^2} \tag{11}$$

To calculate the Euclidian distance for views matching is the most time-consuming process. K-View-V uses an accelerating method based on the efficient Summed Square Image (SSI) scheme as well as Fast Fourier Transform (FFT) to enable overall faster processing. The method is briefly described as following. Through analysis we may find that the d can be calculated as:

$$d = \sqrt{\| V_1 - V_2 \|^2} = \sqrt{V_1^2 + V_2^2 - V_1 \times V_2} \tag{12}$$

Let V_1 denote a view of the original image and V_2 one of characteristic views belonging to a texture class. Just because we should calculate the Euclidian distance between all views in the original image and each characteristic view of each texture class, we may use SSI to calculate the part of V_1 and FFT to calculate the part of $V_1 \times V_2$ more quickly. The second square must be calculated directly.

1) SSI Calculations

Liu et al. proposed an image de-noising method [13] using the SSI method which is based on the integral image concept [14]. Rectangle features can be computed very rapidly using an intermediate representation for an image which is called an integral image. Liu et al. extended the integral image to their SSI [13] method. In SSI, the pixel value at location (x_o, y_o) contains the squared value of each pixel in the original image above and to the left of x, y, inclusively:

$$SSI(x_o, y_o) = \sum_{x \le x_O, y \le y_O} I(x,y)^2, x, y \in (1, m) \tag{13}$$

where *I(x, y)* is the pixel value in the original image.

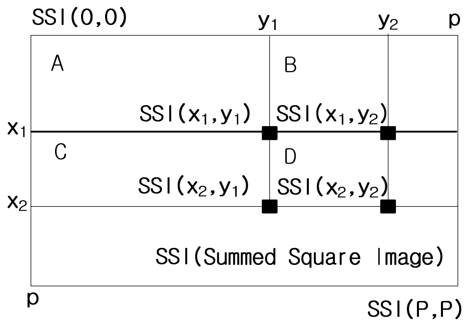


Figure 4. An SSI (Summed Square Image) illustration.

For example, if we need to calculate the sum of squares in region D (this region is denoted the corresponding region in the original image) as it is shown in Figure 4, it can be obtained as follows:

$$S_D = S_{A \cup B \cup C \cup D} + S_A - S_{A \cup C} - S_{A \cup B} \tag{14}$$

From the SSI shown in Figure 4 ,we can see that:

$$\begin{aligned} S_{A \cup B \cup C \cup D} &= SSI(x_2, y_2); S_A = SSI(x_1, y_1); \\ S_{A \cup C} &= SSI(x_2, y_1); S_{A \cup B} = SSI(x_1, y_2) \end{aligned} \tag{15}$$

Therefore, we may obtain an equation as follows:

$$S_D = SSI(x_2, y_2) + SSI(x_1, y_1) - SSI(x_2, y_1) - SSI(x_1, y_2) \tag{16}$$

Therefore, each pixel in the SSI can be calculated in only one pass over the original image. The computational complexity for computing SSI is O (P^2) (P^2 is the original image size). The SSI can be obtained in linear time proportional to the image size.

2) The Fast Fourier Transform (FFT) method

The last term of equation 12 $V_1 \times V_2$ can be calculated quickly with multiplication using the FFT [15]. Assuming that the view size is all m×m (m is odd in general), we flip the characteristic view, V_2, from right to left and up to down, as shown in Figure 5. In this manner we may derive a new view (flipped) which we denote it V_3, then compute two-dimensional convolution of views V_1 and V_3. We can derive a (2×m-1)×(2×m-1) matrix denoted by MAT. According to the Convolution Theorem, the following equation can be easily obtained:

$$MAT(m,m) = V_1 \times V_2 / 2 \tag{17}$$

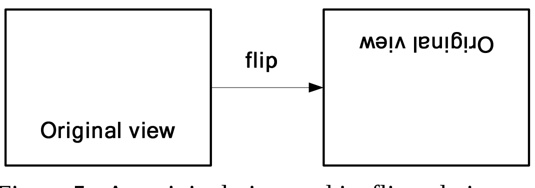


Figure 5. An original view and its flipped view.

Since we need to compare an incoming/being classified view with each characteristic view of each texture class, we can calculate the two-dimensional convolution of V_3 and the padded image which we call PadI. The convolution is formulated below:

$$V_3(x,y) * PadI(x,y) \Leftrightarrow IFFT(FFT(V_3(x,y))FFT(PadI(x,y))) \quad (18)$$

where "*" is convolution operation symbol, IFFT(FFT(V_3(x, y))FFT(PadI(x, y))) means to compute the two-dimensional FFT of V_3 and PadI, then multiply them together, and take the inverse FFT.

E. The K-View Using Gray Level Co-Occurrence Matrix (K-View-G)

The GLCM model is a second-order statistics which calculates how often different combinations of pixel brightness values (grey levels) occur in an image. GLCM matrices are used to record the spatial relationship between pixels and their properties [10], and then the statistics of GLCM will be derived. These statistics include uniformity of energy, entropy, maximum probability, contrast, inverse difference moment, correlation, probability of a run of length, homogeneity and cluster tendency [10]. These features can be used in the classification of texture images. The GLCM is represented in a 2-Dimensional array (i.e. table). We take an image with a size of M×N and let $P(i,j)$ be an element of the GLCM matrix. Hence, $P(i,j)$ represents the frequency of occurrences of pair pixels which are separated by distance δ and angle α . Matrices with different angles and distances can be described as:{e(m,n) = i, e(m+ δ α ,n+ δ α) = j, m $\in (1, M-1)$, n$\in (1, N-1)$, $\alpha \in \{0^o, 45^o, 90^o, 135^o\}$, $\delta \in$ R},where δ is a real number and α is one of four directions (0^o, 45^o, 90^o, 135^o). In this paper, we used the GLCM method in the K-View-G algorithm for our implementation in the experiments, which is briefly described in the following steps:

Step 1-3: Same as Step 1-3 in the K-View-T algorithm. Step 4: In the classification process, each view (a small image block), says V, of an image is classified. Firstly compute the GLCM feature vector (i.e. the vector is compose of Contrast, Correlation, Energy, Homogeneity and Mean) values of K center view of each sub-image. Then compute the GLCM feature vector values of the V. If the best matched characteristic view belongs to characteristic view set M, classify all pixels in the view, V, from the original image to class M. (If the view is regarded as a neighborhood of one pixel, classify that pixel only to class M). We use Euclidean distance as the comparison method.

Step 5: Repeat steps 4 for each pixel in the original image being classified.

F. The Proposed Combinatorial Algorithm

In order to utilize the K-View-V and K-View-R respective advantages in the best measure, we propose a scheme to incorporate those two K-View based method. The scheme of K-View-VR is shown in Fig. 7.

The steps of this new combinatorial method can be described as follows:

Step 1: Selecting sample sub-image randomly for each texture class from the originally classified image;

Step 2: Selecting characteristic views. In this step, two types of characteristic view sets are produced for K-View-R and K-View-V method respectively. This is to say, the first type of characteristic view set will be generated by K-means algorithm or fuzzy c-means algorithm for K-View-V approach. On the other hand, another type of characteristic view set is extracted randomly from each sample sub-image for K-View-R approach.

Step 3: Calculate the loyalty1 for each pixel being classified by using K-View-V approach. Calculate the loyalty2 for each pixel being classified by using K-View-R approach.

Step 4: Voting. In this step, a simple fusion method was used to yield a final loyalty for this pixel being classified. The fusion method is shown in the following equation.

$$\text{Loyalty} = \max(\text{loyalty1}, \text{loyalty2}) \quad (19)$$

Step 5: Classify the pixel to a class when getting the max loyalty.

Step 6: Repeat steps 3, 4 and 5 for each pixel in the original image being classified.

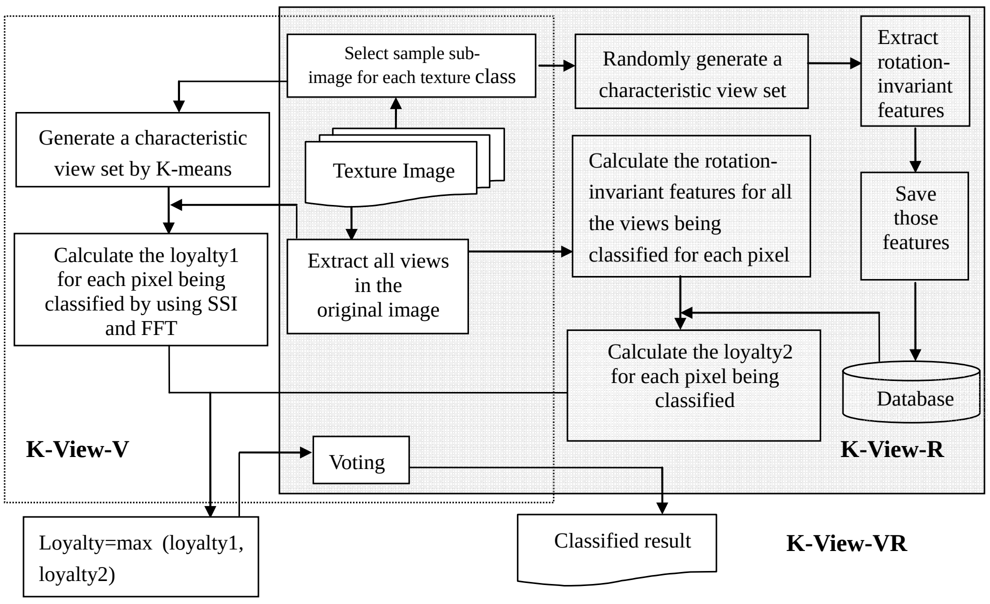
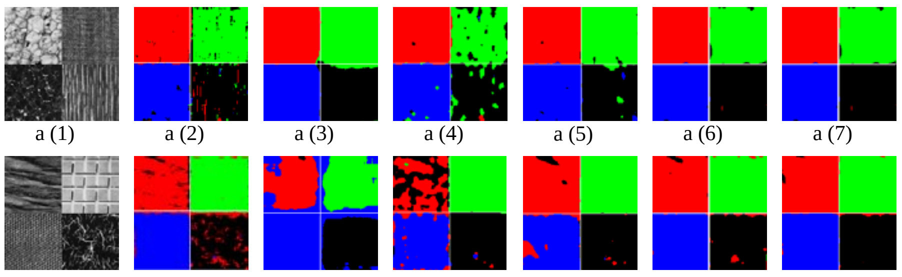

Select sample sub-image for each texture class
Randomly generate a characteristic view set
Extract rotation-invariant features
Generate a characteristic view set by K-means
Texture Image
Calculate the rotation-invariant features for all the views being classified for each pixel
Save those features
Calculate the loyalty1 for each pixel being classified by using SSI and FFT
Extract all views in the original image
Calculate the loyalty2 for each pixel being classified
Database
K-View-V
Voting
K-View-R
Loyalty=max (loyalty1, loyalty2)
Classified result
K-View-VR

Figure 6. The entire training process and classification process of K-View-VR.

IV. Experimental Results and Discussions

In this section, experiments for a number of image textures were performed with the characteristic views texture features and different five K-View based algorithms as stated in the previous sections. Those algorithms were tested on a set of variety representative texture images include coarse texture, irregular texture and regular texture which were obtained randomly from Brodatz Gallery [16]. The size of these artificial images is 150×150 pixels. In our experiments, all K-View based algorithms were implemented with same number of characteristic views (i.e. K) and view size, we choose K=30, which means that there are 30 characteristic views for each texture class. The view size was set to 7×7. The features used in GLCM include contrast, correlation, energy, homogeneity and mean. Other parameters were set as follows: distance $\delta = 1$, $\alpha = \{0^o, 45^o, 90^o, 135^o\}$, gray-level = 16, we ran experiments on GLCM model with 0^o, 45^o, 90^o and 135^o four directions respectively, and selected the optimum one as the final results. Classified results of all texture images with different K-View based algorithms are shown in Figure 6.

By comparing experimental results shown in Figure 6 and 7 ,we can see that the proposed combinatorial K-View-VR algorithm performs better than those of K-View-T, K-View-D, K-View-R, K-View-V and K-View-G, and it can achieve overall better classification accuracy. The average classified right ratio which is obtained through the proposed method obtained is o.9810, while the ratios of K-View-T, K-View-D, K-View-G, K-View-R, K-View-V are 0.9217, 0.8932, 0.8608, 0.9377 and 0.9712 respectively. Classified results of K-View-T, K-View-R and K-View-G are not very good in interior regions as shown in Figure 6 a(2), a(4), a(5),d(2), d(4), d(5) and so on, and many pixels were misclassified in the interior regions. On the contrary, the K-View-D, K-View-V and the K-View-VR gave higher classification accuracy in interior regions; moreover, the pixels near boundary areas classified by K-View-VR are more correctly classified. But K-View-D makes the classified boundaries distorted, such as c (3), h (3), with many pixels misclassified which are all located near the boundaries. From the Figure 7, we also can find that K-View-V and K-View-VR are more robust than other four algorithms. The reason is that K-View-V uses group decisions made by weighted voting, and it makes the decision more reasonable and has a better performance. In addition, the proposed combinatorial K-View-VR can take the advantages of K-View-R and K-View-V, therefore, the proposed method achieves the better performance compared with K-View-R and K-View-V methods.



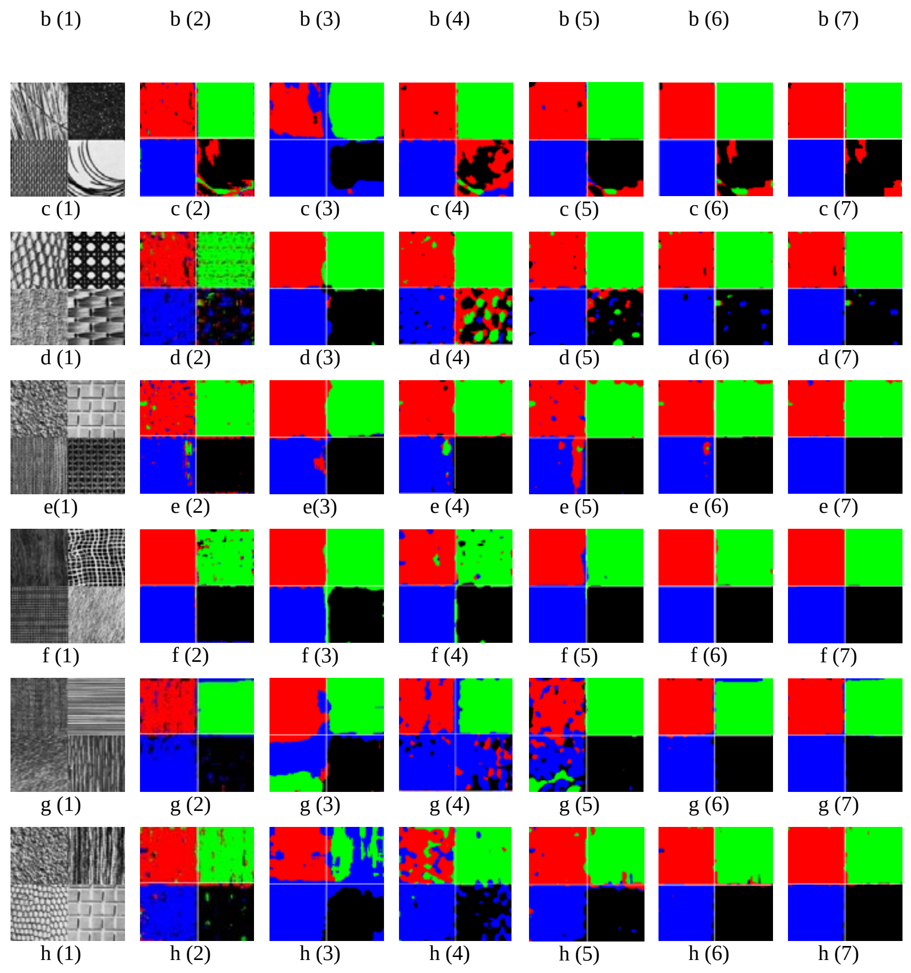

b (1) b (2) b (3) b (4) b (5) b (6) b (7)
c (1) c (2) c (3) c (4) c (5) c (6) c (7)
d (1) d (2) d (3) d (4) d (5) d (6) d (7)
e(1) e (2) e(3) e (4) e (5) e (6) e (7)
f (1) f (2) f (3) f (4) f (5) f (6) f (7)
g (1) g (2) g (3) g (4) g (5) g (6) g (7)
h (1) h (2) h (3) h (4) h (5) h (6) h (7)

Figure 6.The classified results of the texture images with different K-View algorithms. a(1) An original image, classified result with K-View-T, K-View-D, K-View-G, K-View-R, K-View-V and the proposed K-View-VR are shown in a(2),a(3),a(4),a(5), a(7)and a(6) respectively, The classified results of original image Figure 6 b(1)- Figure 6 h(1) may be shown in Figure 6 deduced by analogy. The white lines are drawn on the top of classified results to show the actual boundary.

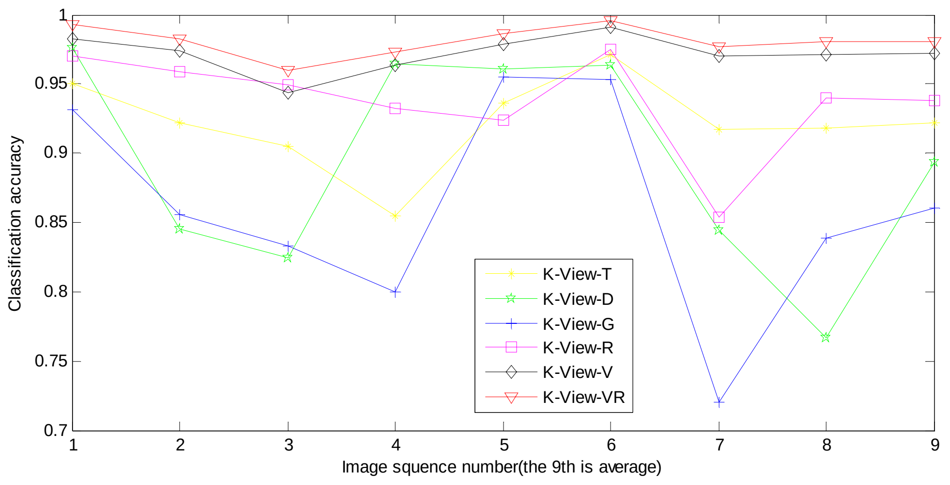


Figure 7. Classified right ratios of each texture image classification with different K-View algorithms

TABLE I.
COMPUTATION TIME(S) OF EACH TEXTURE IMAGE CLASSIFICATION WITH DIFFERENT K-VIEW ALGORITHMS

Sequence number	K-View-T	K-View-D	K-View-G	K-View-V	K-View -R	K-View -VR
1	37.67	1258.72	177.00	17.97	15.78	33.78
2	36.78	1224.26	151.43	17.93	11.84	28.98
3	38.34	1229.12	160.10	18.30	17.69	35.75
4	35.84	1176.76	150.43	17.65	12.28	29.58
5	36.41	1210.24	152.55	17.82	11.02	28.24
6	36.00	1178.20	160.40	17.98	12.22	29.52
7	35.95	1183.10	155.61	18.02	10.20	28.00
8	37.53	1181.59	147.22	17.56	11.98	29.22
Average	36.81	1205.24	156.84	17.9	12.88	30.38

As to the computation time, we know that K-View-V is much faster by using the SSI and FFT methods. From the Table 1, we can also see that K-View-D algorithm takes more computation time which is from 10 to 100 magnitudes of time that is used in other K-View based algorithms, because this algorithm needs to calculate the datagram (D_N) of original texture image. K-View-G should calculate the GLCM and features of each views in original images and characteristic views, so it is also much slower than other algorithms. Due to the combination of K-View-V and K-View-R in K-View-VR, the K-View-VR achieves the best classification accuracy at the cost of computation time.

V. Conclusions

In this paper, a combinatorial K-View based algorithm which attempts to incorporate the advantages of the K-View-R and K-View-V is presented. Experimental results show that the K-View-VR algorithm achieves encouraging classification accuracy compared with other K-View based algorithms, which is more robust and accurate.

From the image process of these algorithms, we may also find that these algorithms have some weaknesses or disadvantages in common. We concluded them and provided the possible improvement of these algorithms as following: firstly, these algorithms have a shortcoming by refering to supervise classification method. Because unsupervised method has more actually applied value, it is the right time to research the method which can determine the number of the texture class auto in future work. Secondly, by increasing the view size and number of characteristic views, the classification accuracy will be increased at the expense of processing time, so we need to explore an intelligent method which can determine these parameters in the future.

Acknowledgments

This study was supported in part by the China International Science and Technology Cooperation Project (Grant No. 2009DFA12290), and by the Huaihai Institute of Technology Natural Science Foundation (grant No. Z2009013).

References

[1] C Palm. Color texture classification by integrative Co-occurrence matrices. *Pattern Recognition*, 2004, 37(2004):965-976.
[2] CaS-CA, R Rangel-Kuoppa, M Reyes-Ayala, et al. high-order statistical texture analysis-font recognition applied. *Pattern Recognition Letters*, 2005, 26(2005):135-145.
[3] M Partio, B Cramariuc, M Gabbouj, et al. Rock Texture Retrieval Using Gray Level Co-occurrence Matrix. In *Proc. of 5th Nordic Signal Processing Symposium*, 2002.
[4] Oriol Pujol. Texture segmentation by statistic deformable models, *International Journal of Image and Graphics*. 4(3) (2004) 433- 452.
[5] J.Havlicek, D.Harding, and A. Bovik. The multi-component AM-FM image representation, *IEEE Transaction on image processing*. 5(6)(1996)1094-1100.
[6] Chen D, Wang L.1991. Texture features based on texture spectrum. *Pattern Recognition*, 24:391-399
[7] Haralick, R. M. and L. G. Shapiro, *Computer and Robot Vision*, (Volume I and II), Addison Wesley, 1993.
[8] C. C. Hung, S. Yang, and C. Laymon, Use of Characteristic Views in Image Classification. In *proceedings of the 16th International Conference on Pattern Recognition*, Quebec, Canada, August 11--15, 2002.
[9] Yang, S. and C.-c. Hung, Texture Classification in Remotely Sensed Images. In *proceedings IEEE Southeast Con* 2002, 2002: p. 62-66.
[10] Yang S., C.C. Hung, Image texture classification using datagrams and characteristic Views. In *proceedings of the 2003 ACM symposium on applied computing*, 2003, pp.22-26.
[11] H Liu, S Dai, E Song, et al., A New K-View Algorithm for Texture Image Classification Using Rotation-Invariant Feature. In *proceedings of the 2009 ACM Symposium on Applied Computing*, 2009: 914-921.
[12] H. Liu, Y. Lan, Q. Wang, R. Jin, E. Song, and C. Hung, "Fast weighted K-view-voting algorithm for image texture classification", *Opt. Eng*. 51, 027004 (2012), DOI:10.1117/1.OE.51.2.027004

[13] Y. L. Liu, J. Wang, X. Chen, Y. W. Guo, and Q. S. Peng. A robust and fast non-local means algorithm for image denoising. *Journal of Computer Science and Technology*, vol. 23, pp. 270-279, 2008.
[14] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple. In *proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1, pp. I511-I518. 2001.
[15] J hne, B. Digital image processing: concepts, algorithms and scientific applications. 1991: Springer-Verlag London, UK.
[16] P. Brodatz, Textures: A Photographic Album for Artists and Designers: New York: Dover Publications, 1966.

Yihua Lan received his Ph.D. degree in Computer Science from the School of Computer Science and Technology, Huazhong University of Science and Technology,Wuhan(HUST) in 2011, now he held teaching and research positions at School of Computer Engineering, Huaihai Institute of Technology (HHIT), Jiangsu, China. His research areas are image processing and analysis. His research interests include PDE methods for image processing, iterative methods, Krylov subspace methods, optimization algorithms, and artificial intelligence.

Yong Zhang received the M.S.degree in in School of computer science, SuZhou University in 2007, China. His M.S. subject is digital image processing. He held teaching and researching positions at the School of Computer Engineering, Huaihai Institute of Technology, where he has been an instructor. His research interests include image processing and machine vision.

Haozheng Ren received the M.S.degree in computer engineering, China, in 2006, from the Lanzhou University of Technology. She is currently a teacher of the School of Computer Engineering, Huaihai Institute of Technology, where she has been an Instructor since 2008. Her research interests include PDE methods for image processing, iterative methods, Krylov subspace methods, and parallel algorithms.